Detecting and Preventing Unauthorized Outbound Traffic

GCIH Gold Certification

Author: Brian Wippich, brian@wippich.com

Adviser: Dominicus Adriyanto Hindarto

Accepted: October 15th 2007

Table of Contents

Table of Figures

1. Introduction

When securing a network, most people are more concerned with controlling inbound traffic than outbound traffic. However, outbound traffic introduces some unique risks which should be considered when designing and securing a network.

This paper will describe some of the risks associated with outbound traffic, methods for securing this traffic, techniques for circumventing these controls, and methods for detecting and preventing these techniques. There is no way to eliminate all risk associated with outbound traffic short of closing all ports. However, a good understanding of these risks should allow you to make informed decisions on securing this traffic.

2. Overview

The following sections describe some typical outbound ports and risks associated with these open ports.

2.1 Typical Outbound Ports

The following list of ports may be open from within a corporate network to the Internet, although usually limited to specific source and destination addresses. Obviously, every network is designed and controlled differently. This is a small sample of some well known ports and their intended purpose. Several of the techniques mentioned in this paper communicate over these ports.

The Internet Assigned Numbers Authority (IANA) is responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources.

According to RFC 2780:

- Well known ports are those from 0 through 1023.
- Registered ports are those from 1024 through 49151.
- The Dynamic and/or Private Ports are those from 49152 through 65535

Protocol	Port Number	Description
TCP	20	FTP - File Transfer [Default Data]
TCP	21	FTP - File Transfer [Control]
TCP	22	SSH Remote Login Protocol
TCP	23	Telnet
TCP	25	Simple Mail Transfer
TCP	43	Who Is
TCP	80	HTTP
UDP	123	Network Time Protocol
TCP	443	HTTPS
TCP	989	FTP protocol, data, over TLS/SSL
TCP	990	FTP protocol, control, over TLS/SSL
TCP	1863	MSNP - Microsoft Instant Messaging (default)
TCP	3389	MS WBT Server (MS Terminal Server)
TCP	5050	Multimedia conference control tool - Default port

		for Yahoo Instant Messaging
TCP	5190	AOL Instant Messaging (default)
TCP	6660-7000	IRC

Figure 1: Sample Network Ports ("IANA", August 16, 2007)

2.2 Risks Associated with Outbound Traffic

To support normal business operations, networks are designed to allow traffic to flow from hosts located within a corporate network to resources on the Internet. This traffic is typically limited to specific TCP and UDP ports and source and destination IP addresses. In many cases, the traffic may communicate over a well known port, although not using the protocol for which is was originally intended. In other cases, the traffic may use the intended protocol but may be used to tunnel other unintended protocols or perhaps transport data for which the protocol was not intended. In all of these examples, there is a potential to circumvent security controls to conduct activities which may be against corporate policy or even worse to conduct illegal or harmful activities.

The following examples highlight a few of the risks associated with outbound network traffic. The risks are constantly changing as new technologies are developed, new techniques for subverting security controls are created and new illegally motivated financial incentives are discovered. This table identifies risks based on the CIA triad consisting of confidentiality, integrity and availability.

Risk	C	I	A

Inappropriate high bandwidth web usage (e.g. streaming media).			X
Access to malicious web sites resulting in host infection or compromise (e.g. worm, virus, compromise resulting in theft or tampering / destruction of sensitive data)	X	X	X
Access to BOT command and control center (BOT capabilities include DDoS, keylogger, SPAM, phishing, malware distribution)	X		X
Covert channels (e.g. insider or outsider information theft)	X		
Unauthorized remote access (insider information theft, weak authentication leading to host compromise)	X	X	
Access to externally compromised unofficial network resources (DNS, NTP, etc.)	X	X	X
Usage of clear text protocols over the Internet (sniffing, man in the middle attacks)	X		
Unmanaged IM usage (host infection or compromise)	X		X

Figure 2: Risk Matrix of Outbound Network Traffic based on CIA Triad

In addition to the above security risks, access to inappropriate web sites (e.g. porn, violence) introduces a legal risk of employee lawsuits.

3 Securing Outbound Traffic

Since this paper focuses on outbound traffic, a brief refresher on network firewall technologies seems appropriate.

Packet filtering devices employ rules based on layer 3 - IP addresses and layer 4 - source and destination ports including TCP flags (e.g. SYN, ACK, SYN ACK) to filter network traffic. The strengths of packet filters are that they are fast and inexpensive. Most routers include packet filtering capabilities. Although less secure then other firewall technologies, packet filtering still plays an important role within firewall architectures.

Stateful inspection firewalls are the most common firewalls in use today. In addition to the capabilities of packet filtering devices, these firewalls track the state of a network connection to further determine if a packet should be filtered. For example, a new request for service (SYN bit set) could be evaluated based on the firewall rule set, while subsequent packets related to this conversation may be automatically allowed based on the state table maintained by the firewall. More granular inspection of UDP and ICMP packets, including some layer 7 support, allow the firewall to more securely handle multi and mapped-port services like FTP and RPC-based services.

Proxy firewalls deny actual end-to-end connections between source and destination. The client establishes a connection to the proxy and the proxy establishes a connection to the destination. There are two types of proxy firewalls. Circuit-level proxies are a generic proxy mechanism that establishes a virtual connection between source and destination based on layer 3 and layer 4 headers. Client software must be adapted to work with this generic proxy. The most common circuit-level proxy in use today is SOCKS. Application-level proxies can inspect network traffic at layer 7 in addition to layers 3 and 4. This allows application-level proxies to filter traffic based on protocol compliance in addition to filtering based on application specific commands (e.g. HTTP methods). For example, a web proxy may only allow HTTP compliant traffic. Application-level proxies are substantially slower than other firewalls and every service requires its own specific proxy ("Network Firewall Architectures and Technologies", February 6, 2007).

4 Preventing Unauthorized Outbound Traffic

In the previous section, I described some typical security controls implemented by enterprises to address the risks associated with outbound traffic. Now I will identify some common techniques for bypassing these controls.

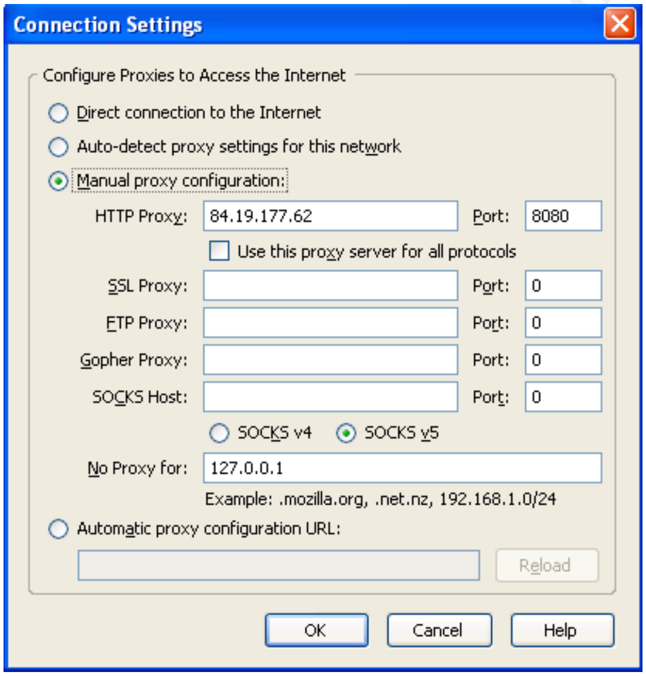
4.1 Proxy / Web Filter Avoidance

Most large enterprises implement some form of web filtering or proxy server solution to restrict access to inappropriate Internet web sites from within a corporate network. In the rare event that no web filtering or proxy server is implemented, firewalls offer almost no protection for malicious or inappropriate web use since http/https ports 80 and 443 are typically open outbound on corporate firewalls for all source and destination IP addresses.

4.1.1 Public Proxy Servers

For this paper I define web filtering as software or a device that interrogates web traffic as it leaves a corporate network. Using a database of categorized URLs, these solutions typically allow you to impose filtering policies based on categories of web sites. Examples of web filtering solutions include Websense, Surf Control and others. On the other hand, a proxy server may allow you to impose similar URL category restrictions, but also provides deeper packet inspection allowing more granular control and authentication. Examples of proxy server solutions include Squid, Blue Coat and Microsoft's web proxy.

One way to circumvent web filtering is to utilize an existing public proxy server available on the Internet. There is no shortage of available proxy sites listening on ports 80 (http), 443 (https), 3128 (squid proxy), 8080 (http alternate) and others. For example, the site http://www.publicproxyservers.com/page1.html lists several public proxy servers available for bypassing filtering solutions or to allow anonymous browsing. Simply add one of these proxy sites to the connection settings in your browser and you may be able to browse to previously filtered sites.



Connection Settings

Configure Proxies to Access the Internet

○ Direct connection to the Internet

○ Auto-detect proxy settings for this network

● Manual proxy configuration:

HTTP Proxy: 84.19.177.62 Port: 8080

□ Use this proxy server for all protocols

SSL Proxy: Port: 0

FTP Proxy: Port: 0

Gopher Proxy: Port: 0

SOCKS Host: Port: 0

○ SOCKS v4 ● SOCKS v5

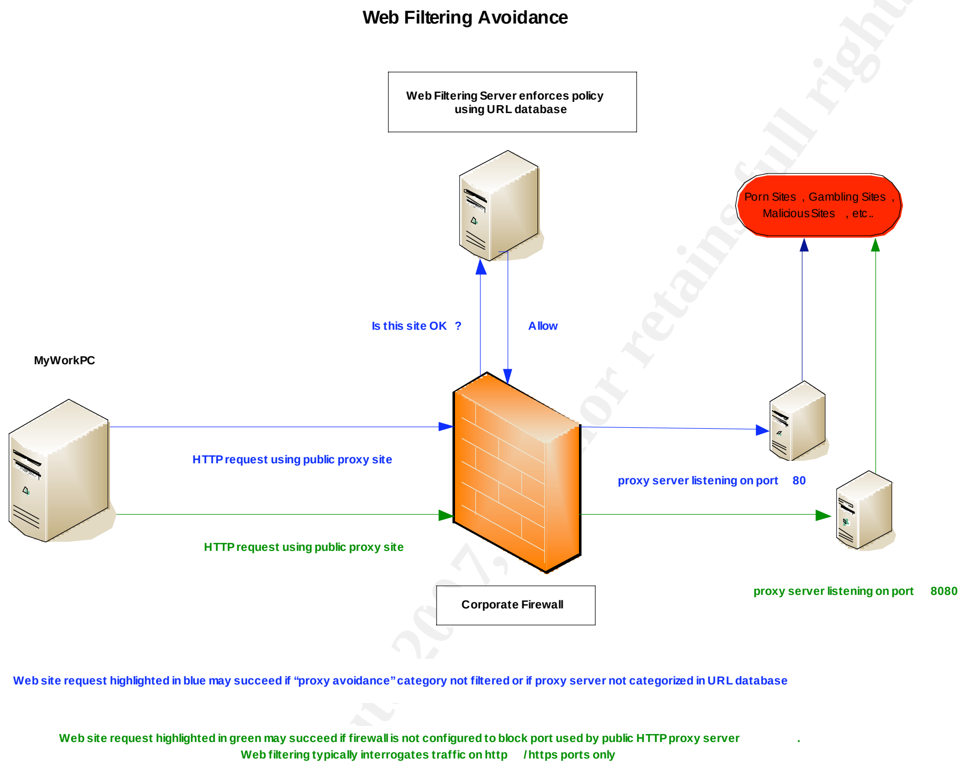
No Proxy for: 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

○ Automatic proxy configuration URL:

Figure 3: Browser Configuration for Manual Proxy Configuration

In a web filtering environment, this technique will work if no proxy server is implemented and either traffic is allowed to traverse directly through the corporate firewall on well known proxy server ports (8080, 3128) or the public proxy server listening on 80 or 443 is not categorized as "proxy avoidance" in the web filtering URL database or blocked by a corporate proxy server.


Web Filtering Avoidance
Web Filtering Server enforces policy using URL database
Porn Sites , Gambling Sites , Malicious Sites , etc..
Is this site OK ?
Allow
MyWorkPC
HTTP request using public proxy site
HTTP request using public proxy site
proxy server listening on port 80
proxy server listening on port 8080
Corporate Firewall

Web site request highlighted in blue may succeed if "proxy avoidance" category not filtered or if proxy server not categorized in URL database

Web site request highlighted in green may succeed if firewall is not configured to block port used by public HTTP proxy server .
Web filtering typically interrogates traffic on http /https ports only

Figure 4: Proxy Avoidance using Public Proxy Server

4.1.2 Personal Proxy Servers

Yet another possibility is to set up a personal proxy server on your own server outside of the corporate network. This could be done by installing open source proxy software like Squid http://www.squid-cache.org or a number of other proxy software solutions like Lozdodge http://www.proxy-avoidance.com/ or others available at http://www.softplatz.com/software/proxy-avoidance. These proxy servers can be configured to listen on any open outbound port in a corporate firewall. A personal proxy server site is more likely to be "uncategorized"instead of categorized within a "proxy avoidance" category, therefore these requests will most likely succeed in an environment which only employs web filtering. Assuming the personal proxy server accepts connections on port 80, this traffic would appear similar to the traffic from figure 4 identified in blue.

4.1.3 Using TOR (The Onion Router) for Proxy Avoidance

TOR (The Onion Router) is a free software implementation of second-generation onion routing – a system enabling its users to communicate anonymously on the Internet (wikepedia.org, 2007).

Using a form of Internet surveillance known as traffic analysis, you can determine who is communicating and to whom they are communicating. Additional information may also be inferred from packet sizes and timing. Encrypted traffic is not immune from traffic analysis since only the payload data is encrypted. The packet headers are typically not encrypted so that packets may be routed successfully to their final destination ("TOR Overview", n.d.).

Using a TOR network, packets traverse several computers (i.e. TOR nodes) taking an indirect path from source to destination. To identify this path, the TOR client must first retrieve a list of TOR nodes from a directory server.

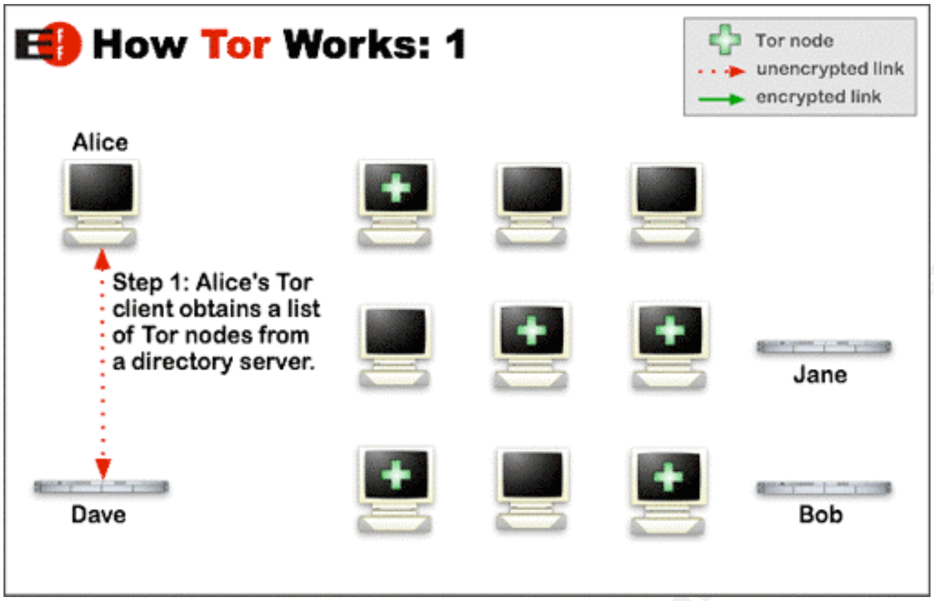

How Tor Works: 1
Tor node
unencrypted link
encrypted link
Alice
Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Jane
Dave
Bob

Figure 5: How TOR Works - Step 1 ("TOR Overview", n.d.)

Using the list of TOR nodes, an indirect path is established between source and final destination. Separate encryption keys are negotiated for each hop to ensure each TOR node can not determine any more than the previous or next hop in the path. No individual TOR node knows the complete path ("TOR Overview", n.d.).

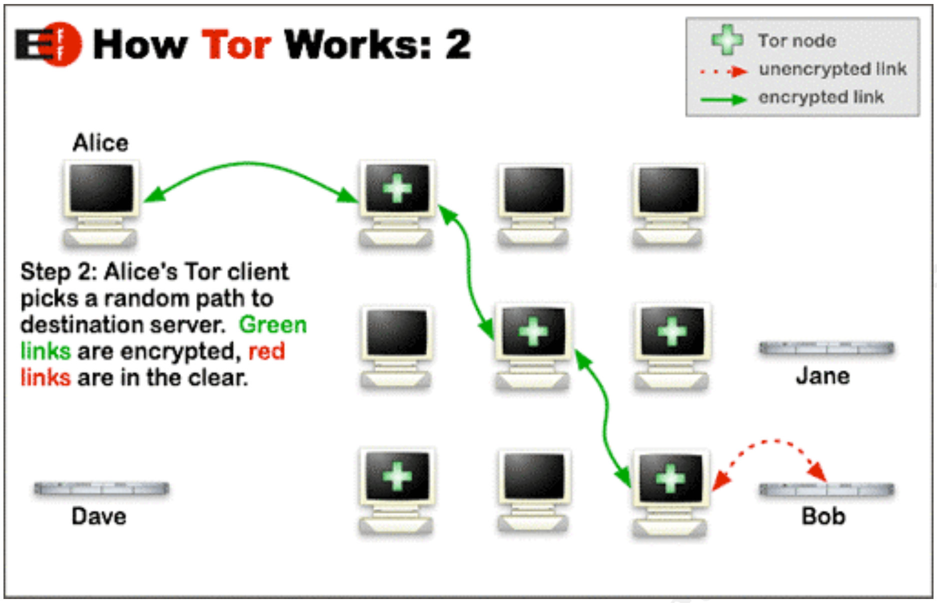

How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Figure 6: How TOR Works – Step 2 ("TOR Overview", n.d.)

There are valid reasons for using TOR to protect personal privacy. However, it's important to understand that TOR can also be used by an employee to circumvent corporate security controls. In the following example, TOR is used to circumvent controls for inappropriate web and IM usage. Although this example highlights an individual employee's desire to circumvent security controls, the same technique can be leveraged for communication by malicious software such as Bots. A Bot herder will often issue commands to the botnet by connecting through a chain of compromised hosts or by using anonymizing networks such as TOR (Brozycki & Bong, 2007)

Vidalia is a cross platform controller GUI for TOR. The Vidalia bundle available from http://vidalia-project.net/ includes Vidalia, TOR and Privoxy. Once installed, the Privoxy HTTP proxy server listens on port 8118 and the TOR client accepts SOCKS v4 connections on port 9050. To browse anonymously, the user's browser must be configured to use the Privoxy HTTP proxy server listening on 127.0.0.1:8118. The Privoxy HTTP proxy is configured by default to forward requests to the TOR client listening on 127.0.0.1:9050. The TOR client encrypts and forwards traffic to the first TOR router hop on its way to the final destination. Since the TOR client accepts SOCKS v4 connections, any socksified client, including most IM clients, can forward traffic directly to the TOR client listening on localhost:9050 ("Installing Vidalia on Windows", n.d.). The following diagram depicts the final installation and how unauthorized web and IM traffic is able to bypass firewall and web filtering restrictions.

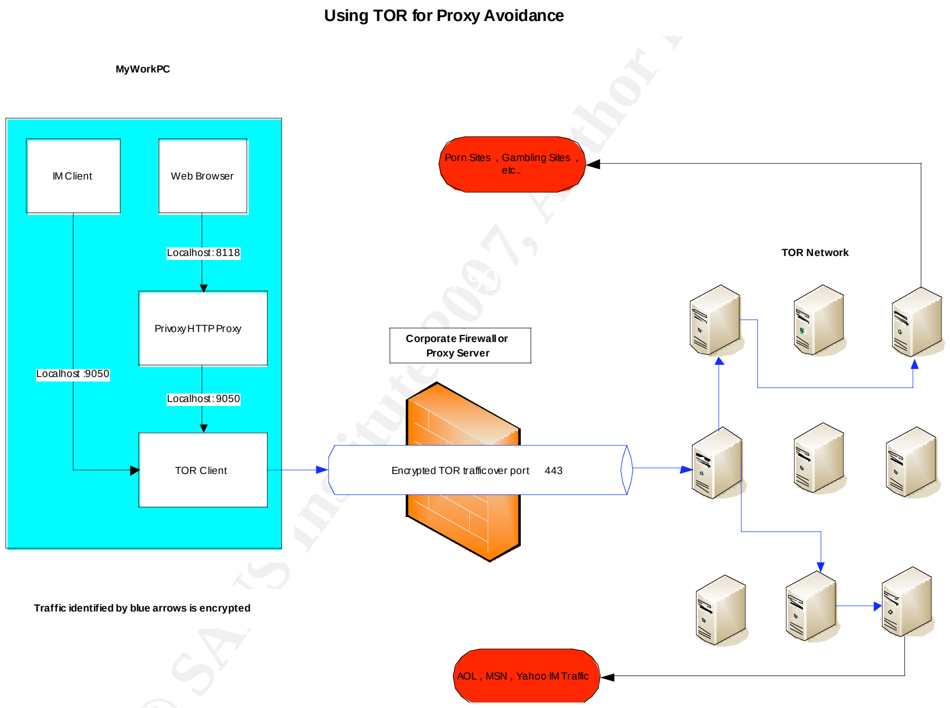
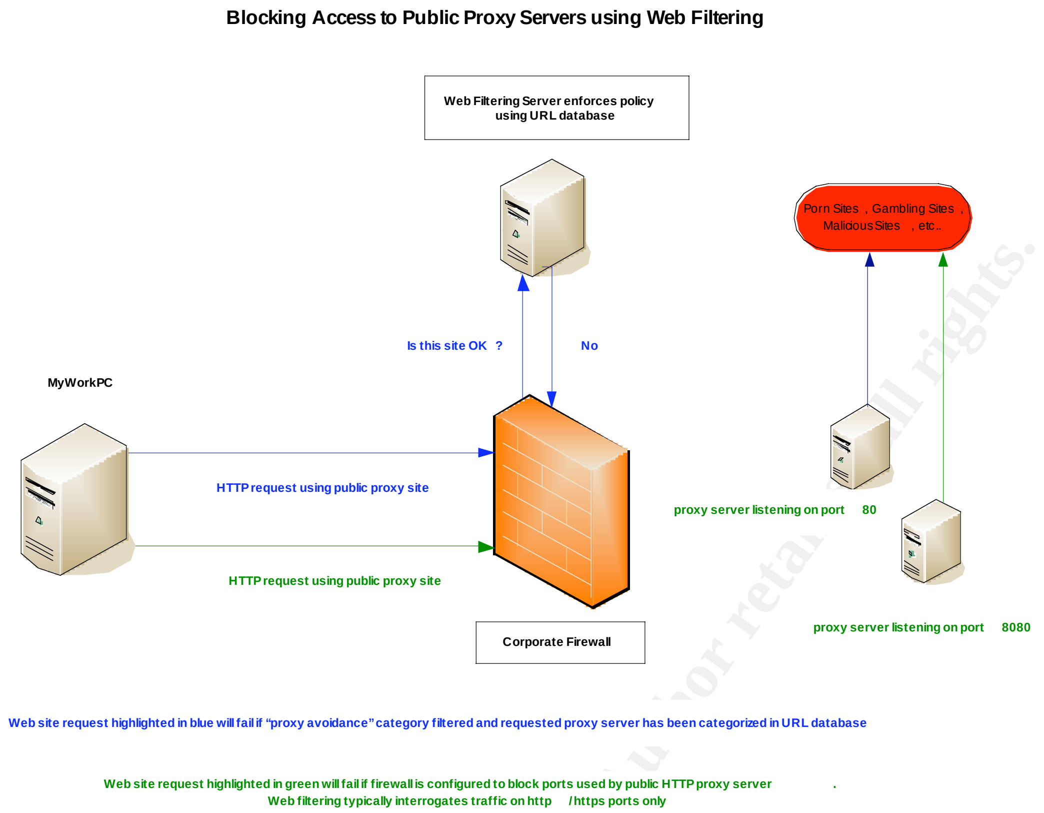

Using TOR for Proxy Avoidance
MyWorkPC
IM Client
Web Browser
Localhost: 8118
Privoxy HTTP Proxy
Localhost :9050
Localhost: 9050
TOR Client
Traffic identified by blue arrows is encrypted
Porn Sites , Gambling Sites , etc.
TOR Network
Corporate Firewall or Proxy Server
Encrypted TOR traffic over port 443
AOL , MSN , Yahoo IM Traffic

Figure 7: Using TOR for Proxy Avoidance

This technique is successful for several reasons. First, TOR can use several ports including 80 and 443, which are typically open on most firewalls. Attempts to block traffic based on native ports (e.g. IM) will not prevent traffic sent over the TOR network. Second, TOR traffic is encrypted therefore attempts to block traffic (e.g. IM) using protocol specific signatures will be unsuccessful. Finally, TOR traffic is routed over several hops. The first hop is a TOR node which may not be identified as an inappropriate destination. The TOR network is very dynamic making it difficult to identify all nodes within the network.

4.1.4 Detecting and Preventing Proxy / Web Filter Avoidance

Preventing access to public proxy servers is fairly simple since most commercial filtering solutions have identified popular "Proxy Avoidance" sites and allow you to restrict access to these sites. However, these same filtering solutions may only monitor traffic on ports 80 and 443, therefore all unnecessary outbound ports on your Internet facing firewalls especially well known proxy ports (e.g. 8080, 3128) must be closed as well. A properly configured firewall and filtering solution will block most attempts to connect using well known public proxy servers, although the list of available proxy servers is constantly changing so you can not be 100% assured that the provider of your URL filtering database has correctly categorized every available public proxy site.


Blocking Access to Public Proxy Servers using Web Filtering
Web Filtering Server enforces policy using URL database
Porn Sites , Gambling Sites , Malicious Sites , etc..
Is this site OK ?
No
MyWorkPC
HTTP request using public proxy site
HTTP request using public proxy site
proxy server listening on port 80
proxy server listening on port 8080
Corporate Firewall

Web site request highlighted in blue will fail if “proxy avoidance” category filtered and requested proxy server has been categorized in URL database

Web site request highlighted in green will fail if firewall is configured to block ports used by public HTTP proxy server . Web filtering typically interrogates traffic on http /https ports only

Figure 8: Blocking Access to Public Proxy Servers using Web Filtering

Preventing unauthorized web traffic via personal proxy servers is slightly more difficult. Once again, all unnecessary outbound ports on your Internet facing firewalls must be closed. Web filtering software can be configured to block traffic to “uncategorized” sites, although this may also have the unintended side affect of blocking access to legitimate sites which have not yet been categorized by your URL filtering vendor. Implementing a corporate proxy server inside your network could prevent this technique assuming users were unable to access the Internet directly. Since you can only define one proxy server in your browser, users would be unable to define both the corporate proxy and personal proxy server.

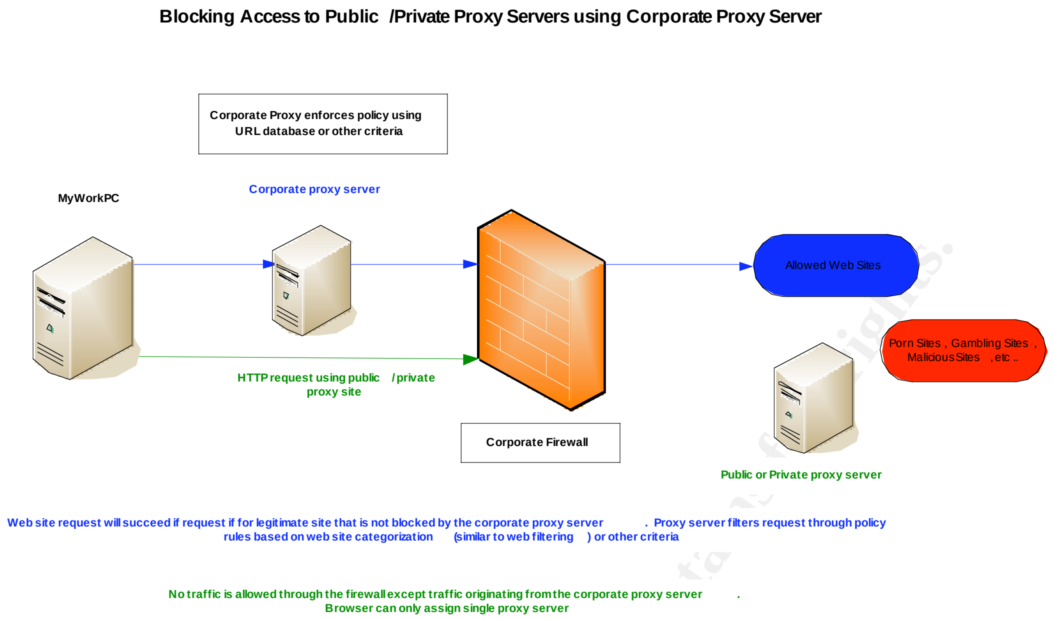

Blocking Access to Public /Private Proxy Servers using Corporate Proxy Server
Corporate Proxy enforces policy using URL database or other criteria
MyWorkPC
Corporate proxy server
Allowed Web Sites
Porn Sites , Gambling Sites , Malicious Sites , etc ..
HTTP request using public /private proxy site
Corporate Firewall
Public or Private proxy server
Web site request will succeed if request if for legitimate site that is not blocked by the corporate proxy server . Proxy server filters request through policy rules based on web site categorization (similar to web filtering) or other criteria
No traffic is allowed through the firewall except traffic originating from the corporate proxy server . Browser can only assign single proxy server

Figure 9: Blocking Access to Public/Private Proxy Servers using Corporate Proxy Server

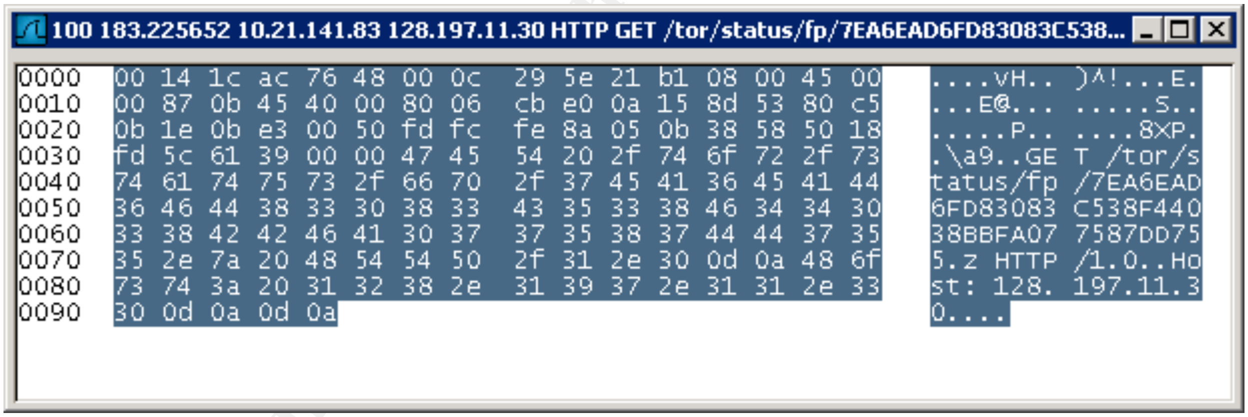
Effectively identifying and preventing TOR traffic on a corporate network is more difficult than preventing the use of unauthorized proxy servers. Blocking all unnecessary ports on the firewall is a necessity since TOR can communicate out on a number of ports. Some IDS, particular those based on analysis of layer 4 protocols like Cisco's NetFlow protocol (e.g. Arbor Network's Peakflow), is difficult since these signatures must be updated to include new Onion routers added to the TOR network.

The TOR protocol requires that TOR clients periodically retrieve live network-status documents from directory authorities and directory mirrors. Each client contains a list of directory authorities, although once they retrieve these documents subsequent requests are made from directory mirrors. Once network-status documents are received from at least half of the directory authorities, the client then requests router descriptors from the directory mirrors. TOR circuits are not built until the client has network-statuses from than half the directory authorities and descriptors for at least 1/4 of the servers believed to be running. ("Tor directory protocol, version 1", 7/20/2006)

The requests for network-status documents and router descriptors are made in clear text therefore signatures are available to identify and prevent this activity. Web URL vendors and signature based IDS can identify and possibly prevent TOR traffic by looking for these requests.

Sample network-status document request

The network-status published by a host with fingerprint
    <F> should be available at:
      http://<hostname>/tor/status/fp/<F>.z



100 183.225652 10.21.141.83 128.197.11.30 HTTP GET /tor/status/fp/7EA6EAD6FD83083C538...

```
0000  00 14 1c ac 76 48 00 0c  29 5e 21 b1 08 00 45 00   ....vH.. )^!...E.
0010  00 87 0b 45 40 00 80 06  cb e0 0a 15 8d 53 80 c5   ...E@... .....S..
0020  0b 1e 0b e3 00 50 fd fc  fe 8a 05 0b 38 58 50 18   .....P.. ....8XP.
0030  fd 5c 61 39 00 00 47 45  54 20 2f 74 6f 72 2f 73   .\a9..GE T /tor/s
0040  74 61 74 75 73 2f 66 70  2f 37 45 41 36 45 41 44   tatus/fp /7EA6EAD
0050  36 46 44 38 33 30 38 33  43 35 33 38 46 34 34 30   6FD83083 C538F440
0060  33 38 42 42 46 41 30 37  37 35 38 37 44 44 37 35   38BBFA07 7587DD75
0070  35 2e 7a 20 48 54 54 50  2f 31 2e 30 0d 0a 48 6f   5.z HTTP /1.0..Ho
0080  73 74 3a 20 31 32 38 2e  31 39 37 2e 31 31 2e 33   st: 128. 197.11.3
0090  30 0d 0a 0d 0a                                     0....
```

Figure 10: TOR Packet Capture – network-status document request

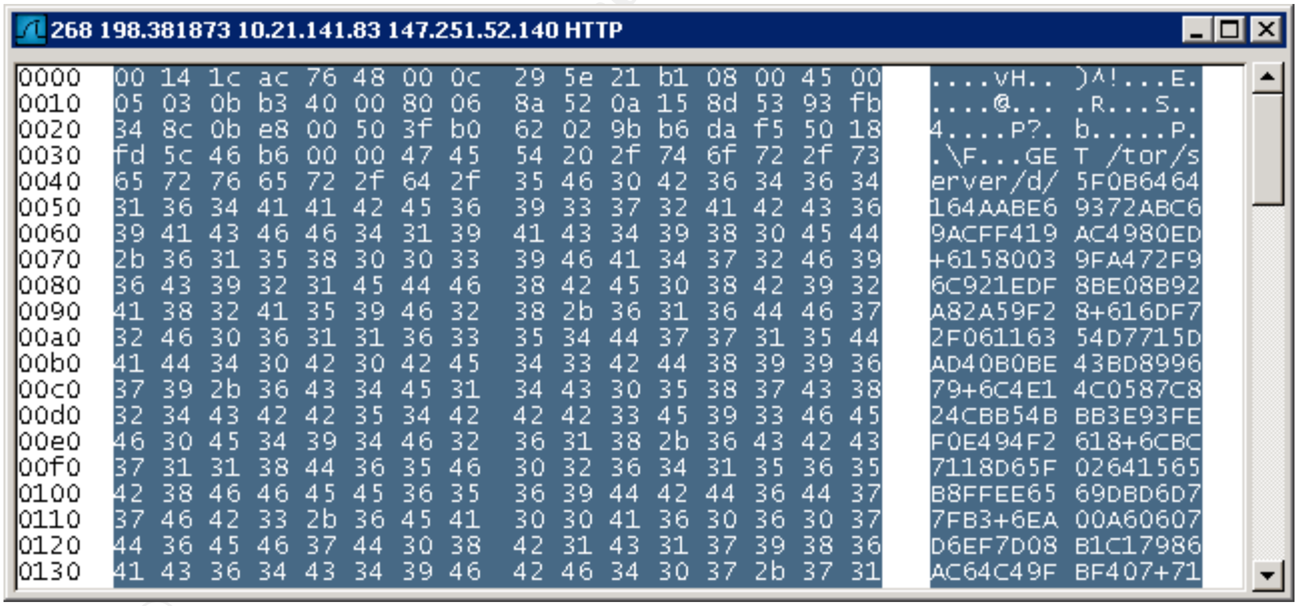
Sample Cisco IDS "TOR Client Activity" Alert

```
signature:   description=TOR Client Activity  id=5816  version=S256
  subsigId: 0
  sigDetails: /tor/status/fp/
  marsCategory: Info/Misc
interfaceGroup: vs0
vlan: 0
participants:
  attacker:
        addr: 10.XXX.XXX.XXX  locality=Inside
        port: 3200
      target:
        addr: 213.161.192.240  locality=Outside
        port: 80
        os:   idSource=unknown  type=unknown  relevance=relevant
    context:
      fromAttacker:
000000  47 45 54 20 2F 74 6F 72  2F 73 74 61 74 75 73 2F  GET /tor/status/
000010  66 70 2F 33 38 44 34 46  35 46 43 46 37 42 31 30  fp/38D4F5FCF7B10
000020  32 33 32 32 38 42 38 39  35 45 41 35 36 45 44 45  23228B895EA56EDE
000030  37 44 35 43 43 44 43 41  46 33 32 2E 7A 20 48 54  7D5CCDCAF32.z HT
000040  54 50 2F 31 2E 30 0D                              TP/1.0.
```

Sample router descriptor request

```
The most recent descriptors for servers with identity fingerprints
   <F1>,<F2>,<F3> should be available at:
      http://<hostname>/tor/server/fp/<F1>+<F2>+<F3>.z
```

Figure 10: TOR Packet Capture – network-status document request

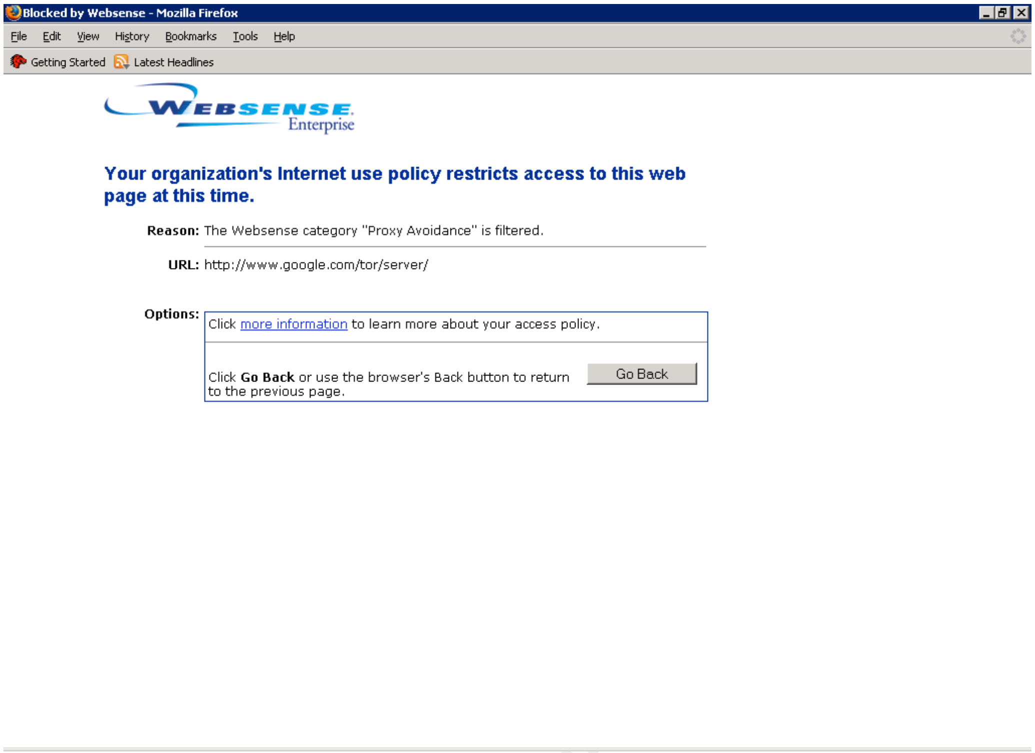


268 198.381873 10.21.141.83 147.251.52.140 HTTP

```
0000  00 14 1c ac 76 48 00 0c  29 5e 21 b1 08 00 45 00  ....vH.. )^!...E.
0010  05 03 0b b3 40 00 80 06  8a 52 0a 15 8d 53 93 fb  ....@... .R...S..
0020  34 8c 0b e8 00 50 3f b0  62 02 9b b6 da f5 50 18  4....P?. b.....P.
0030  fd 5c 46 b6 00 00 47 45  54 20 2f 74 6f 72 2f 73  .\F...GE T /tor/s
0040  65 72 76 65 72 2f 64 2f  35 46 30 42 36 34 36 34  erver/d/ 5F0B6464
0050  31 36 34 41 41 42 45 36  39 33 37 32 41 42 43 36  164AABE6 9372ABC6
0060  39 41 43 46 46 34 31 39  41 43 34 39 38 30 45 44  9ACFF419 AC4980ED
0070  2b 36 31 35 38 30 30 33  39 46 41 34 37 32 46 39  +6158003 9FA472F9
0080  36 43 39 32 31 45 44 46  38 42 45 30 38 42 39 32  6C921EDF 8BE08B92
0090  41 38 32 41 35 39 46 32  38 2b 36 31 36 44 46 37  A82A59F2 8+616DF7
00a0  32 46 30 36 31 31 36 33  35 34 44 37 37 31 35 44  2F061163 54D7715D
00b0  41 44 34 30 42 30 42 45  34 33 42 44 38 39 39 36  AD40B0BE 43BD8996
00c0  37 39 2b 36 43 34 45 31  34 43 30 35 38 37 43 38  79+6C4E1 4C0587C8
00d0  32 34 43 42 42 35 34 42  42 42 33 45 39 33 46 45  24CBB54B BB3E93FE
00e0  46 30 45 34 39 34 46 32  36 31 38 2b 36 43 42 43  F0E494F2 618+6CBC
00f0  37 31 31 38 44 36 35 46  30 32 36 34 31 35 36 35  7118D65F 02641565
0100  42 38 46 46 45 45 36 35  36 39 44 42 44 36 44 37  B8FFEE65 69DBD6D7
0110  37 46 42 33 2b 36 45 41  30 30 41 36 30 36 30 37  7FB3+6EA 00A60607
0120  44 36 45 46 37 44 30 38  42 31 43 31 37 39 38 36  D6EF7D08 B1C17986
0130  41 43 36 34 43 34 39 46  42 46 34 30 37 2b 37 31  AC64C49F BF407+71
```

Figure 11: TOR Packet Capture – router descriptor request

Sample URL Filtering "Proxy Avoidance" Alert (simulated by entering any domain name with /tor/server/ in the path)


Blocked by Websense - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Getting Started
Latest Headlines
WEBSENSE Enterprise

Your organization's Internet use policy restricts access to this web page at this time.

Reason: The Websense category "Proxy Avoidance" is filtered.

URL: http://www.google.com/tor/server/

Options: Click more information to learn more about your access policy.

Click **Go Back** or use the browser's Back button to return to the previous page.

Figure 12: URL Filtering of TOR Traffic

Identifying and blocking the execution of the TOR executable at the desktop may also be possible, although renaming this executable may circumvent this control. Blocking non conforming HTTP traffic on 80 and uncategorized traffic on port 443 using a proxy server in conjunction with a URL database may offer some reasonably good prevention although false positives are inevitable.

The following Snort IDS signatures are also available to identify TOR traffic on your corporate network.

Bleeding Edge Policy TOR 1.0 Server Key Retrieval

http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2002950&scope=text

Bleeding Edge Policy TOR 1.0 Status Update

http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2002951&scope=text

Bleeding Edge Policy TOR 1.0 Inbound Circuit Traffic

http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2002952&scope=text

Bleeding Edge Policy TOR 1.0 Outbound Circuit Traffic

http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2002953&scope=text

4.2 Unauthorized Remote Access

Most corporations provide their users with a method to access network resources remotely. This is typically accomplished using IPSec or SSL VPN solutions. For various reasons, some employees may use alternate unsanctioned methods to access the network. Many remote access solutions will not work since most corporate firewalls are configured to block inbound connections on the required remote access ports. However, there are a few solutions that enable remote access by utilizing ports already open for outbound access. To illustrate this point we will focus on software from LogMeIn.com.

4.2.1 LogMeIn

LogMeIn provides access to a remote host using a standard web browser. There are three key components: Client, Remote Host and LogMeIn Gateway.

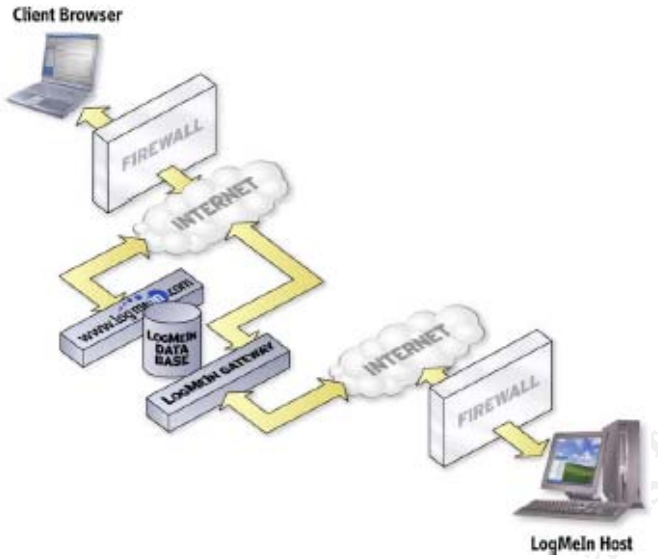
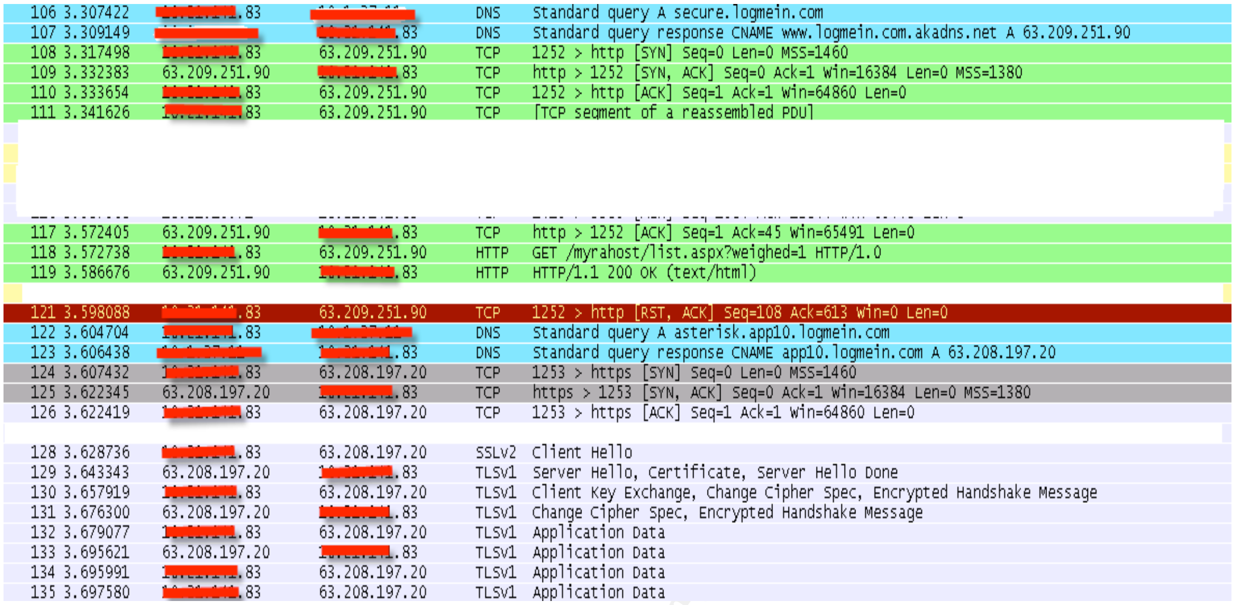


Figure 13: LogMeIn Network Architecture (Anka, Marton, 2006)

The remote host maintains a constant SSL connection with one of the LogMeIn gateways. This connection is initiated by the host initially on http and then on https. The firewall allows this traffic since it appears like normal web traffic initiated from inside the network.

The client browser establishes a connection to www.logmein.com. Once authenticated, the user can select one of their remote hosts to access. The gateway then forwards the encrypted traffic between the client and remote host (Anka, Marton, 2006).

I ran the following packet capture after installing the LogMeIn software which runs on the remote host.



```
106 3.307422  [redacted].83     [redacted]        DNS    Standard query A secure.logmein.com
107 3.309149  [redacted]        [redacted].83     DNS    Standard query response CNAME www.logmein.com.akadns.net A 63.209.251.90
108 3.317498  [redacted].83     63.209.251.90     TCP    1252 > http [SYN] Seq=0 Len=0 MSS=1460
109 3.332383  63.209.251.90     [redacted].83     TCP    http > 1252 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1380
110 3.333654  [redacted].83     63.209.251.90     TCP    1252 > http [ACK] Seq=1 Ack=1 Win=64860 Len=0
111 3.341626  [redacted].83     63.209.251.90     TCP    [TCP segment of a reassembled PDU]
117 3.572405  63.209.251.90     [redacted].83     TCP    http > 1252 [ACK] Seq=1 Ack=45 Win=65491 Len=0
118 3.572738  [redacted].83     63.209.251.90     HTTP   GET /myrahost/list.aspx?weighed=1 HTTP/1.0
119 3.586676  63.209.251.90     [redacted].83     HTTP   HTTP/1.1 200 OK (text/html)
121 3.598088  [redacted].83     63.209.251.90     TCP    1252 > http [RST, ACK] Seq=108 Ack=613 Win=0 Len=0
122 3.604704  [redacted].83     [redacted]        DNS    Standard query A asterisk.app10.logmein.com
123 3.606438  [redacted]        [redacted].83     DNS    Standard query response CNAME app10.logmein.com A 63.208.197.20
124 3.607432  [redacted].83     63.208.197.20     TCP    1253 > https [SYN] Seq=0 Len=0 MSS=1460
125 3.622345  63.208.197.20     [redacted].83     TCP    https > 1253 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1380
126 3.622419  [redacted].83     63.208.197.20     TCP    1253 > https [ACK] Seq=1 Ack=1 Win=64860 Len=0
128 3.628736  [redacted].83     63.208.197.20     SSLv2  Client Hello
129 3.643343  63.208.197.20     [redacted].83     TLSv1  Server Hello, Certificate, Server Hello Done
130 3.657919  [redacted].83     63.208.197.20     TLSv1  Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
131 3.676300  63.208.197.20     [redacted].83     TLSv1  Change Cipher Spec, Encrypted Handshake Message
132 3.679077  [redacted].83     63.208.197.20     TLSv1  Application Data
133 3.695621  63.208.197.20     [redacted].83     TLSv1  Application Data
134 3.695991  [redacted].83     63.208.197.20     TLSv1  Application Data
135 3.697580  [redacted].83     63.208.197.20     TLSv1  Application Data
```

Figure 14: LogMeIn Remote Host Packet Capture

The IP address ending in .83 is the remote host. As you can see, the initial HTTP request is made to secure.logmein.com. Once this connection is established, the remote host requests a page from secure.logmein.com containing the names of several other LogMeIn gateway servers. In this case, asterisk.app10.logmein.com was chosen. Now the remote host negotiates an SSL connection with asterisk.app10.logmein.com. The remote host periodically contacts asterisk.app10.logmein.com to keep this connection open. This persistent connection informs the LogMeIn gateway that this remote host is available to be accessed by an authorized client browser.

The purpose of this section is not to question the inherent security of these types of remote access solution. LogMeIn actually seems to have reasonably strong security controls as described in their security whitepaper https://secure.logmein.com/wp_lmi_security.pdf. However, the security may still not meet corporate security standards for remote access like two-factor authentication requirements. In addition, these solutions also provide a method of moving files potentially containing sensitive data; therefore attempts to monitor data leaving an organization should consider this channel.

4.2.2 Detecting and Preventing Unauthorized Remote Control Software

Preventing LogMeIn activity is fairly easy since the solution requires an initial connection to secure.logmein.com. A similar solution from GotoMyPC also requires an initial connection to poll.gotomypc.com. This traffic can be blocked on the firewall, proxy server or URL filtering solutions. Blocking at the firewall by IP address may be more challenging than blocking based on domain name since it appears LogMeIn is now using Akami for load balancing. As indicated in the packet capture from figure 14 above, the domain secure.logmein.com initially resolved to 63.209.251.90, however a more recent nslookup request returned the following different results.

```
Non-authoritative answer:
Name:      www.logmein.com.akadns.net
Address:   69.25.20.193
Aliases:   secure.logmein.com
```

A web filtering or proxy server solution would be more effective than using a firewall ACL, since this would allow you to block using the fully qualified domain name (i.e. secure.logmein.com) instead of all of the individual IP addresses to which this address may resolve.

4.3 Covert Communication Channels

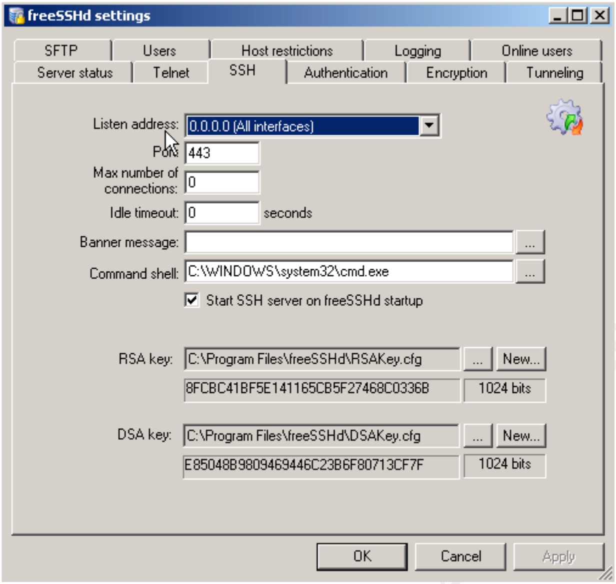
A **tunneling protocol** is a network protocol which encapsulates one protocol or session inside a higher layer protocol or a protocol at the same layer (wikepedia.org, 2007). Some network protocols are especially good candidates for tunneling outbound traffic either because the traffic is considered normal or because the traffic is encrypted which makes detection more difficult.

4.3.1 SSH Tunneling

HTTP (port 80) and HTTPS (port 443) traffic is typically allowed for most if not all employees. The following technique takes advantage of the open HTTPS port by tunneling traffic through an SSH tunnel over port 443.

An SSH tunnel can be used to circumvent corporate firewall and web filtering controls to access any number of services (restricted web sites, IM, FTP, Telnet, etc.) over the Internet from within a corporate network. This requires installing and configuring some software on a computer located on the Internet, perhaps an employee's home computer. The setup on the computer located within the corporate network is trivial.

To test this scenario, I installed an SSH, HTTP proxy and SOCKS proxy server on the home computer MyHomePC. For the SSH server, I installed freeSSHd from http://freesshd.com. I configured this SSH server to listen on port 443 instead of the standard SSH port 22. This would work equally well on port 22 assuming this port was also open on the firewall.

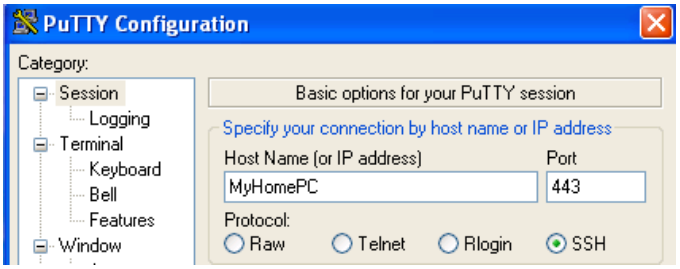


Listen address: 0.0.0.0 (All interfaces)
Port: 443
Max number of connections: 0
Idle timeout: 0 seconds
Banner message:
Command shell: C:\WINDOWS\system32\cmd.exe
☑ Start SSH server on freeSSHd startup
RSA key: C:\Program Files\freeSSHd\RSAKey.cfg
8FCBC41BF5E141165CB5F27468C0336B 1024 bits
DSA key: C:\Program Files\freeSSHd\DSAKey.cfg
E85048B9809469446C23B6F80713CF7F 1024 bits

Figure 15: SSH Tunneling - freeSSHd Configuration

For the HTTP proxy, I installed a Squid proxy server from http://www.squid-cache.org/ using the default port of 3128. For the SOCKS proxy, I installed WinSocks from http://proxylabs.netwu.com/ using the default port of 1080.

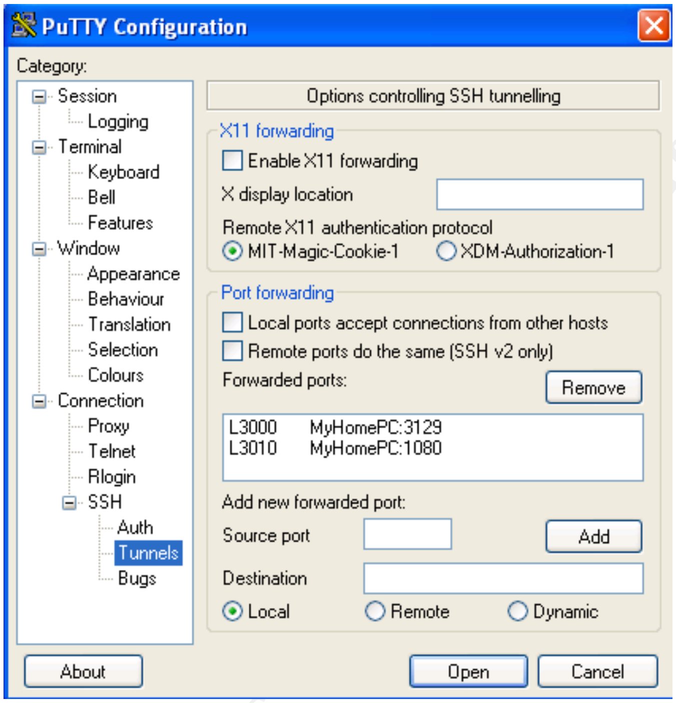
On the work computer MyWorkPC, I downloaded the Putty SSH client from http://www.chiark.greenend.org.uk/~sgtatham/putty. In some cases and SSH client may already exist on standard corporate builds. I set up a Putty session using the following networking and port forwarding rules.


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Basic options for your PuTTY session
Specify your connection by host name or IP address

Host Name (or IP address): MyHomePC

Port: 443

Protocol: ○ Raw ○ Telnet ○ Rlogin ● SSH



□ Enable X11 forwarding

X display location:

Remote X11 authentication protocol: ● MIT-Magic-Cookie-1 ○ XDM-Authorization-1

□ Local ports accept connections from other hosts

□ Remote ports do the same (SSH v2 only)

Forwarded ports:

```
L3000    MyHomePC:3129
L3010    MyHomePC:1080
```

Source port:

Destination:

● Local ○ Remote ○ Dynamic

Figure 16: SSH Tunneling - Putty SSH Client Configuration

To open the SSH tunnel, I initiated an SSH to the MyHomePC over port 443 using the above Putty profile. Once authenticated, an SSH tunnel now exists between MyWorkPC and MyHomePC. This tunnel was established using port 443 and all traffic sent via the tunnel is hidden from network IDS or firewalls. In addition, MyWorkPC is now listening for traffic on localhost ports 3000 and 3010.

- Traffic directed to MyWorkPC:3000 will be sent through the SSH tunnel to the Squid HTTP proxy server on MyHomePC:3129.
- Traffic directed to MyWorkPC:3010 will be sent through the SSH tunnel to the SOCKS proxy server on MyHomePC:1080.

The final step is to configure the browser on MyWorkPC to tunnel all http/https traffic through the SSH tunnel to the HTTP proxy on MyHomePC. This is done by configuring the browser to use 127.0.0.1:3000 as an HTTP proxy. In addition, traffic from any SOCKS aware applications (e.g. Instant Messaging, FTP, Telnet, etc.) can be tunneled through the SSH tunnel to the SOCKS proxy on MyHomePC. This is done by configuring the application to us 127.0.0.1:3010 as a SOCKS proxy (Chirico, Mike, 2007).

The following diagram provides a visual representation of this technique.

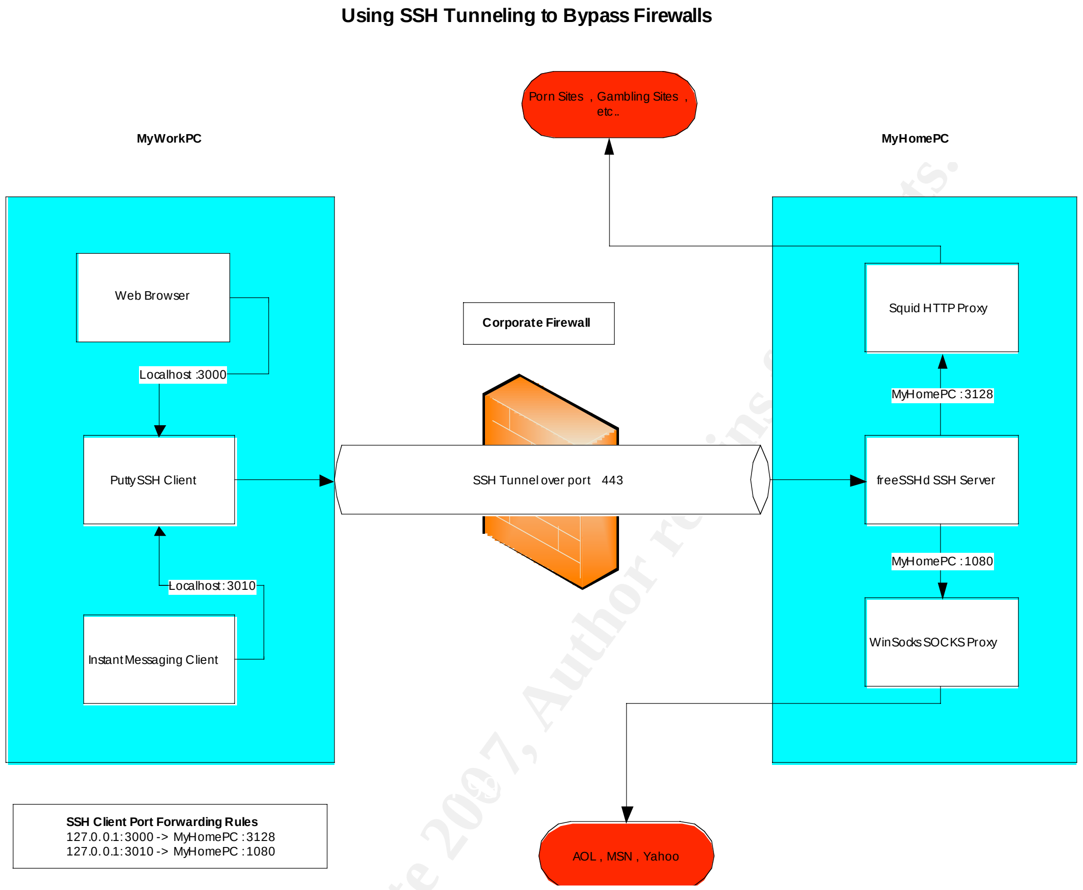


Figure 17: Using SSH Tunneling to Bypass Firewalls

4.3.2 Reverse HTTP Shell

Reverse HTTP Shell is a backdoor that allows an attacker to remotely gain shell access to a computer. Once the backdoor is installed on the compromised computer, the attacker can execute commands and see the output of these commands without logging into the compromised computer. All traffic between the compromised computer and the attacker's computer appears like normal web traffic which is typically allowed out most corporate firewalls.

The Reverse HTTP Shell consists of two software components a SLAVE and MASTER. The SLAVE is a backdoor which runs on the internal host. The SLAVE will communicate with the MASTER running on the external host using HTTP. The SLAVE issues a GET or POST request to the MASTER to retrieve commands. The output of these commands is returned to the MASTER in another GET or POST request. Web traffic is converted to a Base64 type value and passed as a value for a cgi-string to prevent caching (van Hauser). As illustrated below, traffic is allowed to pass through the firewall over port since this port is open and traffic appears like normal web traffic.

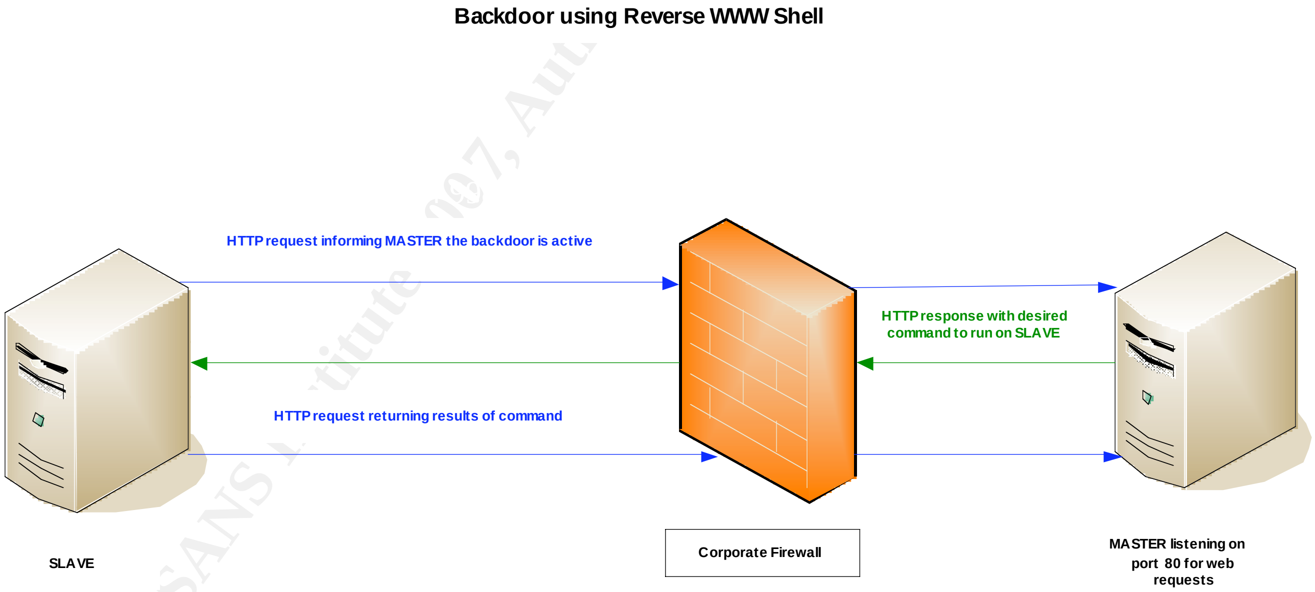

Backdoor using Reverse WWW Shell
HTTP request informing MASTER the backdoor is active
HTTP response with desired command to run on SLAVE
HTTP request returning results of command
SLAVE
Corporate Firewall
MASTER listening on port 80 for web requests

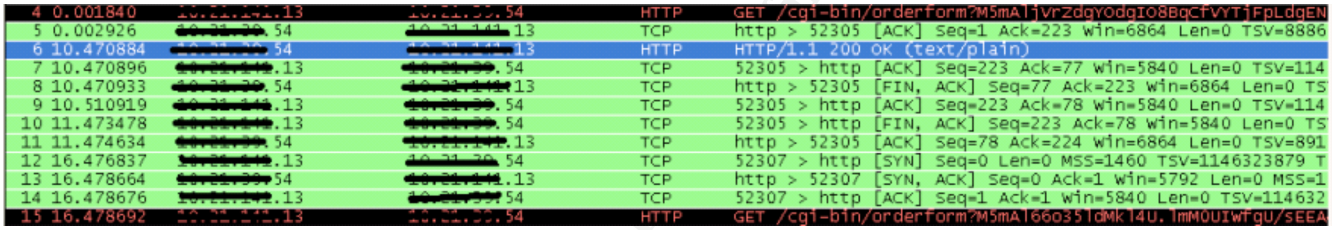
Figure 18: Backdoor using Reverse WWW Shell

The following packet capture further demonstrates the interaction between SLAVE and MASTER.

Line 4 – The SLAVE (.13) issues a GET request to the MASTER (.54) to retrieve a command. Web request is a Base 64 like structure in query string.

Line 6 – The MASTER (.54) sends the "ls"command in an HTTP response

Line 15 – The SLAVE (.13) returns the ls output within another GET request. Web request is Base 64 encoded in query string.



```
 4 0.001840  [redacted].13   [redacted].54   HTTP  GET /cgi-bin/orderform?M5mAljvrZdgYOdgIO8BqCfVYTjFpLdgEN
 5 0.002926  [redacted].54   [redacted]13    TCP   http > 52305 [ACK] Seq=1 Ack=223 Win=6864 Len=0 TSV=8886
 6 10.470884 [redacted].54   [redacted]13    HTTP  HTTP/1.1 200 OK (text/plain)
 7 10.470896 [redacted].13   [redacted].54   TCP   52305 > http [ACK] Seq=223 Ack=77 Win=5840 Len=0 TSV=114
 8 10.470933 [redacted].54   [redacted]13    TCP   http > 52305 [FIN, ACK] Seq=77 Ack=223 Win=6864 Len=0 TS
 9 10.510919 [redacted].13   [redacted].54   TCP   52305 > http [ACK] Seq=223 Ack=78 Win=5840 Len=0 TSV=114
10 11.473478 [redacted].13   [redacted].54   TCP   52305 > http [FIN, ACK] Seq=223 Ack=78 Win=5840 Len=0 TS
11 11.474634 [redacted].54   [redacted].13   TCP   http > 52305 [ACK] Seq=78 Ack=224 Win=6864 Len=0 TSV=891
12 16.476837 [redacted].13   [redacted] 54   TCP   52307 > http [SYN] Seq=0 Len=0 MSS=1460 TSV=1146323879 T
13 16.478664 [redacted]54    [redacted].13   TCP   http > 52307 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1
14 16.478676 [redacted].13   [redacted]54    TCP   52307 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=114632
15 16.478692 [redacted].13   [redacted].54   HTTP  GET /cgi-bin/orderform?M5mAl66o35ldMkl4U.lmM0UIwfgU/sEEA
```

Figure 19: Reverse WWW Shell Packet Capture

The following variables can be configured to avoid detection or communicate through a proxy server.

```
# GENERAL CONFIG
$MODE="GET";                  # GET or POST
$CGI_PREFIX="/cgi-bin/orderform";# should look like a valid cgi.
$MASK="vi";             # for masking the program's process name
$PASSWORD="THC";        # anything, nothing you have to rememeber
                        # (not a real "password" anyway)
#
# MASTER CONFIG (specific for the MASTER)
$LISTEN_PORT=80; # on which port to listen (80 [needs root] or 8080)
$SERVER="localhost";   # the host to run on (ip/dns) (the SLAVE needs this!)
#
# SLAVE CONFIG (specific for the SLAVE)
#
$SHELL="/bin/sh -i";   # program to execute (e.g. /bin/sh)
$DELAY="3";      # time to wait for output after your command(s)
#$TIME="14:39";         # time when to connect to the master (unset if now)
#$DAILY="yes";          # tries to connect once daily if set with something
#$PROXY="127.0.0.1";    # set this with the Proxy if you must use one
#$PROXY_PORT="3128";    # set this with the Proxy Port if you must use one
#$PROXY_USER="user";    # username for proxy authentication
#$PROXY_PASSWORD="pass";# password for proxy authentication
#$DEBUG="yes";          # for debugging purpose, turn off when in production
```

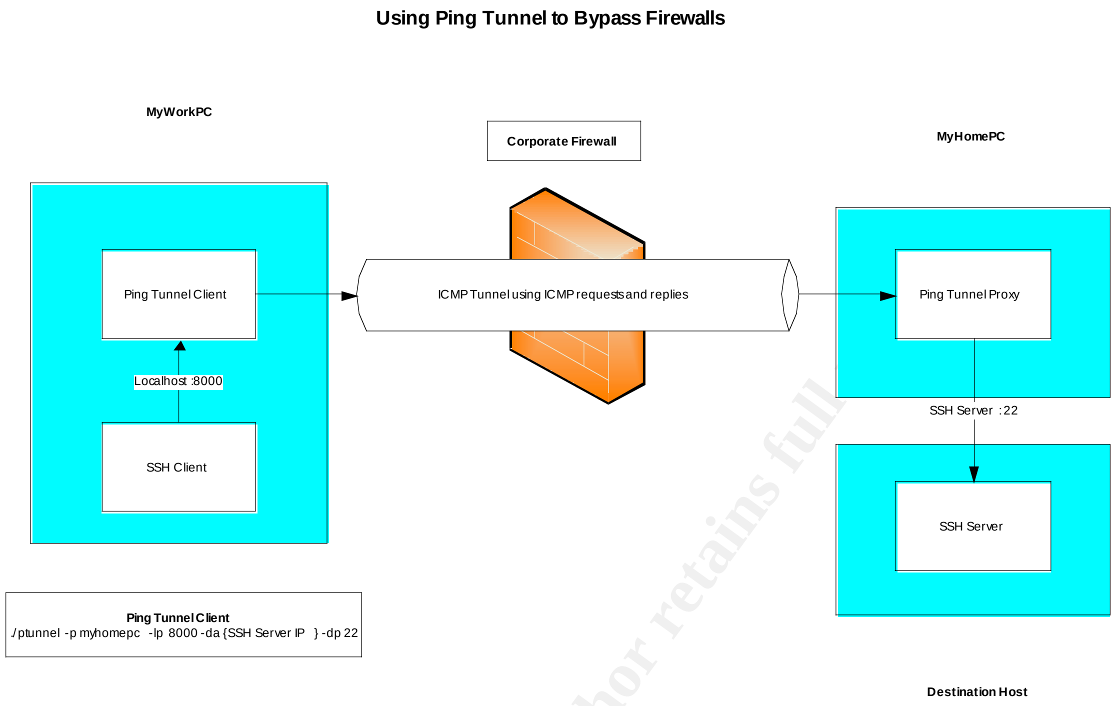
```
        $BROKEN_RECV="yes";     # For AIX & OpenBSD, NOT for Linux & Solaris
```

4.3.3 ICMP Tunneling using Ping Tunnel

Ping Tunnel (Ptunnel) allows you to tunnel TCP packets to a remote host using ICMP echo request and replay packets. Since ICMP is commonly allowed through firewalls for network troubleshooting, this technique provides a way to circumvent firewall and proxy server controls. Ptunnel provides the following functionality:

- Tunnel TCP using ICMP echo request and reply packets
- Connections are reliable (lost packets are resent as necessary)
- Handles multiple connections
- Acceptable bandwidth (150 kb/s downstream and about 50 kb/s upstream)
- Authentication to prevent others from using your proxy

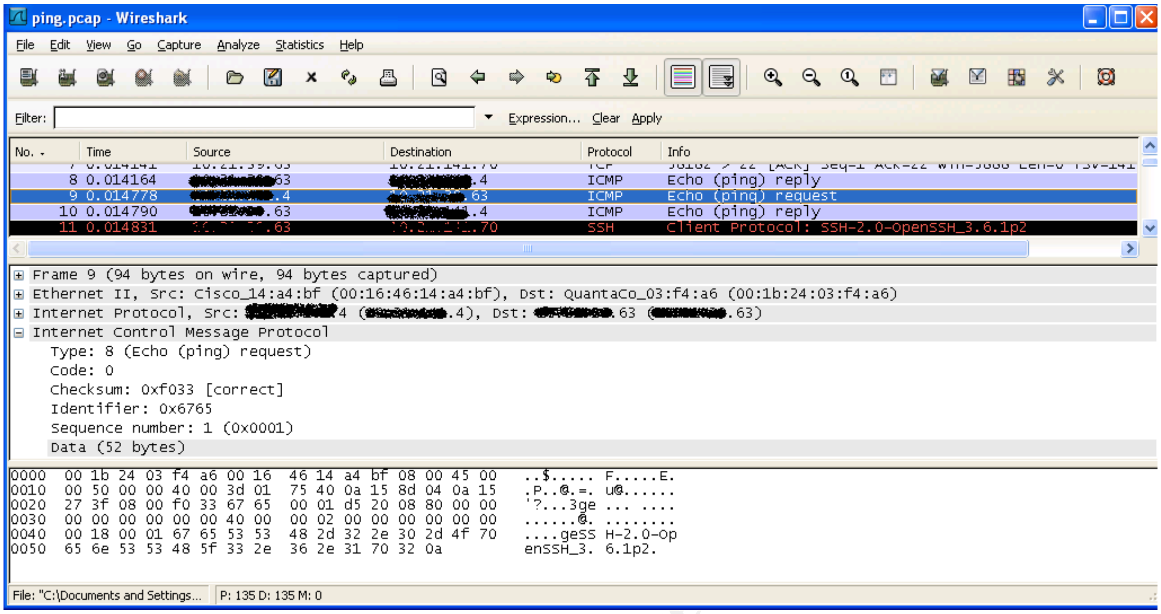
Ptunnel works by running once instance of this application as a proxy on a host outside your firewall and another instance as a client on a host inside your firewall. Once the ICMP tunnel is established, applications may be tunneled by connecting to localhost:port (port 8000 in this example). In the diagram below, SSH traffic normally not allowed through the firewall, is sent through the firewall using the ICMP tunnel (Stodle, 2005).



```
./ptunnel -p myhomepc  -lp 8000 -da {SSH Server IP  } -dp 22
```

Figure 20: Using Ping Tunnel to Bypass Firewalls

The following packet capture shows SSH traffic tunneled within ICMP echo request packets within the data field between the Ping Tunnel client .4 and the Ping Tunnel proxy .63. The traffic is then sent as regular SSH traffic between the Ping Tunnel proxy .63 and the SSH server .70.


ping.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -   Time       Source          Destination     Protocol  Info
  7 0.014141   10.21.39.63     10.21.141.70    TCP       38182 > 22 [ACK] Seq=1 Ack=22 Win=5888 Len=0 TSV=141
  8 0.014164   [redacted]63    [redacted].4    ICMP      Echo (ping) reply
  9 0.014778   [redacted].4    [redacted].63   ICMP      Echo (ping) request
 10 0.014790   [redacted].63   [redacted].4    ICMP      Echo (ping) reply
 11 0.014831   [redacted].63   [redacted].70   SSH       Client Protocol: SSH-2.0-OpenSSH_3.6.1p2

⊞ Frame 9 (94 bytes on wire, 94 bytes captured)
⊞ Ethernet II, Src: Cisco_14:a4:bf (00:16:46:14:a4:bf), Dst: QuantaCo_03:f4:a6 (00:1b:24:03:f4:a6)
⊞ Internet Protocol, Src: [redacted]4 ([redacted].4), Dst: [redacted].63 ([redacted].63)
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf033 [correct]
    Identifier: 0x6765
    Sequence number: 1 (0x0001)
    Data (52 bytes)

0000  00 1b 24 03 f4 a6 00 16  46 14 a4 bf 08 00 45 00   ..$..... F.....E.
0010  00 50 00 00 40 00 3d 01  75 40 0a 15 8d 04 0a 15   .P..@.=. u@......
0020  27 3f 08 00 f0 33 67 65  00 01 d5 20 08 80 00 00   '?...3ge ... ....
0030  00 00 00 00 00 00 40 00  00 02 00 00 00 00 00 00   ......@. ........
0040  00 18 00 01 67 65 53 53  48 2d 32 2e 30 2d 4f 70   ....geSS H-2.0-Op
0050  65 6e 53 53 48 5f 33 2e  36 2e 31 70 32 0a         enSSH_3. 6.1p2.

File: "C:\Documents and Settings...   P: 135 D: 135 M: 0
```

Figure 21: SSH Tunnel Packet Capture

4.3.4 Detecting and Preventing Tunneling Protocols

To limit the potential for SSH tunneling activity, all unnecessary outbound ports should be closed on the firewall. Access outbound via the SSH port 22 should be restricted to specific source and destination IP addresses on the firewall or by implementing a SOCKS proxy. Since ports 80 and 443 are typically open for web browsing, these ports present an opportunity to tunnel SSH.

Port 80 can be restricted to HTTP traffic by implementing an HTTP proxy. Web filtering could restrict access to uncategorized sites, but this may result in blocking legitimate traffic too. IDS signatures like the Cisco example below could identify SSH traffic on ports 80 and 443 based on the SSH banners. These alerts would identify regular SSH shell activity along with SSH tunneling, but even this regular SSH activity is suspicious since legitimate SSH traffic should be using port 22.

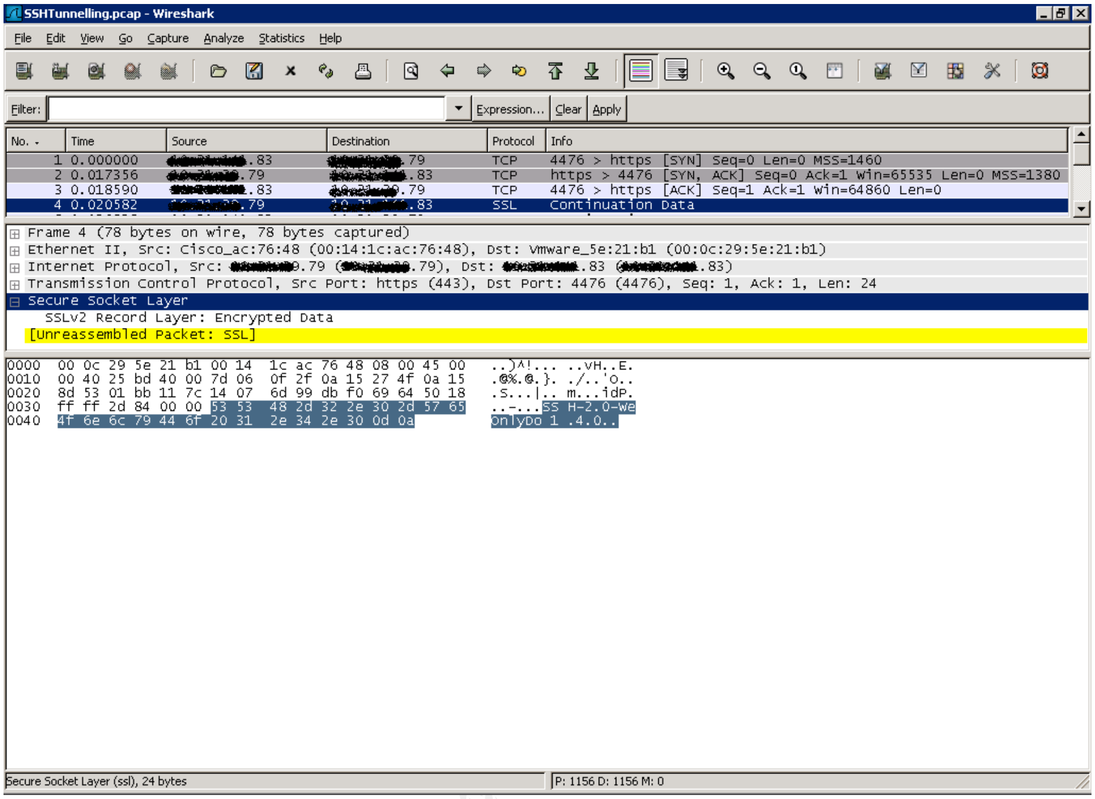
Sample Cisco IDS “SSH over Non-standard Ports” Alert

```
      evIdsAlert: eventId=1183701598144406163  vendor=Cisco  severity=informational
        originator:
          hostId: ISE_NIPS01
          appName: sensorApp
          appInstanceId: 384
        time: August 30, 2007 2:39:51 AM UTC  offset=60  timeZone=GMT-05:00
        signature:   description=SSH Over Non-standard Ports  id=11233
version=S258
          subsigId: 0
          sigDetails: SSH Over Web Ports
          marsCategory: Info/Misc
        interfaceGroup: vs0
        vlan: 0
        participants:
          attacker:
            addr: 10.21.141.83  locality=Inside
            port: 4527
          target:
            addr: 10.21.39.79  locality=Inside
            port: 443
            os:   idSource=learned  type=windows-nt-2k-xp  relevance=relevant
        context:
          fromTarget:
      000000  53 53 48 2D 32 2E 30 2D  57 65 4F 6E 6C 79 44 6F  SSH-2.0-WeOnlyDo
      000010  20 31 2E 34 2E 30 0D 0A                            1.4.0..

          fromAttacker:
      000000  53 53 48 2D                                       SSH-

        riskRatingValue: 26  targetValueRating=medium
      attackRelevanceRating=relevant
        threatRatingValue: 26
        interface: ge0_0
        protocol: tcp
```

The following packet capture shows the packet which triggers the above Cisco alert. This activity must be detected while the SSH session is being established. The activity can't be detected after the SSH session is established as the traffic is now encrypted.


SSHTunnelling.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time      Source          Destination     Protocol  Info
  1 0.000000  [redacted].83   [redacted].79   TCP   4476 > https [SYN] Seq=0 Len=0 MSS=1460
  2 0.017356  [redacted].79   [redacted].83   TCP   https > 4476 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1380
  3 0.018590  [redacted].83   [redacted].79   TCP   4476 > https [ACK] Seq=1 Ack=1 Win=64860 Len=0
  4 0.020582  [redacted].79   [redacted].83   SSL   Continuation Data

⊞ Frame 4 (78 bytes on wire, 78 bytes captured)
⊞ Ethernet II, Src: Cisco_ac:76:48 (00:14:1c:ac:76:48), Dst: Vmware_5e:21:b1 (00:0c:29:5e:21:b1)
⊞ Internet Protocol, Src: [redacted]9.79 ([redacted].79), Dst: [redacted].83 ([redacted].83)
⊞ Transmission Control Protocol, Src Port: https (443), Dst Port: 4476 (4476), Seq: 1, Ack: 1, Len: 24
⊟ Secure Socket Layer
    SSLv2 Record Layer: Encrypted Data
  [Unreassembled Packet: SSL]

0000  00 0c 29 5e 21 b1 00 14  1c ac 76 48 08 00 45 00   ..)^!... ..vH..E.
0010  00 40 25 bd 40 00 7d 06  0f 2f 0a 15 27 4f 0a 15   .@%.@.}. ./..'O..
0020  8d 53 01 bb 11 7c 14 07  6d 99 db f0 69 64 50 18   .S...|.. m...idP.
0030  ff ff 2d 84 00 00 53 53  48 2d 32 2e 30 2d 57 65   ..-...SS H-2.0-We
0040  4f 6e 6c 79 44 6f 20 31  2e 34 2e 30 0d 0a         OnlyDo 1 .4.0..

Secure Socket Layer (ssl), 24 bytes        P: 1156 D: 1156 M: 0
```

Figure 22: SSH Tunnel Packet Capture - SSH Banner

The following Snort IDS signatures are also available to identify potential SSH tunneling traffic on your corporate network.

Bleeding Edge Policy SSH Server Banner Detected on Unusual Port

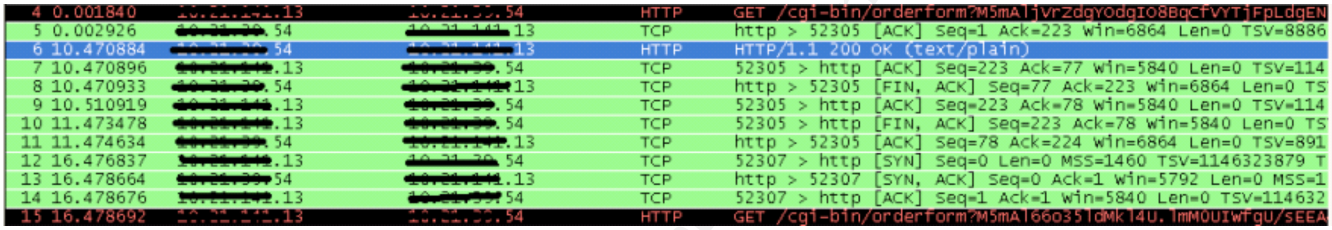
http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2001979&scope=text

Bleeding Edge Policy SSHv2 KEX Detected on Unusual Port

http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2001981&scope=text

Bleeding Edge Policy SSH Session in Progress on Unusual Port http://doc.bleedingthreats.net/bin/view/Main/WebSearch?search=2001984&scope=text

Reverse WWW shell seems more difficult to detect using IDS signatures. The query string is default CGI /cgi-bin/orderform is customizable in the Perl script's configuration file.



```
 4 0.001840   [redacted].13   [redacted].54   HTTP   GET /cgi-bin/orderform?M5mAljvrZdgYOdgIO8BqCfVYTjFpLdgEN
 5 0.002926   [redacted].54   [redacted].13   TCP    http > 52305 [ACK] Seq=1 Ack=223 Win=6864 Len=0 TSV=8886
 6 10.470884  [redacted].54   [redacted].13   HTTP   HTTP/1.1 200 OK (text/plain)
 7 10.470896  [redacted].13   [redacted].54   TCP    52305 > http [ACK] Seq=223 Ack=77 Win=5840 Len=0 TSV=114
 8 10.470933  [redacted].54   [redacted].13   TCP    http > 52305 [FIN, ACK] Seq=77 Ack=223 Win=6864 Len=0 TS
 9 10.510919  [redacted].13   [redacted].54   TCP    52305 > http [ACK] Seq=223 Ack=78 Win=5840 Len=0 TSV=114
10 11.473478  [redacted].13   [redacted].54   TCP    52305 > http [FIN, ACK] Seq=223 Ack=78 Win=5840 Len=0 TS
11 11.474634  [redacted].54   [redacted].13   TCP    http > 52305 [ACK] Seq=78 Ack=224 Win=6864 Len=0 TSV=891
12 16.476837  [redacted].13   [redacted].54   TCP    52307 > http [SYN] Seq=0 Len=0 MSS=1460 TSV=1146323879 T
13 16.478664  [redacted]54    [redacted].13   TCP    http > 52307 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1
14 16.478676  [redacted].13   [redacted]54    TCP    52307 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=114632
15 16.478692  [redacted].13   [redacted].54   HTTP   GET /cgi-bin/orderform?M5mA166o35ldMk14U.ImMOUIwfgU/sEEA
```

Figure 23: Reverse WWW Shell Packet Capture - CGI Request

I suspect you could reverse engineer the encoding used to create a signature, although I'm not aware of any reverse WWW shell signatures provided by IDS vendors. The traffic uses the standard web port 80 and the traffic appears like regular GET or POST requests. The slave can also be configured connect to the master daily at specified times so the HTTP tunnel may only exist for short periods of time. The slave executable process is masked to appear as if vi is running on the host although even this name is customizable (Stodle, 2005).

```
root     22023 21814  0 19:44 pts/0    00:00:00 vi
```

Obviously it would be best if the host was hardened to prevent the installation of this executable. However, once installed and running host based intrusion detection is probably the best method of detection since the http traffic originates from rwwwshell-2.0.pl program instead of the browser. This reverse shell seems reasonably stealthy and difficult to detect.

Ping Tunnel tunnels data within ICMP packets using echo request (Type 8) and echo reply (Type 0) ICMP packets. These are the same ICMP message types used by the ping utility commonly used for network troubleshooting. However, the frequency of these packets exceeds that generated from running the ping utility. While testing Ping Tunnel, an ICMP Flood IDS alert was generated by Cisco's IDS. This alert was configured to identify ICMP packets to the same destination host which exceed 25 per second. This alert could be triggered based on other legitimate or suspicious activity and is obviously also subject to false positives. The important thing to note is that Ping Tunnel generates ICMP echo request packets at a high enough rate to be considered suspicious. While signatures for specific ICMP tunneling applications may exist, the ICMP flood signature provides a generic method of detecting other suspicious activity which may indicate a new ICMP tunneling technique for which no specific signature exists.

Sample Cisco IDS "ICMP Flood" Alert

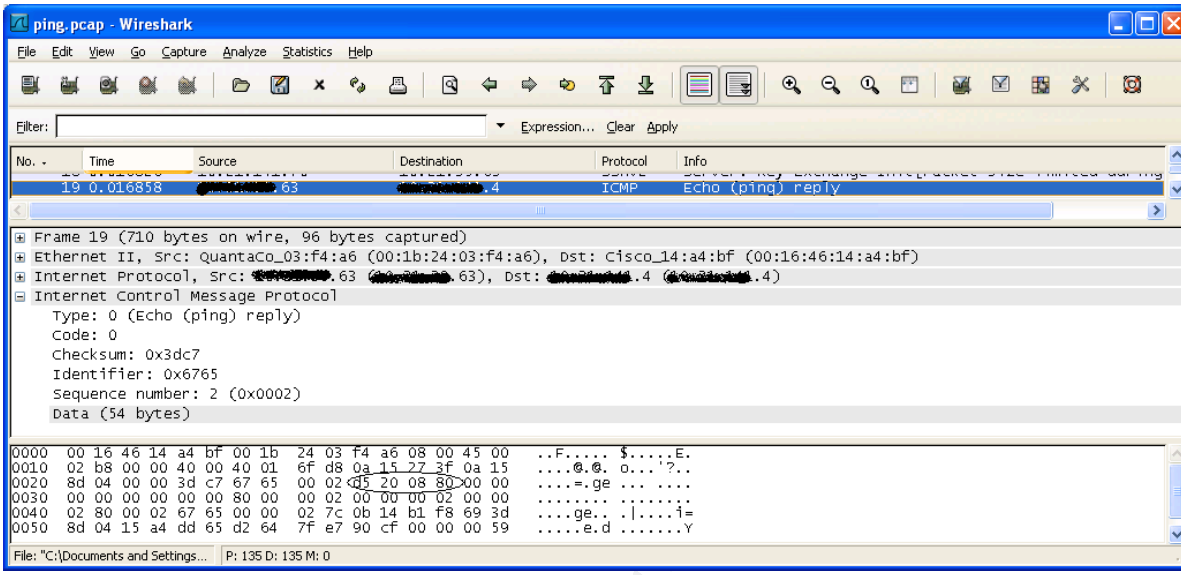
```
evIdsAlert: eventId=1183701598144394830  vendor=Cisco  severity=medium
  originator:
    hostId: XXXXX
    appName: sensorApp
    appInstanceId: 384
  time: August 27, 2007 11:35:27 PM UTC  offset=60  timeZone=GMT-05:00
  signature:   description=ICMP Flood  id=2152  version=S1
    subsigId: 0
    marsCategory: DoS/Network/ICMP
  interfaceGroup: vs0
  vlan: 0
  participants:
    attacker:
      addr: 10.xxx.xxx.xxx  locality=Inside
    target:
      addr: 198.xxx.xxx.xxx  locality=Inside
      os:   idSource=unknown  type=unknown  relevance=relevant
  riskRatingValue: 85  targetValueRating=medium
attackRelevanceRating=relevant
  threatRatingValue: 85
  interface: ge0_0
  protocol: icmp
```

Additionally, the Ping Tunnel traffic generated a PingTunnel ICMP Tunneling alert more specific to this activity. This signature was based on the hexadecimal values \\xD5\\x20\\x08\\x80. Obviously this signature is valid for Ping Tunnel only.

Sample Cisco IDS "Ping Tunnel ICMP Tunneling" Alert

```
evIdsAlert: eventId=1183701598144394829  vendor=Cisco  severity=high
  originator:
    hostId: XXXX
    appName: sensorApp
    appInstanceId: 384
  time: August 27, 2007 11:34:48 PM UTC  offset=60  timeZone=GMT-05:00
  signature:   description=PingTunnel ICMP Tunneling  id=5543  version=S176
    subsigId: 0
    sigDetails: \\xD5\\x20\\x08\\x80
    marsCategory: Penetrate/Backdoor/CovertChannel
  interfaceGroup: vs0
  vlan: 0
  participants:
    attacker:
      addr: 10.XXX.XXX.XXX  locality=Inside
    target:
      addr: 198.XXX.XXX.XXX locality=Inside
      os:   idSource=unknown  type=unknown  relevance=unknown
  riskRatingValue: 80  targetValueRating=medium
  threatRatingValue: 80
  interface: ge0_0
  protocol: icmp
```

The following packet capture shows ICMP traffic containing the hexadecimal values which trigger the above Cisco IDS "Ping Tunnel ICMP Tunneling" alert.


ping.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info

```
19 0.016858 [redacted].63 [redacted].4 ICMP Echo (ping) reply

⊞ Frame 19 (710 bytes on wire, 96 bytes captured)
⊞ Ethernet II, Src: QuantaCo_03:f4:a6 (00:1b:24:03:f4:a6), Dst: Cisco_14:a4:bf (00:16:46:14:a4:bf)
⊞ Internet Protocol, Src: [redacted].63 ([redacted].63), Dst: [redacted].4 ([redacted].4)
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x3dc7
    Identifier: 0x6765
    Sequence number: 2 (0x0002)
    Data (54 bytes)

0000  00 16 46 14 a4 bf 00 1b  24 03 f4 a6 08 00 45 00   ..F..... $.....E.
0010  02 b8 00 00 40 00 40 01  6f d8 0a 15 27 3f 0a 15   ....@.@. o...'?..
0020  8d 04 00 00 3d c7 67 65  00 02 d5 20 08 80 00 00   ....=.ge ... ....
0030  00 00 00 00 00 00 80 00  00 02 00 00 00 02 00 00   ........ ........
0040  02 80 00 02 67 65 00 00  02 7c 0b 14 b1 f8 69 3d   ....ge.. .|....i=
0050  8d 04 15 a4 dd 65 d2 64  7f e7 90 cf 00 00 00 59   .....e.d .......Y

File: "C:\Documents and Settings...   P: 135 D: 135 M: 0
```

Figure 24: Ping Tunnel Packet Capture – Cisco Signature

A simple way to prevent ICMP tunneling is to block all ICMP traffic or only ICMP message types which are required. This may not be possible in all environments, but should be considered when reviewing outbound firewall rules. Some data leakage prevention products may detect suspicious behavior if the tunneled protocol is not encrypted and the payload contains data considered private like SSNs or credit card numbers. Preventing users from running unauthorized software on their computer either by limiting administrator or root privileges or detecting/preventing using desktop agents like McAfee ePO may be another method for limiting this activity.

4.4 Botnets

While the term "botnet" can be used to refer to any group of bots, such as IRC bots, the word is generally used to refer to a collection of compromised computers (called zombie computers) running programs, usually referred to as worms, Trojan horses, or backdoors, under a common command and control infrastructure (wikepedia.org, 2007).

There are many types of Bots used for a variety of malicious purposes which propagate using many methods some of which are described at http://www.honeynet.org/papers/bots/. This section will focus primarily on the various types of network communications between the Bot and command and control center.

4.4.1 Botnet Command & Control Models

To understand Botnet communication, it's important to first understand the centralized and P2P Command & Control (C&C) models. A third random C&C model is potentially interesting, but not really in use today.

The centralized C&C model is the most common. A centralized high bandwidth host is selected to be the contact point for all bots within the network. An infected host joins the botnet by contacting the centralized host and the centralized host issues instructions directly to all member bots. It's common for the C&C to be a compromised computer. It's also possible for the C&C to change to another host if dynamic DNS is used by the bots to identify the C&C ("Taxonomy of Botnet Threats", November 2006).

According to Ed Skoudis (2007), attackers are starting to use Peer-to-peer (P2P) botnets to direct botnets without a central point of control. ("10 Emerging Malware Trends for 2007"). P2P based C&C are more difficult to dismantle since they are not controlled by a single server. On the other hand, P2P based C&C can only support small numbers of bots and P2P does not guarantee message delivery and propagation latency. However, new P2P based botnets will likely appear as knowledge of how to implement the P2P model improves. Some Phatbot variants currently use the P2P C&C model ("Taxonomy of Botnet Threats", November 2006).

4.4.2 Botnet Methods of Communication

The following 2 diagrams from (Botnet Communication Platforms, March 7 2007), show two common methods of communication in a centralized C&C Botnet.

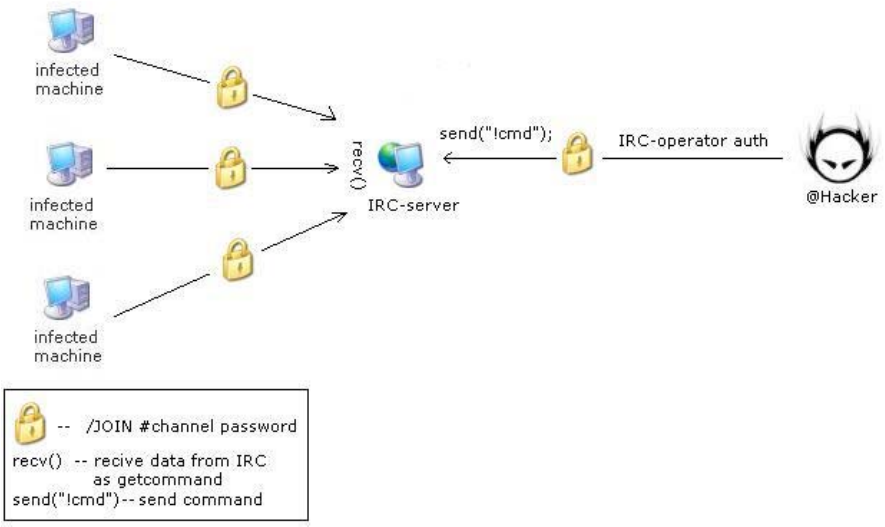
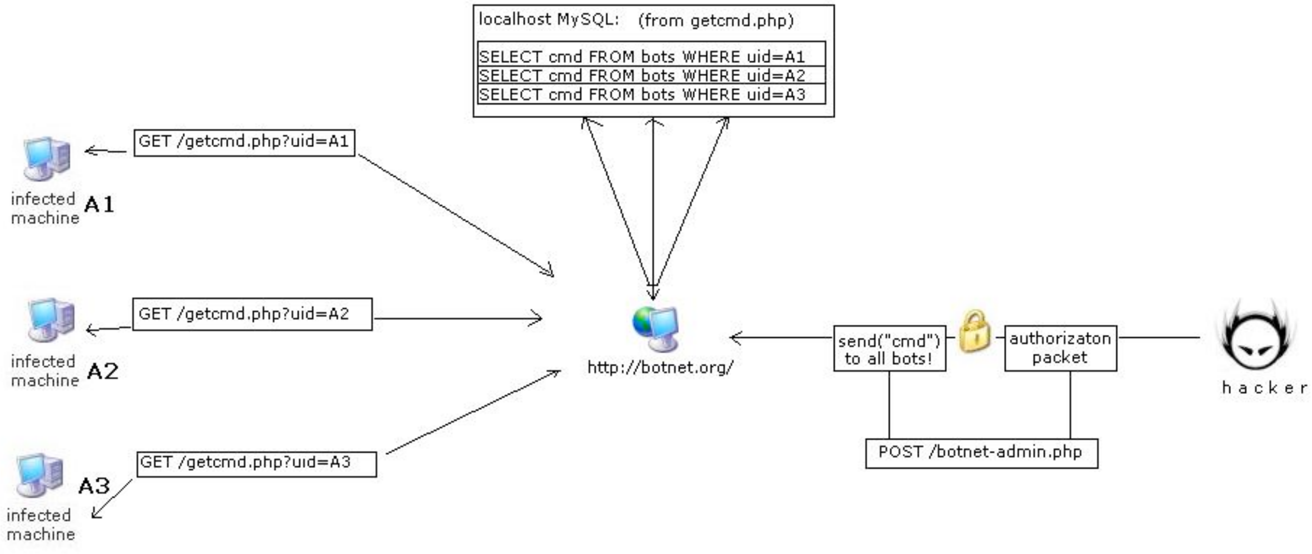


Figure 24: Botnet Network Diagram – Using IRC Protocol

IRC was originally developed as a means of BBS users to chat. Today it is also commonly used as a method for Bots to communicate back to an IRC server acting as a command and control center. The infected machines communicate with the IRC server to retrieve commands. The IRC server may be listening on one of the standard IRC ports 6660, 6661, 6662, 6663, 6664, 6665, 6666, 6667, 6668, 6669 or 7000. Non standard IRC ports like port 80 may also be used to ensure greater success in circumventing a firewall. IRC is extremely popular since it provides a simple, low-latency, widely available and anonymous command and control channel for botnet communication (Cooke, Jahanian, & McPherson)



```
SELECT cmd FROM bots WHERE uid=A1
SELECT cmd FROM bots WHERE uid=A2
SELECT cmd FROM bots WHERE uid=A3
```

```
GET /getcmd.php?uid=A1
GET /getcmd.php?uid=A2
GET /getcmd.php?uid=A3
POST /botnet-admin.php
```

Figure 25: Botnet Network Diagram – Using HTTP Protocol

HTTP is another protocol that can be used to control botnets. In above diagram, the attacker has established a web server running PHP with a SQL Server backend. Once again, this can be installed on a compromised host to avoid detection. The attacker send commands using the standard HTTP post method, which is then stored in the SQL server database. Infected machines retrieve their commands using standard HTTP get requests. All of this traffic is standard HTTP over port 80 so it blends in very nicely with legitimate web traffic.

WASTE is a decentralized chat, instant messaging and file sharing program & protocol. It behaves similar to a virtual private network by connecting to a group of trusted computers, as determined by the users. This kind of network is commonly referred to as a darknet. It employs heavy encryption to ensure that third parties cannot decipher the messages being transferred. The same encryption is used to transmit and receive instant messages, chat, and files, maintain the connection, and browse and search (Wikepedia, 2007).

The WASTE protocol is used by Phatbot for control communication. The encryption within WASTE has been disabled possibly due to the additional complexity of sharing keys. Since no central host is used it is difficult to shut down this Botnet. On the other hand, peer discovery and network responsiveness are a challenge with P2P networks. Phatbot utilizes Gnutella cache servers for bot registration. Bots register pretending to be a GNUT client, but differentiate themselves from standard Gnutella clients by using TCP port 4387 instead of the standard Gnutella port. Bots can retrieve a list of other bots using the same Gnutella cache servers.

To control the Botnet, the attacker uses a custom WASTE client to connect to a peer identified through a Gnutella cache server. With the correct username and password, the attacker can issue commands which will be distributed to other bots in a similar P2P fashion. Since WASTE was designed for small networks (10-50 users), only small botnets are currently supported (Stewart, 2004).

4.4.3 Detection and Prevention of Botnet Communication

The following steps can be implemented to identify and contain Botnet communication on your network.

- Close all unnecessary outbound ports on the firewall.
- Implement proxy server to require properly formed HTTP traffic over port 80 to prevent IRC over 80.
- Implement URL filtering to block known Botnet sites.
- Deploy IPS with signature based on well known Botnet sites or detection of known Botnet activity.
- Deploy behavioral IDS to detect suspicious traffic.
- Perform flow analysis to identify Botnet traffic based on high rate of failed connections. Results in a lot of ICMP "destination unreachable" packets and TCP reset packets (Schoof & Koning, 2007).

5.0 Overview of Techniques for Securing Outbound Traffic

The techniques described in this paper may be used by company employees (insiders), outside attackers (intruders) or both. For insiders, security policy, security awareness and training are important to ensure employees understand the appropriate use of corporate resources including the network.

Preventing attackers from getting on a system in the first place by hardening and patching servers is an obvious first step to minimize outsider threats like Bots, Worms and Viruses or backdoors. This includes considering using AV, host based intrusion prevention and host firewalls as appropriate.

From a network perspective, all unnecessary outbound ports should be closed. Consider implementing a HTTP proxy for web traffic and a SOCKS proxy for other outbound traffic (e.g. SSH). If unable to implement an HTTP proxy, then some other form of web filtering (e.g. Websense, Surf Control) will provide protection against malicious web sites and possibly some undesirable network protocols (e.g. P2P, IM) if supported by the filtering solution. Sensing outbound traffic using a network IDS is another important tool in identifying suspicious traffic like tunneling protocols and TOR traffic.

Stealthy traffic like reverse WWW shell may require some additional protection since this traffic appears like well formed HTTP traffic. Restricting outbound Internet access from sensitive servers and running desktop firewall or HIPS may help to identify HTTP traffic originating from a process other than the web browser.

All of the above tools are important in providing a layered approach to protecting your network. Once these tools are in place, they can be configured to address the threats that exist both today and hopefully in the future.

6.0 References

Anka, Marton (2006). LogMeIn Security - an In-Depth Look. Retrieved June 20th 2007 from https://secure.logmein.com/wp_lmi_security.pdf

BlueCoat (2007). ProxySG TechBrief - Controlling Tunneling Applications. Web site: http://www.bluecoat.com/downloads/support/BCS_tb_tunnelling_applications.pdf

Brozycki, John & Bong, Kevin. (April 23, 2007). Managing Large Botnets. Retrieved July 3rd 2007 from www.sans.edu/resources/student_projects/200704_001.doc

Chirico, Mike. Breaking Firewalls with OpenSSH and Putty. Retrieved August 2nd 2007 from http://souptonuts.sourceforge.net/sshtips.htm

Cooke, Evan, & Jahanian, Farnam, & McPherson, Danny. The Zombie Roundup: Understanding, Detecting and Disrupting Botnets. Retrieved June 14th 2007 from http://www.eecs.umich.edu/~emcooke/pubs/botnets-sruti05.pdf

Danchev, Danco (March 07, 2007). Botnet Communication Platforms. Retrieved June 14th 2007 from http://ddanchev.blogspot.com/2007/03/botnet-communication-platforms.html

Forward, Kenneth (February 6, 2007). Network Firewall Architectures and Technologies. Retrieved August 20th, 2007 from http://www.giac.org/resources/whitepaper/network/7.php

Grof (July 1, 2007) freeSSHd. Retrieved July 15, 2007 from http://www.freesshd.com/

Iana.org (August 16, 2007). Port Numbers. Retrieved August 18, 2007 from http://www.iana.org/assignments/port-numbers

Installing Vidalia on Windows. (n.d.) Retrieved July 3 2007 from http://trac.vidalia-project.net/wiki/InstallWindows

Lars Brinkhoff (October 2006). HTTP Tunnel. Retrieved May 22, 2007. Web site: http://www.nocrew.org/software/httptunnel.html

Luotonen, Ari (August 1998). Tunneling TCP based protocols through Web proxy servers. Retrieved June 15, 2007 from http://tools.ietf.org/id/draft-luotonen-web-proxy-tunneling-01.txt

Schoof, Reinier & Koning, Ralph, (February 4, 2007). Detecting Peer-to-Peer botnets. Retrieved June 14th, 2007 from http://staff.science.uva.nl/~delaat/sne-2006-2007/p17/report.pdf

Skoudis, Ed (January 18, 2007). 10 Emerging Malware Trends for 2007. Retreived June 14th, 2007 from http://searchsecurity.techtarget.com/tip/0,289483,sid14_gci1238948,00.html

Stewart, Joe (March 15, 2004). Phatbot Trojan Analysis. Retrieved June 14th 2007 from http://www.secureworks.com/research/threats/phatbot/

Stodle, Daniel (May 25, 2005). Ping Tunnel. Retrieved August 17, 2007 from http://www.cs.uit.no/~daniels/PingTunnel/

Squid-cach.org (n.d.) Squid: Optimising Web Delivery. Retrieved July 1, 2007 from http://www.squid-cache.org/

Tatham, Simon (n.d.) Putty: A free Telnet/SSH Client. Retrieved July 1, 2007 from http://www.chiark.greenend.org.uk/~sgtatham/putty/

The Free Haven Project. (n.d.). TOR Overview. Retrieved July 3 2007 from http://tor.eff.org/overview.html.en

Thomas, Rob (March 12, 2003). ICMP Packet Filtering v1.2. Retrieved August 29, 2007 from http://www.cymru.com/Documents/icmp-messages.html

Tor directory protocol, version 1. (7/20/2006) Retrieved August 23, 2007 from http://tor.eff.org/cvs/doc/dir-spec.txt

Trend Micro (November 2006). Taxonomy of Botnet Threats. Retrieved June 14th, 2007 from http://us.trendmicro.com/imperia/md/content/us/pdf/**threats**/securitylibrary/**botnettaxonomy**whitepapernovember2006.pdf

Using Putty (n.d.) Retrieved June 15, 2007 from http://the.earth.li/~sgtatham/putty/0.60/htmldoc/Chapter3.html#using-port-forwarding

Van Hauser. (n.d.) Placing Backdoors Through Firewalls. Retrieved July 3, 2007 from http://www.thc.org/papers/fw-backd.htm