

---[Placing Backdoors Through Firewalls]---

v1.5

Author: [van Hauser](#) / *THC*

----[Introduction

This article describes possible backdoors through different firewall architectures. However, the material can also be applied to other environments to describe how hackers (you?) cover their access to a system.

Hackers often want to retain access to systems they have penetrated even in the face of obstacles such as new firewalls and patched vulnerabilities. To accomplish this the attackers must install a backdoor which a) does it's job and b) is not easily detectable. The kind of backdoor needed depends on the firewall architecture used.

As a gimmick and proof-of-concept, a nice backdoor for any kind of intrusion is included, so have fun.

----[Firewall Architectures

There are two basic firewall architectures and each has an enhanced version.

Packet Filters:

This is a host or router which checks each packet against an allow/deny ruletable before routing it through the correct interface. There are very simple ones which can only filter from the origin host, destination host and destination port, as well as good ones which can also decide based on incoming interface, source port, day/time and some tcp or ip flags. This could be a simple router, f.e. any Cisco, or a Linux machine with firewalling activated (ipfwadm).

Stateful Filters:

This is the enhanced version of a packet filter. It still does the same checking against a rule table and only routes if permitted, but it also keeps track of the state information such as TCP sequence numbers. Some pay attention to application protocols which allows tricks such as only opening ports to the interior network for ftp-data channels which were specified in a permitted ftp session. These filters can (more or less) get UDP packets (f.e. for DNS and RPC) securely through the firewall. (That's because UDP is a stateless protocol. And it's more difficult for RPC services.)

This could be a great OpenBSD machine with the ip-filter software, a Cisco Pix, Watchguard, or the (in)famous Checkpoint FW-1.

Proxies / Circuit Level Gateways:

A proxy as a firewall host is simply any server which has no routing activated and instead has proxy software installed.

Examples of proxy servers which may be used are squid for WWW, a sendmail relay configuration and/or just a sockd.

Application Gateways:

This is the enhanced version of a proxy. Like a proxy, for every application which should get through the firewall a software must be installed and running to proxy it. However, the application gateway is smart and checks every request and answer, f.e. that an outgoing ftp only may download data but not upload any, and that the data has got no virus, no buffer overflows are generated in answers etc. One can argue that squid is an application gateway, because it does many sanity checks and let you filter stuff but it was not programmed for the installation in a secure environment and still has/had security bugs. A good example for a freeware kit for this kind is the TIS firewall toolkit (fwtk).

Most firewalls that vendors sell on the market are hybrid firewalls, which means they've got more than just one type implemented; for example the IBM Firewall is a simple packet filter with socks and a few proxies. I won't discuss which firewall product is the best, because this is not a how-to-by-a-firewall paper, but I will say this: application gateways are by far the most secure firewalls, although money, speed, special protocols, open network policies, stupidity, marketing hype and bad management might rule them out.

----[Getting in

Before we talk about what backdoors are the best for which firewall architecture we should shed a light on how to get through a firewall the first time. Note that getting through a firewall is not a plug-n-play thing for script-kiddies, this has to be carefully planned and done.

The four main possibilities:

Insider:

There's someone inside the company (you, girl/boy-friend, chummer) who installs the backdoor. This is the easiest way of course.

Vulnerable Services:

Nearly all networks offer some kind of services, such as incoming email, WWW, or DNS. These may be on the firewall host itself, a host in the DMZ (here: the zone in front of the firewall, often not protected by a firewall) or on an internal machine. If an attacker can find a hole in one of those services, he's got good chances to get in. You'd laugh if you'd see how many "firewalls" run sendmail for mail relaying ...

Vulnerable External Server:

People behind a firewall sometimes work on external machines. If an attacker can hack these, he can cause serious mischief such as the many X attacks if the victim uses it via an X-relay or sshd. The attacker could also send fake ftp answers to overflow a buffer in the ftp client software, replace a gif picture on a web server with one which crashes netscape and executes a command (I never checked if this actually works, it crashes, yeah, but I didn't look through this if this is really an exploitable overflow). There are many possibilities with this but it needs some knowledge about the company. However, an external web server of the company is usually a good start. Some firewalls are configured to allow incoming telnet from some machines, so anyone can sniff these and get it. This is particularly true for the US, where academic environments and industry/military work close together.

Hijacking Connections:

Many companies think that if they allow incoming telnet with some kind of secure authentication like SecureID (secure algo?, he) they are safe. Anyone can hijack these after the authentication and get in ... Another way of using hijacked connections is to modify replies in the protocol implementation to generate a buffer overflow (f.e. with X).

Trojans:

Many things can be done with a trojan horse. This could be a gzip file which generates a buffer overflow (well, needs an old gzip to be installed), a tar file which tampers f.e. ~/.logout to execute something, or an executable or source code which was modified to get the hacker in somehow. To get someone running this, mail spoofing could be used or replacing originals on an external server which internal employees access to update their

software regularly (ftp xfer files and www logs can be checked to get to know which files these are).

----[Placing the Backdoors

An intelligent hacker will not try to put the backdoors on machines in the firewall segment, because these machines are usually monitored and checked regularly. It's the internal machines which are usually unprotected and without much administration and security checks.

I will now talk about some ideas of backdoors which could be implemented. Note that programs which will/would run on a stateful filter will of course work with a normal packet filter too, same for the proxy. Ideas for an application gateway backdoor will work for any architecture. Some of them are "active" and others "passive". "Active" backdoors are those which can be used by a hacker anytime he wishes, a "passive" one triggers itself by time/event so an attacker has to wait for this to happen.

Packet Filters:

It's hard to find a backdoor which gets through this one but does not work for any other. The few ones which come into my mind

a) the ack-telnet. It works like a normal telnet/telnetd except it does not work with the normal tcp handshake/protocol but uses TCP ACK packets only. Because they look like they belong to an already established (and allowed) connection, they are permitted. This can be easily coded with the spoofit.h of Coder's Spoofit project (<http://reptile.rug.ac.be/~coder>).

b) Loki from Phrack 49/51 could be used too to establish a tunnel with icmp echo/reply packets. But some coding would be needed to be done.

c) daemonshell-udp is a backdoor shell via UDP (<http://www.thc.org> look for thc-uht1.tgz)

d) Last but not least, most "firewall systems" with only a screening router/firewall let any incoming tcp connection from the source port 20 to a highport (>1023) through to allow the (non-passive) ftp protocol to work. "netcat -p 20 target port-of-bindshell" is the fastest solution for this one.

Stateful Filters:

Here a hacker must use programs which initiate the connection from the secure network to his external owned server. There are many out there which could be used:
active:

tunnel from Phrack 52.

ssh with the -R option (much better than tunnel ... it's a legitimate program on a computer and it encrypts the datastream).

passive:

netcat compiled with the execute option and run with a time option to connect to the hacker machine (ftp.avian.org).

reverse_shell from the thc-uht1.tgz package (see above) does the same.

Proxies / Circuit Level Gateways:

If socks is used on the firewall, someone can use all those stuff for the stateful filter and "socksify" them. (www.socks.nec.com) For more advanced tools you'd should take a look at the application gateway section.

Application Gateways:

Now we get down to the interesting stuff. These beasts can be intelligent so some brain is needed.

active:

(re-)placing a cgi-script on the webserver of the company, which allows remote access. This is unlikely because it's rare that the webserver is in the network, not monitored/ checked/audited and accessible from the internet. I hope nobody needs an example on such a thing ;-)

(re-)placing a service/binary on the firewall. This is dangerous because those are audited regularly and sometimes even sniffed on permanent ... Loading a loadable module into the firewall kernel which hides itself and gives access to its master. The best solution for an active backdoor but still dangerous.

passive:

E@mail - an email account/mailler/reader is configured in a way to extract hidden commands in an email (X-Headers with weird stuff) and send them back with output if wanted/needed.

WWW - this is hard stuff. A daemon on an internal machine does http requests to the internet, but the requests are in real the answers of commands which were issued by a rogue www server in a http reply. This nice and easy beast is presented below (->[Backdoor Example: The Reverse WWW Shell](#))

DNS - same concept as above but with dns queries and replies.

Disadvantage is that it can not carry much data. (<http://www.icon.co.za/~wosp/wosp.dns-tunnel.tar.gz>, this example needs still much coding to be any effective)

----[Backdoor Example: The Reverse WWW Shell

This backdoor should work through any firewall which has got the security policy to allow users to surf the WWW (World Wide Waste) for information for the sake and profit of the company. For a better understanding take a look at the following picture and try to remember it onwards in the text:

```

+-----+
+-----+
| internal |-----| FIREWALL |-----|
server owned |
| host | internal network +-----+ internet |by
the hacker|
+-----+
+-----+
      SLAVE
MASTER

```

Well, a program is run on the internal host, which spawns a child every day at a special time. For the firewall, this child acts like a user, using his netscape client to surf on the internet. In reality, this child executes a local shell and connects to the www server owned by the hacker on the internet via a legitimate looking http request and sends it ready signal. The legitimate looking answer of the www server owned by the hacker are in reality the commands the child will execute on it's machine it the local shell. All traffic will be converted (I'll not call this "encrypted", I'm not Micro\$oft) in a Base64 like structure and given as a value for a cgi-string to prevent caching.

Example of a connection:

```

Slave
GET /cgi-bin/order?M5mAejTgZdgYOdgIO0BqFfVYTgjFLdgxEdb1He7krj
HTTP/1.0

```

```

Master replies with
g5mAlfbknz

```

The GET of the internal host (SLAVE) is just the command prompt of the shell, the answer is an encoded "ls" command from the hacker on the external server (MASTER). Some gimmicks:

The SLAVE tries to connect daily at a specified time to the MASTER if wanted; the child is

spawned because if the shell hangs for whatever reason you can check & fix the next day; if an administrator sees connects to the hacker's server and connects to it himself he will just see a broken webserver because there's a Token (Password) in the encoded cgi GET request; WWW Proxies (f.e. squid) are supported; program masks it's name in the process listing ...

Best of all: master & slave program are just one 260-lines perl file ... Usage is simple: edit `rwwwshell.pl` for the correct values, execute "`rwwwshell.pl slave`" on the SLAVE, and just run "`rwwwshell.pl`" on the MASTER just before it's time that the slave tries to connect.

Well, why coding it in perl? a) it was very fast to code, b) it's highly portable and c) I like it. If you want to use it on a system which hasn't got perl installed, search for a similar machine with perl install, get the a3 compiler from the perl CPAN archives and compile it to a binary. Transfer this to your target machine and run that one.

The code for this nice and easy tool is appended in the section THE CODE after my last words. If you've got updates/ideas/critics for it drop me an email. If you think this text or program is lame, write me at `root@localhost`. Check out <http://www.thc.org> for updates.

----[The Source

Grab it here ...

[rwwwshell v2.0](#)

----[Security

Now it's an interesting question how to secure a firewall to deny/detect this. It should be clear that you need a tight application gateway firewall with a strict policy. email should be put on a centralized mail server, and DNS resolving only done on the WWW/FTP proxies and access to WWW only prior proxy authentication. However, this is not enough. An attacker can tamper the mailreader to execute the commands extracted from the crypted X-Headers or implement the http authentication into the reverse www-shell (it's simple). Also checking the DNS and WWW logs/caches regularly with good tools can be defeated by switching the external servers every 3-20 calls or use aliases.

A secure solution would be to set up a second network which is connected to the internet, and the real one kept seperated - but tell this the employees ... A good firewall is a big improvement, and also an Intrusion Detection Systems can help. But nothing can stop a dedicated attacker.

----[Last Words

Have fun hacking/securing the systems ...
Greetings to all guys who like + know me ;-) and especially to
those good
chummers I've got, you know who you are.

Ciao...

van Hauser / [THC] - The Hacker's Choice

For further interesting discussions you can email me at
vh@reptile.rug.ac.be with my public pgp key blow:

```
Type Bits/KeyID      Date      User ID
pub  2048/CDD6A571  1998/04/27  van Hauser / THC
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i
```

```
mQENAzVE0A4AAAEIAOzKPhKBDFDyeTvMKQ1xx6781tEdIYgrkrsUEL6VoJ8H8CIU
SeXDuCVu3JlMKITD6nPMFJ/DT0iKHgnHUZGdCQEk/b1YHUYOciGlDPGsg3WeTX7L
XL1M4DwqDvPz5QUQ+U+VHuNOUzgxfcjhHsjj2qorVZ/T5x4k3U960CMJ1leOVNC
meD/+c6a2FfLZJG0sJ/kIZ9HUKY/dvXDInOJaalQc1mYjkvfcPsSzas4ddiXiDyc
QcKX+HAXIdmT7bjq5+JS6yspnBvIZC55tB7ci2axTjwpkdzJBZIkCoBlWsDXNwyq
s70Lo3H9dcaNt4ubz5OMVIvJHFMCEtIGS83WpXEABRG0J3ZhbIBIYXVzZXIgLjBU
SEMgPHZoQHJlCHRpbGUucnVnLmFjLmJlPokAlQMFEDVE0D7Kb9wCOximfQEBvpAD
/3UCDgJs1CNg/zpLhRuUBlYsZ1kimb9cbB/ufL1I4lYM5WMyw+YfGN0p02oY4pVn
CQN6ca50sqeXHWfn7LxBT3lXEPcckd+vb9LPPCzuDPS/zYNOKUXgUQdPo69B04dl
C9C1YXcZjplYso2q3NYnuc0lu7WVD0qT52snNUdkd19ciQEVAwUQNUTQDhLSBkvN
1qVxAQGRTwgA050murXHVByFcvDaBRMhX6pKbTiVKh8HdJa8IdvuqH0cYFZ2L+xZ
PAQy2WCqeakvss9Xn9I28/PQZ+6TmqWUmG0qgxe5MwkaXWxsZKwRsQ8hH+bcppsZ
2/Q3BxSfPege4PPwFWsa jnymsnmhdVvvrvt69grzJDM+iMK0WR33+RvtgjjUj+i22X
lpt5hLHufDatQzukMu4R84M1tbGnUCNF0wICrU4U503yCA4DT/1eMoDXI0BQXmM/
Ygk9b02Icy+lw1WPodrWmg4TJhdIgxuYlNLIu6TyqDYxjA/c525cBbdqwoE+YvUI
o7CN/bJN0bKg1Y/BMTHEK3mpRLLWxVMRYw==
=MdzX
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

----[THE END