

Cluster Security and pfilter

Neil Gorsuch

NCSA

ngorsuch@ncsa.uiuc.edu



Riddle Me This

- **What do these organizations all have in common?**
 - NSA
 - NASA
 - NCSA
 - Library of Congress
 - U.S. Army U.S. Air Force U.S. Navy
 - U.S. Naval Security Group
 - D.O.D.
 - U.S. Senate
 - Library of Congress
 - Yahoo
 - Microsoft
 - www.rootshell.com www.attrition.org www.sans.org



Answer

- **They have all been hacked!**
- **1988 CERT statistics:**
 - 6 reported incidents
- **1999 CERT statistics:**
 - 9,859 reported incidents
 - 417 new security vulnerabilities
- **2003 CERT statistics:**
 - 137,529 reported incidents
 - 3,784 new security vulnerabilities



Perfect Security

- **To have perfect security you need:**
 - Absolute cooperation
 - Free reign to implement draconian measures
 - Usability and performance secondary to security
 - Smarter than hackers



The Loose Security Model

In political terms, the FAR LEFT:

- All cluster nodes accessible from the internet
- No firewalls between the cluster and the internet
- Little or no packet filtering
- No high ports filtered from the internet
- All network services allowed
 - “what, you mean RedHat isn’t secure out of the box?”

The technical name for this type of environment is a



The Loose Security Model

In political terms, the FAR LEFT:

- All cluster nodes accessible from the internet
- No firewalls between the cluster and the internet
- Little or no packet filtering
- No high ports filtered from the internet
- All network services allowed
 - “What, you mean RedHat isn’t secure out of the box?”

The technical name for this type of environment is a
“target rich environment”



The Loose Security Model

In political terms, the FAR LEFT:

- All cluster nodes accessible from the internet
- No firewalls between the cluster and the internet
- Little or no packet filtering
- No high ports filtered from the internet
- All network services allowed
 - “What, you mean RedHat isn’t secure out of the box?”

The technical name for this type of environment is a
“target rich environment” or “script kiddies playground”



The Strict Security Model (FAR RIGHT)

- **Very strict cluster/border firewall**
 - No packet forwarding, all services must use proxies
 - Sometimes, secure login services such as ssh not allowed
 - (because they can't log conversations)
- **Access outside the firewall to the cluster impaired**
- **File transfer difficulties**
- **“Big Brother” mentality**
(logs of all your actions)

The technical term for this is



The Strict Security Model (FAR RIGHT)

- **Very strict cluster/border firewall**
 - No packet forwarding, all services must use proxies
 - Sometimes, secure login services such as ssh not allowed
 - (because they can't log conversations)
- **Access outside the firewall to the cluster impaired**
- **File transfer difficulties**
- **“Big Brother” mentality**
(logs of all your actions)



"I SEE YOU'VE DISCOVERED OUR 'FIREWALL'"

The technical term for this is

“work impairment”

or



The Strict Security Model (FAR RIGHT)

- **Very strict cluster/border firewall**
 - No packet forwarding, all services must use proxies
 - Sometimes, secure login services such as ssh not allowed
 - (because they can't log conversations)
- **Access outside the firewall to the cluster impaired**
- **File transfer difficulties**
- **“Big Brother” mentality**
(logs of all your actions)



"I SEE YOU'VE DISCOVERED OUR 'FIREWALL'"

The technical term for this is
“work impairment”
or “Why can't I use IRC?”



Cluster Security Difficulties

- **Wide range of cluster security hardening requirements**
 - **Some have all nodes directly connected to internet**
 - **Some already behind massive firewalls**
- **Every cluster machine needs hardening in some cases**
- **Widely varying cluster network topologies**
- **Clusters usually have many network services**
- **Clusters usually have many users shared with other sites**
- **Hard to detect attacks spread across clusters**
- **Clusters are prime targets for hackers:**
 - **Powerful computing, large storage, fast networks**
- **Hard to root out successful invaders**



Cluster Security Difficulty Example:

- **Multiple clusters**
- **Many types of clusters**
- **Thousands of users**
- **Users can log in transparently from outside sites**
- **Users can log in from any machine anywhere**
- **Many user administered office machines**
- **Cluster administrators use same office machines**
- **No firewall machines allowed (performance)**



Ideal Cluster Security Layers

- **Router filtering**
- **Firewall(s)?**
- **Secure communications**
- **Machine security layers**
- **Authentication**
- **Intrusion Detection Services**
- **Network monitoring**
- **Common log host with log monitoring**
- **Pattern watching for networks and logs**



Ideal Machine Security Layers

- **Network stack /proc protections**
- **IP Masquerading and NATTING**
- **Host based packet filtering**
- **Hunt and kill un-needed services**
- **TCPwrappers**
- **Application configuration**



Router filtering

- **Block all source spoofed packets.**
- **Block all packets from outside going to RFC1918 addresses or claiming to come from RFC1918.**
- **Block all packets from outside going to cluster machine privileged ports except specific ports on specific machines such as ssh servers on user login nodes.**
- **Optionally block packets from outside going to cluster machine scary ports NFS (portmap, rpc), SMB, http**
- **Block packets from outside going to problem ports such as X, IRC (stop illicit servers from being set up),**
- **Drop ICMP packets from outside for redirect, echo-request, timestamp-request and replies, address-mask-request**



Authentication

- **Passwords are nice, but are not enough**
- **One time passwords are needed**
- **Radius servers**
- **Hardware tokens**
- **Leading vendors supply pam modules**
- **Leading vendors sometimes supply source**



Network Stack /proc Protections

Prevent address spoofing

- echo 0 to `/proc/sys/net/ipv4/conf/*/accept_source_route`
- echo 1 to `/proc/sys/net/ipv4/conf/*/rp_filter`
- echo 1 to `/proc/sys/net/ipv4/conf/*/log_martians`

Disable ICMP redirects

- echo 0 to `/proc/sys/net/ipv4/conf/*/accept_redirects`

Turn off bootp packet relaying

- echo 0 to `/proc/sys/net/ipv4/conf/*/bootp_relay`

Ignore ICMP bad error responses

- 1 to `/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses`

Enable syncookie protection

- echo 1 to `/proc/sys/net/ipv4/tcp_syncookies`



Host Packet Filtering

Examines each network packet, either:

- **ACCEPTS** the packet
- **Nicely REJECTS** the packet
- **Rudely DROPS** the packet

Linux uses iptables command to control

Most difficult of layers to set up, without:

Rule-set compilers such as PFILTER or equivalent

With rule-set compilers, best security return with the least effort because it protects against omissions in other security layers



Hunt and kill un-needed services.

- **Run only the services you need, no more, no fewer.**
- **Most important security layer**
(most network break-ins through service security holes)
- **Difficult way to tighten security**
(different distributions turn on/off services different ways)
(different services conditionally block access different ways)
- **Packet filtering can hide services left on mistakes**
- **Using common tools to find services**
 - **lsof**
 - **chkconfig**
 - **package manager**
 - **initialization scripts**



Using Isof to Find Services

- **Isof shows which network files are open:**
 - `% Isof -i | grep LISTEN | awk '{print $1,$(NF-2),$(NF-1)}' | sort | uniq`
 - `condor_ma TCP hn01.ncsa.uiuc.edu:1026`
 - `identd TCP *:auth`
 - `inetd TCP *:ftp`
 - `inetd TCP *:globus-gatekeeper`
 - `inetd TCP *:gsiftp`
 - `inetd TCP *:klogin`
 - `inetd TCP *:kshell`
 - `inetd TCP *:login`
 - `inetd TCP *:netsaint_remote`
 - `inetd TCP *:shell`



Finding init.d Started Services

- **To find the services that will be started by default at the current runlevel using /etc/rc.d/init.d scripts:**
 - `# chkconfig --list | grep `grep :initdefault: /etc/inittab | awk -F: '{print $2}` :on | awk '{print $1}' | sort | column`
 - atd isdn random
 - autofs keytable reconfig
 - condorg netfs sendmail
 - crond network sshd
 - globus nfslock syslog
 - gm pbs_mom verifyd
 - identd pcp xntpd



Finding xinetd Started Services

- **To find services started by xinetd:**
 - `# chkconfig --list | awk 'NF==2&&$2=="off"{print}' | awk -F: '{print $1}' | sort | column`
 - `echo`



Finding Network Visible Services

- **To find services visible from the network:**

- other-host# nmap host-to-be-looked-at

- **Port State Service**

- **21/tcp open ftp**

- **22/tcp open ssh**

- **23/tcp open telnet**

- **111/tcp open sunrpc**

- **113/tcp open auth**

- **513/tcp open login**

- **514/tcp open shell**

- **1026/tcp open nterm**

- **4321/tcp open rwhois**



Blocking Outside Access To Services

- **Best way to tighten security**
(most network break-ins through service security holes)
- **Difficult way to tighten security**
(different distributions turn on/off services different ways)
(different services conditionally block access different ways)
- **TCPwrappers can block/filter most services**
`/etc/hosts.allow` and `/etc/hosts.deny`



TCPwrappers

- **TCPwrappers can block/filter most services**
- **/etc/hosts.deny**
 - **ALL: ALL**
- **/etc/hosts.allow**
 - **Enable specific host sources for specific services (some are not obvious, NFS uses portmap entry, ssh uses sshd entry)**



Application Configuration

- **NFS (export only to a subset, read-only wherever possible, TCPwrappers only takes care of initial portmap connection filtering)**
- **Samba (separate protections/access lists for separate directory trees)**
- **HTTP**
- **etc.**



Miscellaneous Protections

- **Get rid of (chmod) unneeded SETUID/SETGID binaries.**
- **Encrypt when possible.**
- **Monitor system changes.**
- **“Sandboxes”**



Host Packet filtering

The host examines each network packet, looking at:

- Source and/or destination interface
- Protocol type (TCP or UDP or ICMP)
- Source address and port
- Destination address and port
- Connection state and other parameters depending on type

Each packet is either:

- Passed
- Dropped (nasty, makes port scanning more difficult, and makes harmless mis-guided connection attempts take a long time to timeout)
- Rejected (normal network protocol)



Stateful Packet Filtering

- **The kernel keeps track of the state of each network connection**
- **The kernel knows about appropriate ICMP messages**
- **The kernel keeps a timer on every active connection**



Stateful Packet Filtering

- **Linux kernels 2.4 and later, controlled by iptables**
- **OpenBSD FreeBSD NetBSD BSD Solaris Sunos, controlled by ipfilter**
- **Keeps track of active connections, examines each packet based on their place in a connection**
- **Can block ALL incoming packets and ALL forwarded packets on ALL ports except the ones needed to connect to approved services or ones that are part of an ongoing connection**
- **Rulesets can be very simple and still allow good security**



Stateful Packet Filtering Gotchas

- **Connection tracking will handle any simple port access protocol, but will not handle complicated port protocols such as ftp without special modules.**
- **Linux 2.4/iptables currently has these:**
 - ftp
 - irc
 - rpc/NFS
- **But lacks these:**
 - AFS



More Host Packet Filtering

- **Ideal single NIC situation:**
 - Host can initiate any outgoing network connection
 - No connections initiated from outside to host except as allowed by filtering rules
 - All packets of allowed connections pass
 - All other packets dropped or rejected
- **Simplified iptables ruleset:**
 - `iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`
 - `iptables -A INPUT -p tcp --destination-port ssh -j ACCEPT`
 - `iptables -A INPUT -j REJECT`



Still More Host Packet Filtering

Ideal double NIC situation:

- Firewall host can initiate any outgoing network connection
- Hosts on private subnet behind the firewall host can initiate any outgoing network connection through firewall
- Hosts on private subnet behind the firewall host are hidden from the outside, all connections are NAT'ed
- Some hosts on private subnet behind the firewall host are accessible from the outside on selected ports through pseudo interfaces created by the firewall host
- No connections initiated from outside to firewall host except as allowed by filtering rules
- All packets of allowed connections pass
- All other packets dropped or rejected



Even More host packet filtering

Simplified double NIC iptables ruleset:

- `iptables -A INPUT -p tcp --destination-port ssh -j ACCEPT`
- `iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`
- `iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT`
- `iptables -A pfilter -i INTERNAL_INTERFACE -m state --state NEW -j ACCEPT`
- `iptables -A INPUT -j REJECT`
- `iptables -A FORWARD -j REJECT`



Best Security Return on Invested Time

- **Packet filtering**

- Does the work of most of the other security layers
- Hard to set up
- Easy to mess up

- **What is needed:**

A good way to generate packet filtering rulesets that is:

- Easy to use
- Shields the user from packet filtering details
- (a packet filtering compiler)



PFILTER

a packet filtering firewall compiler



PFILTER Requirements

- Firewall packet filtering should be configured in a high level language, hiding any iptables/ipchains details
- Firewall configuration should be specified in a text configuration file to allow program scripting
- A GUI can be layered on top of the text configuration
- Default configuration should allow nothing in, but still allow any outgoing network connections
- Should be installed as a system service on Linux, with the configuration being re-compiled on (re)starts
- Should be Linux version independent
- Should be available as RPMS
- Should be available as a “make install” tarball



Further PFILTER Requirements

- **The configuration language should support:**
 - defined constants
 - changeable variables
 - macros
 - passing parameters to macros
 - private variables within macro invocations
 - conditional blocks
 - looping
 - flexible packet forwarding
 - passed-through raw iptables/ipchains commands
 - address ranges
 - address masks
 - case independence



More PFILTER Requirements

- **PFILTER should support multiple network interfaces each with these settable attributes:**
 - Whether it is on a trusted network or whether all packets coming in on the NIC are filtered
 - Whether the host machine is providing a NATTING, packet forwarding type firewall for the NIC's network
- **Interface attributes should, if not specified for an interface, be defaulted to:**
 - Filtered (all packets from that interfaces addressed to the host should be filtered, instead of being blindly allowed)
 - Provide NATTING and packet forwarding for machines on the attached network if the interface address is RFC 1918



Even More PFILTER Requirements

- **Aliasing should be supported on hosts with multiple network interfaces. Hosts on a protected private network that are “aliased” have these done for them:**
 - **A pseudo interface with it’s own IP address is created on the public network side for each “aliased” hosts, allowing them to be “visible” on the public network.**
 - **Allowed network packets are forwarded in and out to the protected hidden hosts.**
 - **The PFILTER firewall host should do all packet filtering for the “aliased” hosts, without them having to cooperate.**
 - **Outgoing network connections from an “aliased” host should be NAT’ed so that they appear to be originating from the “aliased” host’s public pseudo IP address.**



And Even More PFILTER Requirements

- **The “compiled” output should:**
 - Be a shell script to allow shell script snippets to be passed through
 - Show the configuration lines that produce output as comments
 - Allow the “glue” code to be changed through text files
 - Be Linux version independent
 - Do the right things with /proc controlled network stack attributes
 - Load needed kernel modules as needed
 - By default block incoming ICMP packets except for “destination unreachable” and redirect packets for error
 - Keep the ordering of filtering specified in the configuration file
- **Once compiled, the output script should be able to be used without re-compiling, so that the administrator then tweak it themselves or use it as a template for their own efforts**



Did You Think I Was Done With Requirements?

- **Network “services” that can be filtered should:**
 - Be able to be defined in text configuration files
 - Be able to be defined with one line if they are simple a list of TCP/UDP/ICMP ports
 - Support shell script snippets
 - Support constants, variables, macros, looping, condition output
- **Standard system services can be filtered should:**
 - Be able to be specified by name or number
 - System files such as /etc/services should be “mined” for names
 - Have nicer names be defined, such as “ping”
- **PFILTER should optimize the compiled output**



PFILTER on a 2 NIC machine

- Automatically determines which is the protected internal interface by looking for RFC 1918 private addresses
- By default allows machines on the internal/protected interface to access any outside network service using NAT'ed IP masquerading for all connections from inside
- Allows internal hosts to have some services be accessible from the outside using translated address/ports. These hosts can appear to respond to a separate external address than the one that PFILTER is running on. This allows hosts behind the firewall machine to not have to be security tightened, while still allowing selected access to them from the outside.



Main Parts of a PFILTER Configuration File

- **ALIAS** – to allow a hidden host to be “visible”
- **CLOSE** - blocks network connections
- **%constant%** - defines a fixed value named constant
- **DROP** – specifies that packets are dropped
- **FORWARD** – to set up packet forwarding/translation
- **%if%** - various conditional generation forms
- **INTERFACE** – specified network interface attributes
- **LOGGING** – specified packet rejection logging options
- **%loop%** - invokes a generated loop
- **%macro%** - defines a macro
- **OPEN** - allows network connections
- **REJECT** – specifies that packets are rejected
- **(UN)TRUSTED** – specifies (un)trusted network/addresses/ranges
- **%variable%** - defines/re-defines a named variable



OPEN and CLOSE directives

This opens/closes network services/ports on the PFILTER host machine or for machines that the host is providing “aliasing” for:

**OPEN/CLOSE [protocol(s)] service(s) port(s) \
[from source(s)] [to destination(s)] [on interface(s)]**

Examples:

- OPEN ssh
- OPEN nfs from 192.168.1.0/24
- OPEN dhcp on eth2
- OPEN tcp 17:19 89 udp 234 from host.domain.name on eth0
- CLOSE all from hackers.heaven.host
- OPEN http ssh tcp 8080 to aliased.host from our.domain/mask



ALIAS directives

This allows hidden hosts on a private subnet to appear to be outside the firewall machine with selected services on the hosts being accessible from outside:

```
ALIAS external_address hidden_address [services/ports] \  
[from source(s)] [on interface(s)]
```

For example, a hidden host is attached to one network interface at an address which is named as hidden1 in the /etc/hosts file, we want the hidden host to be accessible through ssh and be a web server on the public DNS address exposed1.our.domain:

```
ALIAS exposed1.our.domain hidden1 ssh http
```

or we could use two lines instead:

```
ALIAS exposed1.our.domain hidden1  
OPEN ssh http to exposed1.our.domain
```

or we could use these two lines:

```
ALIAS exposed1.our.domain hidden1  
OPEN ssh http to hidden1
```



Simplest Configuration File

- **As provided by the installation as a default:**

```
open  ssh
```

This does the following:

- All outgoing connections allowed
- No incoming connections allowed, except ssh
- If two network interfaces, and one has an RFC1918 address:
 - Sets up NATTING for machines on private network
 - Sets up packet forwarding for machines on private network
- Network stack protections enabled
- Limited packet rejection logging enabled
- Rejected packages nicely rejected instead of being dropped



Default Configuration File on OSCAR Clusters

- On an OSCAR server with a private network for the nodes on eth1, with server name “master” and nodes named “node1” through “node6”, here are some portions of the generated file:

```
# define the main OSCAR server and the nodes
```

```
%define oscar_server master
```

```
%define% nodes node1 node2 node3 node4 node5 node6
```

```
# the server needs to be listed as a dhcp server for the nodes
```

```
# because opening up that service requires opening up some
```

```
# broadcast stuff as well, so simply listing the nodes as
```

```
# trusted is not sufficient
```

```
open dhcp on eth1
```

```
# the server trusts the compute nodes
```

```
trusted %oscar_server% %nodes
```



PFILTER, A Work in Progress

- **PFILTER is in use:**
 - On NCSA clusters
 - In various NCSA X-in-a-box products
 - As part of OSCAR
- **Open sourced (GNU license)**
- **Resides on sourceforge**



PFILTER Future Versions

- **GUI**
- **TCPwrappers <> PFILTER synchronization**
- **System logs analysis for hacking attempts reporting**



Further PFILTER Information

- **Contact me at:**
`ngorsuch@ncsa.uiuc.edu`
- **PFILTER sourceforge project page:**
 - <http://sourceforge.net/projects/pfilter/>
- **PFILTER latest versions and cvs mirror:**
 - <ftp://sponge.ncsa.uiuc.edu/pfilter/>
 - <http://sponge.ncsa.uiuc.edu/ftp/pfilter/>



Checking your work

- **Cracker tools**
 - nmap
- **Admin tools**
 - Nessus
 - Snort
 - Tripwire and friends



Security Rewards

- **Reward for working security:**
 - “Keep on doing your job and don’t bother me again.”
 - “And NO, you can’t close that service, some users want it.”



Security Rewards

- **Reward for less than perfect security:**
 - “And how in the **** did we end up on CNN?”
 - “How did YOU let this all happen?”
 - “How could you let a 14 year old get the better of you?”



The PERFECT Firewall



Questions?

