Kernel Security : Netfilter/iptables

Gateway, Firewalls, LoadBalance

Mr. Sawangpong Muadphet
sawangpong@itbakery.com

IT Bakery Co., Ltd, Bangkok, Thailand

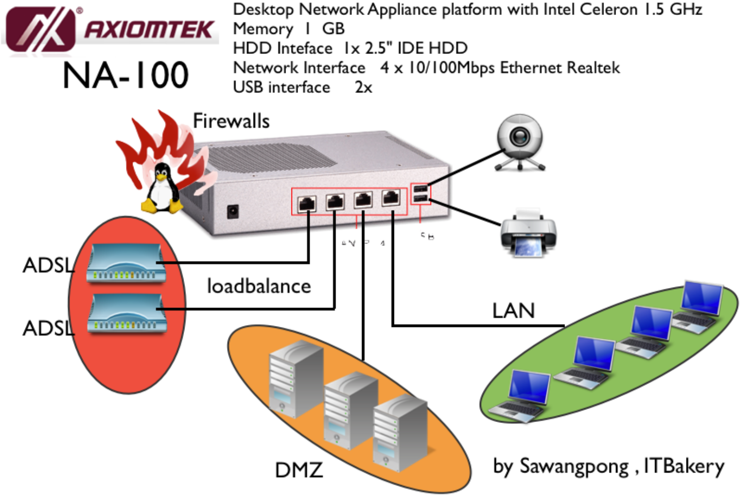
Sixth Asia OSS Workshop
8 to 10 March 2010

Discussion Outline

1. Overview
   - SOHO Firewall Box
   - Netfilter Architecture
   - basic iptables
   - Netfilter Connection Tracking
   - Basic Ipsets
2. Objective
   - CentOS 5.4 with iptables 1.4.5 and L7 Netfilter Support
   - Integrate IP sets to IPTABLES
   - Create Blacklist with IPsets
   - Create tracking with conntrack-tools
   - Improve Logging System with Ulogd
3. Advance Topic: Netfilter Hook function
   - Netfilter is framework
   - register | unregister Callback
4. Conclusion

Netfilter Packet flow with tables

RICE

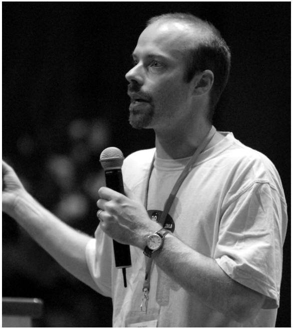


Desktop Network Appliance platform with Intel Celeron 1.5 GHz
Memory 1 GB
HDD Inteface 1x 2.5" IDE HDD
Network Interface 4 x 10/100Mbps Ethernet Realtek
USB interface 2x

by Sawangpong , ITBakery

Who is Rusty Russell?

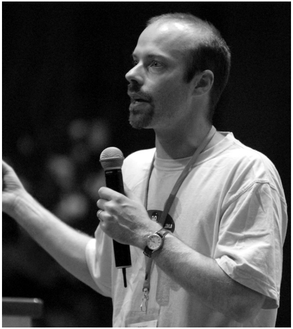
The Netfilter project was found by Paul Rusty Russell during kernel 2.3. Rusty build Netfilter Frame work from scratch. He build a set of hooks over the network protocol stack.



Russell wrote the packet filtering systems ipchains and netfilter/iptables in the Linux operating system kernel. Linus Torvalds has referred to him as one of his "top deputies"

Who is Rusty Russell?

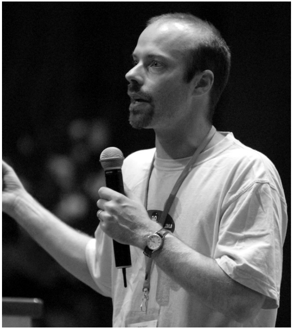
The Netfilter project was found by Paul Rusty Russell during kernel 2.3. Rusty build Netfilter Frame work from scratch. He build a set of hooks over the network protocol stack.



Russell wrote the packet filtering systems ipchains and netfilter/iptables in the Linux operating system kernel. Linus Torvalds has referred to him as one of his "top deputies"

Who is Rusty Russell?

The Netfilter project was found by Paul Rusty Russell during kernel 2.3. Rusty build Netfilter Frame work from scratch. He build a set of hooks over the network protocol stack.



Russell wrote the packet filtering systems ipchains and netfilter/iptables in the Linux operating system kernel. Linus Torvalds has referred to him as one of his "top deputies"

Why Netfilter/Iptables?

Packet filtering policies were based uniquely on the packet header information, such as the IP source, destination, and ports are **OVER**. this is not **insufficient protection** against probes and denial-of-service attacks. Moreover there iss application protocal such as FTP, TFTP, IRC. PPTP has aspects that are hard to track

Netfilter Architecture

netfilter
firewalling, NAT, and packet mangling for linux

Netfilter is a framework inside Linux 2.4.x and 2.6.x kernel series that intercepts network traffic at various predefined points (i.e. hooks) in network protocol stack and facilitates,

1. Packet Filtering
2. Packet Mangling
3. Stateless/stateful Firewalling
4. Network Address Translation (NAT)

Netfilter Architecture

RICE

netfilter
firewalling, NAT, and packet mangling for linux

Netfilter is a framework inside Linux 2.4.x and 2.6.x kernel series that intercepts network traffic at various predefined points (i.e. hooks) in network protocol stack and facilitates,

1. Packet Filtering
2. Packet Mangling
3. Stateless/stateful Firewalling
4. Network Address Translation (NAT)

Netfilter Architecture

netfilter
firewalling, NAT, and packet mangling for linux

Netfilter is a framework inside Linux 2.4.x and 2.6.x kernel series that intercepts network traffic at various predefined points (i.e. hooks) in network protocol stack and facilitates,

1. Packet Filtering
2. Packet Mangling
3. Stateless/stateful Firewalling
4. Network Address Translation (NAT)

Netfilter Architecture

Netfilter is a framework inside Linux 2.4.x and 2.6.x kernel series that intercepts network traffic at various predefined points (i.e. hooks) in network protocol stack and facilitates,

1. Packet Filtering
2. Packet Mangling
3. Stateless/stateful Firewalling
4. Network Address Translation (NAT)

Overview of the Linux Packet filter filter framework RICE

The packet filter framework on linux is divided in to two path

- **Netfilter/Xtables** -- the kernel-space portion
  netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.
- **iptables** -- the user-space portion
  iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).
- **kernel module is key**
  netfilter, ip_tables, connection tracking (ip_conntrack, nf_conntrack) and the NAT subsystem together build the

Overview of the Linux Packet filter filter framework RICE

The packet filter framework on linux is divided in to two path

- **Netfilter/Xtables** -- the kernel-space portion
  netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.
- **iptables** -- the user-space portion
  iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).
- **kernel module is key**
  netfilter, ip_tables, connection tracking (ip_conntrack, nf_conntrack) and the NAT subsystem together build the

Overview of the Linux Packet filter filter framework RICE

The packet filter framework on linux is divided in to two path

- **Netfilter/Xtables** -- the kernel-space portion
  netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.
- **iptables** -- the user-space portion
  iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).
- **kernel module is key**
  netfilter, ip_tables, connection tracking (ip_conntrack, nf_conntrack) and the NAT subsystem together build the

Understand iptable

RICE

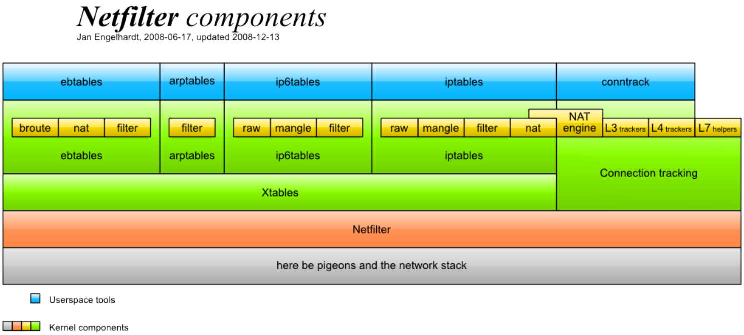
iptables does not register with any netfilter hooks: it relies on other modules to do that and feed it the packets as appropriate; a module must register the netfilter hooks and ip_tables **separately**, and provide the mechanism to call ip_tables when the hook is reached.

lsmod command

```
lsmod | grep ip_tables
ip_tables 11692 1 iptable_filter
x_tables 16544 1 ip_tables
```

lsmod shows information about all loaded modules. The format is name, size, use count, list of referring modules. The information displayed is identical to that available from /proc/modules. /proc/modules shows what kernel modules (drivers) are currently loaded.

Netfilter Packet flow with tables



Netfilter components

Jan Engelhardt, 2008-06-17, updated 2008-12-13

ebtables | arptables | ip6tables | iptables | conntrack

broute | nat | filter

filter

raw | mangle | filter

raw | mangle | filter | nat

NAT engine | L3 trackers | L4 trackers | L7 helpers

ebtables | arptables | ip6tables | iptables

Xtables

Connection tracking

Netfilter

here be pigeons and the network stack

🟦 Userspace tools

▨ Kernel components

tables

- **filter table**
  for doing the actual packet filtering. This is the default table table if we not specify one when create rules.
- **nat table**
  for rewriting packet source and/or destination (IP Address)
- **mangle table**
  for altering headers and/or contents
- **raw table**
  for avoiding connection tracking

tables

- **filter table**
  for doing the actual packet filtering. This is the default table table if we not specify one when create rules.
- **nat table**
  for rewriting packet source and/or destination (IP Address)
- **mangle table**
  for altering headers and/or contents
- **raw table**
  for avoiding connection tracking

tables

- **filter table**
  for doing the actual packet filtering. This is the default table table if we not specify one when create rules.
- **nat table**
  for rewriting packet source and/or destination (IP Address)
- **mangle table**
  for altering headers and/or contents
- **raw table**
  for avoiding connection tracking

tables

- **filter table**
  for doing the actual packet filtering. This is the default table table if we not specify one when create rules.
- **nat table**
  for rewriting packet source and/or destination (IP Address)
- **mangle table**
  for altering headers and/or contents
- **raw table**
  for avoiding connection tracking

Built-in Hook

- **INPUT chain**
  All packets that go to localhost must traverse this hook. present in mangle and filter tables.
- **OUTPUT chain**
  All packets that leaving localhost nust traverse this hook. present in raw,nat,mangle and filter tables.
- **FORWARD**
  All packets that not go to localhost must traverse this chain hook. present in mangle and filter tables.

Built-in Hook

- **INPUT chain**
  All packets that go to localhost must traverse this hook. present in mangle and filter tables.
- **OUTPUT chain**
  All packets that leaving localhost nust traverse this hook. present in raw,nat,mangle and filter tables.
- **FORWARD**
  All packets that not go to localhost must traverse this chain hook. present in mangle and filter tables.

Built-in Hook

- **INPUT chain**
  All packets that go to localhost must traverse this hook. present in mangle and filter tables.
- **OUTPUT chain**
  All packets that leaving localhost nust traverse this hook. present in raw,nat,mangle and filter tables.
- **FORWARD**
  All packets that not go to localhost must traverse this chain hook. present in mangle and filter tables.

Built-in Chains (cont'd)

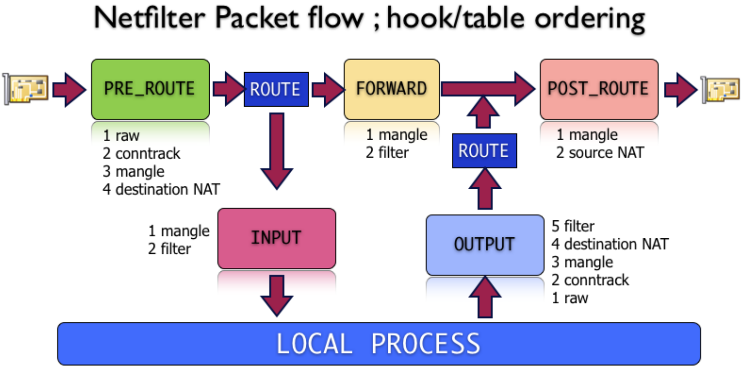
- **PREROUTING chain**
  All packets traverse this chain before a routing decision is made by kernel. present in raw, nat and mangle tables. Destination Network Translation(DNAT) is implement here
- **POSTROUTING chain**
  All packets traverse this hook after a routing decision is made by kernel. present in the nat, mangle table Source Network Address Translation(SNAT) is register to this hook.

Built-in Chains (cont'd)

- **PREROUTING chain**
  All packets traverse this chain before a routing decision is made by kernel. present in raw, nat and mangle tables. Destination Network Translation(DNAT) is implement here
- **POSTROUTING chain**
  All packets traverse this hook after a routing decision is made by kernel. present in the nat, mangle table Source Network Address Translation(SNAT) is register to this hook.

Netfilter Packet flow with tables

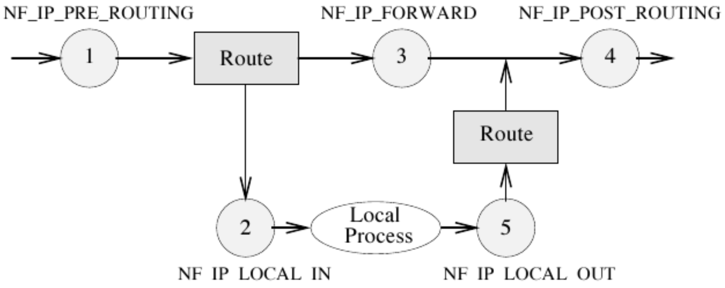
RICE



by Sawangpong, ITbakery. Sun 7 Mar, 2010

many people confuse with iptables because netfilter use also same chain name and hook name.

hooks name order by packets flow



1=NF_IP_PRE_ROUTING
2=NF_IP_LOCAL_IN
3=NF_IP_FORWARD
4=NF_IP_POST_ROUTING
5=NF_IP_LOCAL_OUT

Target in Packet Filtering **filter** table

RICE

Builtin Target to be used in filter table

ACCEPT accept the packets
DROP silently drop the packet
QUEUE enqueue packet to userspace
RETURN return to previous chain
USERDEFINE user defined chain

Target implement as **loadable modules**

REJECT drop the packet but inform sender
MIRROR change source/destination IP and resend.
LOG log via syslog,syslog-ng (facility local1-7)
ULOG log via userspace

Target in Packet Filtering **filter** table

Builtin Target to be used in filter table

ACCEPT accept the packets
DROP silently drop the packet
QUEUE enqueue packet to userspace
RETURN return to previous chain
USERDEFINE user defined chain

Target implement as **loadable modules**

REJECT drop the packet but inform sender
MIRROR change source/destination IP and resend.
LOG log via syslog,syslog-ng (facility local1-7)
ULOG log via userspace

basic iptables syntax

Add and Delete a rule

iptables [-t table] -[AD] chain rule-spec [options]

Examples:

```
iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -D INPUT -p tcp --dport 22 -j ACCEPT
```

Note -A option means we append or add this rule to the end of the chain

basic iptables syntax

Add and Delete a rule

iptables [-t table] -[AD] chain rule-spec [options]

Examples:

```
iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -D INPUT -p tcp --dport 22 -j ACCEPT
```

Note -A option means we append or add this rule to the end of the chain

basic iptables syntax cont'd

Insert a rule into chain

iptables [-t table] -[ID] chain [rulenum] rule-spec [options]

Examples:

iptables -I INPUT 2 -p tcp --dport 110 -j ACCEPT iptables -D INPUT 2

Note -inset rule to chain input at line number 2
we use 'iptable -L' to get line number

basic iptables syntax cont'd

Insert a rule into chain

iptables [-t table] -[ID] chain [rulenum] rule-spec [options]

Examples:

iptables -I INPUT 2 -p tcp --dport 110 -j ACCEPT iptables -D INPUT 2

Note -inset rule to chain input at line number 2
we use 'iptable -L' to get line number

basic iptables syntax cont’d

Flush (Delete) all rule from chain

```
iptables [-t table] -F chain [options]
```

Examples:

```
iptables -t filter -F INPUT
iptables -t nat -F POSTROUTING
```

basic iptables syntax cont'd

Flush (Delete) all rule from chain

```
iptables [-t table] -F chain [options]
```

Examples:

```
iptables -t filter -F INPUT
iptables -t nat -F POSTROUTING
```

basic iptables syntax cont'd

Set the default policy

iptables [-t table] -P chain target [options]

Examples:

iptables -t filter -P INPUT DROP

sets the default action it packets not match any rule in chain

basic iptables syntax cont'd

Set the default policy

iptables [-t table] -P chain target [options]

Examples:

iptables -t filter -P INPUT DROP

sets the default action it packets not match any rule in chain

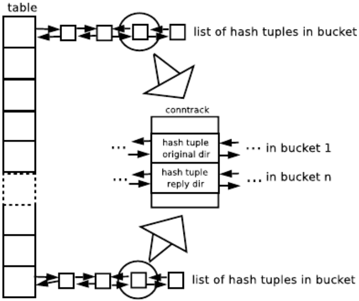
Basic Idea Connection Tracking

The connection tracking system stores information about state of connection in memory structure that contain source and destination ip address, port number, protocol type, state, and time out. The connection tracking does not filter the packets themselves, the default behavior alway let packets go. They just tracking and provide the way to track them. The System admin will define rule to LOG or DROP.

Basic Structure

The connection tracking system is option modular loadable subsystem, alway require by NAT subsystem. every connection have 2 hash tuples represent the relevant information connection. one for original direction and one for reply.

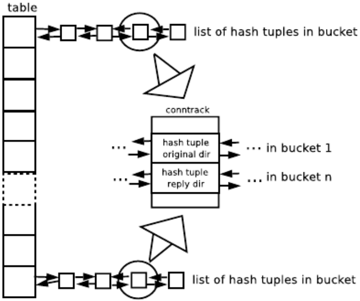
tuple defind in **include/net/netfilter/nf_conntrack_tuple.h** line63 structure nf_conntrack_tuple



Basic Structure

The connection tracking system is option modular loadable subsystem, alway require by NAT subsystem. every connection have 2 hash tuples represent the relevant information connection. one for original direction and one for reply.

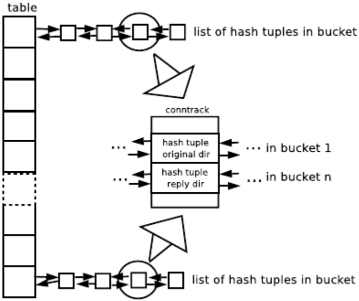
tuple defind in **include/net/netfilter/nf_conntrack_tuple.h** line63 structure nf_conntrack_tuple



Basic Structure

The connection tracking system is option modular loadable subsystem, alway require by NAT subsystem. every connection have 2 hash tuples represent the relevant information connection. one for original direction and one for reply.

tuple defind in **include/net/netfilter/nf_conntrack_tuple.h** line63 structure nf_conntrack_tuple



States

RICE

- **NEW:** The connection starting. valid TCP connection, a SYN packet is received. firewall only seen traffic in on way
- **ESTABLISHED:** The connection has been established. firewall has been seen two-way communication
- **RELATED:** The connection that relate each other. Like FTP use port 21 for control operation. but it use TCP port 1024/65535 to receive the data request. The connections tracking system define **helper** that let system identify the relation.
- **INVALID:** INVALID packets.

States

- **NEW:** The connection starting. valid TCP connection, a SYN packet is received. firewall only seen traffic in on way
- **ESTABLISHED:** The connection has been established. firewall has been seen two-way communication
- **RELATED:** The connection that relate each other. Like FTP use port 21 for control operation. but it use TCP port 1024/65535 to receive the data request. The connections tracking system define **helper** that let system identify the relation.
- **INVALID:** INVALID packets.

States

- **NEW:** The connection starting. valid TCP connection, a SYN packet is received. firewall only seen traffic in on way
- **ESTABLISHED:** The connection has been established. firewall has been seen two-way communication
- **RELATED:** The connection that relate each other. Like FTP use port 21 for control operation. but it use TCP port 1024/65535 to receive the data request. The connections tracking system define **helper** that let system identify the relation.
- **INVALID:** INVALID packets.

States

RICE

- **NEW:** The connection starting. valid TCP connection, a SYN packet is received. firewall only seen traffic in on way
- **ESTABLISHED:** The connection has been established. firewall has been seen two-way communication
- **RELATED:** The connection that relate each other. Like FTP use port 21 for control operation. but it use TCP port 1024/65535 to receive the data request. The connections tracking system define **helper** that let system identify the relation.
- **INVALID:** INVALID packets.

Log report stealth scan by tracking legitimate connection

RICE

Example iptables to report on scan ports

```
iptables -A INPUT -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit-burst 5 -j LOG --log-level info --log-prefix ’###Stealth Scan###’
iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -m limit --limit 5/m -j LOG --log-level info --log-prefix ’###XMAS Scan### ’
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -m limit --limit 5/m -j LOG --log-level info --log-prefix ’###SYN/RST Scan###’
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit --limit 5/m -j LOG --log-level info --log-prefix ’###SYN/FIN Scan###’
```

Don’t miss it.

Format

```
ipset -N [setnam] [settype] --from [ip1] --to [ip2]
```

Examples:

```
ipset -N myset macipmap –from 192.168.0.10 –to
192.168.0.250
ipset -nL
Name: myset
Type: macipmap
References: 0
Header: from: 192.168.0.10 to: 192.168.0.250
Members
```

Don’t miss it.

Format

```
ipset -N [setnam] [settype] --from [ip1] --to [ip2]
```

Examples:

```
ipset -N myset macipmap –from 192.168.0.10 –to
192.168.0.250
ipset -nL
Name: myset
Type: macipmap
References: 0
Header: from: 192.168.0.10 to: 192.168.0.250
Members
```

Basic use ipset

RICE

Add member into List of ipsets

```
ipset -A myset 192.168.0.11,AA:BB:CC:DD:EE:FF
```

Examples:

```
ipset -N myset macipmap –from 192.168.0.10 –to
192.168.0.250
ipset -nL
Name: myset
Type: macipmap
References: 0
Header: from: 192.168.0.10 to: 192.168.0.250
Members
192.168.0.11,AA:BB:CC:DD:EE:FF
```

Basic use ipset

Add member into List of ipsets

```
ipset -A myset 192.168.0.11,AA:BB:CC:DD:EE:FF
```

Examples:

```
ipset -N myset macipmap –from 192.168.0.10 –to 192.168.0.250
ipset -nL
Name: myset
Type: macipmap
References: 0
Header: from: 192.168.0.10 to: 192.168.0.250
Members
192.168.0.11,AA:BB:CC:DD:EE:FF
```

Add ipset to iptables

Add to iptables

```
iptables -A FORWARD -m set –set myset src -j ACCEPT
```

Delete from iptables

```
ipset -D myset 192.168.0.11,AA:BB:CC:DD:EE:FF
```

Add ipset to iptables

Add to iptables

```
iptables -A FORWARD -m set –set myset src -j ACCEPT
```

Delete from iptables

```
ipset -D myset 192.168.0.11,AA:BB:CC:DD:EE:FF
```

Our environment

CentOS 5.4 + iptables 1.4.5 + L7 Netfilter Support

target Linux Distro

CentOS 5.4 Community ENTerprise

target new kernel

linux-2.6.28.9.tar.bz2 (old kernel is 2.6.18-164.11.1.el5)

target new iptables

iptables-1.4.5.tar.gz

Our environment

CentOS 5.4 + iptables 1.4.5 + L7 Netfilter Support

target Linux Distro

CentOS 5.4 Community ENTerprise

target new kernel

linux-2.6.28.9.tar.bz2 (old kernel is 2.6.18-164.11.1.el5)

target new iptables

iptables-1.4.5.tar.gz

Our environment

CentOS 5.4 + iptables 1.4.5 + L7 Netfilter Support

target Linux Distro

CentOS 5.4 Community ENTerprise

target new kernel

linux-2.6.28.9.tar.bz2 (old kernel is 2.6.18-164.11.1.el5)

target new iptables

iptables-1.4.5.tar.gz

target Layer 7 Netfilter kernel patch

netfilter-layer7-v2.22.tar.gz

L7 protocols

l7-protocols-2009-05-28.tar.gz

imq patch

linux-2.6.28.9-imq-test2.diff

target Layer 7 Netfilter kernel patch

netfilter-layer7-v2.22.tar.gz

L7 protocols

l7-protocols-2009-05-28.tar.gz

imq patch

linux-2.6.28.9-imq-test2.diff

target Layer 7 Netfilter kernel patch

netfilter-layer7-v2.22.tar.gz

L7 protocols

l7-protocols-2009-05-28.tar.gz

imq patch

linux-2.6.28.9-imq-test2.diff

Migrate kernel

RICE

build and patch kernel

```
#extract files to /usr/src
tar jxf linux-2.6.28.9.tar.bz2 -C /usr/src/
tar jxf iptables-1.4.5.tar.bz2 -C /usr/src/
tar zxf netfilter-layer7-v2.22.tar.gz -C /usr/src/
tar zxf l7-protocols-2009-05-28.tar.gz -C /usr/src/
cp linux-2.6.28.9-imq-test2.diff /usr/src
#create link cd /usr/src/
ln -s linux-2.6.28.9 linux
ln -s iptables-1.4.5 iptables
ln -s netfilter-layer7-v2.22 netfilter
```

patch kernel

Example

```
patch kernel cd linux
cp /boot/config-2.6.18-164.11.1.el5 .config
patch -p1 < ../netfilter/kernel-2.6.25-2.6.28-layer7-2.22.patch
patch -p1 < ../linux-2.6.28.9-imq-test2.diff
```

menu select select L7 and imq

make menuconfig
— select Networking Support
> Networking options
> Network packet filtering framework (Netfilter)
> Core netfilter configuration

Note I select all module!! for testing

patch kernel

Example

```
patch kernel cd linux
cp /boot/config-2.6.18-164.11.1.el5 .config
patch -p1 < ../netfilter/kernel-2.6.25-2.6.28-layer7-2.22.patch
patch -p1 < ../linux-2.6.28.9-imq-test2.diff
```

menu select select L7 and imq

```
make menuconfig
— select Networking Support
> Networking options
> Network packet filtering framework (Netfilter)
> Core netfilter configuration
```

Note I select all module!! for testing

patch detail

RICE

```
patch -p1 < ../netfilter/kernel-2.6.25-2.6.28-layer7-2.22.patch
```

```
patching file net/netfilter/Kconfig
patching file net/netfilter/Makefile
patching file net/netfilter/xt_layer7.c
patching file net/netfilter/regexp/regexp.c
patching file net/netfilter/regexp/regexp.h
patching file net/netfilter/regexp/regmagic.h
patching file net/netfilter/regexp/regsub.c
patching file net/netfilter/nf_conntrack_core.c
patching file net/netfilter/nf_conntrack_standalone.c
patching file include/net/netfilter/nf_conntrack.h
patching file include/linux/netfilter/xt_layer7.h
```

patch -p1 < ../linux-2.6.28.9-imq-test2.diff

```
patching file drivers/net/imq.c
patching file drivers/net/Kconfig
patching file drivers/net/Makefile
patching file include/linux/imq.h
patching file include/linux/netdevice.h
patching file include/linux/netfilter/xt_IMQ.h
patching file include/linux/netfilter_ipv4/ipt_IMQ.h
patching file include/linux/netfilter_ipv6/ip6t_IMQ.h
patching file include/linux/skbuff.h
patching file include/net/netfilter/nf_queue.h
patching file net/core/dev.c
patching file net/core/skbuff.c
patching file net/netfilter/Kconfig
patching file net/netfilter/Makefile
patching file net/netfilter/nf_queue.c
patching file net/netfilter/xt_IMQ.c
```

build new kernel and iptables

start:

```
make all
make modules_install
make install
```

build iptables with new soure kernel

```
cd /usr/src/iptables
cp ../netfilter/iptables-1.4.3forward-for-kernel-2.6.20forward/* extensions/
./configure --with-kernel=/usr/src/linux
make
make install
iptables -v
mv /usr/src/l7-protocols-2009-05-28 /etc/l7-protocols
modprobe xt_layer7
```

build new kernel and iptables

RICE

start:

```
make all
make modules_install
make install
```

build iptables with new soure kernel

```
cd /usr/src/iptables
cp ../netfilter/iptables-1.4.3forward-for-kernel-2.6.20forward/* extensions/
./configure --with-kernel=/usr/src/linux
make
make install
iptables -v
mv /usr/src/l7-protocols-2009-05-28 /etc/l7-protocols
modprobe xt_layer7
```

build ipp2p module for iptables

prepare all file

ipp2p-0.8.2.tar.gz - IPP2P Version 0.8.2 for kernel 2.4 & 2.6
#patch file ipp2p-0.8.2-kernel-2.6.22.patch
ipp2p-0.8.2-kernel-2.6.28.patch
ipp2p-0.8.2-iptables-1.4.0.patch
ipp2p-0.8.2-iptables-1.4.1.patch
ipp2p-0.8.2-iptables-1.4.3.patch

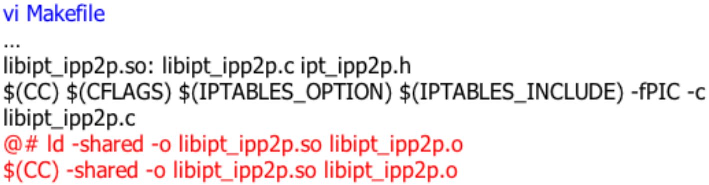
build ipp2p module for iptables cont’d

patch and make

```
tar zxvf ipp2p-0.8.2.tar.gz -C /usr/src
cd /usr/src/ipp2p-0.8.2
patch -p1 < ../ipp2p-0.8.2-kernel-2.6.22.patch
patch -p1 < ../ipp2p-0.8.2-kernel-2.6.28.patch
patch -p1 < ../ipp2p-0.8.2-iptables-1.4.0.patch
patch -p1 < ../ipp2p-0.8.2-iptables-1.4.1.patch
patch -p1 < ../ipp2p-0.8.2-iptables-1.4.3.patch
```

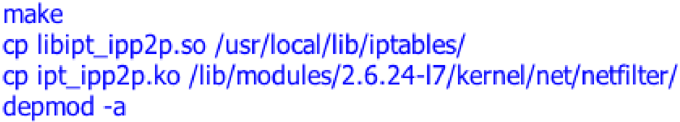
Edit Makefile

vi Makefile
# you have go line 67 and edit



```
vi Makefile
...
libipt_ipp2p.so: libipt_ipp2p.c ipt_ipp2p.h
$(CC) $(CFLAGS) $(IPTABLES_OPTION) $(IPTABLES_INCLUDE) -fPIC -c
libipt_ipp2p.c
@# ld -shared -o libipt_ipp2p.so libipt_ipp2p.o
$(CC) -shared -o libipt_ipp2p.so libipt_ipp2p.o
```

#make it and copy modules to iptables



```
make
cp libipt_ipp2p.so /usr/local/lib/iptables/
cp ipt_ipp2p.ko /lib/modules/2.6.24-l7/kernel/net/netfilter/
depmod -a
```

Enable LAYER 7 to Block bittorrent RICE

add rule to iptable

```
depmod -a
iptables -A FORWARD -m --ipp2p -j DROP
iptables -A FORWARD -m layer7 --proto bittorrent -j DROP
```

Note:'–ipp2p' is equal to '–edk –dc –kazaa –gnu –bit –apple –winmx –soul –ares

Example

```
test iptables -nvL
iptables -m ipp2p –help
lsmod | grep 'ipp2p|layer7'
```

Enable LAYER 7 to Block bittorrent

RICE

add rule to iptable

```
depmod -a
iptables -A FORWARD -m --ipp2p -j DROP
iptables -A FORWARD -m layer7 --proto bittorrent -j DROP
```

Note:’–ipp2p’ is equal to ’–edk –dc –kazaa –gnu –bit –apple –winmx –soul –ares

Example

```
test iptables -nvL
iptables -m ipp2p –help
lsmod | grep ’ipp2p|layer7’
```

What is Ipsets? ipset.netfilter.org

IP sets are a framework inside the Linux 2.4.x and 2.6.x kernel, Ipsets are an extension to Netfilter/iptables. Ipset allows you to create one or more named sets of addresses then use those sets to define Netfilter/iptables rules.

prepare all file

ipset-4.2.tar.bz2

Example

```
build ipset tar jxf ipset-4.2.tar.bz2 -C /usr/src/
cd /usr/src/ipset-4.2
KERNEL_DIR=/usr/src/linux make
KERNEL_DIR=/usr/src/linux make install
cp kernel/include/linux/netfilter_ipv4/ip_set.h
to /usr/src/iptables/include/linux/netfilter_ipv4/
# recompile IPTABLES again
```

What is Ipsets? ipset.netfilter.org

IP sets are a framework inside the Linux 2.4.x and 2.6.x kernel, Ipsets are an extension to Netfilter/iptables. Ipset allows you to create one or more named sets of addresses then use those sets to define Netfilter/iptables rules.

prepare all file

ipset-4.2.tar.bz2

Example

```
build ipset tar jxf ipset-4.2.tar.bz2 -C /usr/src/
cd /usr/src/ipset-4.2
KERNEL_DIR=/usr/src/linux make
KERNEL_DIR=/usr/src/linux make install
cp kernel/include/linux/netfilter_ipv4/ip_set.h
to /usr/src/iptables/include/linux/netfilter_ipv4/
# recompile IPTABLES again
```

Possible uses of ipsets: BlackList

RICE

#take from **PeerGuardian Blocklist**

download the convert script and compile will gcc

```
wget http://www.maeyanie.com/pg2ipset.c
gcc -O3 -o pg2ipset pg2ipset.c
```

Example

```
convert and create ipset name LEVEL1 curl -L
http://www.bluetack.co.uk/config/level1.gz | gunzip -c |
./pg2ipset -- LEVEL1 | ipset -R
#Example Converted 224283 rules.
. . .
-A LEVEL1 222.231.45.0-222.231.45.255
-A LEVEL1 222.233.164.0-222.233.179.255
. . .
```

Possible uses of ipsets: BlackList

#take from **PeerGuardian Blocklist**

download the convert script and compile will gcc

```
wget http://www.maeyanie.com/pg2ipset.c
gcc -O3 -o pg2ipset pg2ipset.c
```

Example

```
convert and create ipset name LEVEL1 curl -L
http://www.bluetack.co.uk/config/level1.gz | gunzip -c |
./pg2ipset -- LEVEL1 | ipset -R
#Example Converted 224283 rules.
. . .
-A LEVEL1 222.231.45.0-222.231.45.255
-A LEVEL1 222.233.164.0-222.233.179.255
. . .
```

Conntrack-tools: conntrack-tools.netfilter.org RICE

The conntrack-tools are a set of free software userspace tools for Linux that allow system administrators interact with the Connection Tracking System, which is the module that provides stateful packet inspection for iptables. The conntrack-tools are the userspace daemon conntrackd and the command line interface conntrack. it's tracking tool.

How to install Conntrack-tools?

prepare all file

conntrack-tools-0.9.7-1.el5.hrb.i386.rpm
conntrack-tools-0.9.7.tar.gz

Example

install: rpm -Uvh conntrack-tools-0.9.7-1.el5.hrb.i386.rpm
which conntrackd
cat /proc/net/ip_contrack

How to install Conntrack-tools?

prepare all file

conntrack-tools-0.9.7-1.el5.hrb.i386.rpm
conntrack-tools-0.9.7.tar.gz

Example

```
install: rpm -Uvh conntrack-tools-0.9.7-1.el5.hrb.i386.rpm
which conntrackd
cat /proc/net/ip_contrack
```

configure

configure file

```
tar zxvf conntrack-tools-0.9.7.tar.gz mkdir /etc/conntrackd cp
conntrack-tools-0.9.7/doc/stats/conntrackd.conf /etc/conntrackd
```

tail -f /var/log/conntrackd-stats.log

```
Fri Mar 5 13:16:08 2010 icmp 1 src=192.168.2.253
dst=192.168.2.200
type=8 code=0 id=34353 packets=1 bytes=84
src=192.168.2.200 dst=192.168.2.253
type=0 code=0 id=34353 packets=1 bytes=84
Fri Mar 5 13:23:21 2010 tcp 6 src=192.168.2.200
dst=134.160.38.1
sport=57858 dport=80 packets=4 bytes=180 src=134.160.38.1
dst=192.168.2.200
sport=80 dport=57858 packets=2 bytes=84 [ASSURED]
```

configure

RICE

configure file

```
tar zxvf conntrack-tools-0.9.7.tar.gz mkdir /etc/conntrackd cp
conntrack-tools-0.9.7/doc/stats/conntrackd.conf /etc/conntrackd
```

tail -f /var/log/conntrackd-stats.log

```
Fri Mar 5 13:16:08 2010 icmp 1 src=192.168.2.253
dst=192.168.2.200
type=8 code=0 id=34353 packets=1 bytes=84
src=192.168.2.200 dst=192.168.2.253
type=0 code=0 id=34353 packets=1 bytes=84
Fri Mar 5 13:23:21 2010 tcp 6 src=192.168.2.200
dst=134.160.38.1
sport=57858 dport=80 packets=4 bytes=180 src=134.160.38.1
dst=192.168.2.200
sport=80 dport=57858 packets=2 bytes=84 [ASSURED]
```

What is ulogd?

The Userspace Logging Daemon

RICE

(ulogd) is a flexible framework for extensive logging of packets on a firewall machine. ulogd uses the **ULOG target** of iptables/netfilter, the packet filtering framework of Linux 2.4 and 2.6. It supports binary plugins for adding packet interpreters and output-targets

How to install Conntrack-tools?

install from git

```
git clone git://git.netfilter.org/ulogd2.git ulogd2
cd ulogd2
./configure --with-mysql
make
make install
cp ulogd.conf /usr/local/etc/
vi /usr/local/etc/ulogd.conf
/usr/local/sbin/ulogd &
```

create rule in iptable

RICE

usr ULOG target

```
iptables -I OUTPUT -d 99.99.99.99 -j ULOG --ulog-nlgroup 1 --ulog-cprange 100
ping -c 5 99.99.99.99
vi /var/log/ulogd_syslogemu.log
```

Example

```
Mar 5 13:03:39 tesla IN= OUT=eth0 MAC= SRC=192.168.2.200
DST=99.99.99.99
LEN=84 TOS=00 PREC=0x00 TTL=64 ID=0 DF PROTO=ICMP
TYPE=8 CODE=0 ID=23678 SEQ=1 MARK=0
Mar 5 13:03:40 tesla IN= OUT=eth0 MAC= SRC=192.168.2.200
DST=99.99.99.99
LEN=84 TOS=00 PREC=0x00 TTL=64 ID=0 DF PROTO=ICMP
TYPE=8 CODE=0 ID=23678 SEQ=2 MARK=0
```

create rule in iptable

usr ULOG target

```
iptables -I OUTPUT -d 99.99.99.99 -j ULOG --ulog-nlgroup 1
--ulog-cprange 100
ping -c 5 99.99.99.99
vi /var/log/ulogd_syslogemu.log
```

Example

```
Mar 5 13:03:39 tesla IN= OUT=eth0 MAC= SRC=192.168.2.200
DST=99.99.99.99
LEN=84 TOS=00 PREC=0x00 TTL=64 ID=0 DF PROTO=ICMP
TYPE=8 CODE=0 ID=23678 SEQ=1 MARK=0
Mar 5 13:03:40 tesla IN= OUT=eth0 MAC= SRC=192.168.2.200
DST=99.99.99.99
LEN=84 TOS=00 PREC=0x00 TTL=64 ID=0 DF PROTO=ICMP
TYPE=8 CODE=0 ID=23678 SEQ=2 MARK=0
```

Netfilter is framework

RICE

- Netfilter API register|unregister hooking function
- fields, Struc in Register Process
- Register Process

Netfilter is framework

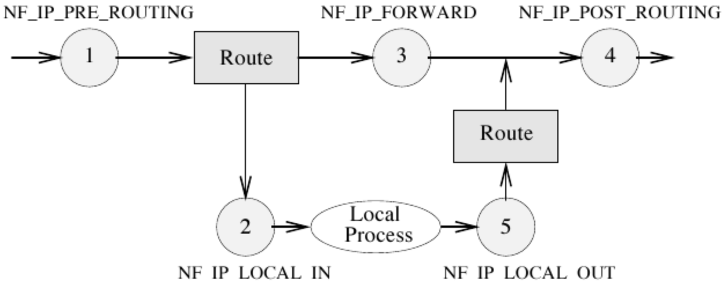
RICE

- Netfilter API register|unregister hooking function
- fields, Struc in Register Process
- Register Process

Netfilter is framework

RICE

- Netfilter API register|unregister hooking function
- fields, Struc in Register Process
- Register Process



1=NF_IP_PRE_ROUTING
2=NF_IP_LOCAL_IN
3=NF_IP_FORWARD
4=NF_IP_POST_ROUTING
5=NF_IP_LOCAL_OUT

- Any Kernel module can register function at any of hooks
- Module has to return the follow constants
  NF_ACCEPT continue traversal as normal
  NF_DROP drop the packet, do not continue
  NF_STOLEN I've taken over the packet , do not continue
  NF_QUEUE sent packet to userspace
  NF_REPEAT call this hook again

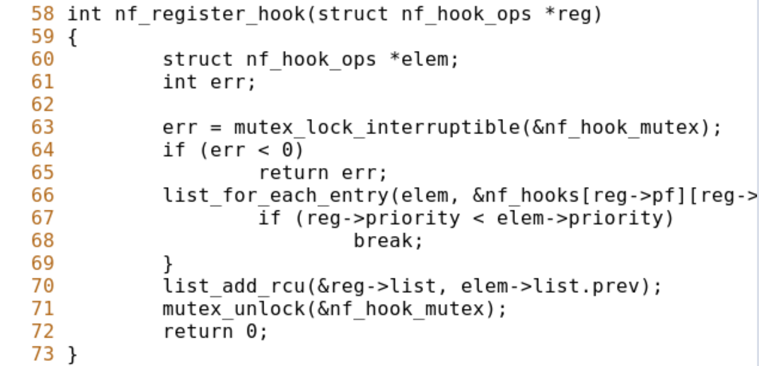
- Any Kernel module can register function at any of hooks
- Module has to return the follow constants
  NF_ACCEPT continue traversal as normal
  NF_DROP drop the packet, do not continue
  NF_STOLEN I've taken over the packet , do not continue
  NF_QUEUE sent packet to userspace
  NF_REPEAT call this hook again

Netfilter API

#net/netfilter/core.c

Register function

nf_register_hook()

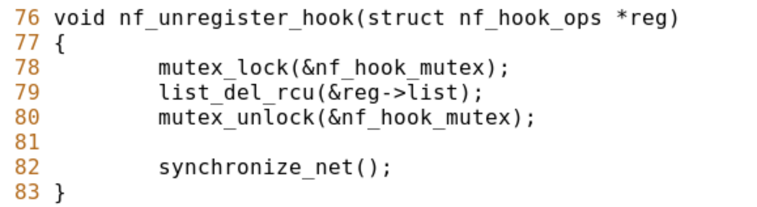


```
int nf_register_hook(struct nf_hook_ops *reg)
{
        struct nf_hook_ops *elem;
        int err;

        err = mutex_lock_interruptible(&nf_hook_mutex);
        if (err < 0)
                return err;
        list_for_each_entry(elem, &nf_hooks[reg->pf][reg->
                if (reg->priority < elem->priority)
                        break;
        }
        list_add_rcu(&reg->list, elem->list.prev);
        mutex_unlock(&nf_hook_mutex);
        return 0;
}
```

Netfilter API cont'd

Unregister function

nf_unregister_hook()



```
void nf_unregister_hook(struct nf_hook_ops *reg)
{
        mutex_lock(&nf_hook_mutex);
        list_del_rcu(&reg->list);
        mutex_unlock(&nf_hook_mutex);

        synchronize_net();
}
```

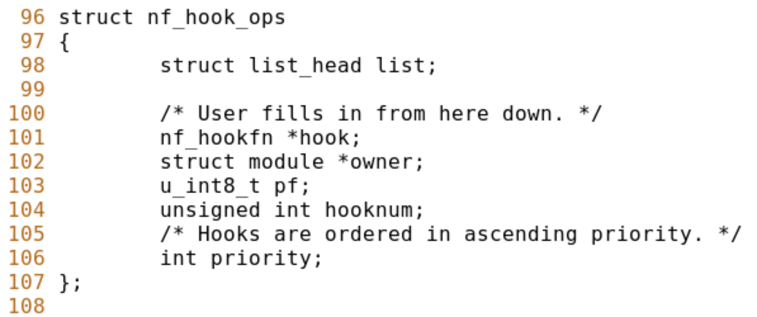
to register|unregister we call nf_register_hook with structure **nf_hook_ops**

Netfilter API cont'd

include/linux/netfilter.h

Callback function Prototype

nf_hook_ops() in netfilter.h



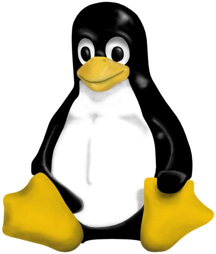
```
struct nf_hook_ops
{
        struct list_head list;

        /* User fills in from here down. */
        nf_hookfn *hook;
        struct module *owner;
        u_int8_t pf;
        unsigned int hooknum;
        /* Hooks are ordered in ascending priority. */
        int priority;
};

```

hook* is a pointer to callback function of kernel module. Netfilter will callback fucntion for registered kernel modules.

pf is a protocol family for which module is interested in. examples are PF_INET, NF_ARP

Rusty's Dream

Netfilter Framework will provide the appropriate mechanisms to allow people to implement their own protocols helper in userspace.



IN LINUX, WE BELIEVE
Thank you