پیاده سازی دیواره ی آتش در لینوکس از طریق نگهداری Iptables

دیواره آتش iptables توسط پروژه ی netfilter توسعه یافته و از زمان linux با هسته ی 4.2 در ژانویه 4002 به عنوان قسمتی از linux در اختیار عموم قرار گرفته، طی سالها ویژگی iptables بهبود یافته و آن را به یک فایروال قدرتمند یا بیشتر قابلیت هایی که عموماً در دیواره های آتش تجاری پیدا می شود تبدیل کرد. برای مثال iptables قابلیت های جامع ردیابی وضعیت پروتکل، بررسی کاربرد بسته ها توسط لایه، کاهش نرخ، و یک مکانیسم قدرتمند جهت تأمین نمودن یک سیاست فیلتر کردن را ارائه می دهد. تمامی نسخه های اصلی linux شامل iptables هستند و خیلی از این نسخه ها نیز از همان ابتدای نصب، کاربر را وادار به استفاده از یک سیاست iptables می کنند. تفاوت اصطلاح iptables و netfilter در جامعه linux منجر به سردرگمی های فراونی شده که اکنون به توضیح این دو اصطلاح می پردازیم. نام رسمی که توسط linux برای تمامی پروژه های فیلتر کردن بسته ها و ابزارهای ایجاد تغییر در بسته ها فراهم شده netfilter است. گرچه این اصطلاح برای یک framework درون هسته ی linux نیز به کاربرده می شود. که از این نوع framework می توان جهت قرار دادن توابع درون پشته های شبکه در مراحل مختلف استفاده کرد. از طرفی دیگر iptables، از netfilter framework به منظور قرار دادن توابع طراحی شده برای اجرای عملیات (مانند فیلتر کردن) بر روی پشته های شبکه استفاده می کند. شما می توانید به netfilter به عنوان ابزاری جهت فراهم نمودن framework هایی که iptables با استفاده از آنها نقش دیوار آتش را ایفا می کند نگاه کنید. اصطلاح iptables همچنین به ابزار بخش کار گفته می شود که خط فرمان را می شکند. یک

سیاست آتش را به هسته القا می کند. اصطلاحاتی مانند جداول، زنجیرها، همتاها و هدف ها در متن iptables معنا پیدا می کنند (Gregor N.Purdy,2004) .

Netfilterخود، ترافیک را فیلتر نمی کند صرفاً به توابعی که قادر به فیلتر کردن ترافیک هستند اجازه می دهد تا در محل مناسب در هسته قرار بگیرند. پروژه ی netfilter همچنین چندین قطعه از شالوده ی هسته (مانند ردیابی ارتباط، logging یا واقعه نگاری) را نیز تأمین می کند. هر سیاست iptables می تواند از این تسهیلات جهت اجرای هرنوع فرایند ویژه ی پردازش استفاه کند (Gregor N.Purdy,2004) .

فیلتر کردن بسته ها به وسیله ی iptables

دیوار آتش iptables به کاربر و یا وسیله اجازه می دهد کنترل زیادی بر روی بسته های ip با یک سیستم linux در ارتباط هستند داشته باشد. که این کنترل درون هسته ی linux اعمال می شود یک سیستم می تواند توسط iptables ساخته شود و به عنوان یک ناظر ترافیک فعال عمل کند نحوه ی کار این ناظر به این صورت خواهد بود که بسته هایی که اجازه ی عبور ندارند از بین می روند و بسته هایی که عبور می کنند جمع می شوند و به مسیر مورد نظرشان می روند یا مطابق نیازمندی های شبکه ی محلی تغییر می کنند. یک سیاست iptables بر اساس دسته ای از قوانین مرتب که عملیات مورد نیاز جهت برخورد با گروه های مختلف بسته ها را توصیف می کند ساخته می شود. هر قانون iptables به یک زنجیر درون یک جدول مربوط می شود. یک هماهنگ و یا مشابه با موارد مربوط به بسته هایی به سمت سیستم linux و یا دور کردن از هسته از جدول های netfilter می گذرند.

زنجیر iptables مجموعه ای قوانین است که با مقایر یا است که ویژگی های مشترکی دارند (مانند مسیر یابی شدن

سیستم). شکل زیر نشان می دهد که چگونه بسته ها درون (Rash. Michael, 2007)

نصب iptables

به دلیل اینکه iptables به دو بخش اساسی تقسیم می شود (ماژول های هسته و برنامه مدیریت بخش کاربر) نصب iptables شامل کامپایل کردن و نصب کردن هسته ی لینوکس و قسمت باینری مربوط به کاربر می شود کد منبع هسته شامل تعداد زیادی زیر سیستم و توانایی های ضروری فیلتر کردن بسته ها دارا می باشد که به صورت پیش فرض فعال شده اند. در برخی از هسته های 4.2 (و همه ی هسته های 2.4) گزینه های کامپایل netfilter به صورت پیش فرض فعال نبودند اما طی سال ها به دلیل رسیدن نرم افزارهایی که توسط پروژه ی netfilter تولید می شوند به سطح کیفیت بالا، نگهدارندگان هسته احساس کردند که نرم افزار به حدی رسیده است که استفاده از iptables در linux که بتواند به عنوان یک فایروال iptables کار کند پیکربندی و کامپایل مناسب هسته ی linux است تمامی پردازش های سنگین شبکه و کارهای مقایسه در iptables درون هسته انجام می شود (Gregor N.Purdy,2004)

پیکربندی هسته

قبل از شروع به کامپایل باید یک فایل پیکربندی هسته ایجاد کنیم. خوشبختانه فرآیند ساخت این فا یل به صورت خودکار توسط توسعه دهندگان هسته ایجاد شده و برای راه اندازی آن تنها نیاز به یک دستور داریم (درون دایرکتوری usr/scr/linux-2.6.20.1) دستور make menuconfig واسط ncurses را راه اندازی می کند که در آن می توان گزینه های مختلف کامپایل را انتخاب کرد (شما می توانید به ترتیب با دستورات make config, make xconfig واسطه های xwindows و termimal را فراخوانی کنید. در اینجا ما واسط ncurses را

انتخاب کریدم چون تعادل خوبی بین واسط نسبتاً گران xwindows ایجاد می کند. واسط ncurses همچنین به آسانی با پیکربندی یک هسته linux ریموت در میان یک جلسه ssh بدون نیاز به ارسال یک ارتباط xwindows هماهنگ می شود (Rash. Michael, 2007).

پس از اجرای make menuconfig چندین بخش پیکربندی از گزینه های میزان پیشرفتگی کد تا روال های کتابخانه ارائه می شوند. بیشتر گزینه های کامپایل netfilter برای هسته سری 2.6 درون بخشی به نام network paket filtering framework در زیر networking option> networking قرار دارد. برخی از مهم ترین گزینه هایی که باید در قسمت فایل پیکربندی هسته فعال شوند عبارتند از: ردیابی ارتباط netfilter، ثبت کردن و فیلتر کردن بسته ها. دو بخش پیکربندی اضافی در بخش network paket filtering framework (netfilter) وجود دارند که عبارتند از پیکربندی مرکز netfilter و پیکربندی ip:netfilter (Rash. Michael, 2007).

پیکربندی مرکز netfilter

قسمت core netfilter configuration شامل چندین گزینه ی مهم است که همگی باید فعال شوند.

Comment match support

FTP support

Length match support

Limit match support

MAC address match support

MARK target support

Netfilter connection tracking support

Netfilter LOG over NFNETLINK interface

Netfilter netlink interface

Netfilter Xtables support

State match support

String match support

پیکربندی ip: netfilter

پس از اتمام پیکربندی بخش core netfilter configuration به بخش netfilter configuration می رویم که گزینه هایی که در این بخش باید فعال شوند به ترتیب زیر هستند:

ECN target support

Full NAT

IP address range match support

IP tables support (required for filtering/masq/NAT)

IPv4 connection tracking support (required for NAT)

LOG target support

MASQUERADE target support

Owner match support

Packet filtering

raw table support (required for NOTRACK/TRACE)

Packet mangling

Recent match support

REJECT target support

TOS match support

TOS target support

TTL match support

TTL target support

ULOG target support

اتمام پیکربندی هسته

پس از پیکربندی هسته 4.2.40.2 با پشتیبانی مورد نیاز توسط واسط menuconfig با انتخاب exit و تأیید پیام do configuration you wish to save your new kernel را ثبت کنید. پس از این مرحله

به قسمت پوسته ی فرمان برمی گردید که در آن قسمت می توانید با استفاده از دستورات زیر پیکربندی netfilter حاصل را امتحان کنید.

$ grep "_NF_" .config

$ grep NETFILTER .config

کامپایل و نصب هسته

اکنون پیکربندی هسته به پایان رسید به سراغ کامپایل کردن و نصب آن می رویم. جهت نصب و کامپایل هسته ی 4.2.40.2 درون قسمت boot دستورات زیر را اجرا کنید (Suehring. S and Zeigler. R,2005).

$ make

$ su -

Password:

mount /boot

cd /usr/src/linux-2.6.20.1

make install && make modules_install

نتیجه ی موفقیت آمیز دستورات بالا اعلام می کند که bootloader باید پیکربندی شود و در نهایت هسته ی 4.2.40.2 جدید را بوت کنیم/ در این قسمت ازbootloade grub و محل dev/had2 برای بخش ریشه استفاده می کنیم با استفاده از ویرایشگر دلخواه خطوط زیر را به قسمت boot/grup/grup.conf اضافه می کنیم (Suehring. S and Zeigler. R,2005).

title linux-2.6.20.1

root (hd0,0)

kernel /boot/vmlinuz-2.6.20.1 root=/dev/hda2

اکنون مجدداً سیستم را راه اندازی می کنیم.

shutdown -r now

نصب باینری قسمت iptables

پس از نصب و راه اندازی هسته ای که دارای قلاب های netfilter کامپایل شده است اکنون باید برنامه iptables قسمت کاربر را نصب کنیم برای اینکار ابتدا منابع iptables را دانلود کرده و سپس در قسمت دایرکتوری user/local/src قرار می دهیم (Suehring. S and Zeigler. R,2005)

$ cd /usr/local/src/

$ wget http://www.netfilter.org/projects/iptables/files/iptables-1.3.7.tar.bz2

$ md5sum 1.3.7.tar.bz2

3

dd965bdacbb86ce2a6498829fddda6b7 iptables-1.3.7.tar.bz2

$ tar xfj iptables-1.3.7.tar.bz2

$ cd iptables-1.3.7

جهت نصب و کامپایل مراحل باینریiptables به یاد داشته باشید که ما هسته را درون دایرکتوری usr/src/linux-2.6.20.1 کامپایل کردیم. کامپایل کردن iptables نیازمند به دسترسی به منبع کد هسته است زیرا در مقایسه با فایل های با سرفصل c در دایرکتورها کامپایل می شود مانند linux/netfilter-ipv4 در درخت منبع هسته. ما از دایرکتوری usr/src/linux-2.6.20.1 جهت تعریف متغیر KERNEL-DIR روی خط فرمان استفاده می کنیم و متغیرهایbindir وlibdir به ما اجازه می دهند تا مسیری که باینری iptables کتابخانه ها نصب می شوند را کنترل کنیم. Iptables به صورت زیر کامپایل و نصب می شود Suehring. S and (Zeigler. R,2005).

$ make KERNEL_DIR=/usr/src/linux-2.6.20.1 BINDIR=/sbin LIBDIR=/lib

$ su -

Password:

cd /usr/local/src/iptables-1.3.7

```
# make install KERNEL_DIR=/usr/src/linux-2.6.20.1 BINDIR=/sbin
LIBDIR=/lib
```

جهت اطمینان از نصب iptables و اینکه می تواند با هسته ی 4.2.40.2 در حال اجرا ارتباط برقرار کند دستورتی می دهیم تا نسخه iptables را ارائه دهد و دستوری می دهیم که مجموعه قوانین input, output,forward زنجیره ه را لیست کند (که در این مراحل قانون فعالی ندارند) Suehring. S and Zeigler. (R,2005).

```
# which iptables
/sbin/iptables
# iptables -V
iptables v1.3.7
# iptables -nL
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

سیاست پیش فرض iptables

نیازمندی های سیاست

بیایید موارد مورد نیاز جهت پیکربندی یک دیوار آتش موثر برای یک شبکه چندین ماشین client و دو سرور را تعریف کنیم. سرورها (یک webserver و یک dnsserver) باید از شبکه ی بیرونی قابل دسترسی باشند و سیستم هایی که درون شبکه قرار دارند باید اجازه داشته باشند انواع ترافیک های ذکر شد در زیر را از فایروال به سرورهای بیرونی انجام دهند (Rash. Michael, 2007).

نام دامنه ی سیستم (dns) پرس و جوها
پروتکل انتقال فایل (ftp) انتقال ها
پروتکل زمان شبکه (ntp) پرس و جوها
پوسته ی امن (ssh) جلسات
پروتکل ساده انتقال mail (smtp) جلسات
جلسه های وب روی http و https
پرس و جوها whois

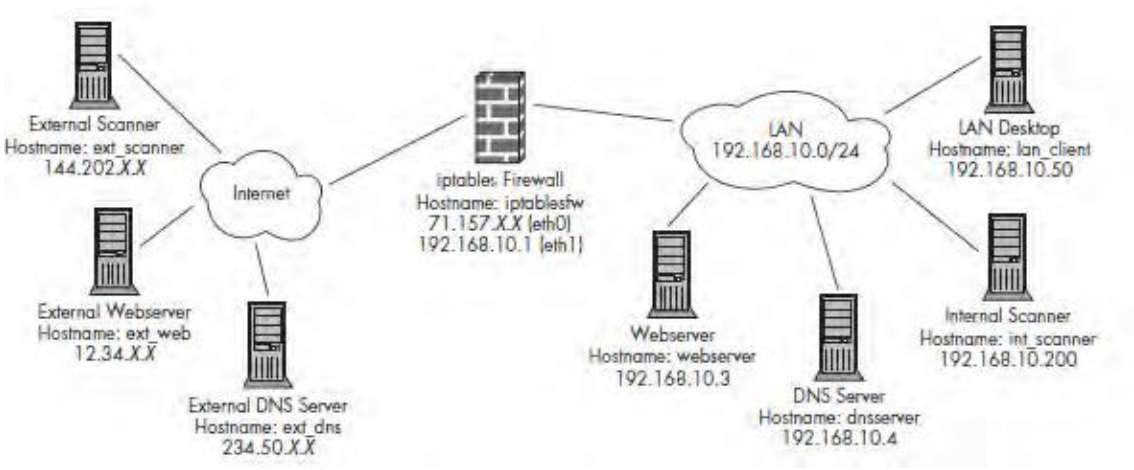
به جز جهت دستیابی به سرویس هایی که در بالا ذکر شد بقیه ترافیک ها باید بسته شوند. جلساتی که از شبکه درونی آغاز می شوند یا مستقیماً از طریق دیوار آتش، باید وضعیت به وضعیت توسط iptables ردیابی شوند (بسته هایی که با وضعیت معتبری همخوانی ندارند باید ثبت شده و دور انداخته شوند) و سرویس های nat نیز باید فراهم شوند. بعلاوه دیوار آتش باید کنترل های برای بسته های جمعی انجام دهد که از درون شبکه به ip آدرس های بیرونی فرستاده می شوند در واقع باید اینگونه باشد که:

دیوارآتش از شبکه ی درونی توسط ssh دسترسی باشد و از جاهای دیگر قابل دسترسی، مگر اینکه fwknop را جهت اعتبار سنجی انجام می دهد. ssh باید تنها پردازشگر سرور باشد که روی سرور دیوار آتش اجرا می شود.

دیوار آتش باید درخواست های icmp echo از شبکه های درونی و بیرونی بپذیرد اما بسته های icmp ناخواسته که درخواست های echo نیستند باید از هر منبع ip ادرس دور ریخته شوند.

در آخر دیوار آتش باید با یک حالت حذف، ثبت پیکربندی شود که هر بسته ی سرگردان چک کردن درگاه و یا هر تلاش جهت ارتباط که به صورت واضح اجاره داده نشده اند ثبت و حذف شود.

جهت ساده سازی کار ایجاد سیاست iptables تصور کنید یک تکه شبکه درونی با یک آدرس شبکه ی غیر قابل مسیریابی 192.168.10.0 و یک subnet mask ازکلاس c (255.255.255.0) وجود دارد. واسط شبکه ی درونی روی دیوار آتش eth1 است با ip آدرس 192.168.10.1 و همه ی میزبان های داخلی این آدرس را به عنوان دروازه پیش فرضشان دارند این امر به سیستم های درونی اجازه می دهد تا تمامی بسته هایی را که برای سیستم هایی که درون subnet 192.168.10.0/24 فرستاده شده اند را از طریق فایروال بیرون بفرستد. واسط بیرونی روی دیوار آتش به این واسط یک ip آدرس بیرونی eth10 است و برای اینکه فرض کنیم شبکه ای وجود ندارد میدهیم. Rash. 71.157.X.X (Michael, 2007).



دو سیستم خرابکار در اینجا در نظر گرفته می شود: یکی روی شبکه درونی ((192.198.10.200, hostname int- scanner و دیگری روی شبکه ی بیرونی (144.202.X.X, hostname ext- scanner) شکل بالا به عنوان مرجع مثال ها قرار خواهد گرفت و بعداً نیز از آن استفاده خواهیم کرد (Rash. Michael, 2007).

مقدمه ی Iptables .sh. Script

جهت شروع iptables . sh script مفید است که سه متغیر تعریف کنیم. Iptables و modprobe (برای مسیرهای به سوی iptables و

باینری های modprobe و int-net (برای آدرس subnet درونی و mask) از این سه متغیر در script استفاده خواهیم کرد. (قسمت 2) در قسمت (4) همه ی قانون های موجود از هسته ی در حال اجرا برداشته می شوند و سیاست فیلتر کردن روی زنجیرهای input, output, forward بر روی drop تنظیم می شود و همچنین ماژول های ردیابی مسیر دستور modprobe پر می شوند. (Gregor N.Purdy,2004).

```
[iptablesfw]# cat iptables.sh
#!/bin/sh
IPTABLES=/sbin/iptables2
MODPROBE=/sbin/modprobe
INT_NET=192.168.10.0/24
### flush existing rules and set chain policy setting to DROP
echo "[+] Flushing existing iptables rules..."
$IPTABLES –F4
$IPTABLES -F -t nat
$IPTABLES -X
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
### load connection-tracking modules
$MODPROBE ip_conntrack
$MODPROBE iptable_nat
$MODPROBE ip_conntrack_ftp
$MODPROBE ip_nat_ftp
```

ترجمه آدرس شبکه

آخرین گام جهت ایجاد سیاست iptables فعال سازی ترجمه کردن آدرس درونی و غیر قابل مسیریابی 192.168.10.0/24 به آدرس

71.157.X.X بیرونی و قابل مسیریابی است. این عمل برای ارتباطات درونی به وب سرورهای dns از client های بیرونی و نیز به ارتباطات خارجی که از سیستم های موجود در شبکه ی درونی شروع می شوند به کار گرفته می شود. برای ارتباطاتی که از سیستم های بیرونی آغاز می شوند از هدف nat مقصد (dnat) استفاده می کنیم. جدول iptables,nat به تمامی قوانین nat اختصاص داده می شود و درون این جدول دو زنجیر prerouting و postrouting وجود دارد. زنجیر prerouting جهت اعمال قوانین موجود در جدول nat بر روی بسته هایی استفاده می شود که هنوز وارد الگوریتم مسیریابی درون هسته نشده اند تا مشخص شود به کدام واسط باید فرستاده شوند. بسته هایی که در این زنجیر پردازش می شوند هنوز با زنجیرهای input یا forward در جدول filter مقایسه نشده اند. زنجیر postrouting مسئول پردازش بسته هایی است که الگوریتم مـ سیریابی درون هسته را گذارنده اند و در شرف فرستاده شدن به واسط فیزیکی محاسبه شده برایشان هستند. بسته هایی که به وسیله ی این زنجیر پردازش شده اند قسمت نیازمندی های زنجیره های output و forward در جدول filter گذارنده اند. (Gregor N.Purdy,2004).

NAT rules

echo "[+] Setting up NAT rules..."

9 $IPTABLES -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j DNAT --to 192.168.10.3:80

$IPTABLES -t nat -A PREROUTING -p tcp --dport 443 -i eth0 -j DNAT --to 192.168.10.3:443

$IPTABLES -t nat -A PREROUTING -p tcp --dport 53 -i eth0 -j DNAT --to 192.168.10.4:53

10 $IPTABLES -t nat -A POSTROUTING -s $INT_NET -o eth0 -j MASQUERADE

با توجه به شکل در تصویرip آدرس های web و سرورهای dns در شبکه های درونی, 192.198.10.4 , 192.168.10.3 هستند. دستورات iptables مورد نیاز جهت ایجاد وظایف nat در بالا نشان داده شده اند. سه قانون prerouting در قسمت 9 به سرویس های وب و درخواست های dns از شبکه ی بیرونی اجازه می دهند تا به سرورهای درونی مناسب فرستاده شود. قانون postrouting آخر در قسمت 20 به ارتباطاتی که از شبکه ی غیر قابل مسیر یابی درونی سرچشمه می گیرند و باید به اینترنت بیرونی بروند اجازه می دهد که به نظر برسند از ip آدرس 71.157.X.X می آیند و آخرین قدم در ساخت یک سیاست iptables این است که فرستادن ip را در هسته ی linux امکان پذیر کند.

forwarding

echo "[+] Enabling IP forwarding..."

echo 1 > /proc/sys/net/ipv4/ip_forward

فعالیت سازی سیاست

اکنون که یک پوسته ی متن داریم که دستورات iptables را گرفته آن را اجرا می کنیم.

[iptablesfw]# ./iptables.sh

[+] Flushing existing iptables rules...

[+] Setting up INPUT chain...

[+] Setting up OUTPUT chain...

[+] Setting up FORWARD chain...

[+] Setting up NAT rules...

[+] Enabling IP forwarding...

ذخیره و باز گرداندن iptables

همه دستورات iptables قبلی در متن iptables.sh قبلی در متن iptables.sh یکی یکی و هرکدام در یک زمان اجرا می شوند تا قوانین جدید را معرفی کنند یا سیاست پیش فرض را روی یک زنجیر تنظیم کنند و با قوانین قدیمی حذف کنند هر دستور نیازمند یک اجرای جداگانه ی باینری قسمت کاربر iptables است تا سیاست iptables را ایجاد کند. بنابراین این راه حل بهینه ای جهت به اجرا در آوردن سریع سیاست هنگام روشن کردن یا boot کردن سیستم نیست مخصوصاً وقتی تعداد قوانین iptables به هزاران قانون می رسد. یک مکانیسم سریعتر به وسیله دستورات iptables-save و iptables- restore ایجاد شده که درون همان دایرکتوری (/sbin) برنامه اصلی iptables وجود دارد نصب شده اند. دستور iptables-save فایلی می سازد که شامل تمام قوانین iptables است که در سیاست اعمال شده وجود دارند این فایل در قالب خواندن برای انسان است. این قالب با یک برنامه ی iptables- restore قابل تغییر است، این برنامه هر قانون لیست شده در فایل ipt.save را می گیرد و آن را درون یک هسته ی در حال اجرا معرفی می کند. تنها با اجرای برنامه ی iptables- restore تمامی سیاست iptables را درون هسته ایجاد می کند. اجرای چندین باره ی برنامه ی iptables ضروری نیست. این روش دستورات iptables-save و restore- iptables را برای اجرای سریع دستورات iptables ایده آل می کند این فرایند را با دو دستور زیر نمایش می دهیم (Gregor N.Purdy,2004).

محتویات فایل ipt,save توسط جدول iptables سازماندهی شده اند و درون هر بخش مختص یک جدول یکتاست. فایل ipt.save بیشتر از این با زنجیر iptables سازماندهی می شود. خطی که با کاراکتر (*) شروع می شود و به همراه آن یک نام جدول (

مانند فیلتر) وجود دارد ابتدای یک بخش را در فایلipt.savei مشخص می کند که یک جدول خاص را توصیف می کند. به دنبال این خطوطی هستند که بسته ها و شمارش های بایت ها را برای هر زنجیر مربوط با جدول ردیابی می کنند. قسمت بعدی فایل ipt.save یک توصیف شامل تمامی قوانین iptables هست. که توسط زنجیر سازمان شده. این خطوط اجازه می دهند تا مجموعه قوانین حقیقی و حتی بسته ها شمارش های بایت برای هر قانون iptables توسط iptables –restore ساخته شوند. تنها اگر گزینه c در iptables-save استفاده شده باشد. در آخر کلمه ی commit در یک خط به تنهایی قسمت فایل ipt.save را که جدول iptables را توصیف می کند خاتمه می دهد. این خط علامت پایانی را برای تمامی اطلاعات مربوط به جدول را می سازد. در زیر مثال کاملی از اینکه قسمت جدول filterهنگامی که تمامی دستورات iptables را تا به اینجا اجرا کردیم چگونه به نظر خواهد رسید را خواهد دید (Gregor N.Purdy,2004).

```
# Generated by iptables-save v1.3.7 on Sat Apr 14 17:35:22 2007*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [2:112]
-A INPUT -m state --state INVALID -j LOG --log-prefix "DROP INVALID "
--log-tcp-options --log-ip-options
-A INPUT -m state --state INVALID -j DROP
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -s ! 192.168.10.0/255.255.255.0 -i eth1 -j LOG --log-prefix
"SPOOFED PKT "
-A INPUT -s ! 192.168.10.0/255.255.255.0 -i eth1 -j DROP
-A INPUT -s 192.168.10.0/255.255.255.0 -i eth1 -p tcp -m tcp --dport 22
--tcp-flags FIN,SYN,RST,ACK SYN -m state --state NEW -j ACCEPT
```

-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
7
-A INPUT -i ! lo -j LOG --log-prefix "DROP " --log-tcp-options
--log-ip-options
-A FORWARD -m state --state INVALID -j LOG --log-prefix "DROP
INVALID "
--log-tcp-options --log-ip-options
-A FORWARD -m state --state INVALID -j DROP
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s ! 192.168.10.0/255.255.255.0 -i eth1 -j LOG
--log-prefix "SPOOFED PKT "
-A FORWARD -s ! 192.168.10.0/255.255.255.0 -i eth1 -j DROP
-A FORWARD -s 192.168.10.0/255.255.255.0 -i eth1 -p tcp -m tcp --dport 21
--tcp-flags FIN,SYN,RST,ACK SYN -m state --state NEW -j ACCEPT
-A FORWARD -s 192.168.10.0/255.255.255.0 -i eth1 -p tcp -m tcp --dport 22
--tcp-flags FIN,SYN,RST,ACK SYN -m state --state NEW -j ACCEPT
-A FORWARD -s 192.168.10.0/255.255.255.0 -i eth1 -p tcp -m tcp --dport 25
--tcp-flags FIN,SYN,RST,ACK SYN -m state --state NEW -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 80 --tcp-flags FIN,SYN,RST,ACK SYN -
m state
--state NEW -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 443 --tcp-flags FIN,SYN,RST,ACK SYN -
m state
--state NEW -j ACCEPT
-A FORWARD -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A FORWARD -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A FORWARD -i ! lo -j LOG --log-prefix "DROP " --log-tcp-options
--log-ip-options
-A OUTPUT -m state --state INVALID -j LOG --log-prefix "DROP INVALID "

--log-tcp-options --log-ip-options
-A OUTPUT -m state --state INVALID -j DROP
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 21 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 22 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 25 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 43 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 80 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 443 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 4321 --tcp-flags FIN,SYN,RST,ACK SYN -m state
--state NEW -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A OUTPUT -o ! lo -j LOG --log-prefix "DROP " --log-tcp-options
--log-ip-options
COMMIT

Completed on Sat Apr 14 17:35:22 2007