Playing with network layers to bypass firewalls' filtering policy

Éric Leblond

OISF

Cansecwest 2012

Eric Leblond a.k.a Regit

- French
- Network security expert
- Free Software enthousiast
- NuFW project creator (Now ufwi), EdenWall co-founder
- Netfilter developer:
  - Ulogd2: Netfilter logging daemon
  - Misc contributions:
    - NFQUEUE library and associates
    - Source NAT randomisation (defeat Kaminsky's DNS attack)
- Currently:
  - Independant security consultant
  - Suricata IDS/IPS funded developer

Netilter

Definition

Packet filtering framework inside the Linux 2.4.x to 3.x kernel series.

Netilter

Definition

Packet filtering framework inside the Linux 2.4.x to 3.x kernel series.

Features

- Stateful and stateless packet filtering (IPv4 and IPv6).
- Network address and port translation.
- Multiple layers of API's for 3rd party extensions.

Netilter

Definition

Packet filtering framework inside the Linux 2.4.x to 3.x kernel series.

Features

- Stateful and stateless packet filtering (IPv4 and IPv6).
- Network address and port translation.
- Multiple layers of API's for 3rd party extensions.

Iptables

- Command line utility to do operation on rules.
- It has access to all Netfilter features.
- Two utilities: iptables for IPv4, ip6tables for IPv6.

```
iptables -A FORWARD -p tcp --syn --dport 80 \
        -m connlimit --connlimit-above 2 -j REJECT
```

Netfilter's stateful inspection

- Netfilter keeps a list of all active connections.
- Packet connection is looked up in connections list (the *"conntrack"*).
- Packet is tagged with one of the following state:
  - NEW
  - ESTABLISHED
  - INVALID
- It can be used to decide on the fate of the packet:

```
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
iptables -A FORWARD -m state --state NEW -p tcp --dport 80 -j ACCEPT
```

Application Level Gateway

Non-linear protocol

One can find protocols such as FTP or SIP:

- They rely on a signalling channel.
- It is used to setup dynamic connections.

Application Level Gateway

Non-linear protocol

One can find protocols such as FTP or SIP:

- They rely on a signalling channel.
- It is used to setup dynamic connections.

Application Level Gateway (ALG)

- ALGs search the traffic for command messages.
- They extract information on the expected connections.
- Each expectation:
  - includes information on a potential connection.
  - is associated to a timeout.
- New connection matching an expectation can be accepted.

The example of FTP

FTP client

```
Logged in to ftp.lip6.fr.
ncftp / > ls
etc/          jussieu/    lip6/
```

Tcpdump

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

The example of FTP

FTP client

```
Logged in to ftp.lip6.fr.
ncftp / > ls
etc/          jussieu/    lip6/
```

Tcpdump

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

Protocol

```
C: PASV
S: 227 Entering Passive Mode (195,83,118,1,199,211)
C: MLSD
S: 150 Opening ASCII mode data connection for 'MLSD'.
S: 226 MLSD complete.
C: QUIT
```

The example of FTP

FTP client

```
Logged in to ftp.lip6.fr.
ncftp / > ls
etc/          jussieu/    lip6/
```

Tcpdump

```
195.83.118.1.21 > 10.62.101.203.52994
195.83.118.1.21 > 10.62.101.203.52994
10.62.101.203.57636 > 195.83.118.1.51155
10.62.101.203.52994 > 195.83.118.1.21
195.83.118.1.51155 > 10.62.101.203.57636
```

Protocol

```
C: PASV
S: 227 Entering Passive Mode (195,83,118,1,199,211)
C: MLSD
S: 150 Opening ASCII mode data connection for 'MLSD'.
S: 226 MLSD complete.
C: QUIT
```

Netfilter

```
# conntrack -E expect
    [NEW] 300 proto=6 src=10.62.101.203 dst=195.83.118.1 sport=0 dport=51155
[DESTROY] 300 proto=6 src=10.62.101.203 dst=195.83.118.1 sport=0 dport=51155
```

Details of Netfilter implementation

ALGs in Netfilter

- ALGs are called *Helpers*.
- Each protocol is implemented as a kernel module.
- Loading options can be used to configure the helper.
- Fine-grained setup can be achieved with the CT iptables target.

Details of Netfilter implementation

ALGs in Netfilter

- ALGs are called *Helpers*.
- Each protocol is implemented as a kernel module.
- Loading options can be used to configure the helper.
- Fine-grained setup can be achieved with the CT iptables target.

Current modules list in Vanilla linux kernel

amanda	pptp	broadcast	proto_dccp
ftp	proto_gre	h323	proto_sctp
ipv4	proto_udplite	ipv6	sane
irc	sip	netbios_ns	snmp
tftp			

Expectations in Netfilter

The expectation table

- Expectations are stored in a specific table.
    - It is similar to the conntrack table.
    - Only one tuple is used.
    - A short timeout is added.
- An entry is destroyed when it matches with a packet.
- As a response, a new connection entry is created.
- It is *RELATED* to the signalling connection.

Expectations in Netfilter

The expectation table

- Expectations are stored in a specific table.
  - It is similar to the conntrack table.
  - Only one tuple is used.
  - A short timeout is added.
- An entry is destroyed when it matches with a packet.
- As a response, a new connection entry is created.
- It is *RELATED* to the signalling connection.

Accepting RELATED connections

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Do I use helpers?

- What happens if I load a helper?

Do I use helpers?

- What happens if I load a helper?
- Can a user send crafted messages and go freely through the firewall?

Do I use helpers?

- What happens if I load a helper?
- Can a user send crafted messages and go freely through the firewall?
- Do helpers transform my firewall in openbar?

Do I use helpers?

- What happens if I load a helper?
- Can a user send crafted messages and go freely through the firewall?
- Do helpers transform my firewall in openbar?

Do I use helpers?

- What happens if I load a helper?
- Can a user send crafted messages and go freely through the firewall?
- Do helpers transform my firewall in openbar?

- A study is needed.
- Let's look at the helpers.

Degree of freedom of Netfilter helpers

Module	Source	Destination	Port Dest	Option
ftp	Fixed	In CMD	In CMD	loose = 1 (dflt)
ftp	**Full**	In CMD	In CMD	loose = 0
h323	Fixed	Fixed	In CMD	
irc	**Full**	Fixed	In CMD	
sip signalling	Fixed	Fixed	In CMD	sip_direct_signalling = 1 (dflt)
sip signalling	**Full**	In CMD	In CMD	sip_direct_signalling = 0

- Legend:
  - Fixed: Value comes from the signalling connection. It can't be forged.
  - In CMD: The value comes from protocol message parsing and can be forged.
  - Full: Freedom is total. All values are accepted.
- Options are specific to Netfilter.
- However the degree of freedom will be similar for any firewall using ALGs.

Global analysis

Sane defaults

- Dangerous extensions of protocols have been disabled.
- If we study the attack of client on a server:
  - It is impossible to open arbitrary connections to the server.
  - The level of security is acceptable.

Global analysis

Sane defaults

- Dangerous extensions of protocols have been disabled.
- If we study the attack of client on a server:
    - It is impossible to open arbitrary connections to the server.
    - The level of security is acceptable.

In the limit of protocols

- Security is ensured with regard to the protocol usability.
- IRC helper is really user-friendly.

FTP analysis

If we follow RFC (*loose* $= 0$).

- A FTP server can participate to the initialization of a connection from client to another server.
- It can open arbitrary connections through the firewall.

FTP analysis

If we follow RFC (*loose* $= 0$).

- A FTP server can participate to the initialization of a connection from client to another server.
- It can open arbitrary connections through the firewall.

If we care about security (*loose* $= 1$).

- Expectation are statically bound to the server address.
- The possible openings are acceptable.
- This is the default value.

IRC analysis

The DCC command

DCC command enables transfer between end-point.

- It is impossible to know the source address.
- Destination port is fixed by the client.

IRC analysis

The DCC command

DCC command enables transfer between end-point.

- It is impossible to know the source address.
- Destination port is fixed by the client.

Consequences

- Allowing DCC is thus allowing client to enable arbitrary connection to his IP.
- Client computer is given a complete freedom of connection opening.

IRC analysis

The DCC command

DCC command enables transfer between end-point.

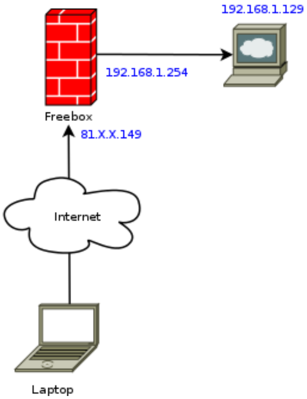
- It is impossible to know the source address.
- Destination port is fixed by the client.

Consequences

- Allowing DCC is thus allowing client to enable arbitrary connection to his IP.
- Client computer is given a complete freedom of connection opening.

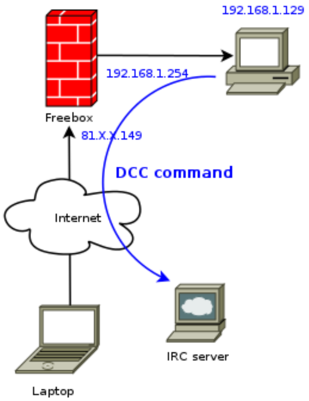
A mistake is simply another way of doing things.
(Katharine Graham)

Using DCC command



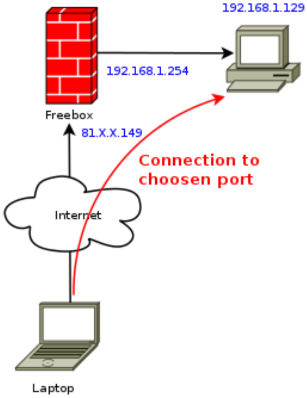
- Client NATed behind firewall, port *N* is closed

Using DCC command



- Client NATed behind firewall, port *N* is closed
- Client sends a DCC command to a valid IRC server

Using DCC command

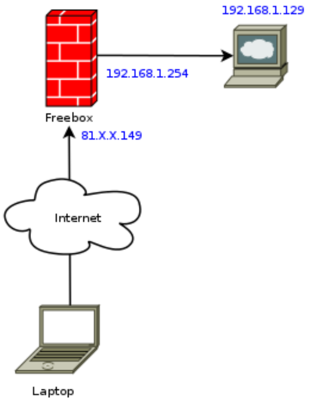


- Client NATed behind firewall, port *N* is closed
- Client sends a DCC command to a valid IRC server
- Firewall creates expectation and laptop can open a connection

“Exploit” code

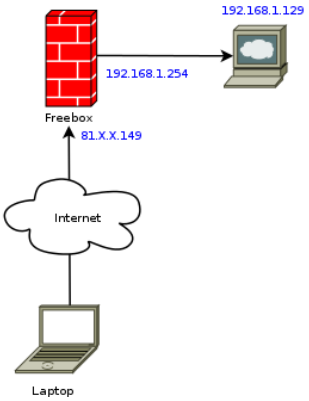
```
import socket

def ipnumber(ip):
    ip=ip.rstrip().split('.')
    ipn=0
    while ip:
        ipn=(ipn<<8)+int(ip.pop(0))
    return ipn

host="irc.freenode.net"
dport=6667 # IRC port
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, dport))

ip="192.168.1.129" # Local address of client
port=6000 # Port to open on Internet
atmsg = 'PRIVMSG opensvp :\x01DCC CHAT CHAT %d %d\x01\r\n' \\
            % (ipnumber(ip), port)

s.send(atmsg)
s.close()
```

Demonstration of DCC usage



Video

Demonstration of DCC usage



Video

Let’s connect from Internet to port 6000 of a NATed client.

Secure use of Netfilter helpers

Disable helper by default

Load with port 0 or use a dedicated *proc* entry (After 3.3 Linux kernel):

```
modprobe nf_conntrack_ftp ports=0
```

Secure use of Netfilter helpers

Disable helper by default

Load with port 0 or use a dedicated *proc* entry (After 3.3 Linux kernel):

```
modprobe nf_conntrack_ftp ports=0
```

Use the CT target

Activate the helper for chosen connections and do explicit authorization of RELATED traffic:

```
iptables -A PREROUTING -t raw -p tcp --dport 21 \\
  -d $MY_FTP_SERVER -j CT --helper ftp
iptables -A FORWARD -m conntrack --ctstate RELATED \\
        -m helper --helper ftp -d $MY_FTP_SERVER \\
        -p tcp --dport 1024: -j ACCEPT
```

Secure use of Netfilter helpers

Disable helper by default

Load with port 0 or use a dedicated *proc* entry (After 3.3 Linux kernel):

```
modprobe nf_conntrack_ftp ports=0
```

Use the CT target

Activate the helper for chosen connections and do explicit authorization of RELATED traffic:

```
iptables -A PREROUTING -t raw -p tcp --dport 21 \\
  -d $MY_FTP_SERVER -j CT --helper ftp
iptables -A FORWARD -m conntrack --ctstate RELATED \\
        -m helper --helper ftp -d $MY_FTP_SERVER \\
        -p tcp --dport 1024: -j ACCEPT
```

More information

See `http://home.regit.org/netfilter-en/secure-use-of-helpers/`

Known attacks

Cisco Bug ID CSCdr09226

- *goal*: Open pinhole in the firewall.
- Force the server to generate a message interpreted as a command by the firewall.
- An error condition can be used to trigger the abnormal behaviour.

`https://listserv.icsalabs.com/pipermail/firewall-wizards/2000-March/008385.html`

Known attacks

Cisco Bug ID CSCdr09226

- *goal*: Open pinhole in the firewall.
- Force the server to generate a message interpreted as a command by the firewall.
- An error condition can be used to trigger the abnormal behaviour.

`https://listserv.icsalabs.com/pipermail/firewall-wizards/2000-March/008385.html`

A Stateful Inspection of FireWall-1

- Panorama of attacks on Checkpoint FireWall-1
- Interesting techniques using FWZ encapsulation.
- T. Lopatic, J. McDonald, D. Song, Black Hat Briefings 2000

Objective

- Determine if it is possible *as client* to trigger unwanted behaviour:
  - Can we open arbitrary pinholes through a firewall?
  - Can we open more ports on a server?
  - Can we access to badly protected service ?
    - Such as an internal database
    - Such as vulnerable services

Objective

- Determine if it is possible *as client* to trigger unwanted behaviour:
  - Can we open arbitrary pinholes through a firewall?
  - Can we open more ports on a server?
  - Can we access to badly protected service ?
    - Such as an internal database
    - Such as vulnerable services
- Our study of helpers has shown that it is not possible out of the box:
  - Client capabilities are always limited.
  - Dangerous extensions have been blocked.

Objective

- Determine if it is possible *as client* to trigger unwanted behaviour:
  - Can we open arbitrary pinholes through a firewall?
  - Can we open more ports on a server?
  - Can we access to badly protected service ?
    - Such as an internal database
    - Such as vulnerable services
- Our study of helpers has shown that it is not possible out of the box:
  - Client capabilities are always limited.
  - Dangerous extensions have been blocked.
- An alternative approach should be found.

Basic idea

- Only the server can send useful messages.

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.
- We can’t force the server to do it.

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.
- We can’t force the server to do it.
- One can consider sending a message for the server.
  - A computer can forge any IP packet
  - and send it to the gateway
  - if the computer is on the same ethernet network as the gateway.

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.
- We can't force the server to do it.
- One can consider sending a message for the server.
  - A computer can forge any IP packet
  - and send it to the gateway
  - if the computer is on the same ethernet network as the gateway.
- An attacker on a **directly connected network** can send packets:

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.
- We can't force the server to do it.
- One can consider sending a message for the server.
  - A computer can forge any IP packet
  - and send it to the gateway
  - if the computer is on the same ethernet network as the gateway.
- An attacker on a **directly connected network** can send packets:
  - to the ethernet address of the firewall

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.
- We can't force the server to do it.
- One can consider sending a message for the server.
    - A computer can forge any IP packet
    - and send it to the gateway
    - if the computer is on the same ethernet network as the gateway.
- An attacker on a **directly connected network** can send packets:
    - to the ethernet address of the firewall
    - with the IP address of the server

Basic idea

- Only the server can send useful messages.
- We need to trigger the server to send a crafted message.
- We can't force the server to do it.
- One can consider sending a message for the server.
  - A computer can forge any IP packet
  - and send it to the gateway
  - if the computer is on the same ethernet network as the gateway.
- An attacker on a **directly connected network** can send packets:
  - to the ethernet address of the firewall
  - with the IP address of the server
- Let's try to use this method.

Attack description

1. The attacker opens a network connection using a given protocol.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
   - Invert source and destination ethernet address.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
    - Invert source and destination ethernet address.
    - Modify payload to a server command choosing parameters.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
    - Invert source and destination ethernet address.
    - Modify payload to a server command choosing parameters.
    - Increment IP id.
    - Set TCP sequence number correctly using traffic data.
    - Update all checksums and lengths.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
   - Invert source and destination ethernet address.
   - Modify payload to a server command choosing parameters.
   - Increment IP id.
   - Set TCP sequence number correctly using traffic data.
   - Update all checksums and lengths.
4. The attacker sends the forged packet to the network.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
    - Invert source and destination ethernet address.
    - Modify payload to a server command choosing parameters.
    - Increment IP id.
    - Set TCP sequence number correctly using traffic data.
    - Update all checksums and lengths.
4. The attacker sends the forged packet to the network.
5. The firewall sees the forged request.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
    - Invert source and destination ethernet address.
    - Modify payload to a server command choosing parameters.
    - Increment IP id.
    - Set TCP sequence number correctly using traffic data.
    - Update all checksums and lengths.
4. The attacker sends the forged packet to the network.
5. The firewall sees the forged request.
6. The firewall creates an expectation with the parameters **“given”** by the server.

Attack description

1. The attacker opens a network connection using a given protocol.
2. He sniffs the network traffic for that protocol.
3. He gets one packet coming from server.
    - Invert source and destination ethernet address.
    - Modify payload to a server command choosing parameters.
    - Increment IP id.
    - Set TCP sequence number correctly using traffic data.
    - Update all checksums and lengths.
4. The attacker sends the forged packet to the network.
5. The firewall sees the forged request.
6. The firewall creates an expectation with the parameters “given” by the server.
7. The attacker opens a connection with the chosen parameters.

FTP protocol study

Dynamic connection

In FTP, dynamic connections can be opened to the server:

FTP protocol study

Dynamic connection

In FTP, dynamic connections can be opened to the server:

- The server sends a message to the client to indicate him what port to use.
- Client then connects to the provided IP and port.

FTP protocol study

Dynamic connection

In FTP, dynamic connections can be opened to the server:

- The server sends a message to the client to indicate him what port to use.
- Client then connects to the provided IP and port.

IPv4 case

- To ask a client to connect to 192.168.2.2 on port 3306:

```
227 Entering Passive Mode (192,168,2,2,12,234)\r\n
```

- The message format is simple, the only trick to know is that $12 * 256 + 334 = 3306$.

FTP protocol study

Dynamic connection

In FTP, dynamic connections can be opened to the server:

- The server sends a message to the client to indicate him what port to use.
- Client then connects to the provided IP and port.

IPv4 case

- To ask a client to connect to 192.168.2.2 on port 3306:

```
227 Entering Passive Mode (192,168,2,2,12,234)\r\n
```

- The message format is simple, the only trick to know is that $12 * 256 + 334 = 3306$.

IPv6 case

- To ask a client to connect on port 3306:

```
229 Extended Passive Mode OK (|||3306|)\r\n
```

Attack description for FTP

1. The attacker sniffs traffic coming from a FTP server.

Attack description for FTP

1. The attacker sniffs traffic coming from a FTP server.
2. He opens a connection to the FTP server.

Attack description for FTP

1. The attacker sniffs traffic coming from a FTP server.
2. He opens a connection to the FTP server.
3. He forges a packet using the last packet received from server:
   - Invert source and destination ethernet address.
   - Increment IP ID IP. Set TCP sequence number correctly using traffic data.
   - Modify payload to a server command choosing parameters.

     ```
     227 Entering Passive Mode (192,168,2,2,12,234)
     ```
   - Update all checksums and lengths.

Attack description for FTP

1. The attacker sniffs traffic coming from a FTP server.
2. He opens a connection to the FTP server.
3. He forges a packet using the last packet received from server:
   - Invert source and destination ethernet address.
   - Increment IP ID IP. Set TCP sequence number correctly using traffic data.
   - Modify payload to a server command choosing parameters.

     ```
     227 Entering Passive Mode (192,168,2,2,12,234)
     ```
   - Update all checksums and lengths.
4. The attacker sends the forged packet to the firewall.

Attack description for FTP

1. The attacker sniffs traffic coming from a FTP server.
2. He opens a connection to the FTP server.
3. He forges a packet using the last packet received from server:
   - Invert source and destination ethernet address.
   - Increment IP ID IP. Set TCP sequence number correctly using traffic data.
   - Modify payload to a server command choosing parameters.

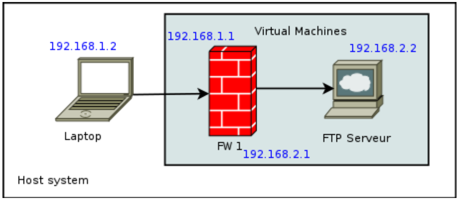
     ```
     227 Entering Passive Mode (192,168,2,2,12,234)
     ```
   - Update all checksums and lengths.
4. The attacker sends the forged packet to the firewall.
5. The firewall creates an expectation for a connection to 192.168.2.2 on port 3306.

Attack description for FTP

1. The attacker sniffs traffic coming from a FTP server.
2. He opens a connection to the FTP server.
3. He forges a packet using the last packet received from server:
    - Invert source and destination ethernet address.
    - Increment IP ID IP. Set TCP sequence number correctly using traffic data.
    - Modify payload to a server command choosing parameters.

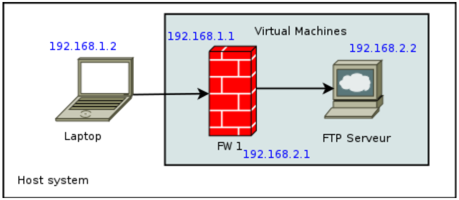
      ```
      227 Entering Passive Mode (192,168,2,2,12,234)
      ```
    - Update all checksums and lengths.
4. The attacker sends the forged packet to the firewall.
5. The firewall creates an expectation for a connection to 192.168.2.2 on port 3306.
6. The attacker connects to 192.168.2.2 on port 3306.

Demonstration on Netfilter



Video

Demonstration on Netfilter



Video

Let’s have firewall with a filtering policy allowing only port 21 and open a connection to port 22 on a FTP server.

Policy violation

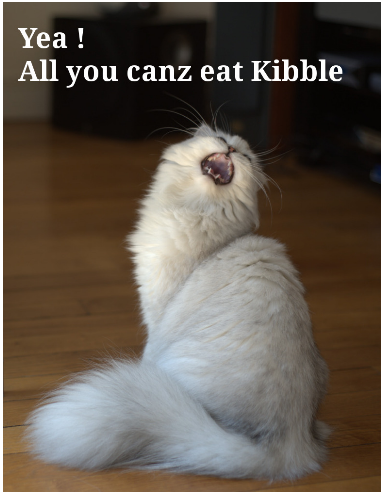
- We've manage to open a connection to port 22

Policy violation

- We've manage to open a connection to port 22
- With a filtering policy that does not allow it.

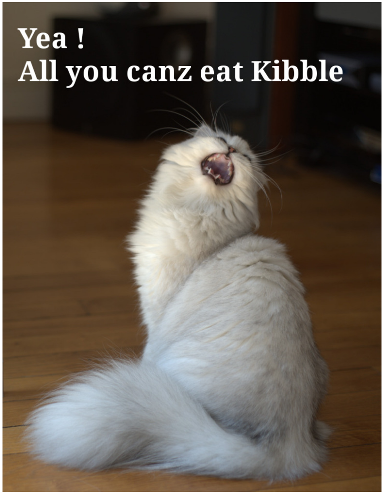
Policy violation

- We've manage to open a connection to port 22
- With a filtering policy that does not allow it.



Policy violation

- We've manage to open a connection to port 22
- With a filtering policy that does not allow it.
- Easy little cat, easy!



Counter-measures

- Anti-spoofing is sufficient to block the attack.
- Reverse path filtering is our friend:
  - Only accept packet coming to an interface if we have a route to the source IP.
  - This will avoid that the kernel handles the attack packet.
- Is this that easy to be protected?

Counter-measures

- Anti-spoofing is sufficient to block the attack.
- Reverse path filtering is our friend:
    - Only accept packet coming to an interface if we have a route to the source IP.
    - This will avoid that the kernel handles the attack packet.
- Is this that easy to be protected? **Yes**

Counter-measures

- Anti-spoofing is sufficient to block the attack.
- Reverse path filtering is our friend:
  - Only accept packet coming to an interface if we have a route to the source IP.
  - This will avoid that the kernel handles the attack packet.
- Is this that easy to be protected? Yes
- But wait, there is still some surprise.

Others protocols

IRC

- As discussed before IRC helper provide the client with great power.
- The issue is inverted: can we act against client?

Others protocols

IRC

- As discussed before IRC helper provide the client with great power.
- The issue is inverted: can we act against client?
- Same technique applies with the following conditions:
  - Attacker and client are separated by firewall.
  - Attacker is on a network directly connected to the firewall.
  - IRC traffic can be sniffed by attacker (MITM or server).

Others protocols

IRC

- As discussed before IRC helper provide the client with great power.
- The issue is inverted: can we act against client?
- Same technique applies with the following conditions:
  - Attacker and client are separated by firewall.
  - Attacker is on a network directly connected to the firewall.
  - IRC traffic can be sniffed by attacker (MITM or server).
- This is not interesting.

Others protocols

IRC

- As discussed before IRC helper provide the client with great power.
- The issue is inverted: can we act against client?
- Same technique applies with the following conditions:
  - Attacker and client are separated by firewall.
  - Attacker is on a network directly connected to the firewall.
  - IRC traffic can be sniffed by attacker (MITM or server).
- This is not interesting.

SIP

- The server sends port parameters in a similar way as FTP.
- The same attack is possible.
- Only the content has to be changed.

Protection for Netfilter

- We only have to use the **rp_filter** feature.
- It is available since last century in all Linux kernel.

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- **Disabled by default**.

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- Disabled by default. Enabled by all decent firewall scripts.

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- Disabled by default. Enabled by all decent firewall scripts.
- To activate it:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- Disabled by default. Enabled by all decent firewall scripts.
- To activate it:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Wait**

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- Disabled by default. Enabled by all decent firewall scripts.
- To activate it:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Wait** and for IPv6?

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- Disabled by default. Enabled by all decent firewall scripts.
- To activate it:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Wait** and for IPv6?
- No problem, let's set value in `/proc`:

```
echo "1"> /proc/sys/net/ipv6/conf/all/rp_filter
  /proc/sys/net/ipv6/conf/all/rp_filter: No such file or directory
```

Protection for Netfilter

- We only have to use the rp_filter feature.
- It is available since last century in all Linux kernel.
- Disabled by default. Enabled by all decent firewall scripts.
- To activate it:

```
echo "1"> /proc/sys/net/ipv4/conf/all/rp_filter
```

- **Wait** and for IPv6?
- No problem, let's set value in `/proc`:

```
echo "1"> /proc/sys/net/ipv6/conf/all/rp_filter
  /proc/sys/net/ipv6/conf/all/rp_filter: No such file or directory
```

Okay, Houston, we've had a problem here.

(Jack Swigert)

IPv6 protection for Netfilter

IPv6 protection for Netfilter

- A manual setup is needed.

IPv6 protection for Netfilter

- A manual setup is needed.
- Dedicated ip6tables rules need to be written.

IPv6 protection for Netfilter

- A manual setup is needed.
- Dedicated ip6tables rules need to be written.
- The network topology needs to be known.

IPv6 protection for Netfilter

- A manual setup is needed.
- Dedicated ip6tables rules need to be written.
- The network topology needs to be known.
- Good implementations already implement these rules.

IPv6 protection for Netfilter

- A manual setup is needed.
- Dedicated ip6tables rules need to be written.
- The network topology needs to be known.
- Good implementations already implement these rules.
- But do they resist to the attack?

IPv6 protection for Netfilter

The bad ruleset

```
ip6tables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

- The attack packet is valid for Netfilter.
- It belongs to an established connection.
- It is accepted by the first rule and never reaches the anti-spoofing rule.

IPv6 protection for Netfilter

The bad ruleset

```
ip6tables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

- The attack packet is valid for Netfilter.
- It belongs to an established connection.
- It is accepted by the first rule and never reaches the anti-spoofing rule.

The good ruleset

```
ip6tables -A PREROUTING -t raw -i $CLIENT_IFACE ! -s $CLIENT_NET -j DROP
```

- Raw table is before the FORWARD chain and even before connection tracking related operations.
- The packet is dropped before causing any problem.

Checkpoint setup

Checkpoint absolute newbie

- I did not read the documentation.
- Why should I? I'm working on firewalls for many years.

Checkpoint setup

Checkpoint absolute newbie

- I did not read the documentation.
- Why should I? I’m working on firewalls for many years.
- That’s the newbie reflex.

Checkpoint setup

Checkpoint absolute newbie

- I did not read the documentation.
- Why should I? I'm working on firewalls for many years.
- That's the newbie reflex.

Used software

- Demonstration version.
- Minimal features installed.

Checkpoint setup

Checkpoint absolute newbie

- I did not read the documentation.
- Why should I? I'm working on firewalls for many years.
- That's the newbie reflex.

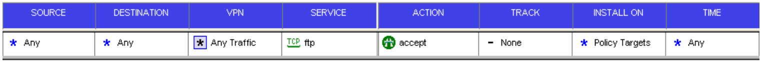
Used software

- Demonstration version.
- Minimal features installed.
- Per default installation.

Demonstration setup

Demonstration setup

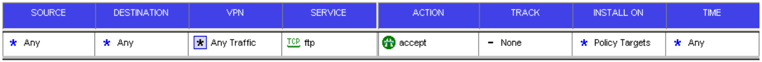
- Let’s do a filtering policy with a single FTP allowed rule ;



SOURCE	DESTINATION	VPN	SERVICE	ACTION	TRACK	INSTALL ON	TIME
* Any	* Any	Any Traffic	TCP ftp	accept	– None	* Policy Targets	* Any

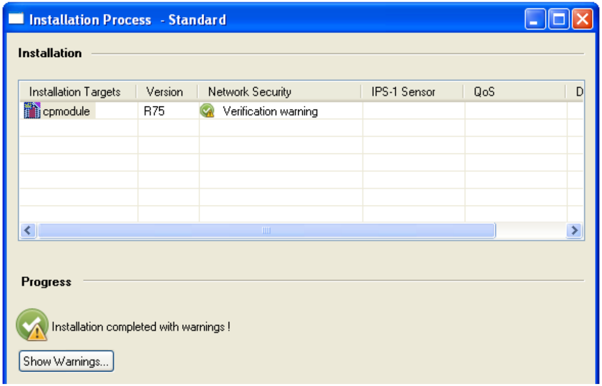
Demonstration setup

- Let's do a filtering policy with a single FTP allowed rule ;



SOURCE	DESTINATION	VPN	SERVICE	ACTION	TRACK	INSTALL ON	TIME
* Any	* Any	* Any Traffic	TCP ftp	accept	- None	* Policy Targets	* Any

- And install the resulting policy.



Installation Process - Standard

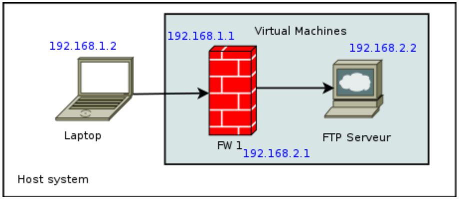
Installation

Installation Targets	Version	Network Security	IPS-1 Sensor	QoS	D
cpmodule	R75	Verification warning			

Progress

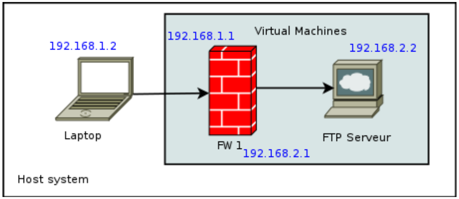
Installation completed with warnings !

Demonstration



Video

Demonstration



Video

Let’s have a firewall with a filtering policy allowing only port 21 and open a connection to port 22 on a FTP server.

Policy violation

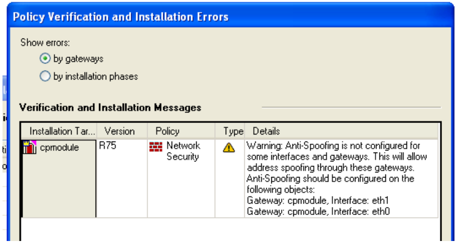
- One managed to open a connection to port 22
- With a filtering policy not allowing this

Policy violation

- One managed to open a connection to port 22
- With a filtering policy not allowing this
- But the connection was blocked after a few packets.

Policy violation

- One managed to open a connection to port 22
- With a filtering policy not allowing this
- But the connection was blocked after a few packets.
- Checkpoint GUI displays a warning about anti-spoofing.



Policy Verification and Installation Errors

Show errors:
[✓ by gateways]
by installation phases

Verification and Installation Messages

Installation Tar...	Version	Policy	Type	Details
cpmodule	R75	Network Security	⚠	Warning: Anti-Spoofing is not configured for some interfaces and gateways. This will allow address spoofing through these gateways. Anti-Spoofing should be configured on the following objects: Gateway: cpmodule, Interface: eth1 Gateway: cpmodule, Interface: eth0

There is no problem

Swift reaction of Checkpoint security team

Configuring anti-spoofing is a basic requirement.

Them

Are you planning some action regarding this issue?

Me

Anti-spoofing exists exactly for such issues. So [we] don't think that we need to do anything.

Them

There is no problem

Swift reaction of Checkpoint security team

Configuring anti-spoofing is a basic requirement.

Them

Are you planning some action regarding this issue?

Me

Anti-spoofing exists exactly for such issues. So [we] don't think that we need to do anything.

Them

Basic requirement

Choose well you contractor: the security level depends on his skills.

Other products

A generic attack

- The attack may impact other firewall brands using ALGs.
- Many of these firewalls remain untested:
    - Netfilter based firewall,
    - Iptables frontend,
    - Firewalls using ALG.

Testing

- Easy to do with *opensvp* script.
- Contact me if you are interested in using it.

Low level attacks are not dead

IPv6 Linux teaches the hard way

- For the sake of performance, rp_filter for IPv6 was not developed.
- Two patch proposals were refused.
- Hopefully, a Netfilter Reverse Path filtering module will be available in Linux 3.3.

Checkpoint default configuration

- Usability intails insecure default values.
- Anti-spoofing on Checkpoint Cluster seems problematic to manage.
- See: `http://rivald.blogspot.com/2011/01/checkpoint-utm-firewall-clusters-part-2.html`

Firewall survival guide

Getting up is dangerous

- Getting up in the OSI layer is dangerous.
- Old protocols such as FTP are dangerous.
- “New” ones such as SIP continue in the same vein.

Firewall survival guide

Getting up is dangerous

- Getting up in the OSI layer is dangerous.
- Old protocols such as FTP are dangerous.
- “New” ones such as SIP continue in the same vein.

About the security level

- Secure by default is a myth:
  - Default configuration can be vulnerable to attacks.
  - Don’t leave any warning unpunished.
- Defense In Depth should not remain a myth:
  - Protect “internal” services even if they are behind a firewall.
  - Physically separated router and firewall was a good idea.
  - Using both rp_filter and iptables-based anti-spoofing was also a good one.

Handling a generic attack

A really difficult task

- It is impossible for one individual
  - to get the list of potentially vulnerable products.
  - to contact all the relevant people.
- It is even worse when custom iptables script are vulnerable.

Possible help

- Contact CERT
  - If you get no response, send them a second e-mail.
  - Try to contact CERT Luxembourg, CERT Finland.
  - Microsoft Vulnerability Research (MSVR) is an alternative to CERT.
- Contact OSS security mailing list if open source software is involved.

Questions

Do you have any questions?

Thanks to

- Pablo Neira, Patrick McHardy: kernel developers can be friendly.
- Sebastien Tricaud, Alexandre Dulaunoy: for their help and because APT can be fun.

More information

- My blog : `http://home.regit.org`
- Secure use of Iptables and connection tracking helpers: `http://home.regit.org/netfilter-en/secure-use-of-helpers/`

Contact me

- E-mail: eric@regit.org
- Twitter: @Regiteric