

CS397 Network System Labs

Project 5: Port Knocking

Luke Kung and Jennifer C. Hou
Department of Computer Science
University of Illinois at Urbana Champaign

Distribution date: April 8, 2004

Due date: April 27, 2004 (hard deadline)

1 Introduction

Port knocking is a method of establishing connections to a remote computer without an open port. The protected port remains closed unless a particular sequence of packets are received. In this project, you will implement a Netfilter module to support port knocking on the server-side.

First, you should get yourself familiar with the firewall subsystem of Linux, i.e. Netfilter. [2] is a good starting-point. In the design of Netfilter, firewall rules are categorized into tables. There are 3 pre-defined tables: *filter*, *nat*, *mangle*. Inside each table, you can define chains, which consist of lists of rules. Each rule has a match part and a target part. The match part specifies the criteria of packets and the target part specifies what action to take if the a packet does match. In this project, you are going to define a new match called *pknock*, to guard a particular TCP port like 22(SSH) against unauthorized connection attempts. The protected port will be open after a pre-defined sequence of connection attempts to server is received. The port remains open for a certain period so that the authorized user can get service from the server during that period.

Your *pknock* match module will maintain information of incoming packets in a hash table, with the key being the source address. Each entry will also include the timestamp of the last received knocking packet and how many correct knocking packets have been received. Whenever a TCP connection packet (SYN) arrives, you attempt to find a matching entry by its source address in the hash table. If not found and the packet matches the first expected knocking port, we create a new entry. Otherwise the timestamp of the entry is checked to ensure freshness and state is updated if the packet matches the next expected knocking port. If all the knocking packets are received correctly, the state of the entry is changed to open and connection attempt to the protected port will be passed. The protected port is kept open for a certain period of time and is closed afterwards.

2 Write A Netfilter Match Module

Before working on this project, you should read [3] to understand the internals of Netfilter. The following is a step-by-step procedure to create a new Netfilter match module.

1. Write a new netfilter match in *ipt_pknock.c* under *net/ipv4/netfilter*.

To define a new Netfilter match module, you need to define a *ipt_match* structure. The most important field of this strucutre is a match function, which has the following prototype:

```
int (*match)(const struct sk_buff *skb, const struct net_device *in,
             const struct net_device *out, const void *matchinfo, int offset,
             const void *hdr, u_int16_t datalen, int *hotdrop);
```

The match function takes the packet, the input/output device information and the match data. It should return non-zero if the packet *skb* is a match and return zero otherwise. You should register your match using

ipt_register_match in the `module_init` function and call *ipt_unregister_match* to unregister it in the `module_exit` function.

2. Create a header file *ipt_pknock.h* under the `include/linux/netfilter_ipv4/` directory and define *struct ipt_pknock_info*.

This structure contains the specification of a match and is filled by iptables in userspace (See next section).

3. Add the following line to `net/ipv4/netfilter/Makefile` at an appropriate place.

```
obj-$(CONFIG_IP_NF_MATCH_PKNOCK) += ipt_pknock.o
```

4. Add the following line to `net/ipv4/netfilter/Config.in` at an appropriate place.

```
dep_tristate ' pknock match support' CONFIG_IP_NF_MATCH_PKNOCK
$CONFIG_IP_NF_IPTABLES
```

5. Do *make menuconfig* or *make xconfig* and select the *pknock* module in “Networking Options” -> “IP: Netfilter Configuration” -> “pknock match support”. You should also select other modules you want to play with.

3 Adding Support of a Match to Iptables

In order to use the pknock module, you need a way to specify a match and create the corresponding *ipt_pknock_info*. The *iptables* program is designed to take care of this. It creates this match structure in userspace and then pass it to kernel by converting the arguments in command line to corresponding data fields. To add an extension to iptables, you need to support a module that understands your own matchinfo structure and argument formats. The following is a step-by-step procedure:

1. Download the latest iptables source from <http://www.netfilter.org/>.
2. Write a new extension module *libipt_pknock.c* in `$itable_source/extensions/`.

You can use the source file of another match extension (eg. *libipt_tcp.c*) as a starting template. The core data structure of an extension is a *struct iptables_match* which has the following fields:

- `parse`
To convert command-line arguments into fields in *ipt_pknock_info*.
- `help`
Print out a help message which is shown when you execute *iptables -m pknock -help*.
- `init`
Set the default value of the fields of *ipt_pknock_info*.
- `print`
This function is called when you do a *iptables -L [chain]* to show the specification of a match.

3. Add *pknock* to the `PF_EXT_SLIB` target in `extensions/Makefile`

In an extension, you need to specify the format of the options. Your module will have to support at least the four following options:

-pknock-secured-port port The port protected by port knocking.

-pknock-delay-threshold delay The maximum delay in ms allowed between knocking packets.

-pknock-open-duration duration The length of duration in ms for the port to remain open.

-pknock-port-seq p1,...,pN The port numbers of the secret pknocking sequence.

After the extension is done, you should be able to execute the following commands:

- Getting help messages:
`iptables -m --help`
- Adding a rule using pknock module:
`iptables -A INPUT -p tcp -m pknock --pknock-secured-port 22 --pknock-delay-threshold 5000 --pknock-open-duration 180000 --pknock-port-seq 10001,10900,10500,10100,10501`
- Showing a pknock module:
`iptables -L INPUT`

4 Exporting Kernel Information Using `proc_fs`

`Proc fs` is a virtual file system that is commonly used as an interface between the userspace programs and the kernel. In this project, you are **required** to show your hash table information through `proc_fs` interface. Please check [5] for more information. Your module should use `proc_net_create` to create an `/proc/net/ip_pknock` entry and define an output function to dump the contents of the hash table. Each entry corresponds to a line of output. The format of the output will be:

```
[age] src=[src_addr] match=[match_count] state=[state]
```

age The age since the last matching packet was received or the port was opened

src_addr The source IP address of the host

match_count Number of matched packets

state Open if the port is opened to this host

You will write a callback function `list_info` that has the following prototype:

```
int list_info(char *page, char **start, off_t off, int count)
```

The above function prints each live entry of the hash table into a buffer pointed by `page`. This function should return the number of bytes printed.

5 Demo and Report

After your program is fully debugged, you will demo your program to TA during the assigned time slot. You will have to show the following:

- Use `iptables` to set up rules using `pknock match`
- Use another knocking client (will be provided by TA) to send knocking packets
- Show the status of your module in `/proc/net/ip_pknock`

- Show that your module correctly handles the timeout issue, i.e., the opened port will be closed after *open_duration* ms and a knocking sequence is invalid if the interval between two packets is longer than *delay_threshold*
- Use tcpdump to show the packet sequences

You will also have to hand in a lab report. The report should include:

- Overview of what you have done
- Debugging experience
- The commented source code
- Sample output of iptables commands
- Sample */proc/net/ip_pknock* output
- Suggestion to how to improve the lab

6 References

1. *Port Knocking Howto*, <http://www.linuxjournal.com/article.php?sid=6811>
2. Oskar Andreasson, *Iptables Tutorial*, <http://ip-tables-tutorial.frozentux.net/>
3. Rusty Russell and Harald Welta, *Netfilter Hacking Howto*, <http://www.netfilter.org/documentation/index.html>
4. Alessandro Rubini & Jonathan Corbet, *Linux Device Driver 2nd Ed*, O'Reilly 2001
Online version available at <http://www.xml.com/ldd/chapter/book/>
5. Erick Mouw, *Linux Kernel Procfs Guide*, <http://kernelnewbies.org/documents/kdoc/procfs-guide/lkprocfsguide.html>