LABORATORY

Data Compression and Encryption

Semester VII (EXTC 2016)

Experiment 6:	**DCE 6 –Firewall Configurtion**

Note: Students are advised to read through this lab sheet before doing experiment. On-the-spot evaluation may be carried out during or at the end of the experiment. Your performance, teamwork effort, and learning attitude will count towards the marks.

Experiment 6: System Security: Firewall Configuration

1 OBJECTIVE

To study personal firewall of host system using *iptables* configuration

2. INTRODUCTION

A firewall is a device or set of devices designed to permit or deny network transmissions based upon a set of rules and is frequently used to protect networks from unauthorized access while permitting legitimate communications to pass. Many personal computer operating systems include software-based firewalls to protect against threats from the public Internet.

3. LAB ENVIRONMENT

iptables:

iptables is a user space application program that allows a system administrator to configure the tables provided by the Linux kernel firewall (implemented as different Netfilter modules) and the chains and rules it stores. Different kernel modules and programs are currently used for different protocols; *iptables* applies to IPv4, *ip6tables* to IPv6, *arptables* to ARP, and *ebtables* for Ethernet frames.

Iptables requires elevated privileges to operate and must be executed by user root, otherwise it fails to function. On most Linux systems, iptables is installed as */usr/sbin/iptables* and documented in its man page, which can be opened using man iptables when installed. It may also be found in /sbin/iptables, but since iptables is more like a service rather than an "essential binary", the preferred location remains */usr/sbin*.

Xtables allows the system administrator to define *tables* containing *chains* of *rules* for the treatment of packets. Each table is associated with a different kind of packet processing. Packets are processed by sequentially traversing the rules in chains. A rule in a chain can cause a go to or jump to another chain, and this can be repeated to whatever level of nesting is desired. (A jump is like a “call”, i.e. the point that was jumped from is remembered.) Every network packet arriving at or leaving from the computer traverses at least one chain.

The origin of the packet determines which chain it traverses initially. There are five *predefined chains* (mapping to the five available Netfilter hooks), though a table may not have all chains. Predefined chains have a *policy*, for example DROP, which is applied to the packet if it reaches the end of the chain. The system administrator can create as many other chains as desired. These chains have no policy; if a packet reaches the end of the chain it is returned to the chain which called it. A chain may be empty.

- “PREROUTING”: Packets will enter this chain before a routing
- “INPUT”: Packet is going to be locally delivered. (N.B.: It does not have anything to do with processes having a socket open. Local delivery is controlled by the “localdelivery”routing table: `ip route show table local`.)
- “FORWARD”: All packets that have been routed and were not for local delivery will traverse this chain.
- “OUTPUT”: Packets sent from the machine itself will be visiting this chain.

- POSTROUTING”: Routing decision has been made. Packets enter this chain just before handing them off to the hardware.

Each rule in a chain contains the specification of which packets it matches. It may also contain a *target* (used for extensions) or *verdict* (one of the built-in decisions). As a packet traverses a chain, each rule in turn is examined. If a rule does not match the packet, the packet is passed to the next rule. If a rule does match the packet, the rule takes the action indicated by the target/verdict, which may result in the packet being allowed to continue along the chain or it may not. Matches make up the large part of rulesets, as they contain the conditions packets are tested for. These can happen for about any layer in the OSI model, as with e.g. the *--mac-source* and *-p tcp* –dportparameters, and there are also protocol independent matches, such as *-m time*.

The packet continues to traverse the chain until either

1. a rule matches the packet and decides the ultimate fate of the packet, for example by calling one of the ACCEPT or DROP, or a module returning such an ultimate fate; or
2. a rule calls the RETURN verdict, in which case processing returns to the calling chain; or
3. the end of the chain is reached; traversal either continues in the parent chain (as if RETURN was used), or the base chain policy, which is an ultimate fate, is used.

Targets also return a verdict like ACCEPT (NAT modules will do this) or DROP (e.g. the “REJECT” module), but may also imply CONTINUE (e.g. the "LOG" module;CONTINUE is an internal name) to continue with the next rule as if no target/verdict was specified at all.

4. LAB TASKS

Write correct syntax to perform following tasks and observe the output do analysis in your own words.

Task 1: Rejecting all request from other host.

Task 2: Stopping SSH connection

Taks 3: Rejecting only particular host request

Task 4 Allowing only particular host request

5. SUBMISSION

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanation to the observations that are interesting or surprising. In your report, you need to answer all the questions listed in this lab.