LABORATORY WORK NO. 8
SECURITY

1. Objectives

The aims of this laboratory are: understanding the goals of network security, understanding the vulnerabilities and threats, learning about the solutions to security problems, understanding iptables, configure packet filtering using iptables.

2. Theoretical considerations

2.1 Introduction

The Internet continues to grow exponentially. As personal and business-critical applications become more prevalent on the Internet, there are many immediate benefits. However, these network-based applications and services can pose security risks to individuals as well as to the information resources. In many cases, the rush to get connected comes at the expense of adequate network security.

Network security is the process by which digital information assets are protected. The goals of security are to protect confidentiality, maintain integrity, and assure availability. With this in mind, it is imperative that all networks be protected from threats and vulnerabilities.
Threats are unauthorized access of a network or network device. Typically, these threats are persistent due to vulnerabilities, which can arise from misconfigured hardware or software, poor network design, inherent technology weaknesses, or end-user carelessness.

The word security means protection against malicious attack by outsiders, and involves controlling the effects of errors and equipment failures. Anything that can protect against an attack will probably prevent random misfortune as well.

2.2 The goals and key elements of network security

There are three primary network security goals: confidentiality, integrity and availability. Confidentiality refers to the protection of data from unauthorized disclosure to a third party. Whether it is customer data or internal company data, a business is responsible for protecting the privacy of its data. Integrity refers to the assurance that data is not altered or destroyed in an unauthorized manner. For example, integrity is maintained when the message sent is identical to the message received. Availability is defined as the continuous operation of computing systems. Applications require differing availability levels, depending on the business impact of downtime. For an application to be available, all components must provide continuous service. These components include application and database servers, storage devices, and the end-to-end network.

A security solution contains five key elements:

- identity: refers to the accurate and positive identification of network users, hosts, applications, services, and resources. Standard technologies that enable identification include authentication protocols such as Remote Access Dial-In User Service (RADIUS) and Terminal Access Controller Access Control System Plus (TACACS+), Kerberos, and one-time password (OTP) tools. Other technologies such as digital certificates, smart cards, and directory services are beginning to play increasingly important roles in identity solutions;
- perimeter security**:** this element provides the means to control access to critical network applications, data, and services so that only legitimate users and information can pass through the network. Routers and switches with packet filtering or stateful firewalling, as well as dedicated firewall appliances, provide this control. Complementary tools, including virus scanners and content filters, also help control network perimeters;
- data privacy. When information must be protected from eavesdropping, the ability to provide authenticated, confidential communication on demand is crucial. Sometimes, data separation using tunneling technologies, such as generic routing encapsulation (GRE) or Layer 2 Tunneling Protocol (L2TP), provides effective data privacy. However, additional privacy requirements often call for the use of digital encryption technology and protocols such as IP Security (IPSec). This added

protection is especially important when implementing Virtual Private Networks (VPNs);

- security monitoring: to ensure that a network remains secure, it is important to regularly test and monitor the state of security preparation. Network vulnerability scanners can proactively identify areas of weakness, and intrusion detection systems can monitor and respond to security events as they occur.
- policy management: as networks grow in size and complexity, the requirement for centralized policy management tools grows as well. Sophisticated tools that can analyze, interpret, configure, and monitor the state of security.

2.3 Vulnerabilities and threats

2.3.1 Reconnaissance

Reconnaissance is the unauthorized discovery and mapping of systems, services, or vulnerabilities. It is also known as information gathering and, in most cases, it precedes an actual access or Denial of Service attack. The malicious intruder typically ping sweeps the target network to determine which IP addresses are alive. After this, the intruder uses a port scanner to determine what network services or ports are active on the live IP addresses. From this information, the intruder queries the ports to determine the application type and version, as well as the type and version of operating system running on the target host. Based on this information, the intruder can determine if a possible vulnerability exists that can be exploited.

2.3.2 Eavesdropping

Network snooping and packet sniffing are common terms for eavesdropping. A common method for eavesdropping on communications is to capture TCP/IP or other protocol packets and decode the contents using a protocol analyzer or similar utility. Network or protocol analyzers and packet capturing utilities on networked computer are the tools used for eavesdropping.

2.3.3 Access

System access is the ability for an unauthorized intruder to gain access to a device for which the intruder does not have an account or a password. Entering or accessing systems to which one does not have access usually involves running a hack, script, or tool that exploits a known vulnerability of the system or application being attacked.

Some examples of methods used by hackers include the following: exploit easily guessed passwords by brute force, cracking tools, dictionary attacks; exploit misconfigured services such as anonymous FTP, TFTP, remote registry access, trust relationships through spoofing and r-services and file sharing services such as NFS and Windows File Sharing; exploit application holes: mishandled input data; access outside application domain, buffer overflows, race conditions; fragmentation, TCP session hijack; trojan horses: programs that introduce an inconspicuous backdoor into a host; social engineering (posing as a network administrator to gain information from users).

2.3.3.1 Masquerading/IP spoofing

With a masquerade attack, the network intruder can manipulate TCP/IP packets by IP spoofing, falsifying the source IP address, there by appearing to be another user. The intruder assumes the identity of a valid user and gains that user’s access privileges by IP spoofing. IP spoofing occurs when intruders create IP data packets with falsified source addresses. During an IP spoofing attack, an attacker outside the network pretends to be a trusted computer. The attacker may either use an IP address that is within the range of IP addresses for the network or use an authorized external IP address that is trusted and provides access to specified resources on the network. Normally, an IP spoofing attack is limited to the injection of data or commands into an existing stream of data passed between a client and server application or a peer-to-peer network connection. The attacker simply does not worry about receiving any response from the applications.

2.3.3.2 Session replay

A sequence of packets or application commands can be captured, manipulated, and replayed to cause an unauthorized action. Mercenary Messages are designed to use mobile code to penetrate e-mail systems in order to gain private and confidential information. Mobile technologies are easy to use and most traditional security solutions, such as firewalls or anti-virus software, do not detect these security violations. Some mechanisms used to perform these attacks are as follows: cookies, JavaScript or Active X scripts.

2.3.3.3 Back doors

Back doors are paths into systems that can be created during an intrusion. The back door, unless detected, can be used again and again by an intruder to enter a computer or network. An intruder will often use the computer to gain access to other systems or to launch DoS attacks when they have no further use for the computer.

2.3.4 Denial of Service

Denial of Service (DoS) implies that an attacker disables or corrupts networks, systems, or services with the intent to deny services to intended users. DoS attacks involve either crashing the system or slowing it down to the point that it is unusable. But DoS can also be as simple as deleting or corrupting information. In most cases, performing the attack simply involves running a hack or script. The attacker does not need prior access to the target because a way to access it is all that is usually required. For these reasons, DoS attacks are the most feared. DoS attacks take many forms. Ultimately, they prevent authorized people from using a service by using up system resources. The following are some examples of common DoS threats:

- ping of death**:** this attack modifies the IP portion of the header, indicating that there is more data in the packet than there actually is, causing the receiving system to crash;
- SYN flood attack**:** this attack randomly opens up many TCP ports, tying up the network equipment or computer with so many bogus requests that sessions are thereby denied to others. This attack is accomplished with protocol analyzers or other programs;
- packet fragmentation and reassembly**:** this attack exploits a buffer–overrun bug in hosts or internetworking equipment;
- e-mail bombs: programs can send bulk e-mails to individuals, lists, or domains, monopolizing e-mail services;

- CPU hogging**:** these attacks constitute programs such as Trojan horses or viruses that tie up CPU cycles, memory, or other resources;
- malicious applets**:** these attacks are Java, JavaScript, or ActiveX programs that act as Trojan horses or viruses to cause destruction or tie up computer resources.;
- the chargen attack: this attack establishes a connection between UDP services, producing a high character output. The host chargen service is connected to the echo service on the same or different systems, causing congestion on the network with echoed chargen traffic;
- out-of-band attacks such as WinNuke - These attacks send out-of-band data to port 139 on Windows 95 or Windows NT machines. The attacker needs the victim's IP address to launch this attack;
- Land.c: this program sends a TCP SYN packet that specifies the target host address as both source and destination. The program also uses the same port (such as 113 or 139) on the target host as both source and destination, causing the target system to stop functioning;
- Teardrop.c: in this attack, the fragmentation process of the IP is implemented in such a way that reassembly problems can cause machines to crash.

2.3.5 Distributed Denial of Service

Distributed DoS (DDoS) attacks are designed to saturate network links with spurious data. This data can overwhelm an Internet link, causing legitimate traffic to be dropped. DDoS uses attack methods similar to standard DoS attacks but operates on a much larger scale. Typically hundreds or thousands of attack points attempt to overwhelm a target. Examples of DDoS attacks include the following: Smurf, Tribe Flood Network (TFN), Stacheldraht.

The Smurf attack starts with a perpetrator sending a large number of spoofed ICMP echo, or ping, requests to broadcast addresses, hoping that these packets will be magnified and sent to the spoofed addresses. If the routing device delivering traffic to those broadcast addresses performs the Layer 3 broadcast-to-Layer 2 broadcast function, most hosts on that IP network will each reply to the ICMP echo request with an ICMP echo reply, multiplying the traffic by the number of hosts responding. On a multi-access

broadcast network, there could potentially be hundreds of machines replying to each echo packet.
Turning off directed broadcast capability in the network infrastructure prevents the network from being used as a bounce site.

Tribe Flood Network (TFN) and Tribe Flood Network 2000 (TFN2K) are distributed tools used to launch coordinated DoS attacks from many sources against one or more targets. A TFN attack has the capability to generate packets with spoofed source IP addresses. An intruder instructing a master to send attack instructions to a list of TFN servers or daemons carries out a DoS attack using a TFN network. The daemons then generate the specified type of DoS attack against one or more target IP addresses. Source IP addresses and source ports can be randomized, and packet sizes can be altered. Use of the TFN master requires an intruder-supplied list of IP addresses for the daemons.

Stacheldraht combines features of several DoS attacks and also adds features such as encryption of communication between the attacker and stacheldraht masters, and automated update of the agents. There is an initial mass-intrusion phase, in which automated tools are used to remotely root-compromise large numbers of systems to be used in the attack. This is followed by a DoS attack phase, in which these compromised systems are used to attack one or more site.

2.4 Security solutions

2.4.1 Firewalls

The most well-known security device is the firewall which is a system or group of systems that enforces an access control policy between two or more networks. Firewalls fall within three classes: dedicated firewalls, server firewalls which runs on a network operating system (UNIX, NT or Win2K, Novell - it is generally an all-in-one solution that combines a firewall, access control, and virtual private networking features into one package - examples of a server-based security solution include Microsoft ISA Server, Novell BorderManager, and Check Point Firewall-1; personal firewalls resides on the user’s PC and attempt to prevent attacks and they are designed for LAN implementations (personal firewall software vendors include McAfee, Symantec).

2.4.1.1 Packet filtering

Packet filtering is a type of security technology used to control what kinds of packets are allowed to enter or leave a system or network. Packet filtering is typically used for blocking malicious traffic based on source address, port number, protocol type, and other criteria. It is a standard feature of most operating systems, including UNIX/Linux and Microsoft Windows versions, and in most routers and firewall products.

A packet filter is a piece of software which looks at the header of packets as they pass through, and decides the fate of the entire packet. It might decide to accept the packet (i.e., let the packet go through), ignore the packet (i.e., discard the packet as if it had never received it and do not notify the source of the packet), drop the packet (i.e., discard the packet and send a notification to the source of the packet).

Packet filters come in two types: static and dynamic. Static packet filters determine whether to accept or block each packet based on information stored in the header of the packet. These kind of packet filters are typically found operating systems and routers and use a series of rules for determining the fate of the packet. Administrators create these rules as an ordered list and, each packet that arrives at the filter is compared to each rule in succession until a match is found. If no match is found, the default rule, which is typically deny all, is applied.

Dynamic packet filters operate similarly to static filters but also maintain session information that enables them to control the two-way flow of packets in a session between two hosts by dynamically opening and closing ports as required. Dynamic packet filters are commonly implemented in firewalls products where they can be used to control the flow of traffic into and out of a network. Dynamic filtering is possible only with TCP packets and not with UDP or ICM packets because ICMP and UDP are connectionless protocols that do not establish sessions for communications.

Under Linux, packet filtering is built into the kernel (as a kernel module, or built right in), under Windows, packet filtering feature is into Routing and Remote Access.

2.4.1.2 Network Address Translation

Network Address Translation is a mechanism for translating the IP addresses of hosts on a private network into IP addresses belonging to a different (public) network. NAT is usually used at the boundary of two networks, especially where a private network such as a corporate network meets a public network such as the Internet.

In a typical scenario, a NAT-enabled router connects an internal corporate network with the Internet. The internal network has multiple IP hosts using private network addresses, while the router has a similar private IP address on its near-side (internal) interface and a public (global) address on its fair-side (external) interface. NAT operates by examining traffic that passes through the router and building a table that maps the connections between hosts inside the network and hosts outside on the Internet.
Implementing NAT on a router or firewall thus involves creating and configuring a NAT table containing these private/public IP address mappings.

2.4.2 Intrusion Detection Systems

Intrusion detection is the ability to detect attacks against a network. Intrusion protection should provide the following active defense mechanisms: detection - identifies malicious attacks on network and host resources, prevention – stops the detected attack from executing, reaction - immunizes the system from future attacks from a malicious source.

A host-based intrusion detection system (HIDS) audits host log files and host file systems and resources. An advantage of HIDS is that it can monitor operating system processes and protect critical system resources, including files that may exist only on that specific host. This means it can notify network managers when some external process tries to modify a system file in a way that may include a hidden back door program.
HIDS can support both passive and active detection. Active detection can be set to shut down the network connection or to stop the impacted services. This has the benefit of being able to quickly analyze an event and take corrective action.

A network-based intrusion detection system (NIDS) involves the deployment of probing devices, or Sensors throughout the network, which capture and analyze the traffic as it traverses the network.

2.5 Iptables

Netfilter and iptables are building blocks of a framework inside the Linux 2..4.x and 2.6.x kernel. This framework enables packet filtering, network address (and port) translation (NAT/NAPT) and other packet mangling.

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 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).

2.5.1 Tables, chains and targets

Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a `target', which may be a jump to a user-defined chain in the same table.

There are current three independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

The tables are as follows:

- filter: this is the default table. It contains the built-in chains INPUT (for packets coming into the host itself), FORWARD (for packets being routed through the host), and OUTPUT (for locally-generated packets);
- nat: this table is consulted when a packet that creates a new connection is encountered. It consists of three built-ins chains: PREROUTING (for altering packets as soon as they come in), OUTPUT (for altering locally-generated packets before routing),

and POSTROUTING (for altering packets as they are about to go out);
- mangle: this table is used for specialized packet alteration. It has two built-in chains: PREROUTING (for altering incoming packets before routing) and OUTPUT (for altering locally-generated packets before routing).

A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is the examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values.

Targets: *ACCEPT* (accept the packet), *DROP* (reject the packet), REJECT (same effect as DROP, except that the sender is send an ICMP ‘port unreachable’ error message), *QUEUE* (queues the packet for userspace processing) or *RETURN* (same effect as falling off the end of a chain: for a rule in a built-in chain, the policy of the chain is executed, for a rule in a user-defined chain, the traversal continues at the previous chian, just after the rule which jumped to this chain).

In the NAT table the extension targets are: DNAT (used in cases where you have a public IP and want to redirect accesses to the firewall to some other host), SNAT (used for changing the source address of packets), MASQUERADE (used in exactly the same way as SNAT, but the MASQUERADE target takes a little bit more overhead to compute - the reason for this, is that each time that the MASQUERADE target gets hit by a packet, it automatically checks for the IP address to use, instead of doing as the SNAT target does - just using the single configured IP address).

In the mangle table the extension targets are: MARK - used to set the netfilter mark value associated with the packet, TOS - used to set the 8-bit Type of Service field in the IP header, TTL - used to modify the time to live field in the IP header.

2.5.2 Using Iptables

There are several things you can do with iptables. You can create a new chain, delete an empty chain, change the policy for a built-in chain, list the rules in a chain, flush the rules out of a chain, zero the packet and byte counters on all rules in a chain.

There are several ways to manipulate rules inside a chain: append a new rule to a chain, insert a new rule at some position in a chain, replace a rule at some position in a chain, delete a rule at some position in a chain, and delete the first rule that matches in a chain.

Manipulating rules is the bread and butter of packet filtering. Most commonly, you will probably use the append (-A) and delete (-D) commands. The others (-I for insert and -R for replace) are simple extensions of these concepts.

Commands for manipulating rules:

//add a new rule in *table,* in *chain*, packet is a set of options to identify a class of packets
```
ipchains -t table -A chain packet -j target
```

//delete a rule with rule number rule_no (numbering starts from 1) from *table*
```
ipchains -t table -D chain rule_no
```

//replace the rule with number rule_no with a new rule defined `by` *`packet`* `and target`
```
ipchains -t table -R chain rule_no packet -j target
```

//insert a new rule in table in the position indicated by *rule_no*, the new rule is defined by *packet* and *target*
```
ipchains -t table -I chain rule_no packet -j target
```

Commands for manipulating chains:

//create a new chain in *table*
```
ipchains -t table -N chain_name
```

//delete a chain from *table*
```
ipchains -t table -X chain_name
```

//set the policy of a chain
```
ipchains -t table -P target
```

A packet may be identified by source address, destination address, type of packet, port (TCP, UDP) or type of message (ICMP), input/output interface, if that packet is divided in fragments. The options for identifying a packet are:

//used as negation
```
! argument
```

//identify the protocol, protocol can be icmp, tcp, udp
```
-p [!] protocol
```

//source address
```
-s [!] address[/mask] [!] [icmp_type_no | [port][:port]]
```

//destination address
```
-d [!] address[/mask] [!] [icmp_type_no | port][:port]]
```

//type of icmp message: *destination-unreachable, port-unreachable, echo-request, echo-reply*; to find out all the types execute *iptables -p icmp -h*
```
-icmp-type icmp_name
```

//destination port and source port
```
-destination-port port
-source-port port
```

//incoming/outcoming interface (ppp0, eth0, eth1 etc)
```
-i [!] interface_name[+]
-o [!] interface_name[+]
```

//this packet has fragments
```
 [!] -f
```

//MAC address of the packet
```
-mac-source [!] mac-address
```

2.5.3 A simple example

Remember that each rule specifies a set of conditions the packet must meet, and what to do if it meets them (a ‘target’). For example, you might want to drop all ICMP packets coming from the IP address 127.0.0.1. So in this case our conditions are that the protocol must be ICMP and that the source address must be 127.0.0.1. Our target is `DROP'.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms
```

```
--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms

# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP

# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
```

You can see here that the first ping succeeds (the `-c 1' tells ping to only send a single packet). Then we append (-A) to the `INPUT' chain, a rule specifying that for packets from 127.0.0.1 (`-s 127.0.0.1') with protocol ICMP (`-p icmp') we should jump to DROP (`-j DROP'). Then we test our rule, using the second ping. There will be a pause before the program gives up waiting for a response that will never come.

We can delete the rule in one of two ways. Firstly, since we know that it is the only rule in the input chain, we can use a numbered delete, to delete rule number 1 in the INPUT chain.

```
        # iptables -D INPUT 1
```

The second way is to mirror the -A command, but replacing the -A with -D. This is useful when you have a complex chain of rules and you don't want to have to count them to figure out that it's rule 37 that you want to get rid of. In this case, we would use:

```
        # iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
```

The syntax of -D must have exactly the same options as the -A (or -I or -R) command. If there are multiple identical rules in the same chain, only the first will be deleted.

3. Lab activity

3.1 Read the commands that you can use within Iptables.
3.2 Run the example from 2.5.3.
3. Write the commands to complete the following exercise.
3.3.1 Configure a virtual interface eth0:1 with the address 192.168.1.10X.

3.3.2 Configure a rule for filtering ssh traffic so that you can connect from host1 to host2 but you can not connect from host2 to host1.
3.3.3 Configure a filtering rule so that the established traffic is allowed and ne-established traffic is not allowed.
3.3.4 Configure a filtering rule to permit access only form host 192.168.1.10Y and reject the access from other hosts.
3.3.5 Configure a filtering rule to allow the access only to cs.utcluj.ro and reject the access to other addresses.
3.3.6 Configure NAT for host 192.168.1.10Y.
4. Write a set of filtering rules so that the users can have only ssh, web and ftp traffic.

Notes