INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INP TOULOUSE ENSEEIHT

ENAC ECOLE NATIONALE DE L'AVIATION CIVILE

Security Cheatsheet v1.0

Contents

Part I
Network

1 NMAP

Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing.

Nmap Target Selection	
Scan a single IP	`nmap 192.168.1.1`
Scan a host using domain name	`nmap www.testhostname.com`
Scan a range of IPs	`nmap 192.168.1.1-20`
Scan a subnet	`nmap 192.168.1.0/24`
Scan targets from a text file	`nmap -iL list_of_ips.txt`
Nmap Port Selection	
Scan a single Port	`nmap -p 22 192.168.1.1`
Scan a range of ports (here 1 to 100)	`nmap -p 1-100 192.168.1.1`
Scan 100 most common ports (Fast)	`nmap -F 192.168.1.1`
Scan all 65535 ports	`nmap -p- 192.168.1.1`
Nmap Service and OS Detection	
Detect OS and Services	`nmap -A 192.168.1.1`
Standard service detection	`nmap -sV 192.168.1.1`
Nmap outputs to file	
Save default output to file	`nmap -oN outputfile.txt 192.168.1.1`
Save results as XML	`nmap -oX outputfile.xml 192.168.1.1`

Official documentation : https://nmap.org/docs.html

2 Configurations

2.1 iptables

General command syntax:

```
iptables command CHAIN options action
```

Commands:

`-L`	List all currently inserted rules.
`-F / --flush`	Delete all rules currently inserted.
`-A`	"Append": add a rule to a specific chain.

Chains:

`INPUT`	Chain for packets targeted to your machine.
`FORWARD`	Chain for packets that your machine will route.
`OUTPUT`	Chain for packets that your machine sends out.

`--policy CHAIN DROP/ACCEPT`	Set default behaviour for a chain.

Options:

`-d / -s`	Filter by destination/source IPs.
`-p TCP`	Filter TCP packets.
`--dport #/name`	Filter by destination port number (or service, such as http, ssh ...).
`--sport #/name`	Filter by source port number (or service, such as http, ssh ...).
`--tcp-flags ALL FLAG`	Filter by flag. `ALL` means "inspect all packets", substitute `FLAG` with the flag you want to filter.
`-m MODULE`	Load an extension.
`--string "pattern"`	Match "pattern" in packets.
`--algo bm`	Use Boyer-Moore algorithm for pattern matching.
`--state OPTION`	Filter by connection state. Can be `NEW`, `RELATED`, `ESTABLISHED` or `INVALID`.

Additional references :

- Iptables Tutorial 1.2.2 : http://homes.di.unimi.it/sisop/qemu/iptables-tutorial.pdf

2.2 `ip` vs `ifconfig`

The command `ifconfig` is the old command provided by package `net-tools`. You should mainly use the `ip` command today. A great cheatsheet for the `ip` command can be found on RedHat web site (**here**). This cheatsheet is also attached in this document, in appendix A.

Part II
Binary Exploitation

1 Registers

1.1 General Purpose Registers (GPRs)

There is 8 General Purpose Registers (GPRs) in x86 and x64 architectures.

x86	x64	Name	Function
EAX	RAX	Accumulator	Used in arithmetic operations
EBX	RBX	Base	Used as a pointer to data
ECX	RCX	Counter	Used in shift/rotate instructions and loops
EDX	RDX	Data	Used in arithmetic operations and I/O operations
ESP	RSP	Stack Pointer	Pointer to the top of the stack
EBP	RBP	Stack Base Pointer	Used to point to the base of the stack
ESI	RSI	Source Index	Used as a pointer to a source in stream operations
EDI	RDI	Destination Index	Used as a pointer to a destination in stream operations

1.2 Segment Registers

There is 6 Segment Registers (extremely useful in Operating Systems course) :

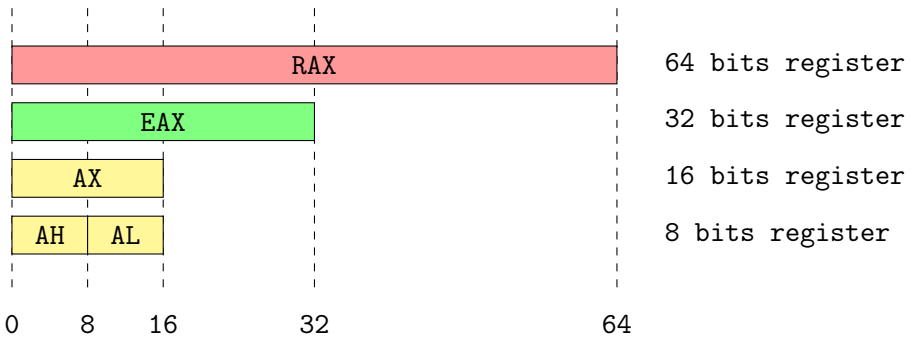
Symbol	Name	Function
SS	Stack Segment	Pointer to the stack
CS	Code Segment	Pointer to the code
DS	Data Segment	Pointer to the data
ES	Extra Segment	Pointer to extra data ('E' stands for 'Extra')
FS	F Segment	Pointer to more extra data ('F' comes after 'E')
GS	G Segment	Pointer to still more extra data ('G' comes after 'F')

1.3 EFLAGS Register

The EFLAGS is a 32-bit register used as a collection of bits representing Boolean values to store the results of operations and the state of the processor.

1.4 Register sizes

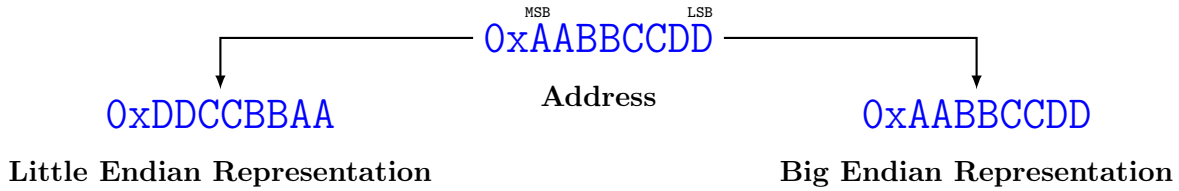
Registers have different names according to their sizes, let's see an example with register **A** :



2 Addresses (Endianness)

2.1 Little Endian and Big Endian

Addresses are represented in different ways in memory. It's called **endianness**.



2.2 Addresses manipulation in Python

In order to manipulate addresses (useful in your python scripts for exploiting binaries), you can use the python `struct` library :

```
1 import struct
2 # Writes bytes of an integer (I)
3 ## => in Litte Endian (<) representation
4 bytes_addr_le = struct.pack('<I', 0xAABBCCDD)
5 ## => in Big Endian (>) representation
6 bytes_addr_be = struct.pack('>I', 0xAABBCCDD)
```

Common struct formats :

Name	Format String
Big Endian Representation	>
Little Endian Representation	<
Others : https://docs.python.org/2/library/struct.html#byte-order-size-and-alignment	
Parse as a `signed int`	i
Parse as an `unsigned int`	I
Parse as a `char`	c
Others : https://docs.python.org/2/library/struct.html#format-characters	

Complete documentation can be found here : https://docs.python.org/2/library/struct.html

3 GDB useful commands

Description	Command
Set a breakpoint at a line number (in C code listing)	`break <line_number>`
Set a breakpoint at address (in assembly code)	`break * <address>`
List all set breakpoints.	`info breakpoints`
Delete a breakpoint	`delete <breakpoint_number>`
Disassemble a function	`disassemble <function>`
Start the program being debugged.	`run <program arguments>`
List lines of C source code (if program compiled with `-g` flag).	`list`
Continue execution up to the next breakpoint or termination.	`continue`

There is some interesting plugins for GDB:

- **GDB-Peda**, useful when you're starting with GDB. It adds lots of features and is clearer than 'raw' GDB for beginners. You can find it here : `https://github.com/longld/peda`
- **GDB-Dashboard** : Provides a modular interface showing relevant information about the program being debugged. You can find it here : `https://github.com/cyrus-and/gdb-dashboard`

Part III
SwissKnife

1 Swissknife Tools and Commands

1.1 Quickly share files, HTTP server in the current folder

Start an HTTP server in the current folder to share files :

```
user@machine:~$ cd /path/to/share/
user@machine:/path/to/share/$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

1.2 Generating a SSH key

You can generate a SSH key with the following command :

```
ssh-keygen -t <algorithm> -b <number_of_bits>
```

Example for generating an RSA 4096 bits SSH key :

```
ssh-keygen -t rsa -b 4096
```

1.3 Copying your SSH key to a remote server for future connections

Now that you have a SSH key, you need to copy id to the remote server to be able to connect to it in the future. To do this, use this command :

```
ssh-copy-id -i /path/to/ssh_key.pub login@server.com
```

1.4 SFTP file transfers

```
sftp -P PORT login@server.com
```

1.4.1 Getting a remote file

```
sftp> get /remote/path/to/dir/file.txt
```

1.4.2 Getting a remote folder recursively

```
sftp> get -r /remote/path/to/dir/
```

1.4.3 Putting a file to a remote server

```
sftp> put /local/path/to/dir/file.txt /remote/path/to/dir/file.txt
```

1.4.4 Putting a folder recursively to a remote server

```
sftp> put -r /local/path/to/dir/ /remote/path/to/
```

2 General Culture

2.1 Cybersecurity Conferences

SSTIC	
It is a francophone conference on information security.	
Dates	June in Rennes, FRANCE
Website	`https://www.sstic.org/`

Toulouse Hacking Convention (THC)	
It is your cybersecurity conference. Takes place in ENSEEIHT !	
Dates	1 day in March in Toulouse, FRANCE
Website	`https://thcon.party/`

USENIX Security Symposium	
Dates	Mid August in Boston, Massachusetts, USA
Website	`https://www.usenix.org/conference/usenixsecurity20`

Chaos Communications Congress (CCC)	
The Chaos Communication Congress is an annual conference organised by the Chaos Computer Club in Germany. The congress features a variety of lectures and workshops on technical and political issues related to security, cryptography, privacy and online freedom of speech. It's not always about security, but it's always about hacking. You can see all the recorded talks for free on their website.	
Dates	27-30 December, in Leipzig or Hamburg, GERMANY
Website	`https://media.ccc.de/`

DEFCON	
It is one of the world's largest hacker conventions, held annually in Las Vegas, Nevada, with the first DEF CON taking place in June 1993.	
Dates	August, in Las Vegas, USA
Website	`https://defcon.org/`

BlackHat	
It is a big information security event, providing attendees with the very latest in research, development and trends. Usually there is a few days for technical trainings, and a few days for more traditional talks.	
Dates	August, in Las Vegas, USA
Website	`https://www.blackhat.com/us-20/`

Appendix

A ip Command Cheat Sheet

IP QUERIES

SUBCOMMAND	DESCRIPTIONS AND TASKS
`addr`	Display IP Addresses and property information (abbreviation of address) **`ip addr`** Show information for all addresses **`ip addr show dev em1`** Display information only for device em1
`link`	Manage and display the state of all network interfaces **`ip link`** Show information for all interfaces **`ip link show dev em1`** Display information only for device em1 **`ip -s link`** Display interface statistics
`route`	Display and alter the routing table **`ip route`** List all of the route entries in the kernel
`maddr`	Manage and display multicast IP addresses **`ip maddr`** Display multicast information for all devices **`ip maddr show dev em1`** Display multicast information for device em1
`neigh`	Show neighbour objects; also known as the ARP table for IPv4 **`ip neigh`** Display neighbour objects **`ip neigh show dev em1`** Show the ARP cache for device em1
`help`	Display a list of commands and arguments for each subcommand **`ip help`** Display ip commands and arguments **`ip addr help`** Display address commands and arguments **`ip link help`** Display link commands and arguments **`ip neigh help`** Display neighbour commands and arguments

MULTICAST ADDRESSING

SUBCOMMAND	DESCRIPTIONS AND TASKS
`maddr add`	Add a static link-layer multicast address **`ip maddr add 33:33:00:00:00:01 dev em1`** Add mutlicast address 33:33:00:00:00:01 to em1
`maddr del`	Delete a multicast address **`ip maddr del 33:33:00:00:00:01 dev em1`** Delete address 33:33:00:00:00:01 from em1

MODIFYING ADDRESS AND LINK PROPERTIES

SUBCOMMAND	DESCRIPTIONS AND TASKS
`addr add`	Add an address **`ip addr add 192.168.1.1/24 dev em1`** Add address 192.168.1.1 with netmask 24 to device em1
`addr del`	Delete an address **`ip addr del 192.168.1.1/24 dev em1`** Remove address 192.168.1.1/24 from device em1
`link set`	Alter the status of the interface **`ip link set em1 up`** Bring em1 online **`ip link set em1 down`** Bring em1 offline **`ip link set em1 mtu 9000`** Set the MTU on em1 to 9000 **`ip link set em1 promisc on`** Enable promiscuous mode for em1

ADJUSTING AND VIEWING ROUTES

SUBCOMMAND	DESCRIPTIONS AND TASKS
`route add`	Add an entry to the routing table **`ip route add default via 192.168.1.1 dev em1`** Add a default route (for all addresses) via the local gateway 192.168.1.1 that can be reached on device em1 **`ip route add 192.168.1.0/24 via 192.168.1.1`** Add a route to 192.168.1.0/24 via the gateway at 192.168.1.1 **`ip route add 192.168.1.0/24 dev em1`** Add a route to 192.168.1.0/24 that can be reached on device em1
`route delete`	Delete a routing table entry **`ip route delete 192.168.1.0/24 via 192.168.1.1`** Delete the route for 192.168.1.0/24 via the gateway at 192.168.1.1
`route replace`	Replace, or add if not defined, a route **`ip route replace 192.168.1.0/24 dev em1`** Replace the defined route for 192.168.1.0/24 to use device em1
`route get`	Display the route an address will take **`ip route get 192.168.1.5`** Display the route taken for IP 192.168.1.5

MANAGING THE ARP TABLE

SUBCOMMAND	DESCRIPTIONS AND TASKS
`neigh add`	Add an entry to the ARP Table **`ip neigh add 192.168.1.1 lladdr 1:2:3:4:5:6 dev em1`** Add address 192.168.1.1 with MAC 1:2:3:4:5:6 to em1
`neigh del`	Invalidate an entry **`ip neigh del 192.168.1.1 dev em1`** Invalidate the entry for 192.168.1.1 on em1
`neigh replace`	Replace, or adds if not defined, an entry to the ARP table **`ip neigh replace 192.168.1.1 lladdr 1:2:3:4:5:6 dev em1`** Replace the entry for address 192.168.1.1 to use MAC 1:2:3:4:5:6 on em1

USEFUL NETWORKING COMMANDS (NOT NECESSARILY PROVIDED FROM IPROUTE)

SUBCOMMAND	DESCRIPTIONS AND TASKS
`arping`	Send ARP request to a neighbour host **`arping -I eth0 192.168.1.1`** Send ARP request to 192.168.1.1 via interface eth0 **`arping -D -I eth0 192.168.1.1`** Check for duplicate MAC addresses at 192.168.1.1 on eth0
`ethtool`	Query or control network driver and hardware settings **`ethtool -g eth0`** Display ring buffer for eth0 **`ethtool -i eth0`** Display driver information for eth0 **`ethtool -p eth0`** Identify eth0 by sight, typically by causing LEDs to blink on the network port **`ethtool -S eth0`** Display network and driver statistics for eth0
`ss`	Display socket statistics. The below options can be combined **`ss -a`** Show all sockets (listening and non-listening) **`ss -e`** Show detailed socket information **`ss -o`** Show timer information **`ss -n`** Do not resolve addresses **`ss -p`** Show process using the socket

COMPARING NET-TOOLS VS. IPROUTE PACKAGE COMMANDS

NET-TOOLS COMMANDS	IPROUTE COMMANDS
`arp -a`	`ip neigh`
`arp -v`	`ip -s neigh`
`arp -s 192.168.1.1 1:2:3:4:5:6`	`ip neigh add 192.168.1.1 lladdr 1:2:3:4:5:6 dev eth1`
`arp -i eth1 -d 192.168.1.1`	`ip neigh del 192.168.1.1 dev eth1`
`ifconfig -a`	`ip addr`
`ifconfig eth0 down`	`ip link set eth0 down`
`ifconfig eth0 up`	`ip link set eth0 up`
`ifconfig eth0 192.168.1.1`	`ip addr add 192.168.1.1/24 dev eth0`
`ifconfig eth0 netmask 255.255.255.0`	`ip addr add 192.168.1.1/24 dev eth0`
`ifconfig eth0 mtu 9000`	`ip link set eth0 mtu 9000`
`ifconfig eth0:0 192.168.1.2`	`ip addr add 192.168.1.2/24 dev eth0`
`netstat`	`ss`
`netstat -neopa`	`ss -neopa`
`netstat -g`	`ip maddr`
`route`	`ip route`
`route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0`	`ip route add 192.168.1.0/24 dev eth0`
`route add default gw 192.168.1.1`	`ip route add default via 192.168.1.1`

redhat.

TLS-SEC