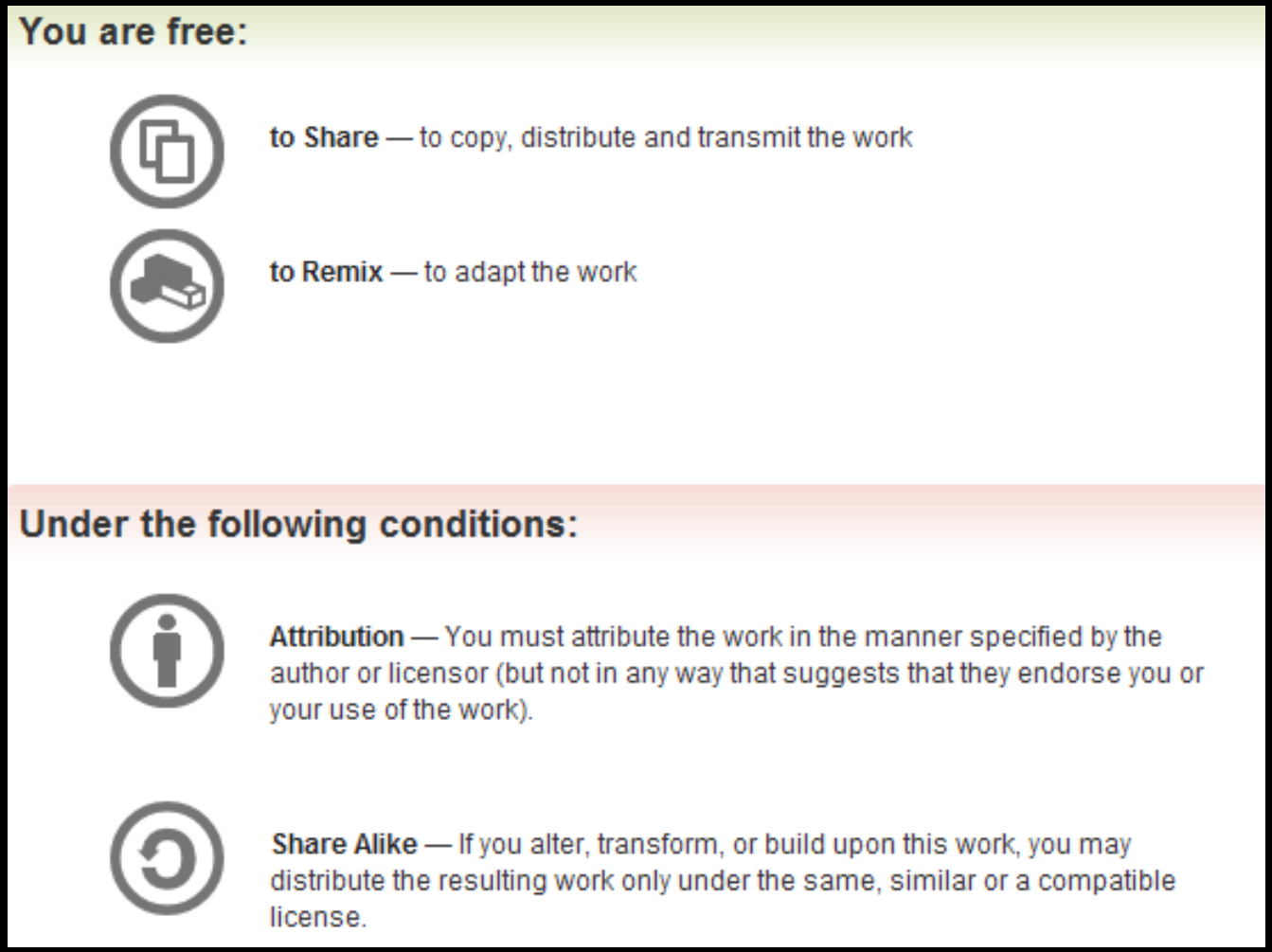
Hacking Techniques & Intrusion Detection

Ali Al-Shemery
arabnix [at] gmail

All materials is licensed under a Creative Commons "Share Alike" license.

- http://creativecommons.org/licenses/by-sa/3.0/



whoami

- Ali Al-Shemery
- Ph.D., MS.c., and BS.c., Jordan
- More than 14 years of Technical Background (mainly Linux/Unix and Infosec)
- Technical Instructor for more than 10 years (Infosec, and Linux Courses)
- Hold more than 15 well known Technical Certificates
- Infosec & Linux are my main Interests

Scanning and Fingerprinting

Outline

- Diving into Important Network Protocols (TCP, UDP, ICMP, ARP, etc)
- Nmap – Intro.
- Host Discovery
- Tracing the Route
- Port Scanning
- OS and Service Fingerprinting
- Learning Python in 4 Slides
- Packet Crafting

Diving into Important Network Protocols

- Diving into Important Network Protocols:
  - TCP
  - UDP
  - ICMP
  - ARP
  - HTTP
  - etc

Nmap

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

  - Fyodor

- IMO: #1 tool in your security arsenal!

Important Note:

A huge difference between running Nmap as a privileged/unprivileged user!

Host Discovery

- Identifying Live Systems
- Also called “Network Sweep”

- Nmap ping sweeps:
  - Ping Only (-sP)
  - ARP Ping (-PR)
  - ICMP Echo Request Ping (-PE)
  - TCP SYN Ping (-PS)
  - TCP ACK Ping (-PA)
  - UDP Ping (-PU)

DEMO

Assignment #1

- Why do host discovery or network sweeping if we already have the target list of IP(s)?

Tracing the Route

- Nmap --traceroute option
- DEMO

DEMO

Port Scanning

- The act of testing a remote port to know in which state it is.
- Common port states:
  - Open,
  - Closed,
  - and Filtered.

DEMO

Port Scanning - Techniques

- TCP SYN or Stealth Scan (-sS)
- TCP Connect Scan (-sT)
- TCP ACK Scan
- UDP Scan (-sU)
- TCP FIN Scan (-sF)
- TCP NULL Scan (-sN)
- XMAS Scan Scan (-sX)
- Custom Scan (--scanflags)
- IP Protocol Scan (-sO)

DEMO

OS and Service Fingerprinting

- Operating System Detection (-O)
- Service Version Detection (-sV)

Or

- Enables OS detection and Version detection, Script scanning and Traceroute (-A)

DEMO

Evasion Techniques

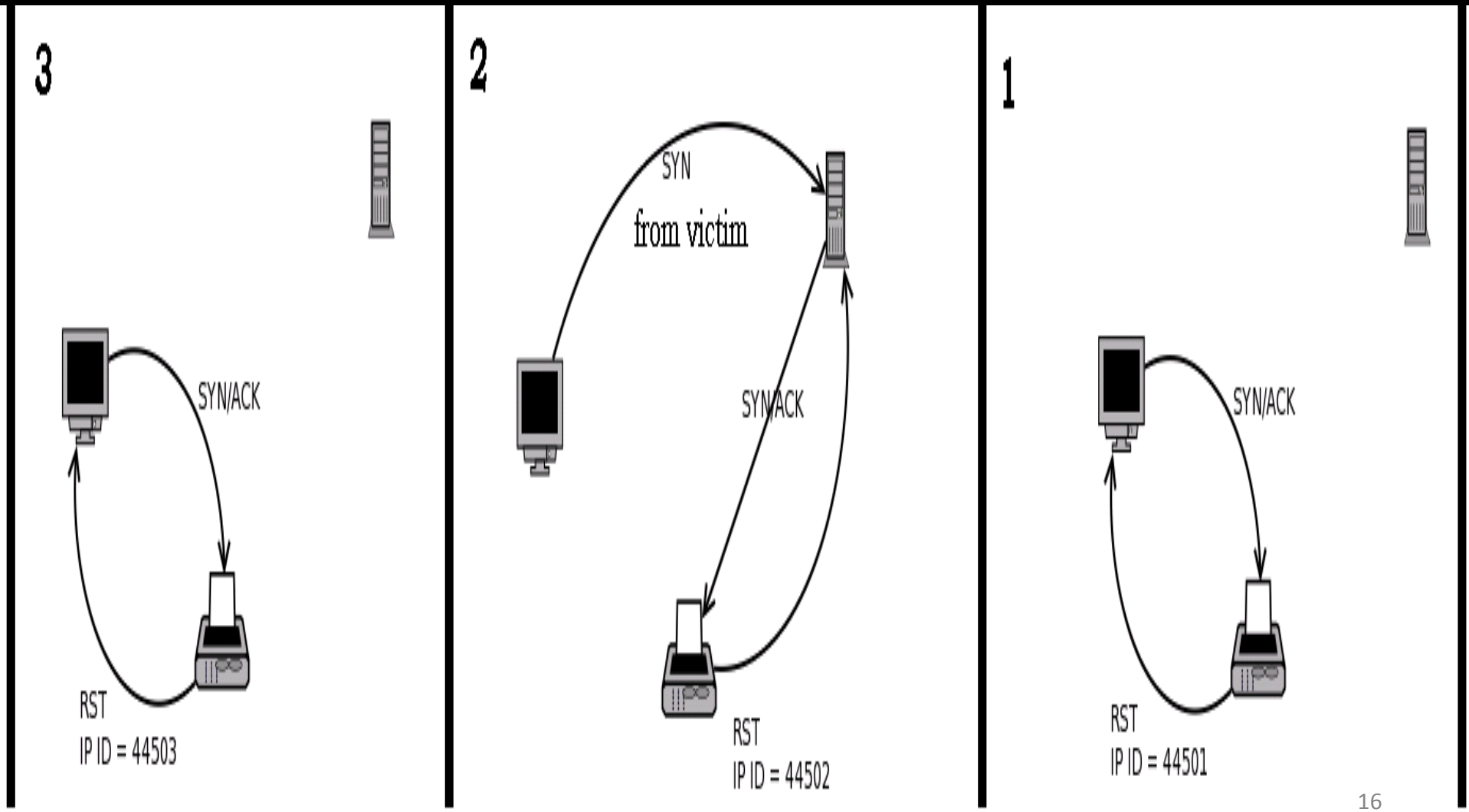
- Fragment Packets (-f)
- Specific MTU (--mtu)
- Using a Decoy (-D)
- Specify source port (--source-port)
- Append Random Data (--data-length)
- Spoof MAC Address (--spoof-mac)
- Send Bad Checksum (--badsum)

- That's all? Nope, check the next slide.

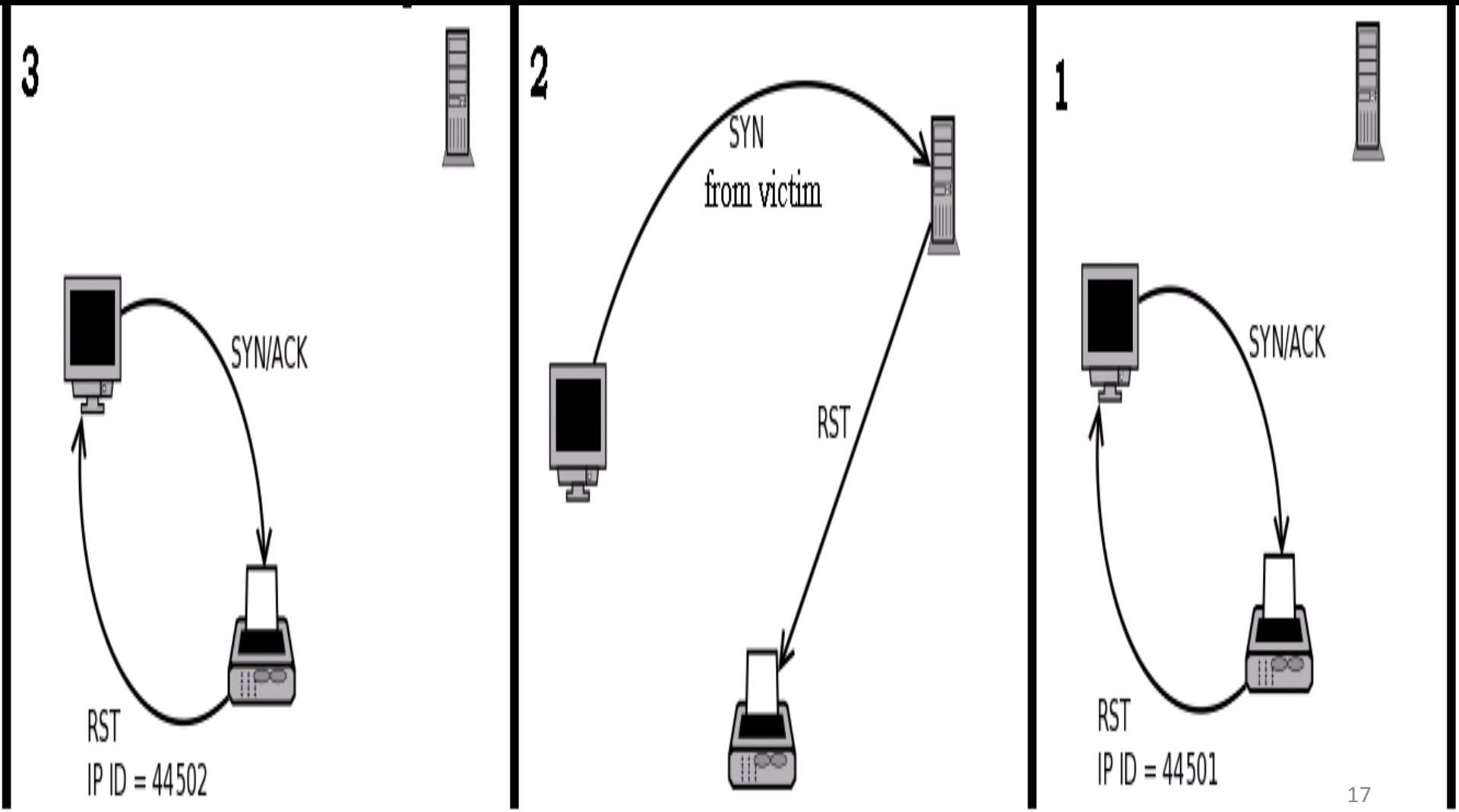
IDLE Scan

- A completely blind port scanning technique, that attackers can use to scan a target without sending a single packet to the target from their own IP address!
- Nmap IDLE Scan (-sI)

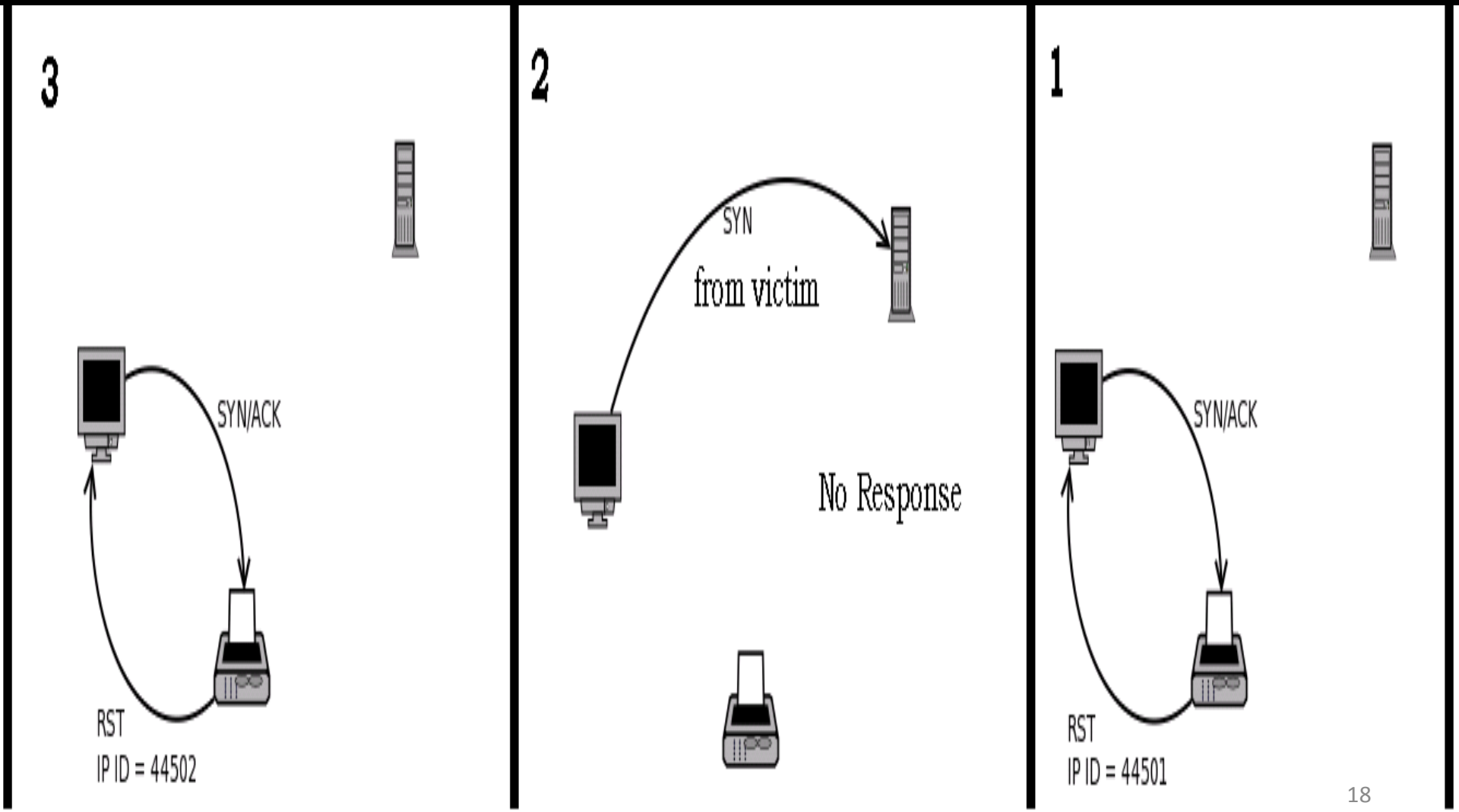
IDLE Scan – Open Port



IDLE Scan – Closed Port



IDLE Scan – Filtered Port



Assignment #2

Choose One:

- How can we find an IDLE machine?
- What is Nmap's Scripting Engine? And how can we benefit from it?

Learning Python in 4 Slides!!!

Python in 4 Slides (1/4)

- Python is an open source scripting language.
- Developed by Guido van Rossum in the early 1990s.
- Name came from TV series “Monty Python’s Flying Circus”.
- Python is cross platform (Linux, Windows, Mac, etc).
- Ideal language for scripting and rapid application development in many areas on most platforms.
- If you’re involved in vulnerability research, reverse engineering or penetration testing, I suggest “Python” for you.

Learning Python in 4 Slides (2/4)

Why Python for Penetration Testers?

- Simple, and easy to learn,
- Free and Open Source,
- powerful high-level programming language
- relatively fast,
- widely used, and
- Portable,
- Extensive Libraries,
- Interpreted, Extensible, Embeddable

Learning Python in 4 Slides (3/4)

- This is an int (signed, 32bits) : 88
- This is a long (signed, infinite): 88L
- This is a str : "bell\x07\n" or ’bell\x07\n’ (" ⇐⇒ ’)
- This is a tuple (immutable): (0,1,“33")
- This is a list (mutable): [8,4,2,"1"]
- This is a dict (mutable): { "one":1 , "two":2 }

Learning Python in 4 Slides (4/4)

```
if condition1:
    instruction1
    instruction2
elif condition2:
    instruction
else:
    instruction
```

```
try:
    instruction
except exception:
    instruction
else:
    instruction
```

```
def fact(x):
    if x == 0:
        return 1
    else:
        return x*fact(x-1)
```

```
while condition:
    instruction
    instruction
```

```
lambda x,y: x+y
```

```
import httplib
from scapy.all import ARP
from scapy.all import *
import scapy.all as scapy
```

```
for variable in set:
    instruction
```

Packet Crafting

Packet Crafting

What is Packet Crafting?

- The art of manually generating packets to test network devices,
- Packets are crafted to test Firewalls, IDS, TCP/IP Stack,….,etc,
- Auditing network protocols looking for vulnerabilities to exploit,
- Find inconsistencies and poor network protocol implementations.

Packet Crafting – Cont.

- Crafting test Packets is an Art!
- Different tools are available to Craft Packets,
- BUT the process of Crafting Packets in such a way that will stress test protocols, firewalls and any other network devices for the purpose of uncovering faults, is an Art.

Packet Crafting Composition

- Packet Crafting consist of:
  - Packet Assembly,
  - Packet Editing,
  - Packet Re-Play,
  - and Packet Decoding

Packet Crafting Tools

Best Packet Crafters:

- Scapy - http://www.secdev.org/projects/scapy/
- hping3 - http://www.hping.org/
- Netdude - http://netdude.sourceforge.net/
- tcpreplay - http://tcpreplay.synfin.net/trac/

Packet Crafting with Scapy

Scapy Overview

- Scapy is a Python program that enables the user to send, sniff and dissect and forge network packets.
- This capability allows construction of tools that can probe, scan or attack networks.
- It can replace hping, arpspoof, arp-sk, arping, p0f and even some parts of Nmap, tcpdump, and tshark.

Scapy Overview – Cont.

- Scapy was created by Philippe Biondi and runs in Python:
  - Can be used interactively at a Python prompt
  - Included within Python scripts for more complex interactions
- Must run with root privileges to craft packets,
- Don't need to be a Python Guru to use Scapy!

Scapy Basics - 1

- Supported protocols:

```
>>> ls()
```

- Details about a specific protocol:

```
>>> ls(TCP)
```

- Available commands/functions:

```
>>> lsc()
```

Scapy Basics - 2

Crafting a SYN/ACK Packet

```
>>> pkt = IP(dst="192.168.122.101")
>>> pkt /= TCP(dport=80, flags="SA")
```

Crafting ICMP Host Unreachable Packet

```
>>> pkt = IP(dst="192.168.122.101")
>>> pkt /= ICMP(type=3,code=1)
```

Scapy Basics - 3

Single Line:

- ICMP echo request Packet

```
>>> mypkt = IP(dst="192.168.122.101")
/ICMP(code=0,type=8)
```

- TCP FIN, Port 22, Random Source Port, and Random Seq#

```
>>> mypkt = IP(dst="192.168.122.101")
/TCP(dport=22,sport=RandShort(),seq=RandShort(),flags="F")
```

Sending and Receiving Packets – @L3

- Send packet at layer 3

```
>>> send(packet)
```

- Send packet at L3 and receive one response

```
>>> resp = sr1(packet)
```

- Send packet at L3 and receive all responses

```
>>> ans,unans = sr(packet)
```

Sending and Receiving Packets – @L2

- Send packet at layer 2

```
>>> sendp(Ether()/packet)
```

- Send packet at L2 and receive one response

```
>>> resp = srp1(packet)
```

- Send packet at L2 and receive all responses

```
>>> ans,unans = srp(packet)
```

Displaying Packets

- Get a summary of each packet:

```
>>> pkts.summary()
```

- Get the whole packet list:

```
>>> pkts.show()
```

Scapy Host Discovery

```
>>> ans,unans =
  srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst
  ="192.168.122.0/24"),timeout=2)


>>> ans.summary(lambda(s,r):
  r.sprintf("Ether: %Ether.src% \t\t
  Host: %ARP.psrc%"))
```

Scapy Port Scanning

- TCP SYN Scanner

```
>>> sr1(IP(dst="192.168.122.101")
/TCP(dport=90,flags="S"))

>>> a,u = sr(IP(dst="192.168.122.101")
/TCP(dport=(80,100),flags="S"))

>>> a.summary(lambda(s,r): r.sprintf("Port:
%TCP.sport% \t\t Flags: %TCP.flags%"))
```

Scapy Sniffing - 1

Scapy has powerful capabilities to capture and analyze packets.

Configure the network interface to sniff packets from:

```
>>> conf.iface="eth0“
```

Configure the scapy sniffer to sniff only 20 packets

```
>>> pkts=sniff(count=20)
```

Scapy Sniffing - 2

Sniff packets and stop after a defined time:

```
>>> pkts=sniff(count=100,timeout=60)
```

Sniff only packets based on a filter:

```
>>> pkts = sniff(count=100,filter="tcp port 80")
```

Scapy Sniffing - 3

```
>>> pkts = sniff(count=10,prn=lambda
x:x.sprintf("SrcIP={IP:%IP.src% ->
DestIP=%IP.dst%} |
Payload={Raw:%Raw.load%\n}"))
```

- What is that doing ???

Exporting Packets

- Sometimes it is very useful to save the captured packets in a PCAP file for future work:

```
>>> wrpcap("file1.cap", pkts)
```

Dumping packets in HEX format:

```
>>> hexdump(pkts)
```

- Dump a single packet in HEX format:

```
>>> hexdump(pkts[2])
```

- Convert a packet to hex string:

```
>>> str(pkts[2])
```

- Exporting to Base64 encoded packets:

```
>>> export_object(pkts)
```

Importing Packets

To import from a PCAP file:

```
>>> pkts = rdpcap(“file1.cap")
```

Or use the scapy sniffer but with the offline argument:

```
>>> pkts2 = sniff(offline="file1.cap")
```

Create your own tools

```
>>> def handler(packet):
        hexdump(packet.payload)
>>> sniff(count=20, prn=handler)

>>> def handler2(packet):
        sendp(packet)
>>> sniff(count=20, prn=handler2)
```

Create your own tools – 2

arpping.py

listpacket.py

arppoisonor.py

Assignment #3
Create your own tools

Choose any two:

[1] Create a TCP ACK Port Scanner

[2] Create a TCP Replay Tool

[3] Create a UDP Ping Tool

[4] Create a Sniffer that filters based on user input

SUMMARY

- Diving into Important Network Protocols (TCP, UDP, ICMP, HTTP, etc)
- Sweep Networks to discover hosts
- Scan systems to discover open ports
- Fingerprint OS's and services
- Craft your own packets using Scapy

References

[1] William Zereneh, http://www.scs.ryerson.ca/~zereneh/cn8822/PacketCrafting.pdf

[2] Mike Poor, Packet Craft for Defense-in-Depth, http://www.inguardians.com/research/docs/packetfoo.pdf

[3] SecTools.Org: Top 125 Network Security Tools, http://sectools.org/tag/packet-crafters/

[4] Scapy Documentation, http://www.secdev.org/projects/scapy/doc/

[5] Python, http://www.python.org/

[6] Python tools for penetration testers, http://www.dirk-loss.de/python-tools.htm

[7] Nmap Book Online, http://nmap.org/book/