Sofacy APT hits high profile targets with updated toolset

By GReAT

Published: 2015-12-04 · Archived: 2026-04-05 14:52:11 UTC

Sofacy (also known as “Fancy Bear”, “Sednit”, “STRONTIUM” and “APT28”) is an advanced threat group that has been active since around 2008, targeting mostly military and government entities worldwide, with a focus on NATO countries. More recently, we have also seen an increase in activity targeting Ukraine.

Back in 2011-2012, the group used a relatively tiny implant (known as “Sofacy” or SOURFACE) as its first stage malware. The implant shared certain similarities with the old Miniduke implants. This led us to believe the two groups were connected, at least to begin with, although it appears they parted ways in 2014, with the original Miniduke group switching to the CosmicDuke implant.

At some point during 2013, the Sofacy group expanded its arsenal and added more backdoors and tools, including CORESHELL, SPLM (aka Xagent, aka CHOPSTICK), JHUHUGIT (which is built with code from the Carberp sources), AZZY (aka ADVSTORESHELL, NETUI, EVILTOSS, and spans across four to five generations) and a few others. We’ve seen quite a few versions of these implants and they were relatively widespread for a time.

> #Sofacy group has been active since 2008, targeting mostly military and government entities in NATO countries
>
> Tweet

Earlier this year, we noticed a new release of the AZZY implant which, at the time, was largely undetected by anti-malware products. We observed several waves of attacks using this version, most recently in October. The new waves of attacks also included a new generation of USB stealers deployed by the Sofacy actor, with the first versions dating back to February 2015, and which appear to be geared exclusively towards high profile targets.

Sofacy’s August 2015 attack wave

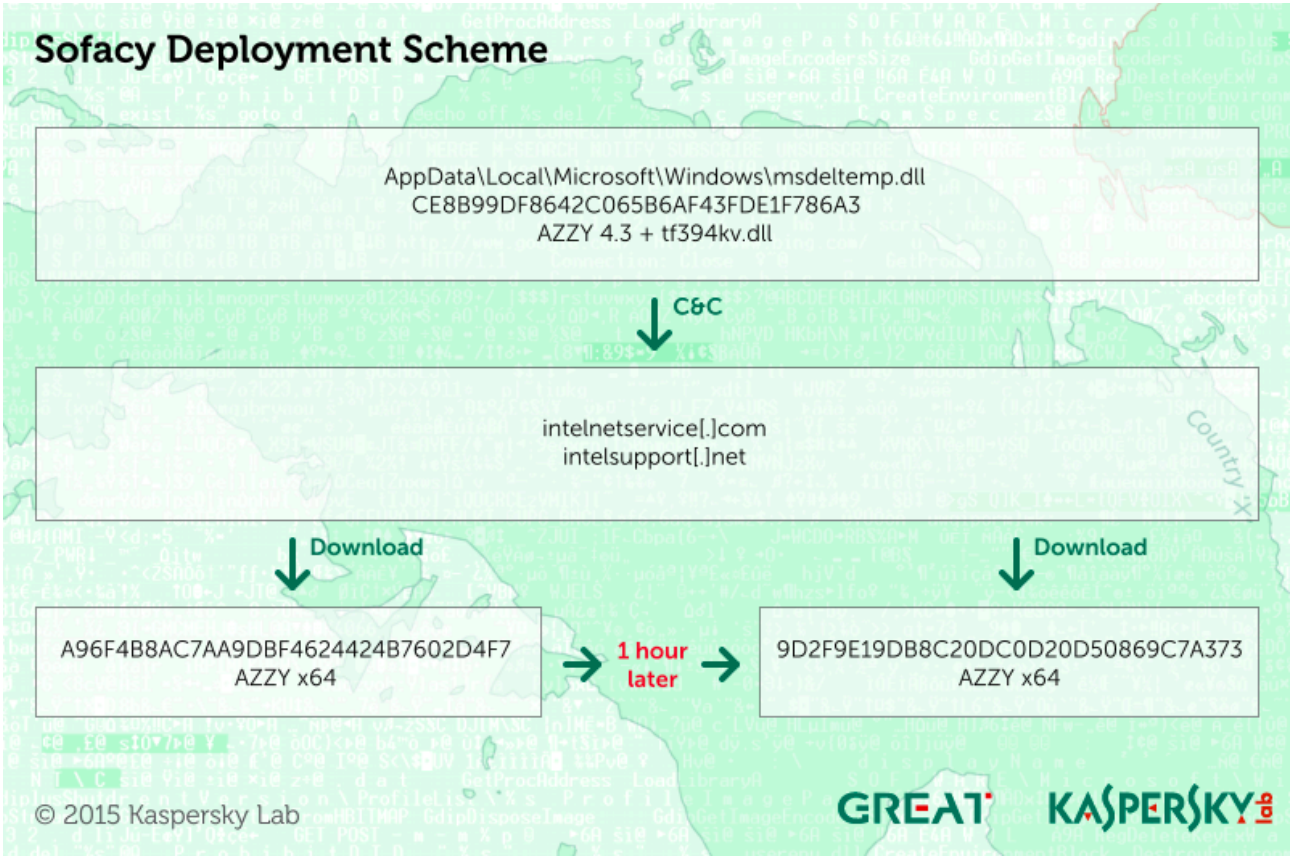
In the months leading up to August, the Sofacy group launched several waves of attacks relying on zero-day exploits in Microsoft Office, Oracle Sun Java, Adobe Flash Player and Windows itself. For instance, its JHUHUGIT implant was delivered through a Flash zero-day and used a Windows EoP exploit to break out of the sandbox. The JHUHUGIT implant became a relatively popular first stage for the Sofacy attacks and was used again with a Java zero-day (CVE-2015-2590) in July 2015.

While the JHUHUGIT (and more recently, “JKEYSKW”) implant used in most of the Sofacy attacks, high profile victims are being targeted with another first level implant, representing the latest evolution of their AZZYTrojan.

> Two recurring characteristics of the #Sofacy group are speed and the use of multi-backdoor packages
>
> Tweet

The first versions of the new AZZY implant appeared in August of this year. During a high profile incident we investigated, our products successfully detected and blocked a “standard” Sofacy “AZZY” sample that was used to target a range of defense contractors. The sample used in this attack (md5 A96F4B8AC7AA9DBF4624424B7602D4F7, compiled July 29th, 2015) was a pretty standard Sofacy x64 AZZY implant, which has the internal name “advshellstore.dll”.

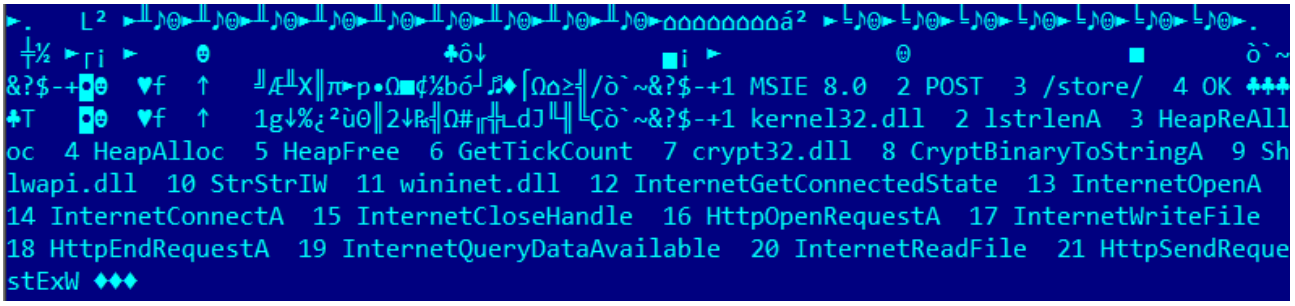
Interestingly, the fact that the attack was blocked didn’t appear to stop the Sofacy team. Just an hour and a half later they had compiled and delivered another AZZY x64 backdoor (md5: 9D2F9E19DB8C20DC0D20D50869C7A373, compiled August 4th, 2015). This was no longer detectable with static signatures by our product. However, it was detected dynamically by the host intrusion prevention subsystem when it appeared in the system and was executed.



This recurring, blindingly-fast Sofacy attack attracted our attention as neither sample was delivered through a zero-day vulnerability — instead, they appeared to be downloaded and installed by another malware. This separate malware was installed by an unknown attack as “**AppData\Local\Microsoft\Windows\msdeltemp.dll**” (md5: CE8B99DF8642C065B6AF43FDE1F786A3).

The top level malware, CE8B99DF8642C065B6AF43FDE1F786A3 (named by its authors “msdeltemp.dll” according to internal strings, and compiled July 28th, 2015) is a rare type of the Sofacy AZZY implant. It has been modified to drop a separate C&C helper, (md5: 8C4D896957C36EC4ABEB07B2802268B9) as “**tf394kv.dll**“.

The dropped “**tf394kv.dll**” file is an external C&C communications library, compiled on July 24th, 2015 and used by the main backdoor for all Internet-based communications.



*Decrypted configuration block of the C&C helper library "**tf394kv.dll**"*

This code modification marks an unusual departure from the typical AZZY backdoors, with its C&C communication functions moved to an external DLL file. In the past, the Sofacy developers modified earlier AZZY backdoors to use a C&C server encoded in the registry, instead of storing it in the malware itself, so this code modularisation follows the same line of thinking.

In addition to the new AZZY backdoors with side-DLL for C&C, we observed a new set of data-theft modules deployed against victims by the Sofacy group. Among the most popular modern defense mechanisms against APTs are air-gaps — isolated network segments without Internet access, where sensitive data is stored. In the past, we've seen groups such as Equation and Flame use malware to steal data from air-gapped networks. The Sofacy group uses such tools as well.

The first versions of these new USB stealer modules appeared around February 2015 and the latest appear to have been compiled in May 2015. Older versions of these USBSTEALER modules were previously described by our colleagues from ESET.

One example of the new Sofacy USBSTEALER modules is 8b238931a7f64fddcad3057a96855f6c, which is named internally as msdetltemp.dll.

Count of sections	5	Machine	Intel386
Symbol table	00000000[00000000]		Fri May 29 14:20:32 2015
Size of optional header	00E0	Magic optional header	010B
Linker version	11.00	OS version	5.01
Image version	0.00	Subsystem version	5.01
Entry point	00002E93	Size of code	00008800
Size of init data	0000C600	Size of uninit data	00000000
Size of image	00019000	Size of header	00000400
Base of code	00001000	Base of data	0000A000
Image base	10000000	Subsystem	Console
Section alignment	00001000	File alignment	00000200
Stack	00100000/00001000	Heap	00100000/00001000
Checksum	00000000	Number of dirs	16

This data theft module appears to have been compiled in May 2015 and is designed to watch removable drives and collect files from them, depending on a set of rules defined by the attackers. The stolen data is copied into a

hidden directory as "**%MYPICTURES%\%volume serial number%**", from where it can be exfiltrated by the attackers using one of the AZZY implants. More details on the new USB stealers are available in the section on technical analysis.

Conclusions

Over the last year, the Sofacy group has increased its activity almost tenfold when compared to previous years, becoming one of the most prolific, agile and dynamic threat actors in the arena. This activity spiked in July 2015, when the group dropped two completely new exploits, an Office and Java zero-day.

At the beginning of August, Sofacy began a new wave of attacks, focusing on defense-related targets. As of November 2015, this wave of attacks is ongoing. The attackers deploy a rare modification of the AZZY backdoor, which is used for the initial reconnaissance. Once a foothold is established, they try to upload more backdoors, USB stealers as well as other hacking tools such as "Mimikatz" for lateral movement.

> Over the last year, the #Sofacy group has increased its activity almost tenfold, that spiked in July 2015
>
> Tweet

Two recurring characteristics of the Sofacy group that we keep seeing in its attacks are speed and the use of multi-backdoor packages for extreme resilience. In the past, the group used droppers that installed both the SPLM and AZZY backdoors on the same machine. If one of them was detected, the other one provided the attacker with continued access.

As usual, the best defense against targeted attacks is a multi-layered approach. Combine traditional anti-malware technologies with patch management, host intrusion detection and, ideally, allowlisting and default-deny strategies. According to a study by the Australian DSD, 85% of the targeted attacks analysed could have been stopped by four simple defense strategies. While it's impossible to achieve 100% protection, in practice and most cases all you have to do is increase your defenses to the point where it becomes too expensive for the attacker – who will just give up and move on to other targets.

More information about the Sofacy group is available to customers of **Kaspersky Intelligent Services**.

Is there a 'silver bullet' to protect yourself against Sofacy? Learn more on Kaspersky Business blog.

Technical analysis

Internal name: DWN_DLL_MAIN.dll
File format: PE32 DLL
MD5: ce8b99df8642c065b6af43fde1f786a3
Linker version: 11.0, Microsoft Visual Studio
Linker timestamp: 2015.07.28 13:05:20 (GMT)
Exported functions:

- 10003F30: ?Applicate@@YGHXZ
- 10004270: ?SendDataToServer_2@@YGHPAEKEPAPAEPAK@Z

- 10003F60: ?k@@YGPAUHINSTANCE__@@PBD@Z

The library starts its main worker thread from the DllMain function.

Most of the strings inside the module are encrypted with a homebrew XOR-based algorithm. In addition to that, API function names are reversed, presumably to avoid detection in memory.

Once started, the code in the main thread resolves the basic API functions it needs and loads an additional library from the following location: **“%TEMP%\tf394kv.dll”**. If this file is not present, it is recreated from a hardcoded encrypted array inside the body of the DLL.

Next, the module enters an infinite loop. Every five minutes it collects basic system information and sends it to the C2 server:

- Windows version number
- Hardcoded string “4.3” (the backdoor’s internal version number)
- List of running processes

The main thread also spawns a separate thread for receiving new commands from the C2 servers. Every 10 minutes, it sends a new request to the server. The server is expected to send back executable code and one of the following commands:

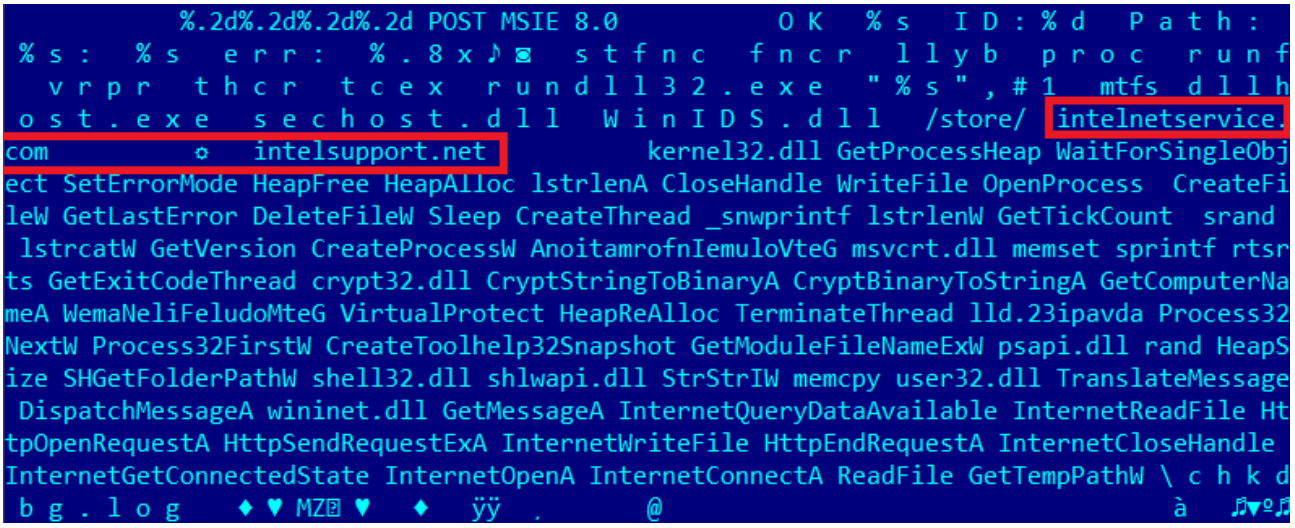
- Write a new file “%LOCAL_APPDATA%\dllhost.exe” or “%TEMP%\dllhost.exe” and execute it, then delete the file
- Write a new file “%LOCAL_APPDATA%\sechost.dll” or “%TEMP%\sechost.dll” and call its first exported function using “rundll32.exe” or Windows API, then delete the file
- Run shellcode provided by the server in a new thread

While processing the commands, the backdoor logs all errors and execution results. The module also reads the contents of the file “**%APPDATA%\chkdbg.log**” and appends it to the results. It then sends the aggregated log back to the C2 server.

The module aborts the thread receiving C2 command after it fails to correctly execute commands more than six times in a row, i.e. if file or process creation fails.

The export called “**k**” is a wrapper for the “LoadLibraryA” API function.

The export called “**SendDataToServer_2**” does exactly what the name means: it encrypts all collected data, encodes it using Base64 encoding and calls its additional library to send the data to the C2 server. The names of the C2 servers are hardcoded.



Hardcoded C&C servers in the main module

The two C&C's hardcoded in the configuration block of the main binary are:

- **intelnetservice[.]com**
- **intelsupport[.]net**

The export called "**Applicate**" runs a standard Windows application message loop until a "WM_ENDSESSION" message is received. It then terminates the main thread.

Internal name: snd.dll
File format: PE32 DLL
MD5: 8c4d896957c36ec4abeb07b2802268b9
Linker version: 11.0, Microsoft Visual Studio
Linker timestamp: 2015.07.24 12:07:27 (GMT)
Exported functions:

- 10001580: Init
- 10001620: InternetExchange
- 10001650: SendData

This external library implements a simple Wininet-based transport for the main module.

The strings inside the binary are encrypted using 3DES and XOR and reversed.

The DllMain function initializes the library and resolves all required Windows API functions.

The "**Init**" export establishes connection to port 80 of a C2 server using Wininet API. The user agent string employed is **"MSIE 8.0"**.

The "**SendData**" export sends a HTTP POST request using a hardcoded URI "**/store/**". The reply, if its length is not equal to six and its contents do not contain "OK" is returned back to the caller.

The "**InternetExchange**" export closes the established connection and frees associated handles.

Sofacy AZZY 4.3 dropper analysis

File format: PE32 EXE
File size: 142,336 bytes
MD5: c3ae4a37094ecfe95c2badecf40bf5bb
Linker version: 11.0, Microsoft Visual Studio
Linker timestamp: 2015.02.10 10:01:59 (GMT)

Most of the strings and data in the file are encrypted using 3DES and XOR.

The code makes use of the Windows Crypto API for 3DES and the decryption key is stored as a standard Windows PUBLICKEYSTRUC structure:

```
CryptAcquireContextA(&phProv, "{wdfr}", "Microsoft Enhanced Cryptographic Provider v1.0", 1u, 0x10u);
CryptAcquireContextA_0(&phProv, "{wdfr}", "Microsoft Enhanced Cryptographic Provider v1.0", 1, 16);
result = CryptAcquireContextA_0(&phProv, "{wdfr}", "Microsoft Enhanced Cryptographic Provider v1.0", 1, 8);
if ( result )
{
  result = CryptImportKey(phProv, v8, v21, 0, 0, &v22);
  if ( result )
  {
    v12 = *v20 % v10;
    v13 = 0;
    v14 = 0;
    v18 = *v20 % v10;
    v15 = *v20 / v10 + (v12 != 0) - 1;
    v21 = v15;
    v16 = v15 == 0;
    while ( 1 )
    {
      if ( !v16 || (v23 = v12) == 0 )
        v23 = v10;
      result = CryptDecrypt(v22, 0, v13 == v15, 0, v14 + *v19, &v23);
```

Part of the decryption algorithm

```
                db    0
encryptedBlob   db PLAINTEXTKEYBLOB     ; DATA XREF: XorOnDES+52↑o XorOnDES
                                        ; main+2FC↑o
                db CUR_BLOB_VERSION
                dw 0
                dd CALG_3DES
                dd 24
+               db 4Ch, 86h, 19h, 25h, 0B9h, 1, 0A4h, 75h, 67h, 0C7h, 51h
                db 0E5h, 32h, 0C8h, 3Eh, 0D6h, 0F2h, 34h, 9Eh, 0C4h, 0C7h
                db 25h, 15h, 0D6h
```

Header of one encrypted data buffer containing the hardcoded 3DES key

First, it creates a new directory: “%LOCAL_APPDATA%\Microsoft\Windows”. If the directory creation fails it tries to install into “%TEMP%” directory instead.

Next it writes a hardcoded binary from its body to **“msdeltemp.dll”** into the target directory**.** If the file exists it then moves it to **“__tmpdt.tmp”** in the same directory and continues the installation. Sets file creation timestamp to that of “%SYSTEM%\sfc.dll”

To ensure the dropped payload starts automatically on user log-in it creates the following registry key:

[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]

StartUpChekTemp=RUNDLL32.EXE "%path to msdeltemp.dll%",#1

Next, it starts the dropped dll using the same command line:

RUNDLL32.EXE *"%path to msdeltemp.dll%",#1*

Finally, the program removes itself by starting the following command: "cmd /c DEL *%path to self%*"

The MD5 of the dropped file is **f6f88caf49a3e32174387cacfa144a89**

Dropper payload – downloader DLL

Internal name: msdetltemp.dll
File format: PE32 DLL
File size: 73 728 bytes
MD5: f6f88caf49a3e32174387cacfa144a89
Linker version: 11.0, Microsoft Visual Studio
Linker timestamp: 2015.02.10 07:20:02 (GMT)
Exported functions:
10002B55: Applicate

Most of the strings inside the binary are encrypted using a homebrew XOR-based algorithm and reversed.

The library is an older version of the "DWN_DLL_MAIN.dll" (md5: ce8b99df8642c065b6af43fde1f786a3).

The DllMain function is identical and starts the main thread; the "**Applicate**" function is identical to the one in the newer library. This version of the module does not rely on an external transport DLL for communicating with its C2 servers; instead it directly uses Wininet API functions.

The module contains the following hardcoded C2 server names:

- **drivres-update[.]info**
- **softupdates[.]info**

The module uses a hardcoded URL ("**/check/**") for sending HTTP POST requests to its C2 servers.

The server is expected to send back executable code and one of the following commands:

- Write a new file "%LOCAL_APPDATA%\svchost.exe" or "%TEMP%\svchost.exe" and execute it, then delete the file
- Write a new file "%LOCAL_APPDATA%\conhost.dll" or "%TEMP%\conhost.dll" and call its first exported function using "rundll32.exe" or Windows API, then delete the file
- Run shellcode provided by the server in a new thread

File collection module ("USB Stealer")

Internal name: msdetltemp.dll (from resources)
File size: 50,176 bytes

File format: PE32 EXE
MD5: 0369620eb139c3875a62e36bb7abdae8
Linker version: 10.0, Microsoft Visual Studio
Linker timestamp: 2015.02.09 11:48:01 (GMT)

Most of the strings inside the binary are encrypted using 3DES and XOR and reversed.

The program creates the mutex “mtx” and an event named “WerTyQ34C”.

Upon start, it creates a window with the class name “Hello” and title “Program”, subscribes for device arrival notifications for that window and enters a standard Windows message processing loop. The message processing function waits for the WM_DEVICECHANGE event and starts a new thread on its arrival.

The module inspects every new disk volume attached to the system. For every disk it creates a “stash” directory in “*%root stash directory location%\%volume serial number in hex%”* with attributes FILE_ATTRIBUTE_HIDDEN and FILE_ATTRIBUTE_SYSTEM. The root stash directory location may be read from the configuration file or set to the default location which is “%MYPICTURES%”.

Then, it traverses the filesystem of the volume looking for files. Depending on the settings read from the configuration file, the scope may be narrowed to files with particular extensions and/or files created after a specified date.

It is worth noting that if the source file location contains the string “System Volume Information\S-1-5-21-1315235578-283289242\”, then the file is deleted after copying to the “stash” directory. There are known related samples that are able to create such directories on removable drives, i.e. the sample with md5: 8cb08140ddb00ac373d29d37657a03cc.

The configuration file is loaded from the same directory as the module and is expected to have a name “**NvCpld.dat**“. It is a text file that may contain the following configuration parameters:

- Path – location of the root “stash” directory
- Ext – search for files with one of these extensions only
- Date – search for files not earlier than this date

Internal name: NvCpld.dll (from export table), msdetltemp.dll (from resources), IGFSRVC.dll (from resources)
File format: PE32 DLL
File size: 76,288 bytes
MD5s: 8b238931a7f64fddcad3057a96855f6c, ce151285e8f0e7b2b90162ba171a4b90
Linker version: 11.0, Microsoft Visual Studio
Linker timestamps: 2015.05.29 11:20:32 (GMT), 2006.11.25 04:39:15 (GMT)
Exported functions:
10002500: NvMswt
10002860: NvReg
10002880: NvStart
10002A80: NvStop

This library is a newer version of the file collection module (md5: 0369620eb139c3875a62e36bb7abdae8) wrapped in a DLL file.

There are two known variants of this module; they only differ in timestamp values and version information in the resource section.

The DllMain function only decrypts the data structures and initializes Windows API pointers.

The function "**NvMswt**" is a wrapper for the API function MsgWaitForMultipleObjects.

The function "**NvReg**" is a wrapper for the API function RegisterClassW.

The function "**NvStart**" is similar to the main function of the older module; it creates a window and enters the message loop waiting for device arrival notifications. The only difference introduced is that an event named "**WerTyQ34C**" can be signalled by the function "**NvStop**" to terminate the message loop and stop processing.

Indicators of compromise:

AZZY 4.3 installer:

c3ae4a37094ecfe95c2badecf40bf5bb

New generation (4.3) AZZY implants:

ce8b99df8642c065b6af43fde1f786a3
f6f88caf49a3e32174387cacfa144a89

Dropped C&C helper DLL for AZZY 4.3:

8c4d896957c36ec4abeb07b2802268b9

File collectors / USB stealers:

0369620eb139c3875a62e36bb7abdae8
8b238931a7f64fddcad3057a96855f6c
ce151285e8f0e7b2b90162ba171a4b90
f6f88caf49a3e32174387cacfa144a89

Stand-alone AZZY backdoors:

a96f4b8ac7aa9dbf4624424b7602d4f7
9d2f9e19db8c20dc0d20d50869c7a373

C&C hostnames:

- drivres-update[.]info
- intelnetservice[.]com

- intelsupport[.]net
- softupdates[.]info

Kaspersky Lab products detect the malware mentioned here with the following names:

- Trojan.Win32.Sofacy.al
- Trojan.Win32.Sofacy.be
- Trojan.Win32.Sofacy.bf
- Trojan.Win32.Sofacy.bg
- Trojan.Win32.Sofacy.bi
- Trojan.Win32.Sofacy.bj
- Trojan.Win64.Sofacy.q
- Trojan.Win64.Sofacy.s
- HEUR:Trojan.Win32.Generic
