Dissecting Ardamax Keylogger » TrainSec

By Uriel Kosayev

Published: 2024-07-04 · Archived: 2026-04-06 00:28:48 UTC

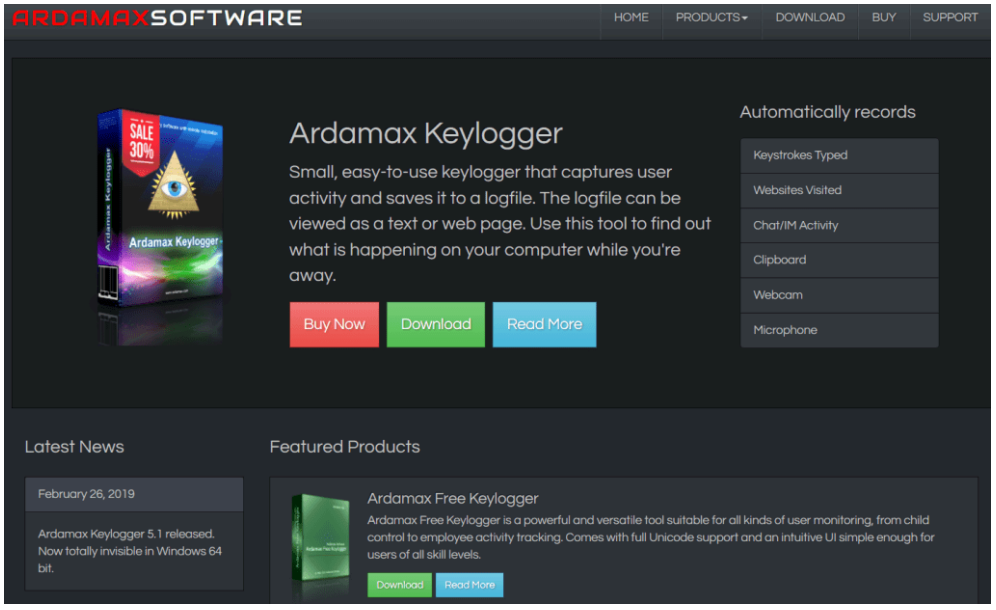
Last month, we decided to enrich our knowledge by delving into research on a popular hacking tool. We decided to go with one that was only covered in the past. It is called – Ardamax Keylogger. In this blog post, we present the methods and operations analyzed, including key capabilities of the infection mechanism. We will also cover what data is being collected and how. In addition to the infection flow, we discovered a vulnerability in the Ardamax Keylogger that may allow attackers to exploit the keylogger's DLL loading mechanism.

Liked the content?

Subscribe to the free TrainSec knowledge library, and get insider access to new content, discounts and additional materials.

Executive Summary

The Ardamax Keylogger developers have an official website that gives users the option to buy their product or only use it for a "test drive".



We tried to find out when they first started and the oldest sample we were able to gather was generated somewhere around the year of 2013. In addition, these old versions are easily detected by existing AV engines. However, newer versions are still in question. From a quick overview of samples uploaded to VirusTotal, we noticed that the developers successfully evaded detection in most cases. The detection rates are not that great compared to the fact that Ardamax is a "noisy" keylogger with an extensive resume in the wild.

We've analyzed dozens of samples, from all versions we were able to find, and discovered that the vast majority of them were vulnerable to the flaw we found.

Kill Chain

The infection kill chain comprises the following steps:

1. Execution of the dropper **Ardamax.exe,** which drops several files, including a randomly named DLL to the **%temp%** folder.

2. The malicious process **Ardamax.exe** loads the dropped DLL that is used to drop the keylogger files under a hidden folder in the system folder.

3. Finally, the keylogger **DPBJ.exe** is executed, logging keystrokes and capturing screenshots.

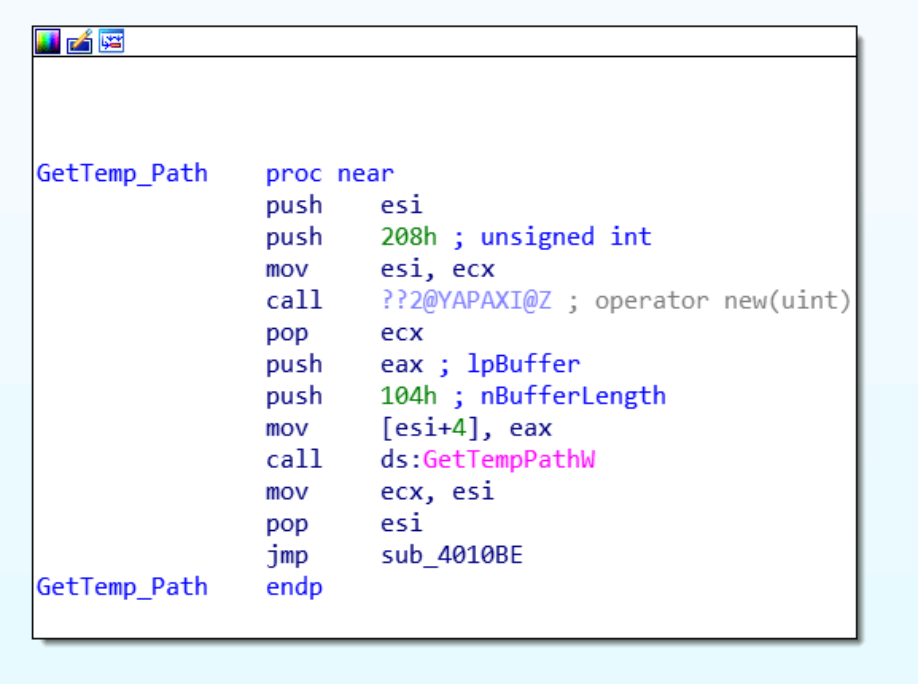
Ardamax Dropper

Filename	Ardamax.exe
Size	784 KB
MD5	E33AF9E602CBB7AC3634C2608150DD18
SHA1	8F6EC9BC137822BC1DDF439C35FEDC3B847CE3FE

Once a victim launches the dropper, Ardamax.exe executes its initial routine **GetTemp_Path (sub_401230)**, which obtains the Windows %temp% path for later use, as shown in the following screenshot:

```
push    ebp
mov     ebp, esp
sub     esp, 3Ch
push    ebx
push    esi
mov     esi, offset off_4050B4
mov     ecx, esi
call    GetTemp_Path
mov     ebx, eax
test    ebx, ebx
jz      short loc_401479
```

By looking at the disassembly output from IDA, the **GetTemp_Path** function calls **GetTempPathW** to retrieve the system's temporary folder.



```
GetTemp_Path    proc near
                push    esi
                push    208h ; unsigned int
                mov     esi, ecx
                call    ??2@YAPAXI@Z ; operator new(uint)
                pop     ecx
                push    eax ; lpBuffer
                push    104h ; nBufferLength
                mov     [esi+4], eax
                call    ds:GetTempPathW
                mov     ecx, esi
                pop     esi
                jmp     sub_4010BE
GetTemp_Path    endp
```

The next routine calls **CreateFileW**. In this routine, Ardamax drops several files to the temp folder, including a randomly named DLL, as we mentioned earlier.

```
push    edi ; hTemplateFile
push    80h ; dwFlagsAndAttributes
push    5 ; dwCreationDisposition
push    edi ; lpSecurityAttributes
push    1 ; dwShareMode
mov     [esi+8], eax
push    40000000h ; dwDesiredAccess
lea     eax, [ebp+Filename]
push    eax ; lpFileName
call    ebx ; CreateFileW
```

Using Process Monitor, it is possible to see the randomly named DLL being copied into the folder. In this case, the DLL filename is **@F9CD.tmp**:

Time ...	Process Name	PID	Operation	Path	Result	Detail
9:58:5...	DPBJ.exe	2860	CreateFile	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	Desired Access: R...
9:58:5...	DPBJ.exe	2860	QueryBasicInformation...	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	CreationTime: 29/0...
9:58:5...	DPBJ.exe	2860	CloseFile	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	
9:58:5...	DPBJ.exe	2860	CreateFile	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	Desired Access: R...
9:58:5...	DPBJ.exe	2860	CreateFileMapping	C:\Users\nir\Desktop\28463\DPBJ.006	FILE LOCKED WI...	SyncType: SyncTy...
9:58:5...	DPBJ.exe	2860	CreateFileMapping	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	SyncType: SyncTy...
9:58:5...	DPBJ.exe	2860	Load Image	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	Image Base: 0x100...
9:58:5...	DPBJ.exe	2860	QueryNameInformation...	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	Name: \Users\nir\...
9:58:5...	DPBJ.exe	2860	CreateFile	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	Desired Access: G...
9:58:5...	DPBJ.exe	2860	CloseFile	C:\Users\nir\Desktop\28463\DPBJ.006	SUCCESS	

Ardamax DLL

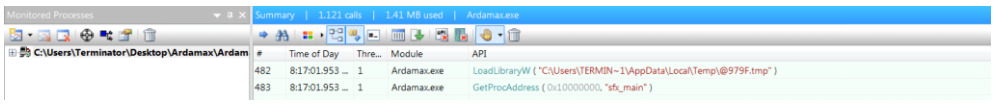
Filename	**[rand_name].tmp**
Size	**4 KB**
MD5	**D73D89B1EA433724795B3D2B524F596C**
SHA1	**213514F48ECE9F074266B122EE2D06E842871C8C**

Ardamax loads the randomly named DLL using **LoadLibraryW**.

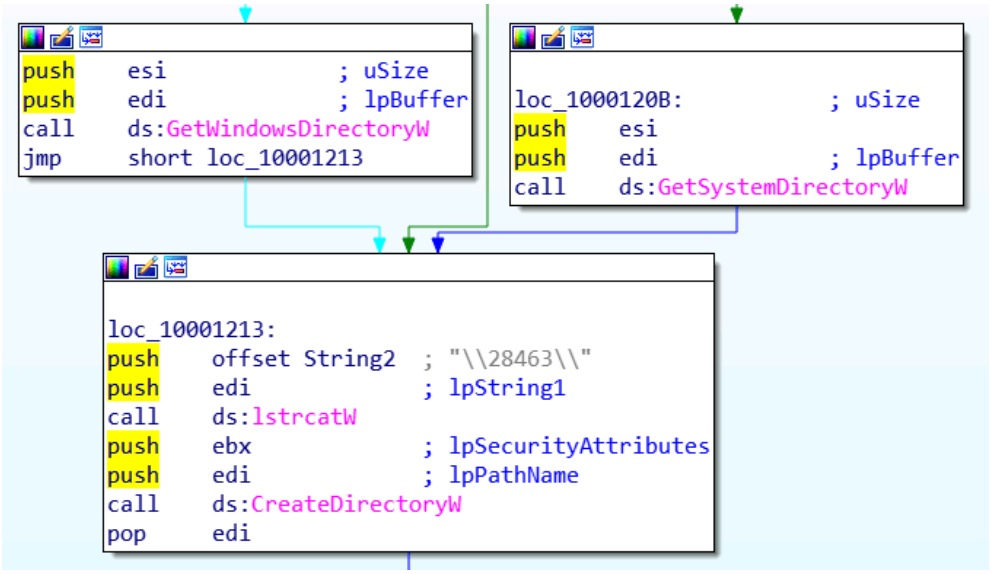
If the DLL load is successful, the dropper will call **GetProcAddress** to get the DLL's **sfx_main** address.

```
sfx_main_proc_addr proc near
                push    esi
                mov     esi, ecx
                push    dword ptr [esi+8] ; lpLibFileName
                call    ds:LoadLibraryW
                mov     [esi+10h], eax
                test    eax, eax
                jz      short loc_401024

                push    offset ProcName ; "sfx_main"
                push    eax ; hModule
                call    ds:GetProcAddress
                push    esi
                call    eax ; Calls the sfx_main function
                pop     esi
                retn

loc_401024:
                push    0FFFFFF9Fh
                pop     eax
                pop     esi
                retn
sfx_main_proc_addr endp
```

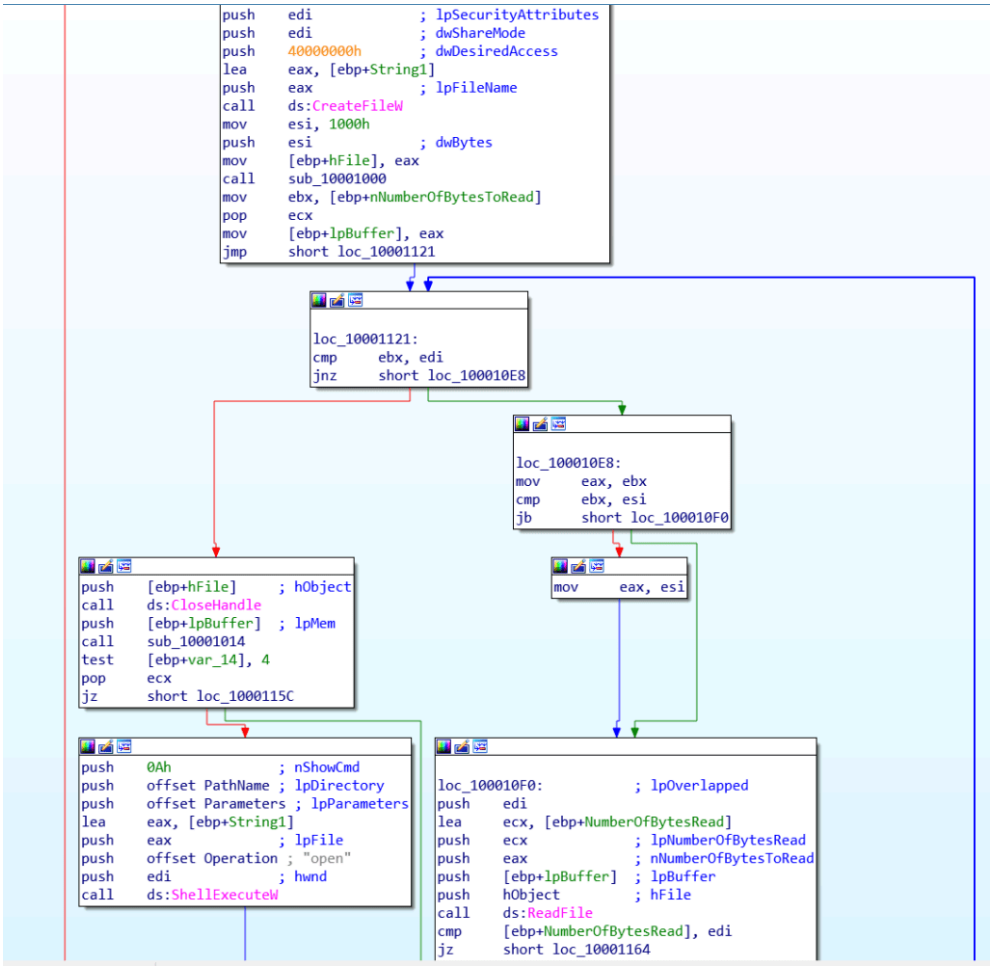
The following screenshot illustrates the operation above in a dynamic execution flow:

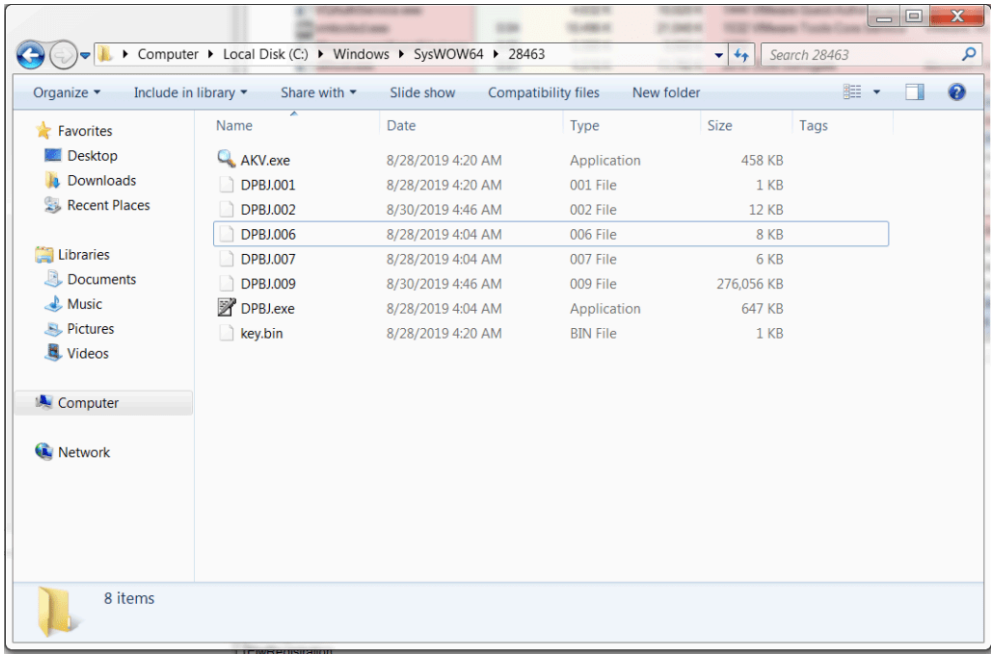


Next, the dropper gets a string containing the system's main directory, which is either **System32** or **SysWow64,** depending on the system architecture. Then, according to a hardcoded string, the dropper creates a hidden folder with the hardcoded name "**28463**" and copies several files into it. The batch of files will include **DPBJ.exe**, which is the actual keylogger.



Once all files are copied to the designated folder, the main file of the keylogger, dubbed **DPBJ.exe** is executed with **ShellExecuteW**:

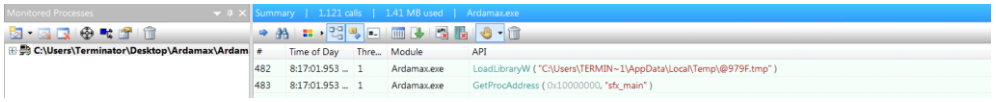




The list includes the following files:

- DPBJ.006 & DPBJ.007 – DLL files that are loaded by DPBJ.exe at runtime.
- DPBJ.exe – Keylogger's main executable.
- key.bin – Keylogger's license serial key (Still in research).
- Other files that didn't seem interesting for our research purposes.

Using Process Monitor again, it is possible to track the behavior of the hacking tool and its use of the hidden folder. The screenshot below illustrates how the "DPBJ.exe" file is being called for execution:

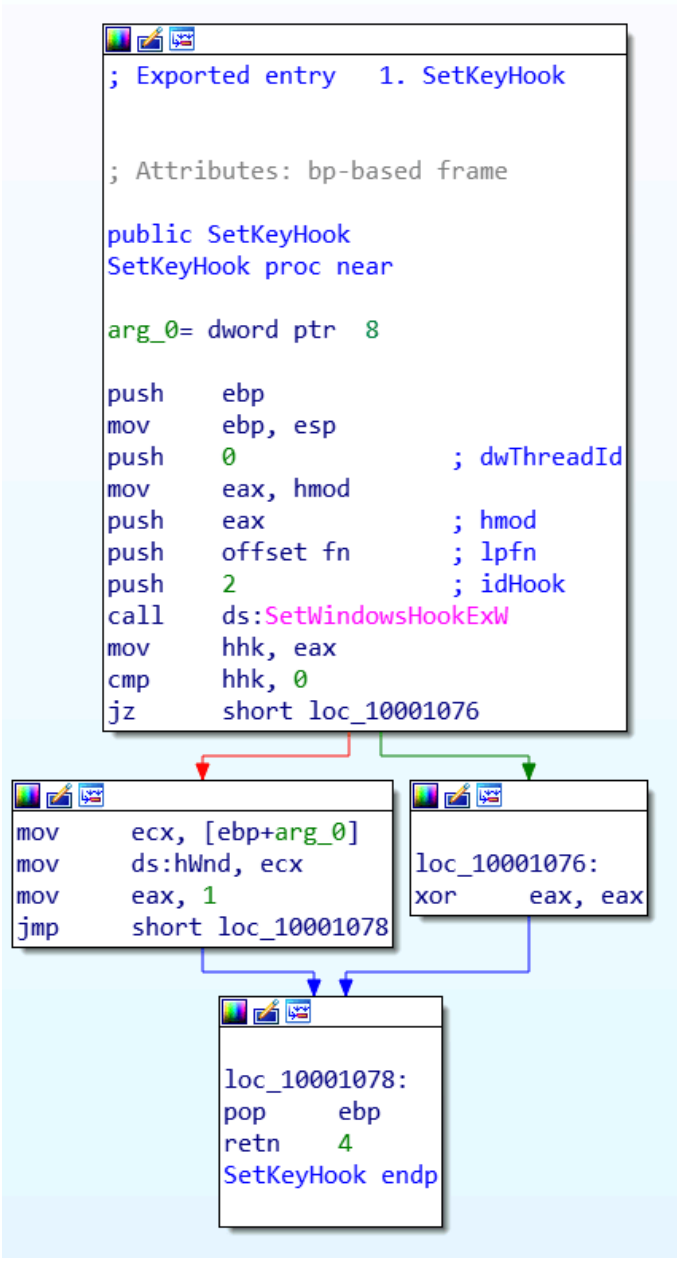


Ardamax Keylogger

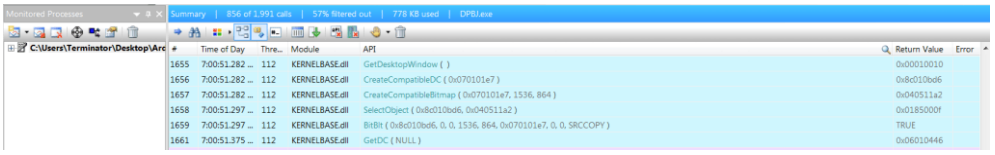
Filename	[rand_name].tmp
Size	646.5 KB
MD5	B863A9AC3BCDCDE2FD7408944D5BF976
SHA1	4BD106CD9AEFDF2B51F91079760855E04F73F3B0

After the keylogger is executed, it starts to collect the victim's keystrokes and screenshots.

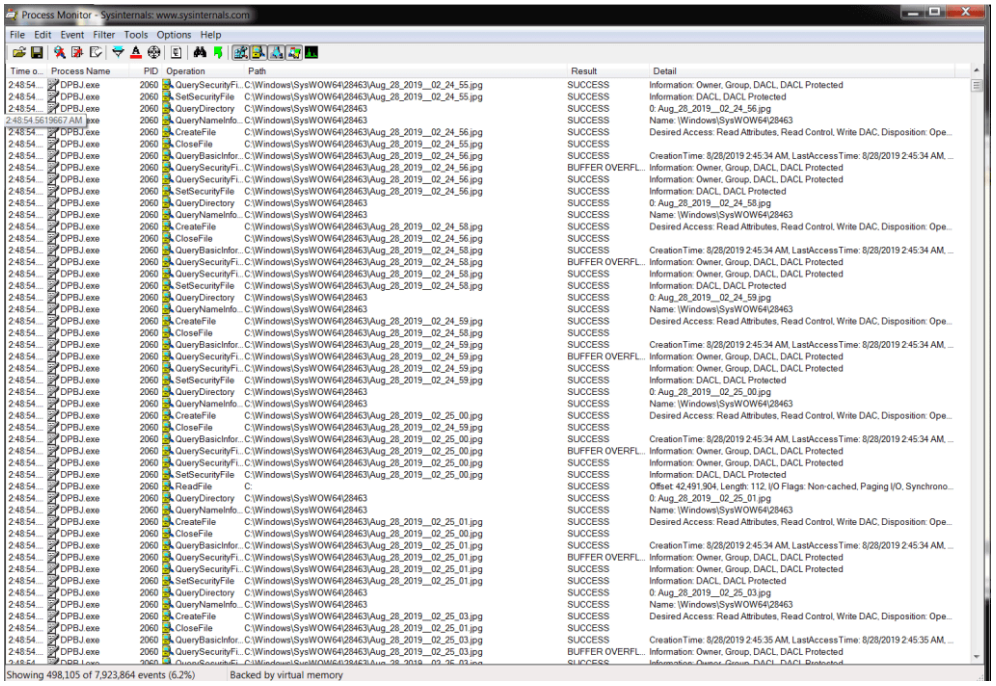
In the following routine, the **SetWindowsHookEx** function is being utilized with idHook of 2 (WH_KEYBOARD) that handles keystroke events and thus logs them:



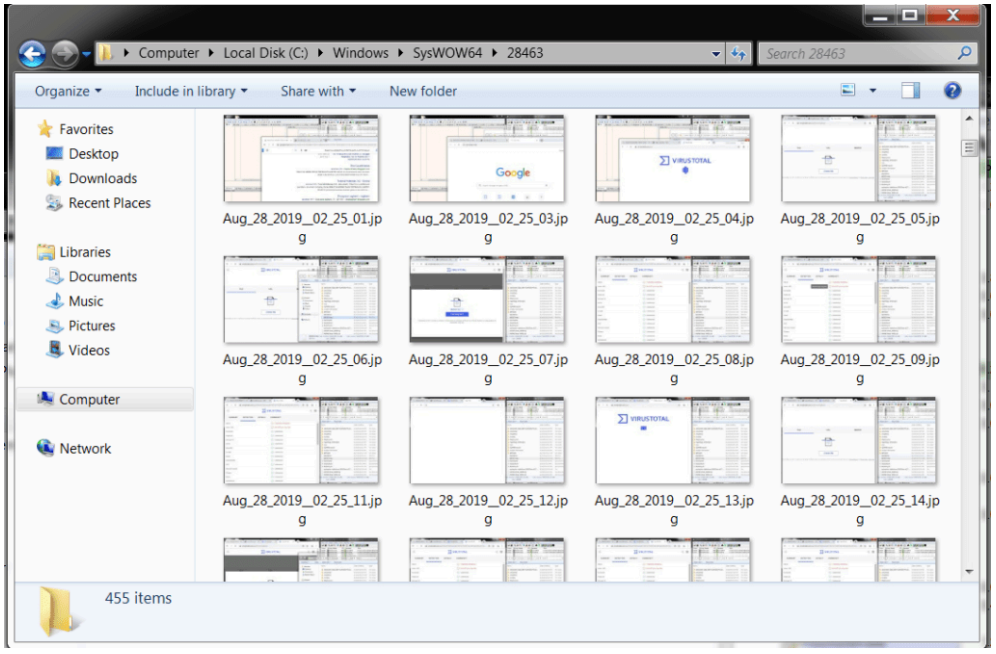
Below, we can see the use of several WinAPI functions to capture screenshots at runtime:



Next, we see that the captured screenshots are stored under **C:\Windows\SysWOW64\28463** with the naming format of **[Date_Hour].jpg**:



Here is an example of how the screenshots are being stored under the hidden folder discussed earlier:



The interesting part here is that the keylogger (DPBJ.exe) loads *DPBJ.006* and DPBJ.007, DLLs that call the WinAPI functions below:

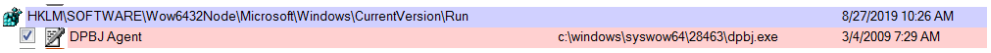
- **SetWindowsHookEx –** Keystrokes logging
- **GetDesktopWindow –** Screenshot taking and more

This can, in some way, make a researcher's job harder while using dynamic tools, such as ProcMon, that have limited insight into the process's behavior. Furthermore, it can also fool some sandbox solutions that try to intercept system calls to understand the malware's behavior.

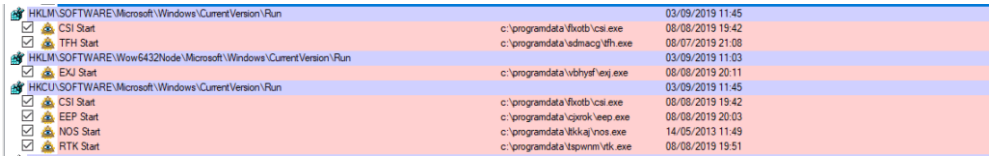
Also, we can see, the "DPBJ Agent" persistency login object is created under the following registry element:

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

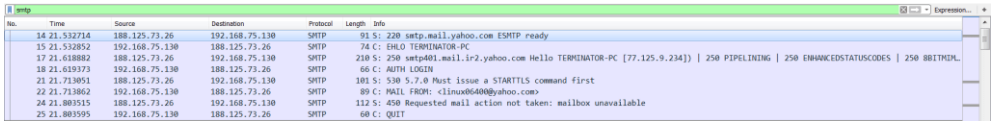
Using Autorun, it is possible to exhibit the Agent's registry path and the executable location:



Note: New versions of the Ardamax keylogger have the same behavior in the system, only with a different persistence path. It drops the keylogger files into a randomly named folder under the %ProgramData% folder, which is also a hidden folder by default:

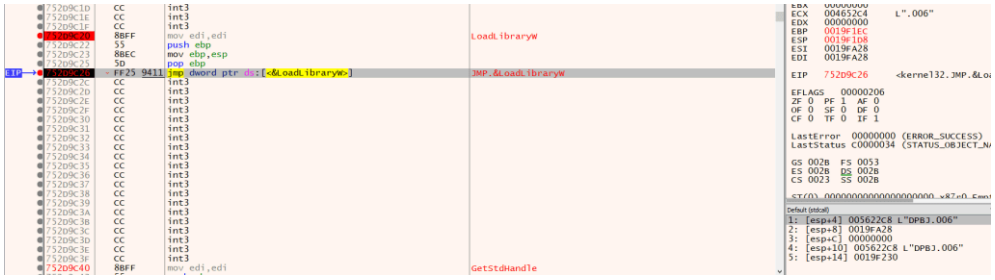


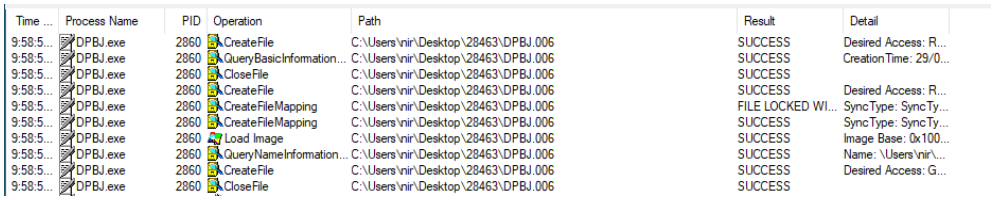
Eventually, the keylogger tries to communicate with a Yahoo-based SMTP server but the mailbox is unavailable:



Keylogger Exploitation – DLL Hijacking

As we wrote earlier, “DPBJ.exe” is loading DLLs with **LoadLibraryW**. In other words, it is looking for a specific DLL file name (in this case DPBJ.006 and DPBJ.007) to load them:

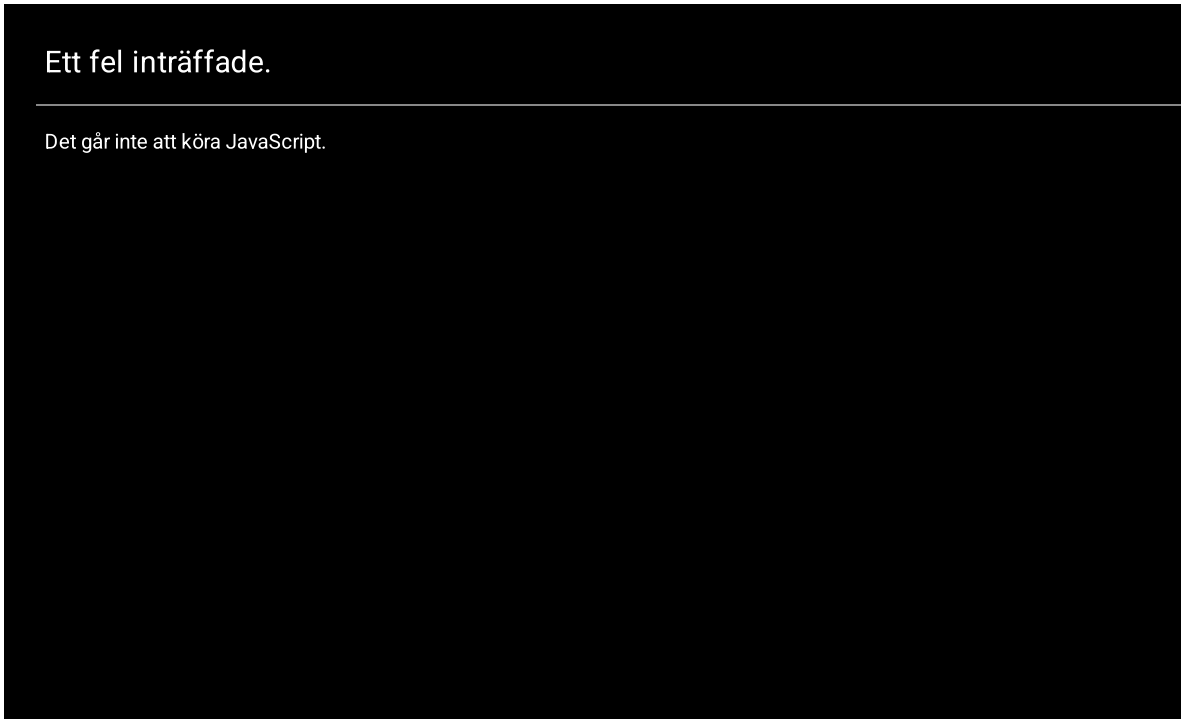




Vulnerability

A lack of validation in the call to **LoadLibraryW** creates a possible backdoor for the generated executable. It allows for loading a DLL only based on its name, hence any third-party attacker can abuse this mechanism by crafting their own malicious DLL and replace it with the exact name (“DPBJ.006”, in this example). In conclusion, when the DPBJ.exe is executed, it’ll load the replaced attacker’s crafted DLL.

To make it even more visual for you, we generated a PoC video where we show a live detonation of the keylogger to get a reverse TCP shell on the victim system:



Note: New versions of the Ardamax keylogger are also vulnerable to this attack including the latest version of 5.1 which was released in February 2019.

Conclusion

We saw that the Ardamax Keylogger has existed in the wild for over 6 years now. Although it's been in the market for so long, secure coding practices were not part of the process, hence it creates more threats for infected victims. In addition, we saw a rather simple flow with features that are not unique enough to rate Ardamax as a strong offensive tool, but enough to evade some detections.

Indicators of Compromise

- Ardamax.exe – md5: E33AF9E602CBB7AC3634C2608150DD18
- [Rand Name].tmp – md5: D73D89B1EA433724795B3D2B524F596C
- DPBJ.exe – md5: B863A9AC3BCDCDE2FD7408944D5BF976

Related samples created recently

cfd015112356dba7c4e81a6449e37e3d

a2b833052cb2743ec60f422f0e7bc185

340b5cc3eb29cdabc4e9647dddc7dea6

b0400b1bf445f8ad5aa978212b04ab94

ed866bf88b059caa4f73211ee62685ab

21fd08a181b865e9b34db69590056dab

83d597b8db70ccf56528a96c1aa48a22

0ea2a4502f86f58fde206f7fe2f8d084

d983a4f16933d3a9cee74283b7a5514b

a258f4a843decccf0e14026b8af4ffb3

c9c546f94488025839760d02514b979b

0f94f7c7ee0e1966c1f3eea4b22a61fe

73a3c61a7272485b9826fac769f0e95d

6cbe9e7fd502e785b43c3e2ba1e66b7f

6bf292ef5e463a40edc9ab4008c242b7

4a57ce1565f05454e9b5a4a80d048865

352e021537a6edfb9d5fa10084d43c96

d49b103ecc47c5e619594dc9e623ff02

568d70bc8109785fb50a92dbf1c173cc

68a7db6168393d289982d3935b8e0f53
