ATM malware targets Wincor and Diebold ATMs

insights.oem.avira.com/atm-malware-targets-wincor-and-diebold-atms/

April 30, 2020

Cyber criminals continue to target cash points (ATMs) causing significant loss to the banking and financial industry. They target the vulnerabilities and security flaws found in the outdated (and unpatched) operating systems still used in many Automated Teller Machines, stealing financial information and dispensing large amounts of unauthorised cash.

Anatoly Kazantsev, specialist threat researcher at Avira Protection Labs, recently found an ATM malware in the wild that specifically targets Wincor and Diebold ATM machines. Further investigation of these samples suggests that the campaigns originated from Portuguese-speaking countries.

In this blog, we will analyze the ATM malware samples that were recently found.

Malware sample analysis

SHA256: *7cea6510434f2c8f28c9dbada7973449bb1f844cfe589cdc103c9946c2673036*

This sample uses CEN/XFS API to communicate with the ATM cash dispenser.

The CEN/XFS standard defines a common language to speak with the hardware components of the ATM. It shares a common conceptual background with Microsoft Windows device drivers. XFS stands for eXtension for Financial Services and has the following architecture:

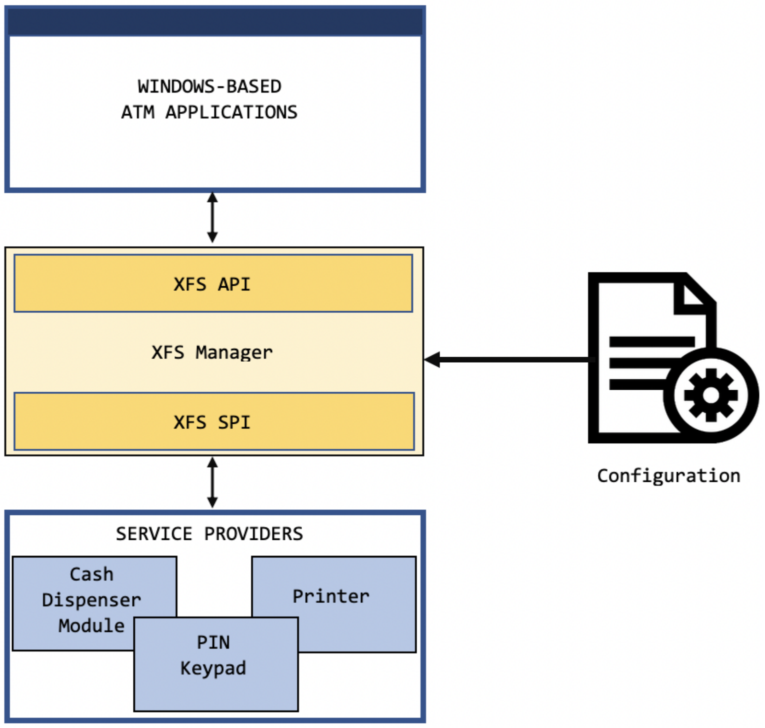
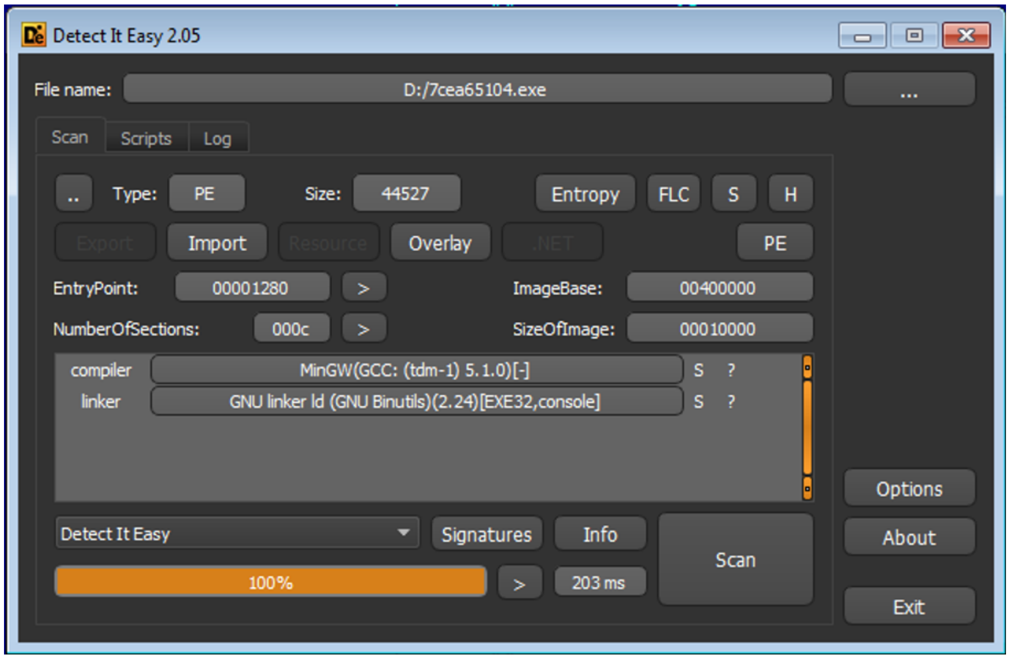


figure 1: XFS architecture

The ATM malware sample is written in C language and compiled using a MinGW GCC compiler as a 32-bit console application:



File name: D:/7cea65104.exe

Type: PE   Size: 44527

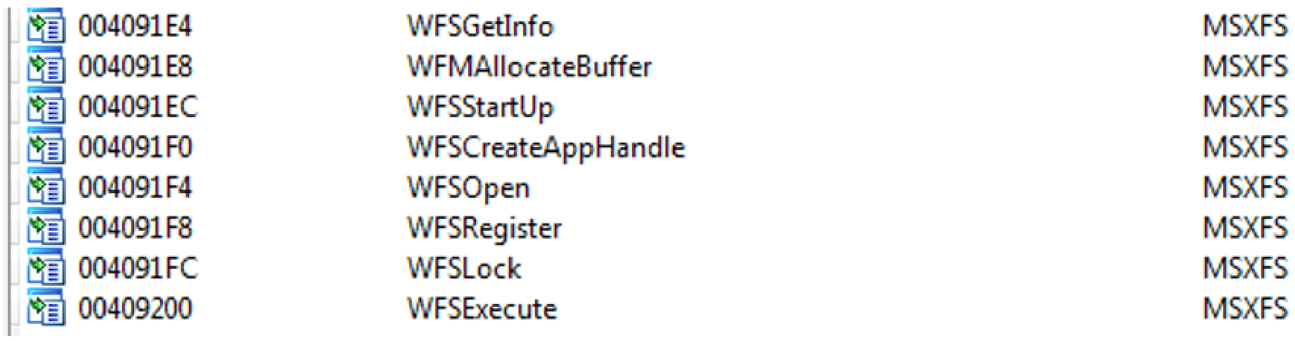
EntryPoint: 00001280   ImageBase: 00400000

NumberOfSections: 000c   SizeOfImage: 00010000

```
compiler   MinGW(GCC: (tdm-1) 5.1.0)[-]                       S  ?
linker     GNU linker ld (GNU Binutils)(2.24)[EXE32,console]  S  ?
```

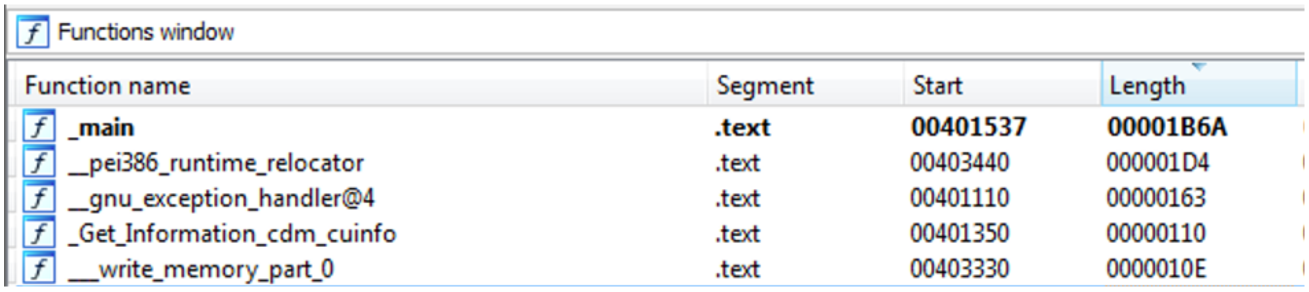
100%   203 ms

For further analysis, we upload the malware sample into IDA Pro and import MSXFS.dll library that implements XFS API functions:



004091E4	WFSGetInfo	MSXFS
004091E8	WFMAllocateBuffer	MSXFS
004091EC	WFSStartUp	MSXFS
004091F0	WFSCreateAppHandle	MSXFS
004091F4	WFSOpen	MSXFS
004091F8	WFSRegister	MSXFS
004091FC	WFSLock	MSXFS
00409200	WFSExecute	MSXFS

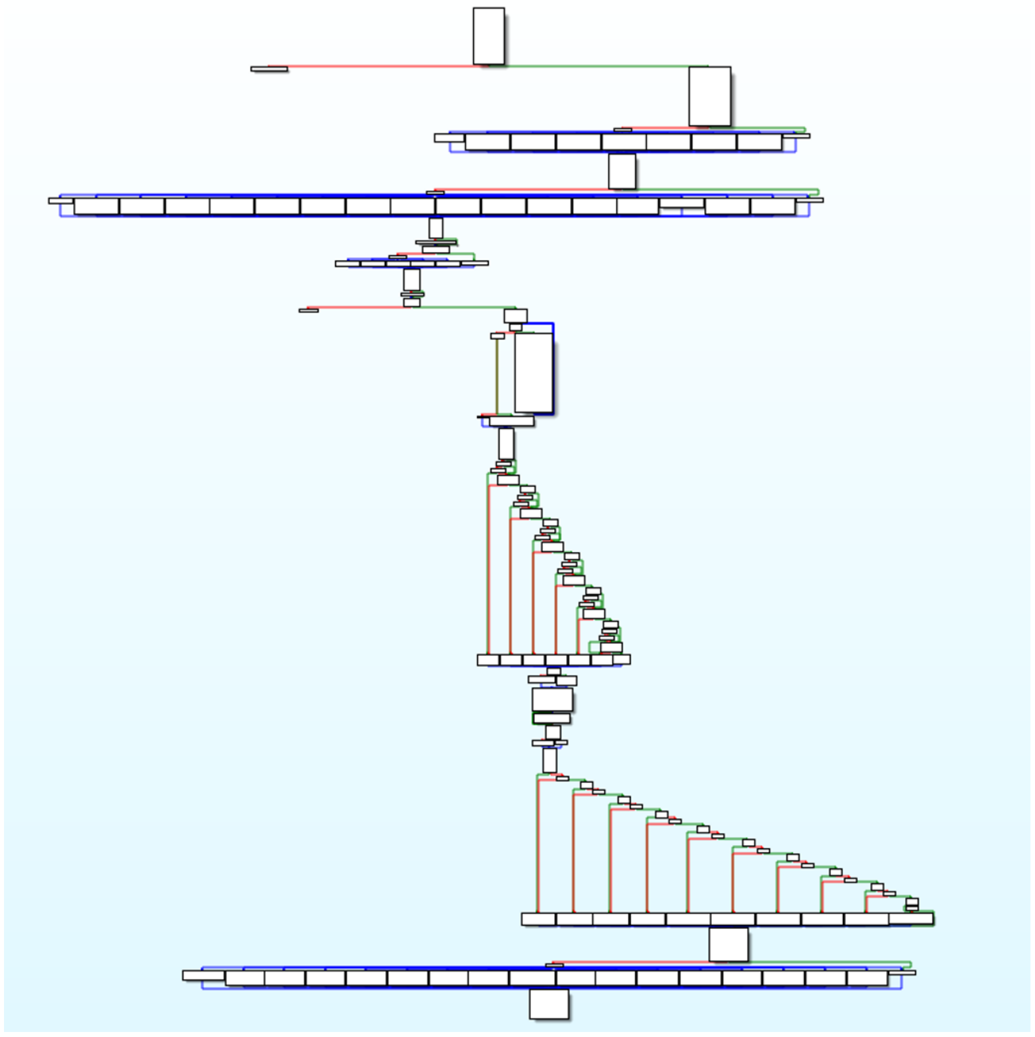
These functions are called from **_main** function. In general, the malware logic is straightforward and mostly implemented inside the **_main** function. The function is the biggest part of the sample's executable code:



Functions window

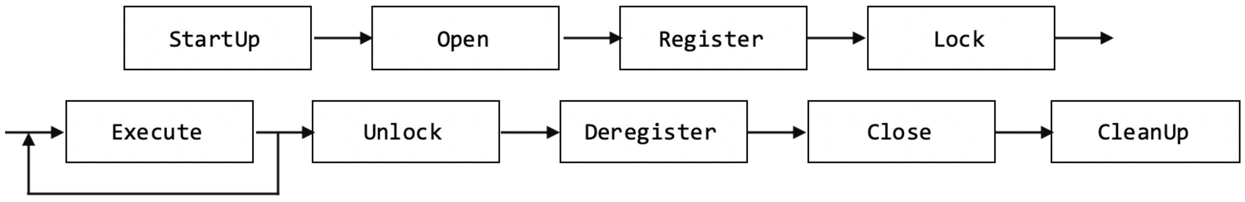
Function name	Segment	Start	Length
_main	**.text**	**00401537**	**00001B6A**
__pei386_runtime_relocator	.text	00403440	000001D4
__gnu_exception_handler@4	.text	00401110	00000163
_Get_Information_cdm_cuinfo	.text	00401350	00000110
___write_memory_part_0	.text	00403330	0000010E

The function's graph (a few *if-else* and *switch-case* constructions):



These if-else and switch-case constructions handle the return values and print error or information messages to a console. We will not describe **_main** function in detail, only concentrate on WFS* functions call and its parameters.

Now, let's review a typical XFS API usage scenario and map malware's behavior to this workflow:

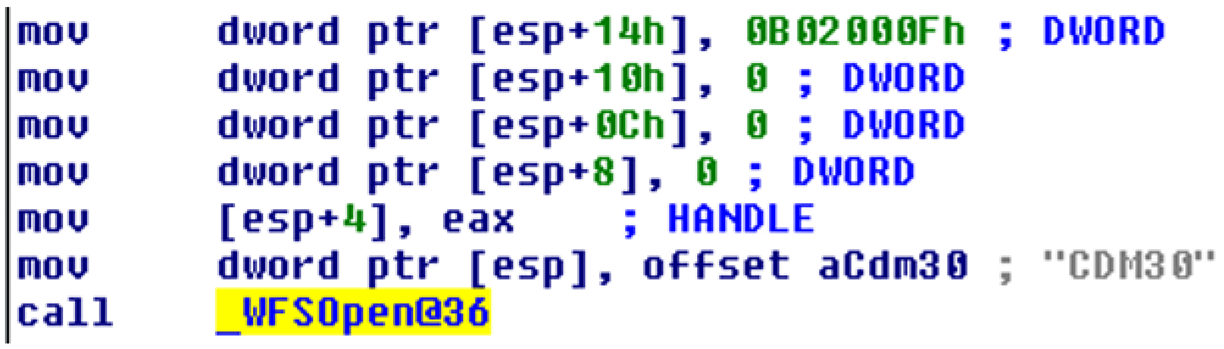


As we can see from the import table, the malware doesn't care about clean-up after itself. It only uses necessary functions (*marked bold in the below table*) to execute a specific command.

Action	Function name	Description
StartUp	**WFSStartUp**	Connects to XFS Manager
Open	**WFSOpen**	Establishes a session between an application and the Service Provider
Register	**WFSRegister**	Enable monitoring of a class of service events by an application
Lock	**WFSLock**	Establish exclusive control by an application of a service
Execute	**WFSExecute**	Send service-specific commands to a Service Provider
Unlock	WFSUnlock	Release exclusive control by an application of a service
Deregister	WFSDeregister	Disable monitoring of a class of service events by an application
Close	WFSClose	Close a session between an application and the Service Provider
CleanUp	WFSCleanUp	Terminate a connection between an application and the XFS Manager

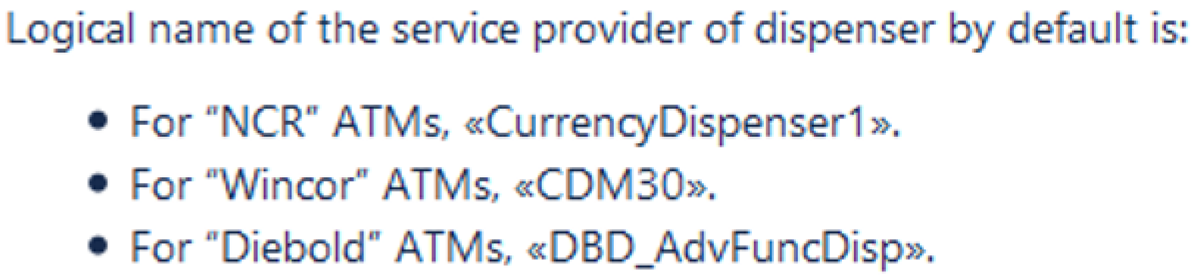
Let's take a closer at the most important functions to understand malware goals:

First, WFSOpen function takes its first argument the string "**CDM30**":



```
mov     dword ptr [esp+14h], 0B02000Fh ; DWORD
mov     dword ptr [esp+10h], 0 ; DWORD
mov     dword ptr [esp+0Ch], 0 ; DWORD
mov     dword ptr [esp+8], 0 ; DWORD
mov     [esp+4], eax     ; HANDLE
mov     dword ptr [esp], offset aCdm30 ; "CDM30"
call    _WFSOpen@36
```

A quick Google search gives us the meaning of the string:

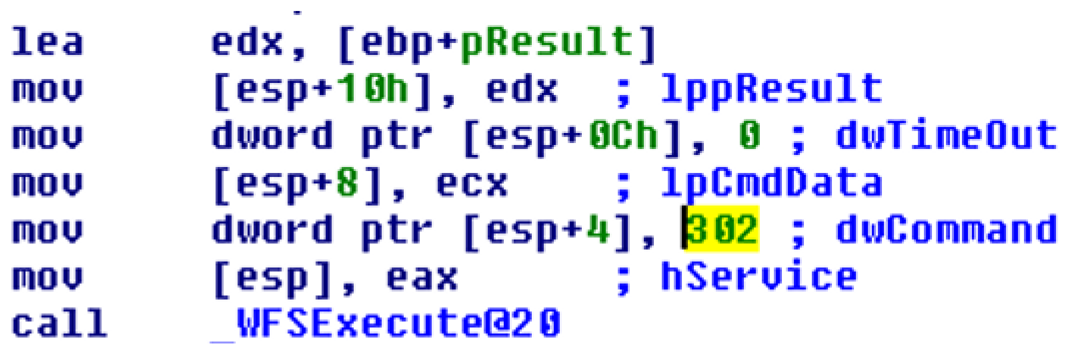


Logical name of the service provider of dispenser by default is:

- For "NCR" ATMs, «CurrencyDispenser1».
- For "Wincor" ATMs, «CDM30».
- For "Diebold" ATMs, «DBD_AdvFuncDisp».

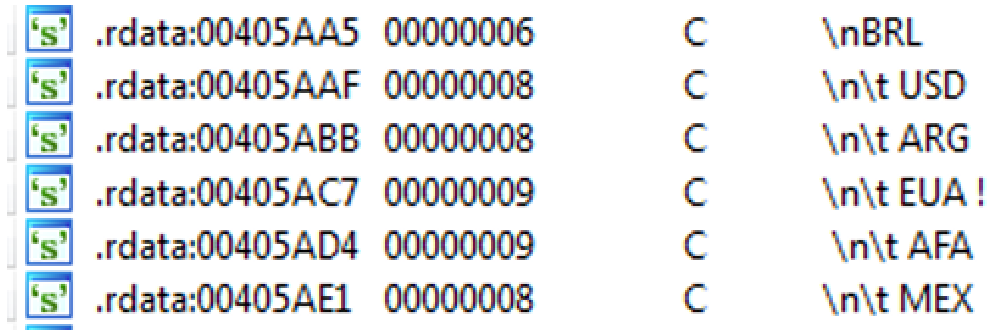
Now we know that this particular sample targets Wincor ATMs and establishes a session with Cash Dispenser Module (CDM) service provider.

The next interesting XFS API call is WFSExecute. This function takes 5 arguments. Two of them define command ID and command data – *dwCommand* and *lpCmdData* accordingly. In our case, *dwCommand* equals 302 and it is WFS_CMD_CDM_DISPENSE command ID:



```
lea     edx, [ebp+pResult]
mov     [esp+10h], edx  ; lppResult
mov     dword ptr [esp+0Ch], 0 ; dwTimeOut
mov     [esp+8], ecx    ; lpCmdData
mov     dword ptr [esp+4], 302 ; dwCommand
mov     [esp], eax      ; hService
call    _WFSExecute@20
```

Command name describes itself – malware tries to dispense some cash. Amount and currency are defined in a special structure *lpCmdData.* Inside the **_main** function there are some interesting string:



```
.rdata:00405AA5  00000006  C  \nBRL
.rdata:00405AAF  00000008  C  \n\t USD
.rdata:00405ABB  00000008  C  \n\t ARG
.rdata:00405AC7  00000009  C  \n\t EUA !
.rdata:00405AD4  00000009  C   \n\t AFA
.rdata:00405AE1  00000008  C  \n\t MEX
```

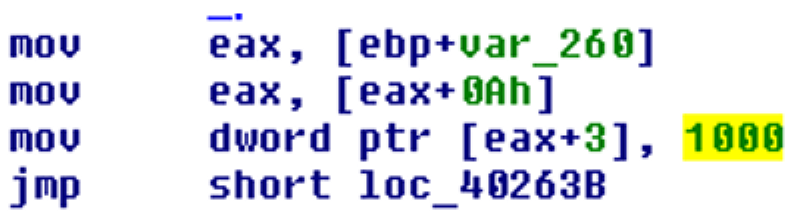
According to ISO4217 (*https://en.wikipedia.org/wiki/ISO_4217*):

- BRL – Brazilian real
- USD – United States dollar
- ARG – seems malware authors meant ARS (Argentine peso)
- EUA – Euro (EUR)?

- AFA – Afghan afghani (AFN)?
- MEX – seems malware authors meant MXN (Mexican peso)

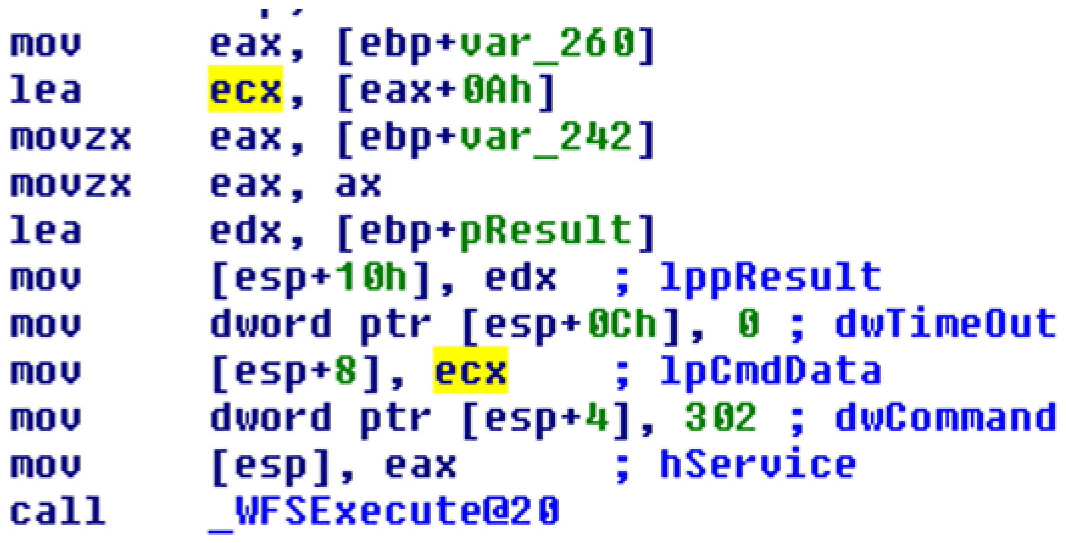
If the malware didn't find one of the above currency code, it will use a value returned by WFSGetInfo call.

To set an amount of money, malware fills a field of special structure (var_260) with value 1000:



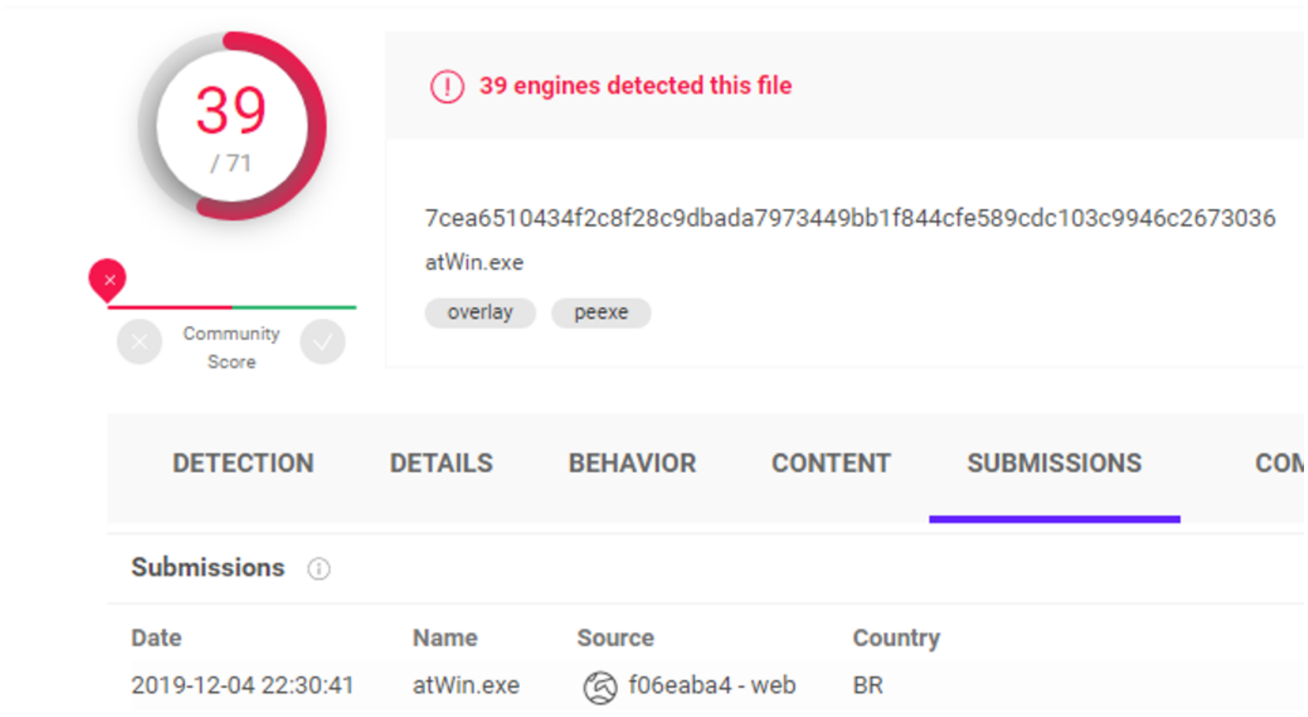
```
mov     eax, [ebp+var_260]
mov     eax, [eax+0Ah]
mov     dword ptr [eax+3], 1000
jmp     short loc_40263B
```

After some manipulations this structure is passed as *lpCmdData* argument to WFSExecute function:



```
mov     eax, [ebp+var_260]
lea     ecx, [eax+0Ah]
movzx   eax, [ebp+var_242]
movzx   eax, ax
lea     edx, [ebp+pResult]
mov     [esp+10h], edx  ; lppResult
mov     dword ptr [esp+0Ch], 0 ; dwTimeOut
mov     [esp+8], ecx    ; lpCmdData
mov     dword ptr [esp+4], 302 ; dwCommand
mov     [esp], eax      ; hService
call    _WFSExecute@20
```

It is interesting to see that the first upload to VirusTotal originated from Brazil:


39 engines detected this file

7cea6510434f2c8f28c9dbada7973449bb1f844cfe589cdc103c9946c2673036

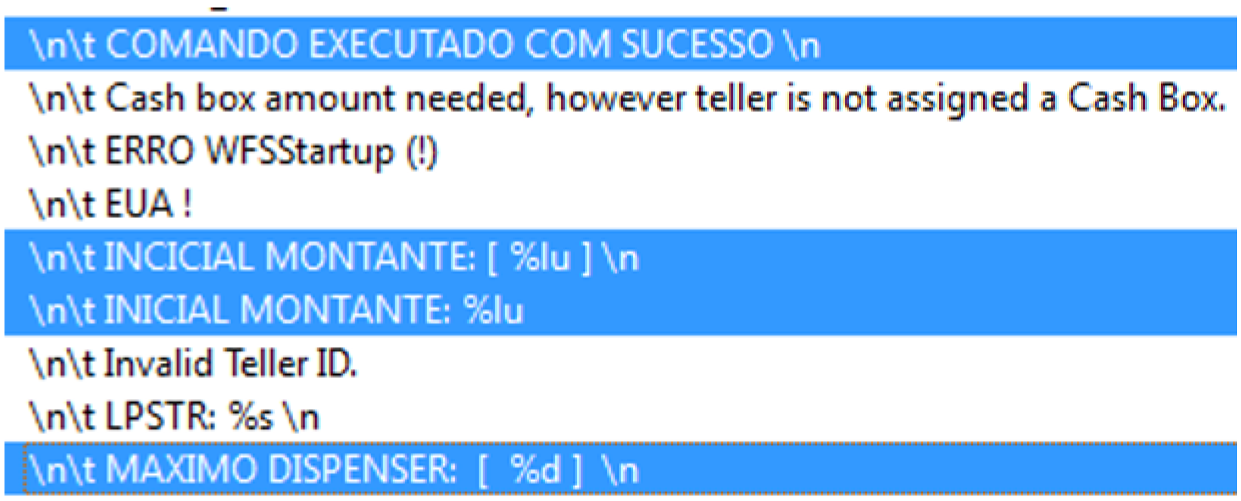
atWin.exe

overlay peexe

Submissions

Date	Name	Source	Country
2019-12-04 22:30:41	atWin.exe	f06eaba4 - web	BR

Similarly, the currency we saw earlier and strings in Portuguese tell us the malware targets Latin America countries (and possibly written by Portuguese-speaking attackers).



```
\n\t COMANDO EXECUTADO COM SUCESSO \n
\n\t Cash box amount needed, however teller is not assigned a Cash Box.
\n\t ERRO WFSStartup (!)
\n\t EUA !
\n\t INCICIAL MONTANTE: [ %lu ] \n
\n\t INICIAL MONTANTE: %lu
\n\t Invalid Teller ID.
\n\t LPSTR: %s \n
\n\t MAXIMO DISPENSER:  [  %d ]  \n
```

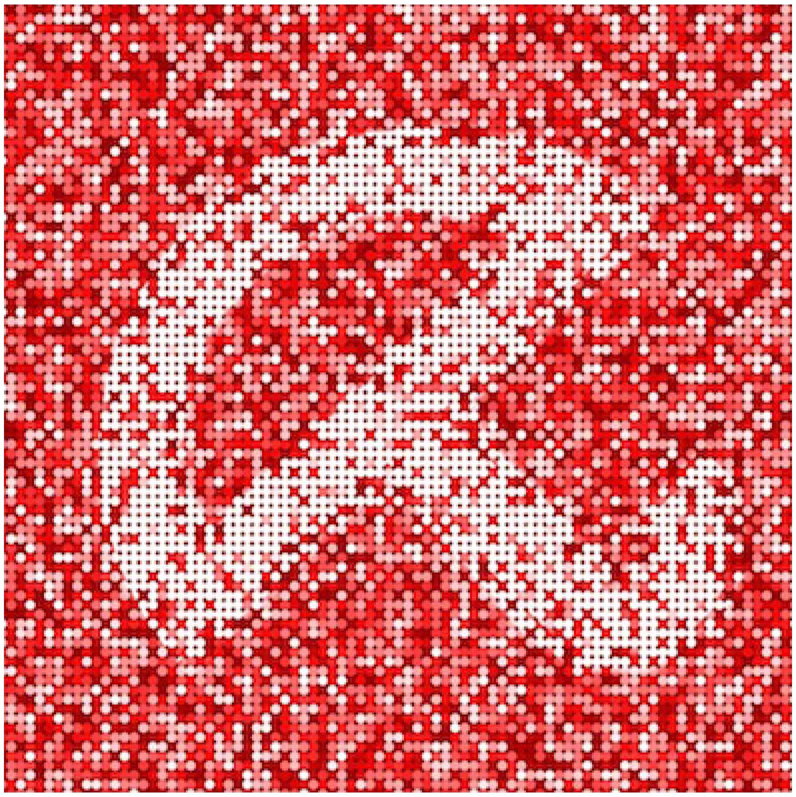
Conclusion

The results of the analysis indicate that the authors behind these attacks target Wincor and Diebold ATMs. We regularly see malware attacks on ATMs mostly because of un-addressed critical issues found in the systems. The detection and prevention of these kinds of threats can be done in multiple ways, and is made easier because the focus of the malware is on executing some specific commands and doesn't clean-up after itself. Avira's range of detection mechanisms protects its customers from such attacks; powerful machine learning based anti-malware technologies help to extend the security level of ATMs and provide protection from known, unknown and advanced threats.

IOCs

5c002870698258535d839d30f15c58934869c337add65c9b499aca93fb1c8692

7cea6510434f2c8f28c9dbada7973449bb1f844cfe589cdc103c9946c2673036


Avira Protection Labs

Protection Lab is the heart of Avira’s threat detection and protection unit. The researchers at work in the Labs are some of the most qualified and skilled anti-malware researchers in the security industry. They conduct highly advance research to provide the best detection and protection to nearly a billion people world-wide.