SUCEFUL: Next Generation ATM Malware

September 11, 2015 | By Daniel Regalado | Threat Research, Advanced Malware

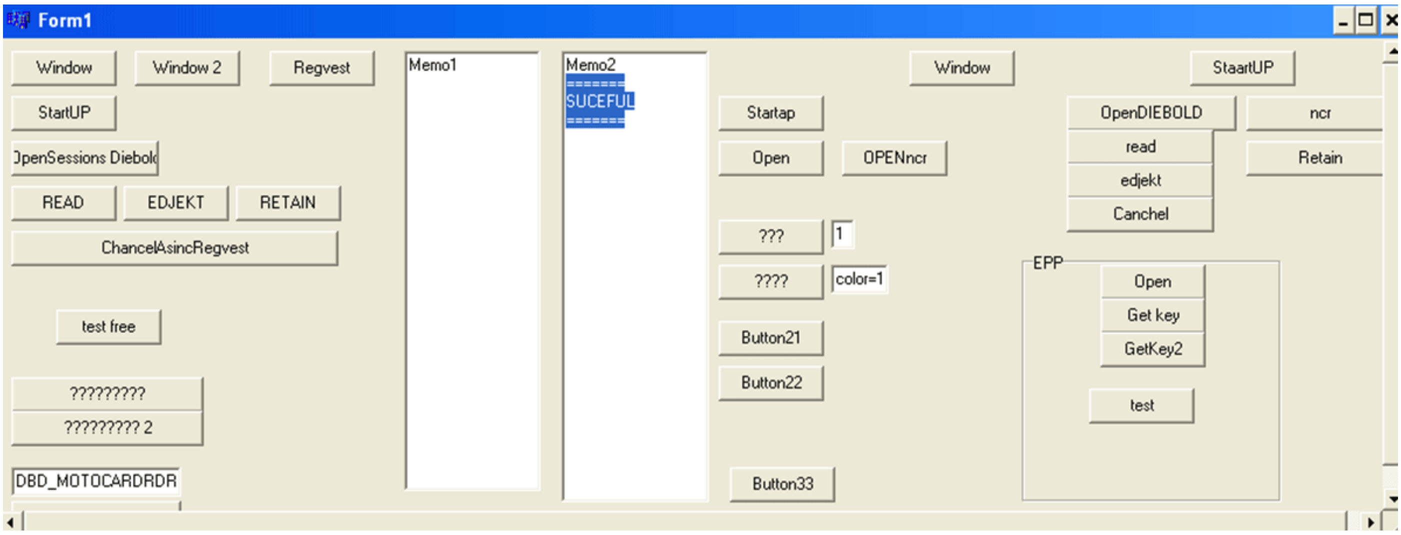
You dip your debit card in an automated teller machine (ATM) and suddenly realize it is stuck inside, what happened?

a) You took too much time entering details.
b) There was an error in the network connection to the bank.
c) The machine is infected with malware and your card was intentionally retained to be ejected to the crooks once you walk away asking for help.

If you answered 'c' you might be correct! FireEye Labs discovered a new piece of ATM malware (4BDD67FF852C221112337FECD0681EAC) that we detect as Backdoor.ATM.Suceful (the name comes from a typo made by the malware authors), which targets **cardholders** and is able to retain debit cards on infected ATMs, disable alarms, or read the debit card tracks.

ATM malware is not new, back in 2013 and 2014 threats like Ploutus[1] or PadPin[2] (Tyupkin) were used to empty ATMs in Mexico, Russia and other countries, but SUCEFUL offers a new twist by targeting the cardholders.

SUCEFUL was recently uploaded to VirusTotal (VT) from Russia, and based on its timestamp, it was likely created on August 25, 2015. It might still be in its development phase; however, the features provided are shocking and never seen before in ATM malware.



```
=======
SUCEFUL
=======
```

Figure 1. SUCEFUL Testing Interface

By clicking on different buttons in the GUI shown in Figure 1. The malware authors can test if the malware operates properly; the word “SUCEFUL” is displayed in the text box indicating success.

Potential SUCEFUL capabilities in Diebold or NCR ATMs include:

1. Reading all the credit/debit card track data
2. Reading data from the chip of the card
3. Control of the malware via ATM PIN pad
4. Retention or ejection of the card on demand: This could be used to steal physical cards
5. Suppressing ATM sensors to avoid detection

XFS Manager

Similar to Ploutus and PadPin, SUCEFUL interacts with a middleware called XFS Manager which is part of the WOSA/XFS[3] Standard that major vendors comply with. The XFS Manager is the interface between the application (malware in this case) and the peripheral devices (e.g., printer, dispenser, card reader, in pad) as shown at Figure 2.

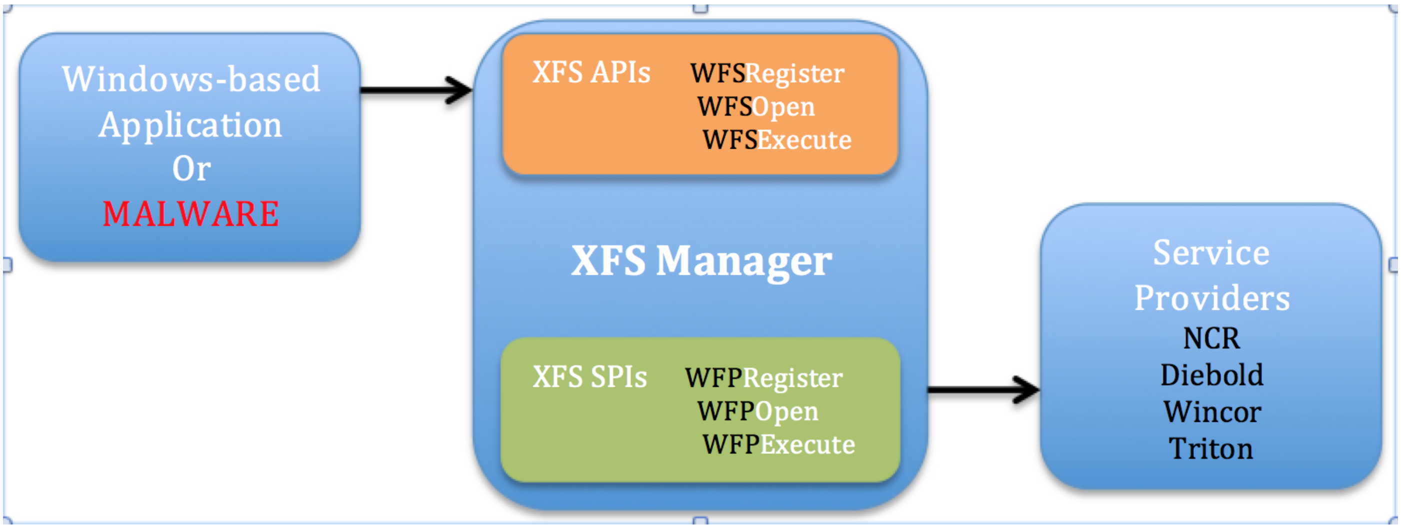


Figure 2. WOSA/XFS Architecture

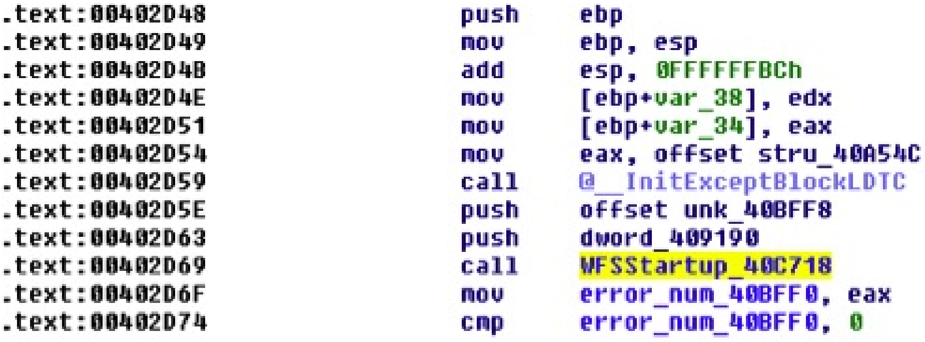
One benefit of the XFS Manager is that it is vendor independent, similar to Java's "Write once, run anywhere" mantra. This means that it can be used maliciously by ATM malware, so that it can run transparently in multiple hardware vendors. This is the case of SUCEFUL, which is targeted for Diebold and NCR.

Every vendor has its own implementation of the XFS Manager with proper security controls in place; however, they also support the default XFS Manager template provided by WOSA/XFS Standard allowing the attackers to create their own interface with the ATM.

The Service Provider Interfaces (XFS SPIs) are vendor-dependent developed to provide the functionality of their own hardware.

Establishing a connection with the XFS Manager

As shown in Figure 3, the first step before starting interacting with the ATMs peripheral devices is to establish a connection with the XFS Manager via WFSStartup API.



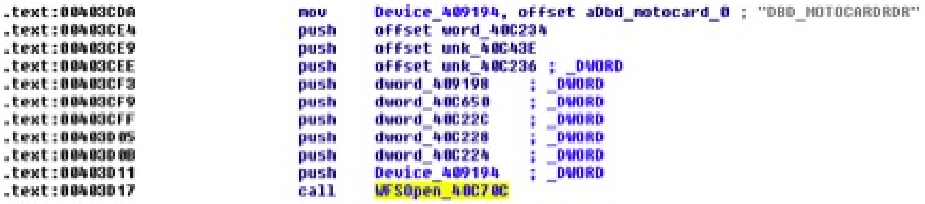
```
.text:00402D48                  push    ebp
.text:00402D49                  mov     ebp, esp
.text:00402D4B                  add     esp, 0FFFFFFBCh
.text:00402D4E                  mov     [ebp+var_38], edx
.text:00402D51                  mov     [ebp+var_34], eax
.text:00402D54                  mov     eax, offset stru_40A54C
.text:00402D59                  call    @__InitExceptBlockLDTC
.text:00402D5E                  push    offset unk_40BFF8
.text:00402D63                  push    dword_409190
.text:00402D69                  call    WFSStartup_40C718
.text:00402D6F                  mov     error_num_40BFF0, eax
.text:00402D74                  cmp     error_num_40BFF0, 0
```

Figure 3. Connecting with the XFS Manager via XFSStartUp API

Opening sessions with the peripheral devices

The next step is to open sessions with the peripheral devices via the Service Providers (XFS SPIs) through the XFS Manager by calling WFSOpen or WFSAsyncOpen APIs where the first parameter is the Logical Device Name.

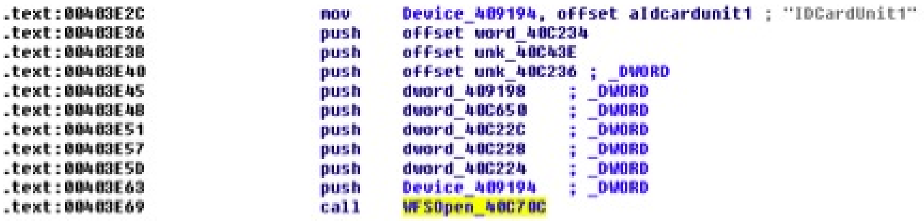
In Figure 4, a session with Diebold Card Reader is being initiated where the logical device name is "DBD_MotoCardRdr".



```
.text:00403CDA                  mov     Device_409194, offset aDbd_motocard_0 ; "DBD_MOTOCARDRDR"
.text:00403CE4                  push    offset word_40C234
.text:00403CE9                  push    offset unk_40C43E
.text:00403CEE                  push    offset unk_40C236 ; _DWORD
.text:00403CF3                  push    dword_409198     ; _DWORD
.text:00403CF9                  push    dword_40C650     ; _DWORD
.text:00403CFF                  push    dword_40C22C     ; _DWORD
.text:00403D05                  push    dword_40C228     ; _DWORD
.text:00403D0B                  push    dword_40C224     ; _DWORD
.text:00403D11                  push    Device_409194    ; _DWORD
.text:00403D17                  call    WFSOpen_40C70C
```

Figure 4. Diebold Card Reader

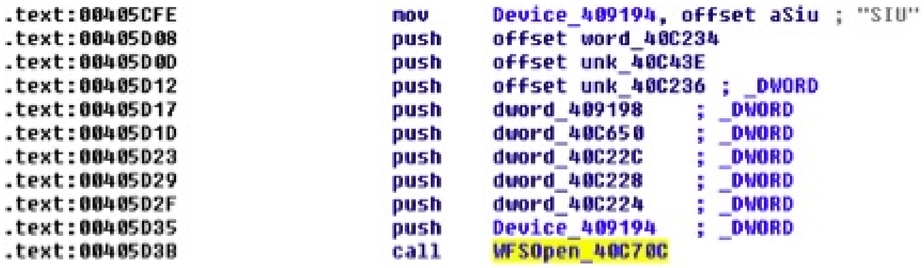
In Figure 5, a session with NCR Card Reader is being initiated where the logical device name is "IDCardUnit1":



```
.text:00403E2C                 mov     Device_409194, offset aIdcardunit1 ; "IDCardUnit1"
.text:00403E36                 push    offset word_40C234
.text:00403E3B                 push    offset unk_40C43E
.text:00403E40                 push    offset unk_40C236 ; _DWORD
.text:00403E45                 push    dword_409198    ; _DWORD
.text:00403E4B                 push    dword_40C650    ; _DWORD
.text:00403E51                 push    dword_40C22C    ; _DWORD
.text:00403E57                 push    dword_40C228    ; _DWORD
.text:00403E5D                 push    dword_40C224    ; _DWORD
.text:00403E63                 push    Device_409194   ; _DWORD
.text:00403E69                 call    WFSOpen_40C70C
```

Figure 5. NCR Card Reader

In Figure 6, a session with the Sensors and Indicators Unit (SIU) is being initiated:



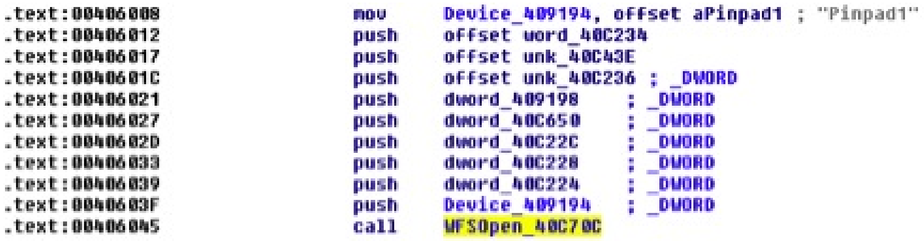
```
.text:00405CFE                 mov     Device_409194, offset aSiu ; "SIU"
.text:00405D08                 push    offset word_40C234
.text:00405D0D                 push    offset unk_40C43E
.text:00405D12                 push    offset unk_40C236 ; _DWORD
.text:00405D17                 push    dword_409198    ; _DWORD
.text:00405D1D                 push    dword_40C650    ; _DWORD
.text:00405D23                 push    dword_40C22C    ; _DWORD
.text:00405D29                 push    dword_40C228    ; _DWORD
.text:00405D2F                 push    dword_40C224    ; _DWORD
.text:00405D35                 push    Device_409194   ; _DWORD
.text:00405D3B                 call    WFSOpen_40C70C
```

Figure 6. Sensors and Indicators Unit

The SIU provides functions to operate port (indicators) categories including but not limited to:

- Door Sensors: cabinet, safe, or vandal shield doors
- Alarm Sensors: tamper, seismic, or heat sensors
- Proximity Sensors

In Figure 7, a session with NCR PIN pad is being initiated where the device logical name is “Pinpad1”.



```
.text:00406008                 mov     Device_409194, offset aPinpad1 ; "Pinpad1"
.text:00406012                 push    offset word_40C234
.text:00406017                 push    offset unk_40C43E
.text:0040601C                 push    offset unk_40C236 ; _DWORD
.text:00406021                 push    dword_409198    ; _DWORD
.text:00406027                 push    dword_40C650    ; _DWORD
.text:0040602D                 push    dword_40C22C    ; _DWORD
.text:00406033                 push    dword_40C228    ; _DWORD
.text:00406039                 push    dword_40C224    ; _DWORD
.text:0040603F                 push    Device_409194   ; _DWORD
.text:00406045                 call    WFSOpen_40C70C
```

Figure 7. Connecting with the ATM PIN pad

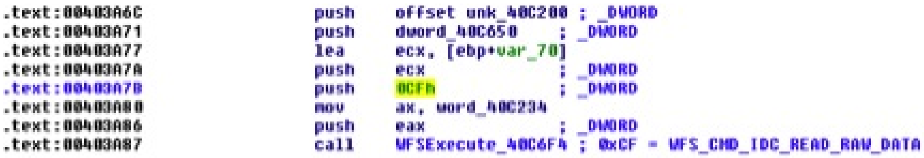
By reading information from the PIN pad, the crooks could interact with the ATM malware.

Interacting with the peripheral devices

Once a session has been opened, the APIs WFSExecute or WFSAsyncExecute can be used to request specific operations to the peripheral devices where the second parameter is the command to be executed.

Reading debit card track data

In Figure 8, the WFS_CMD_IDC_READ_RAW_DATA command instructs the card reader to read all the track data and chip if a card is inserted or wait to read it as soon as the card has been inserted or pulled through.



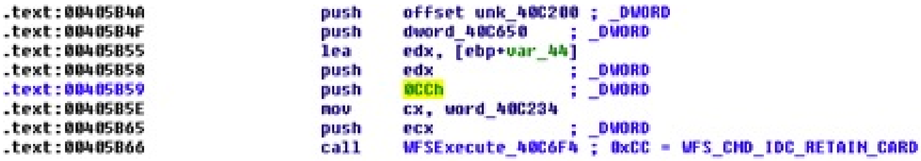
```
.text:00403A6C                 push    offset unk_40C200 ; _DWORD
.text:00403A71                 push    dword_40C650    ; _DWORD
.text:00403A77                 lea     ecx, [ebp+var_70]
.text:00403A7A                 push    ecx             ; _DWORD
.text:00403A7B                 push    0CFh            ; _DWORD
.text:00403A80                 mov     ax, word_40C234
.text:00403A86                 push    eax             ; _DWORD
.text:00403A87                 call    WFSExecute_40C6F4 ; 0xCF = WFS_CMD_IDC_READ_RAW_DATA
```

Figure 8. WFS_CMD_IDC_READ_RAW_DATA Command

Track 1 & 2 contain information like cardholder’s name, account number, expiration date, encrypted PIN, etc.

Retain and/or Eject debit card

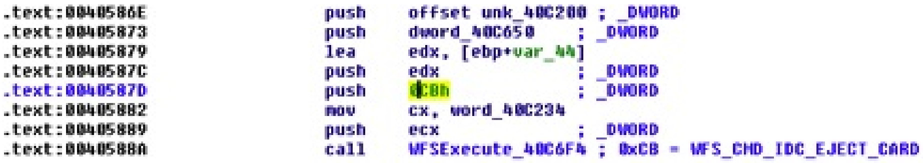
The WFS_CMD_IDC_RETAIN_CARD command in Figure 9 instructs the Card Reader to retain the card:



```
.text:0040584A                 push    offset unk_40C200 ; _DWORD
.text:0040584F                 push    dword_40C650     ; _DWORD
.text:00405855                 lea     edx, [ebp+var_44]
.text:00405858                 push    edx              ; _DWORD
.text:00405859                 push    0CCh             ; _DWORD
.text:0040585E                 mov     cx, word_40C234
.text:00405865                 push    ecx              ; _DWORD
.text:00405866                 call    WFSExecute_40C6F4 ; 0xCC = WFS_CMD_IDC_RETAIN_CARD
```

Figure 9. WFS_CMD_IDC_RETAIN_CARD

In Figure 10, the WFS_CMD_IDC_EJECT_CARD command instructs the Card Reader to eject the card:



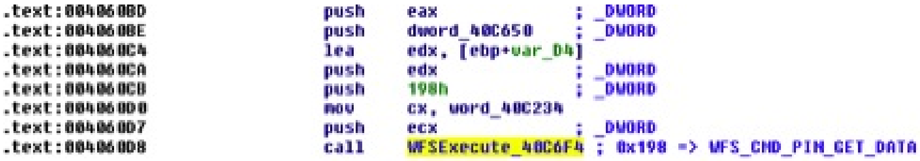
```
.text:0040586E                 push    offset unk_40C200 ; _DWORD
.text:00405873                 push    dword_40C650     ; _DWORD
.text:00405879                 lea     edx, [ebp+var_44]
.text:0040587C                 push    edx              ; _DWORD
.text:0040587D                 push    0CBh             ; _DWORD
.text:00405882                 mov     cx, word_40C234
.text:00405889                 push    ecx              ; _DWORD
.text:0040588A                 call    WFSExecute_40C6F4 ; 0xCB = WFS_CMD_IDC_EJECT_CARD
```

Figure 10. WFS_CMD_IDC_EJECT_CARD

This RETAIN and EJECT commands suggest that the malware authors can retain debit cards inserted into the ATM and eject them whenever they want stealing the physical card from the victims.

Interact with the Malware via PIN pad

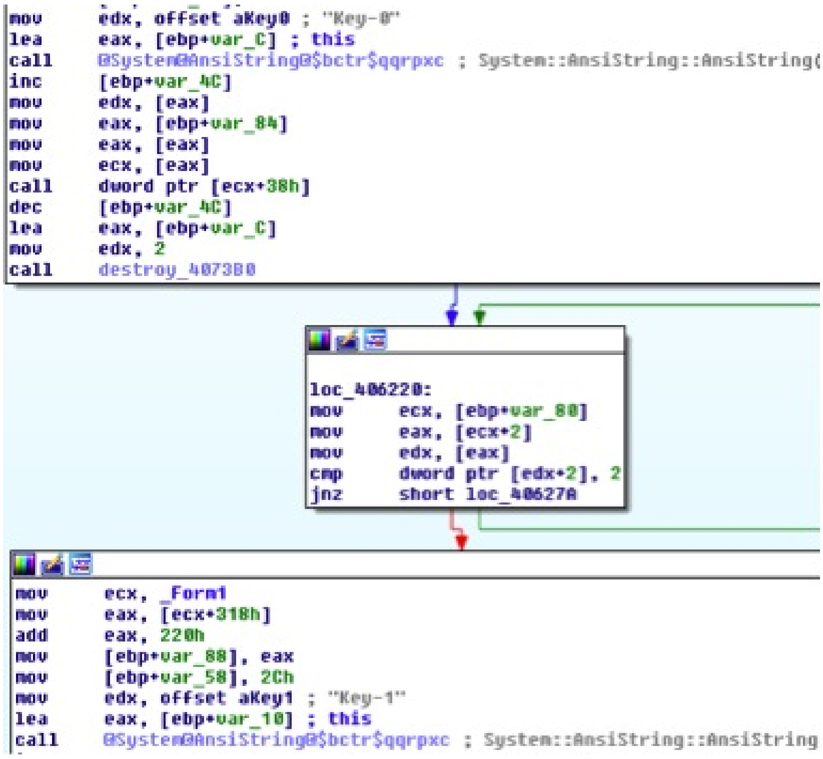
In Figure 11, the WFS_CMD_PIN_GET_DATA command is used to read the keystrokes entered by the cardholder (or attacker) in the PIN pad.



```
.text:004060BD                 push    eax              ; _DWORD
.text:004060BE                 push    dword_40C650     ; _DWORD
.text:004060C4                 lea     edx, [ebp+var_D4]
.text:004060CA                 push    edx              ; _DWORD
.text:004060CB                 push    198h             ; _DWORD
.text:004060D0                 mov     cx, word_40C234
.text:004060D7                 push    ecx              ; _DWORD
.text:004060D8                 call    WFSExecute_40C6F4 ; 0x198 => WFS_CMD_PIN_GET_DATA
```

Figure 11. WFS_CMD_PIN_GET_DATA

Once the input is read, a loop will run to identify the keys typed in the Pin pad, which can be Key0-9, Key-ENTER, Key-CANCEL or KEY-CLEAR. In Figure 12, the Key-0 and Key-1 are being checked:



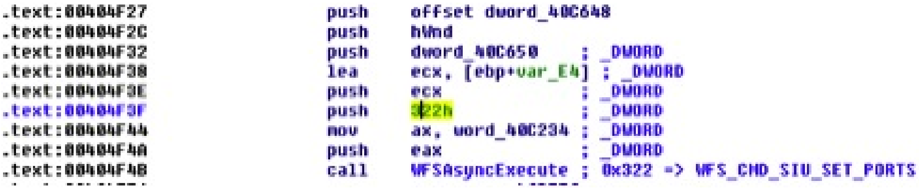
```
mov     edx, offset aKey0 ; "Key-0"
lea     eax, [ebp+var_C] ; this
call    @System@AnsiString@$bctr$qqrpxc ; System::AnsiString::AnsiString(
inc     [ebp+var_4C]
mov     edx, [eax]
mov     eax, [ebp+var_84]
mov     eax, [eax]
mov     ecx, [eax]
call    dword ptr [ecx+38h]
dec     [ebp+var_4C]
lea     eax, [ebp+var_C]
mov     edx, 2
call    destroy_4073B0

loc_406220:
mov     ecx, [ebp+var_80]
mov     eax, [ecx+2]
mov     edx, [eax]
cmp     dword ptr [edx+2], 2
jnz     short loc_40627A

mov     ecx, _Form1
mov     eax, [ecx+318h]
add     eax, 220h
mov     [ebp+var_88], eax
mov     [ebp+var_58], 2Ch
mov     edx, offset aKey1 ; "Key-1"
lea     eax, [ebp+var_10] ; this
call    @System@AnsiString@$bctr$qqrpxc ; System::AnsiString::AnsiString
```

Figure 12. Pin pad Keys check

Disabling ATM Sensors

In Figure 13, the WFS_CMD_SIU_SET_PORTS command could be able to set or clear ATM output ports (indicators) in order to avoid triggering the alarms, some of the sensors that can be controlled are:

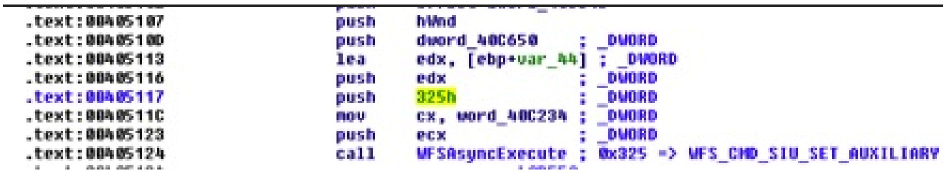
- Turn on/off the Audible Alarm device
- Turn on/off the Facial light
- Turn on/off the Audio indicator
- Turn on/off the Internal Heating device



```
.text:00404F27                 push    offset dword_40C648
.text:00404F2C                 push    hWnd
.text:00404F32                 push    dword_40C650     ; _DWORD
.text:00404F38                 lea     ecx, [ebp+var_E4] ; _DWORD
.text:00404F3E                 push    ecx              ; _DWORD
.text:00404F3F                 push    322h             ; _DWORD
.text:00404F44                 mov     ax, word_40C234  ; _DWORD
.text:00404F4A                 push    eax              ; _DWORD
.text:00404F4B                 call    WFSAsyncExecute  ; 0x322 => WFS_CMD_SIU_SET_PORTS
```

Figure 13. WFS_CMD_SIU_SET_PORTS

In Figure 14 the WFS_CMD_SIU_SET_AUXILIARY command is used to set the status of an Auxiliary indicator including but not limited to:

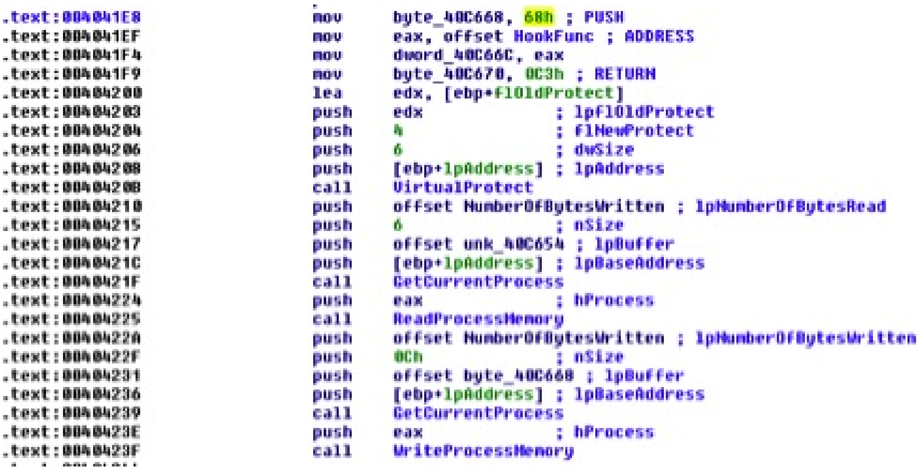


```
.text:00405107                 push    hWnd
.text:0040510D                 push    dword_40C650     ; _DWORD
.text:00405113                 lea     edx, [ebp+var_44] ; _DWORD
.text:00405116                 push    edx              ; _DWORD
.text:00405117                 push    325h             ; _DWORD
.text:0040511C                 mov     cx, word_40C234  ; _DWORD
.text:00405123                 push    ecx              ; _DWORD
.text:00405124                 call    WFSAsyncExecute  ; 0x325 => WFS_CMD_SIU_SET_AUXILIARY
```

Figure 14. WFS_CMD_SIU_SET_AUXILIARY

DLL Hooking

Although DLL Hooking is not a novel technique, it is interesting to understand the reason this is being done inside an ATM. SUCEFUL is able to hook the WFSAsyncExecute API in order to control and monitor all the commands issued to the peripheral devices, this is done by replacing the first 6 bytes of the API Entry point with a classical push <malware_func>, ret instruction (see Figure 15) to redirect execution, as well as patching the RVA address in the Export Directory pointing to WFSAsyncExecute Entry point.



```
.text:004041E8                 mov     byte_40C668, 68h ; PUSH
.text:004041EF                 mov     eax, offset HookFunc ; ADDRESS
.text:004041F4                 mov     dword_40C66C, eax
.text:004041F9                 mov     byte_40C670, 0C3h ; RETURN
.text:00404200                 lea     edx, [ebp+flOldProtect]
.text:00404203                 push    edx             ; lpflOldProtect
.text:00404204                 push    4               ; flNewProtect
.text:00404206                 push    6               ; dwSize
.text:00404208                 push    [ebp+lpAddress] ; lpAddress
.text:0040420B                 call    VirtualProtect
.text:00404210                 push    offset NumberOfBytesWritten ; lpNumberOfBytesRead
.text:00404215                 push    6               ; nSize
.text:00404217                 push    offset unk_40C654 ; lpBuffer
.text:0040421C                 push    [ebp+lpAddress] ; lpBaseAddress
.text:0040421F                 call    GetCurrentProcess
.text:00404224                 push    eax             ; hProcess
.text:00404225                 call    ReadProcessMemory
.text:0040422A                 push    offset NumberOfBytesWritten ; lpNumberOfBytesWritten
.text:0040422F                 push    0Ch             ; nSize
.text:00404231                 push    offset byte_40C668 ; lpBuffer
.text:00404236                 push    [ebp+lpAddress] ; lpBaseAddress
.text:00404239                 call    GetCurrentProcess
.text:0040423E                 push    eax             ; hProcess
.text:0040423F                 call    WriteProcessMemory
```

Figure 15. Hooking WFSAsyncExecute API

Conclusion

Since it is impossible to ascertain whether a retained card is due to this malware, keep the contact number for your bank in your phone and call it while keeping eyes on the ATM.

SUCEFUL is the first multi-vendor ATM Malware targeting cardholders, created to steal the tracks of the debit cards but also to steal the actual physical cards, which is definitely raising the bar of sophistication of this type of threats.

List of known MD5s

4bdd67ff852c221112337fecd0681eac - Backdoor.ATM.Suceful
f74755b92ffe04f97ac506960e6324bb - Backdoor.ATM.Suceful

[1] Ploutus: http://www.symantec.com/connect/blogs/texting-atms-cash-shows-cybercriminals-increasing-sophisticatio

[2] Padpin: https://www.symantec.com/security_response/writeup.jsp?docid=2014-051213-0525-99&tabid=2

[3] WOSA/XFS: http://www.cen.eu/work/areas/ict/ebusiness/pages/ws-xfs.aspx

This entry was posted on Fri Sep 11 08:00:00 EDT 2015 and filed under Advanced Malware, Bank security, Blog, Daniel Regalado, Latest Blog Posts, Threat Research and credit card security.

OCTOBER 12-14, 2015
2015
FIREEYE
CYBER DEFENSE
SUMMIT
MIRcon
GET THE LATEST THREAT INTELLIGENCE AND SECURITY INNOVATIONS FROM EXPERTS ON THE FRONT LINES.
REGISTER NOW AND SAVE ▶

FireEye Alerts

Be the first to receive information on major cyber attacks from the industry leader!

First Name

Last Name

Email Address

Subscribe

Connect

- Facebook
- LinkedIn
- Twitter
- Google+
- YouTube
- Glassdoor

Privacy & Cookies Policy | Safe Harbor