black hat EUROPE 2020

DECEMBER 9–10
BRIEFINGS

POSWorld

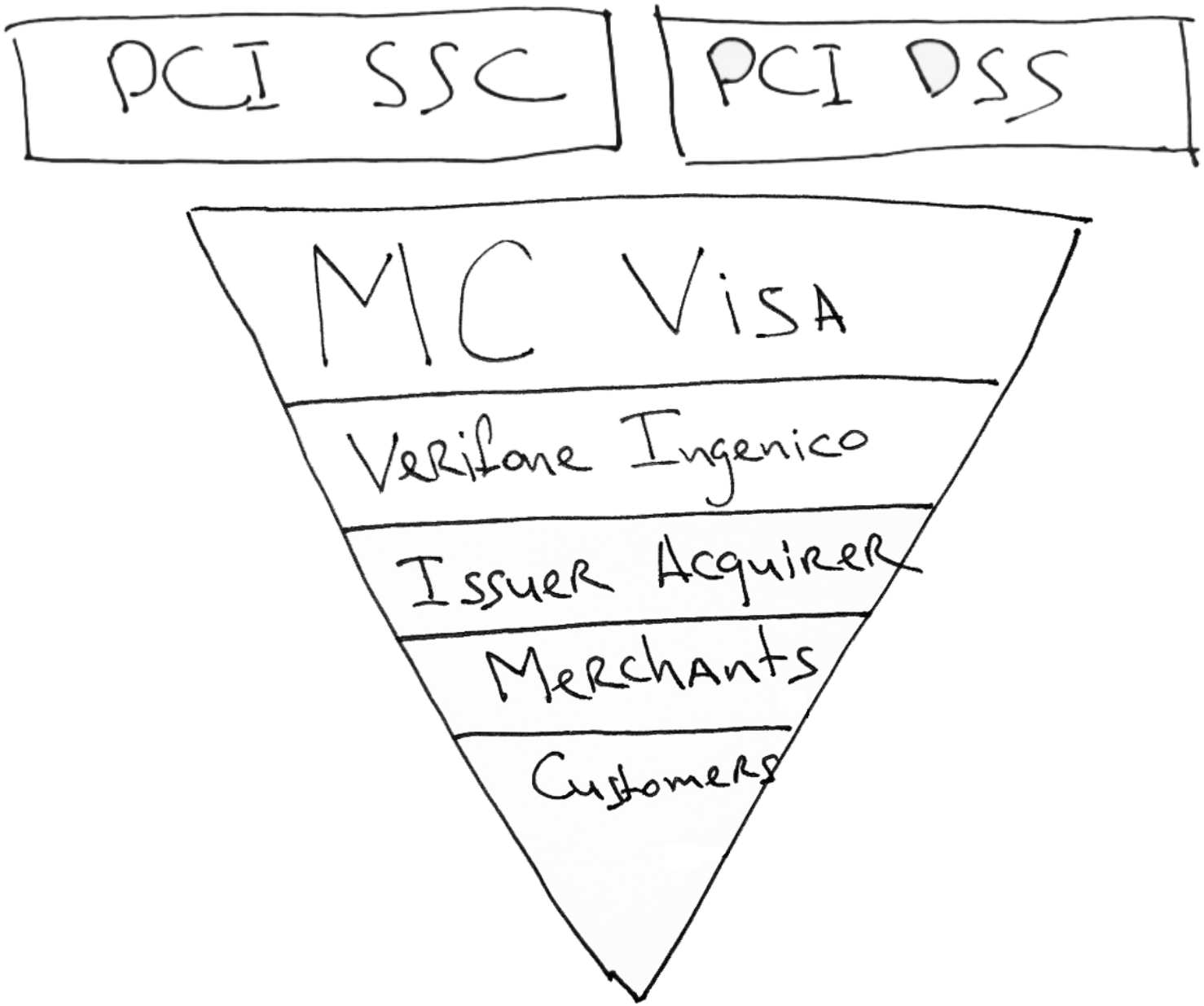
Should you be afraid of hands-on payment devices

CYBER R&D LAB

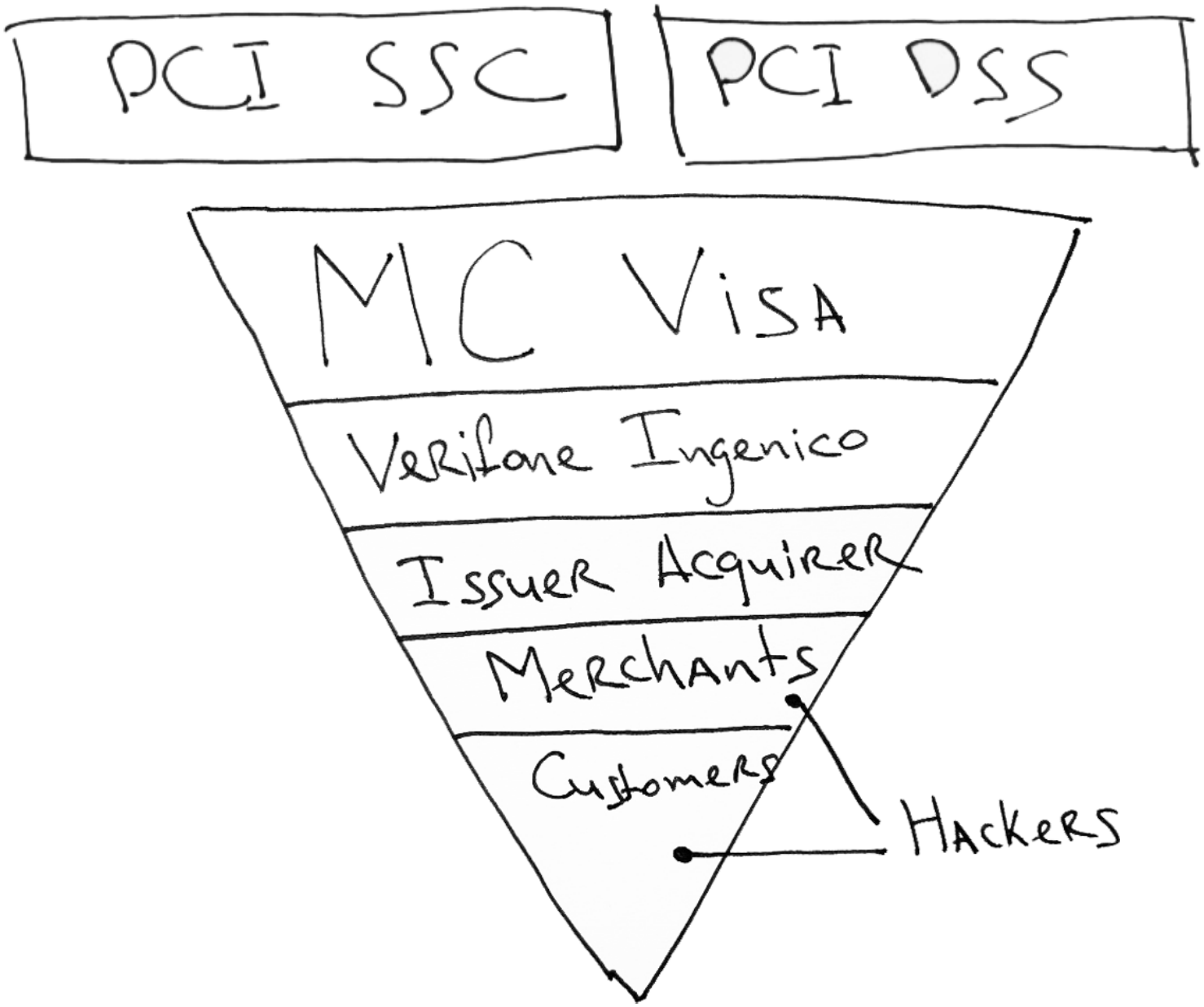
POWERED BY EPAM | POSITIVE

#BHEU @BLACKHATEVENTS

What the POSWorld consists of

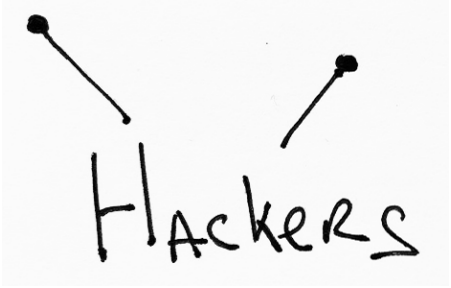


What the POSWorld consists of



What the POSWorld consists of

MERCHANT **ACQUIRER** **CARD BRANDS** **ISSUER**



Evaluation Module 1: Physical and Logical Requirements

A – Physical Security Requirements

Note: *In the following requirements, the device under evaluation is referred to as the "device."*

Number	Description of Requirement	Yes	No	N/A
A1	The device uses tamper-detection and response mechanisms that cause it to become immediately inoperable and result in the automatic and immediate erasure of any sensitive data that may be stored in the device, such that it becomes infeasible to recover the sensitive data. These mechanisms protect against physical penetration of the device by means of (but not limited to) drills, lasers, chemical solvents, opening covers, splitting the casing (seams), and using ventilation openings.	☐	☐	☐
A2	There is no demonstrable way to disable or defeat the tamper mechanism/s and insert a sensitive key-press-disclosing bug. Keypads used for PIN entry require an attack potential of at least 26 per device for identification and initial exploitation, with a minimum of 13 for exploitation, exclusive of the IC card reader, as defined in Appendix B. Keypads used for manual PAN entry, but not PIN entry—e.g., a non-PED—require an attack potential of at least 16 per device for identification, with a minimum of 8 points for exploitation.[B]	☐	☐	☐
A3	The security of the device is not compromised by altering: ▪ Environmental conditions ▪ Operational conditions *(An example includes subjecting the device to temperatures or operating voltages outside the stated operating ranges.)*	☐	☐	☐
A4	Sensitive functions or data are only used in the protected area(s) of the device. Sensitive data and functions dealing with sensitive data are protected from unauthorized modification without requiring an attack potential of at least 26 for identification and initial exploitation, with a minimum of 13 for exploitation, exclusive of the IC card reader, for identification and initial exploitation.[B]	☐	☐	☐

POSWorld requirements

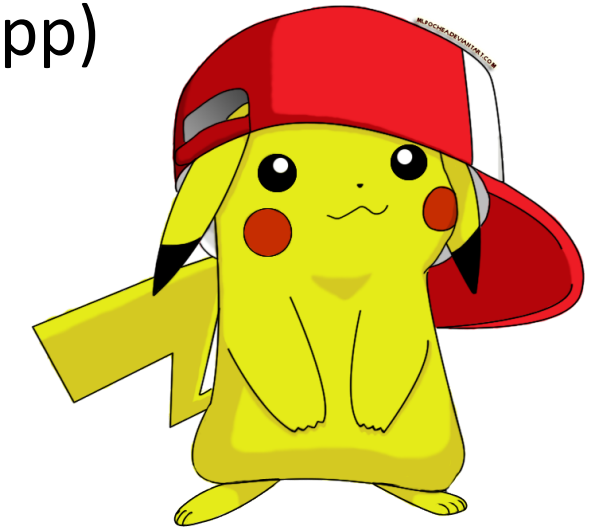
https://www.pcisecuritystandards.org/documents/PCI_PTS_POI_SRs_v4_Final.pdf

https://www.pcisecuritystandards.org/documents/PCI_PTS_POI_SRs_v6.pdf

https://www.pcisecuritystandards.org/documents/pos_ped_security_requirements.pdf

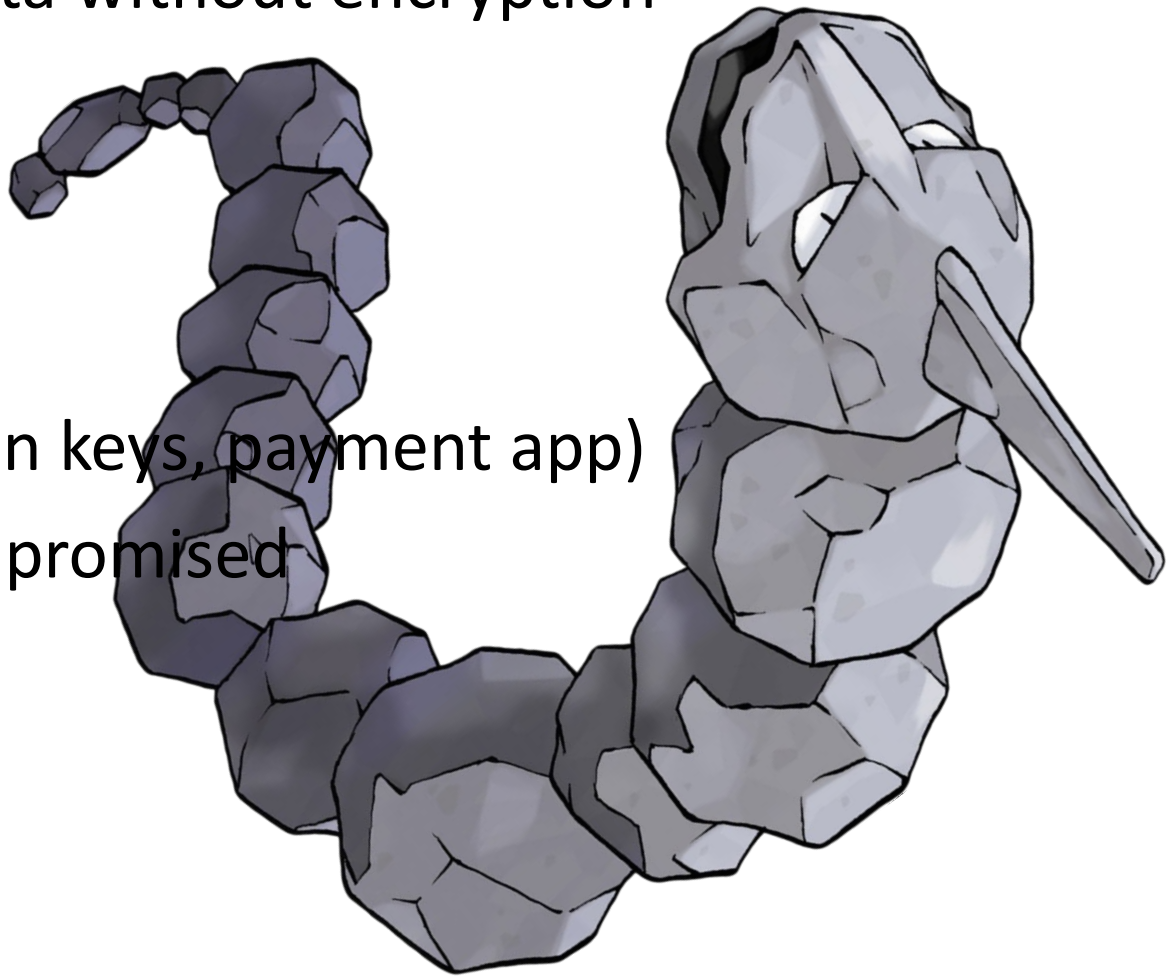
What do PCI Council require?

- Don’t store/send/process sensitive data without encryption
  - PIN
  - Track2 data
  - CVV2/CVC3/CID
- Use tamper detection
- Act: delete “sensitive data” (encryption keys, payment app)
- Indicate that the device has been compromised



What do PCI Council require?

- Don’t store/send/process sensitive data without encryption
  - PIN
  - Track2 data
  - CVV2/CVC3/CID
- Use tamper detection
- Act: delete “sensitive data” (encryption keys, payment app)
- Indicate that the device has been compromised



Hackers

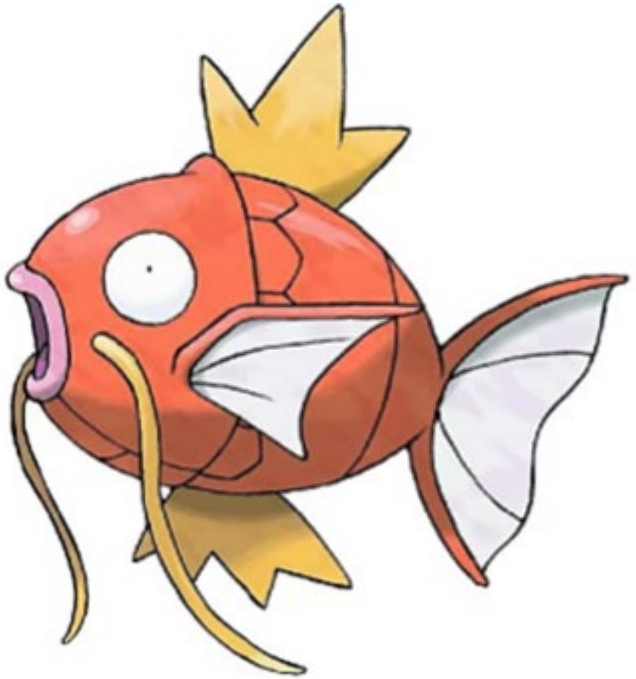
POS terminals vs Cash Registers

POS

Pin pad

PDQ

Terminal



≠

Cash Registers

PoS systems

POS terminals vs Cash Registers

≠

ADVANTIO

Malware is a successful method to hack PoS and steal Customers Data

Igor Mancini **December 9, 2015**

Target

Target is one of the best known American brands, famous for selling miscellaneous goods that range from clothing to CDs to groceries. But in 2013 the retailer hit the headlines for an entirely different reason when it emerged that the company had been affected by a hack that left over 40 million Target customers at risk of credit card fraud, while 70 million others had personal information (such as email addresses) stolen during the breach.

pos hack

About 367,000 results (0.29 seconds)

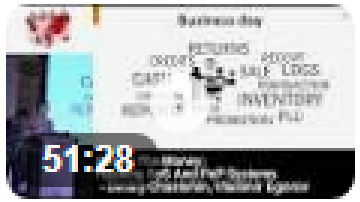
www.youtube.com › watch

#CyberHeadlines - POS Hack - YouTube

15 Nov 2016 · Uploaded by Social27

www.youtube.com › watch

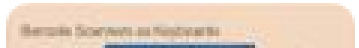
#HITBGSEC 2017 Conf D1 - Get To The Money: Hacking PoS ...



27 Sep 2017 · Uploaded by Hack In The Box Security Conference

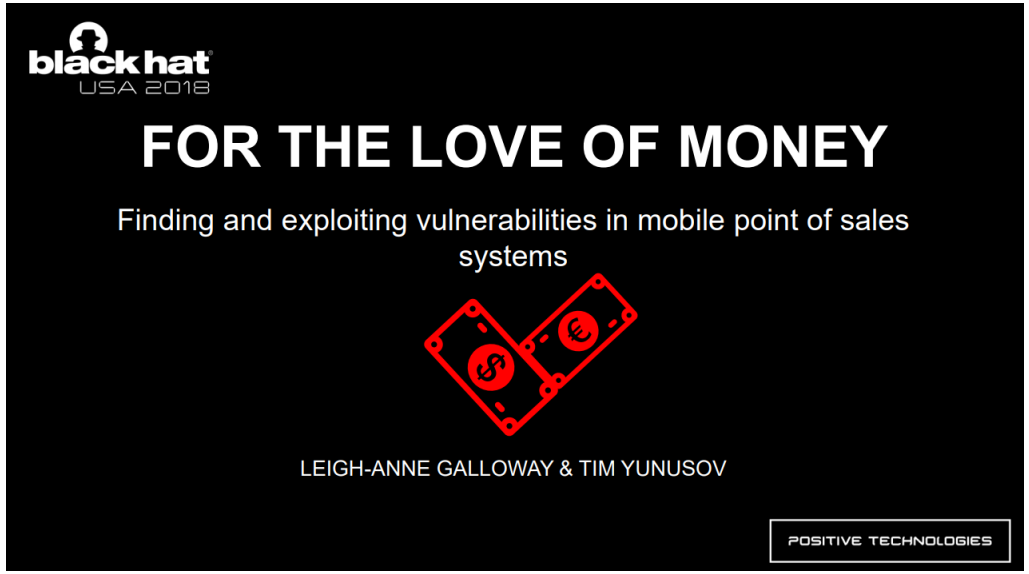
www.youtube.com › watch

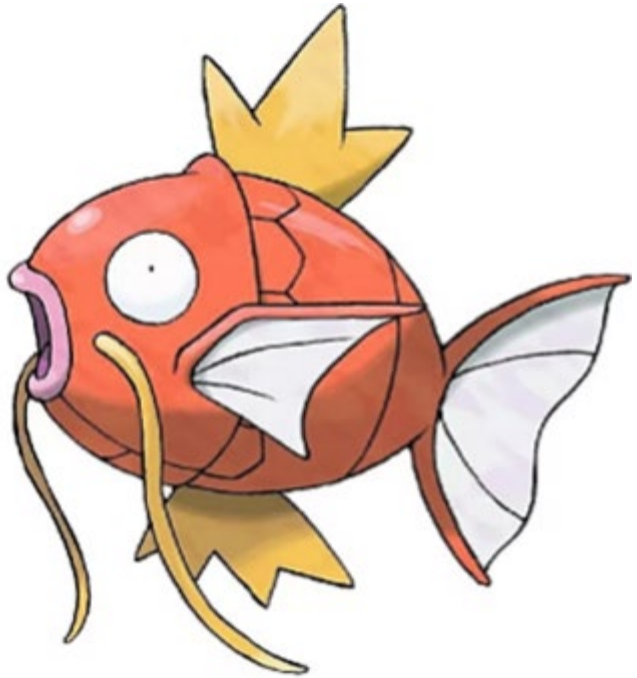
Jackson Thuraisamy & Jason Tran - Hacking POS PoS ...



https://www.researchgate.net/figure/Modified-Chip-and-PIN-terminal-playing-Tetris_fig3_230839731

Payment terminals allow for remote PIN capture and card cloning





On the Origin of POSWorld Species

Instead of hardware intro

PoS Terminal Security Uncovered - Aleksei Stennikov

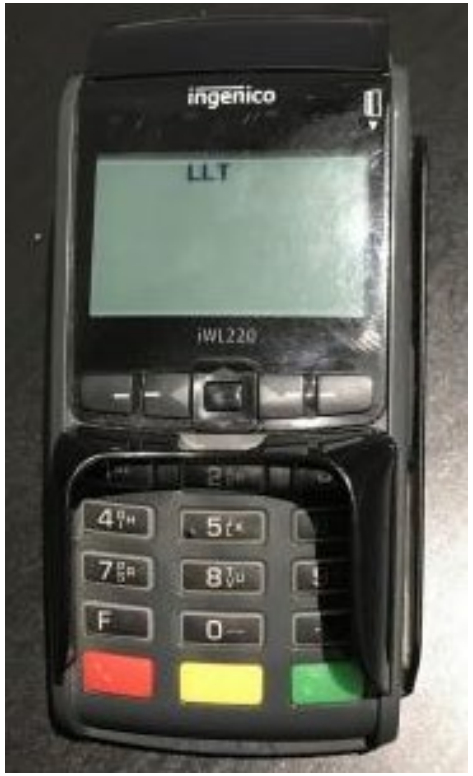
https://www.youtube.com/channel/UCivO-5rpPcv89Wt8okBW21Q - DEFCON Payment Village
https://youtu.be/oyUD7RDJsJs - POS HW details

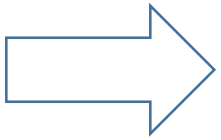
Weaponizing your POS

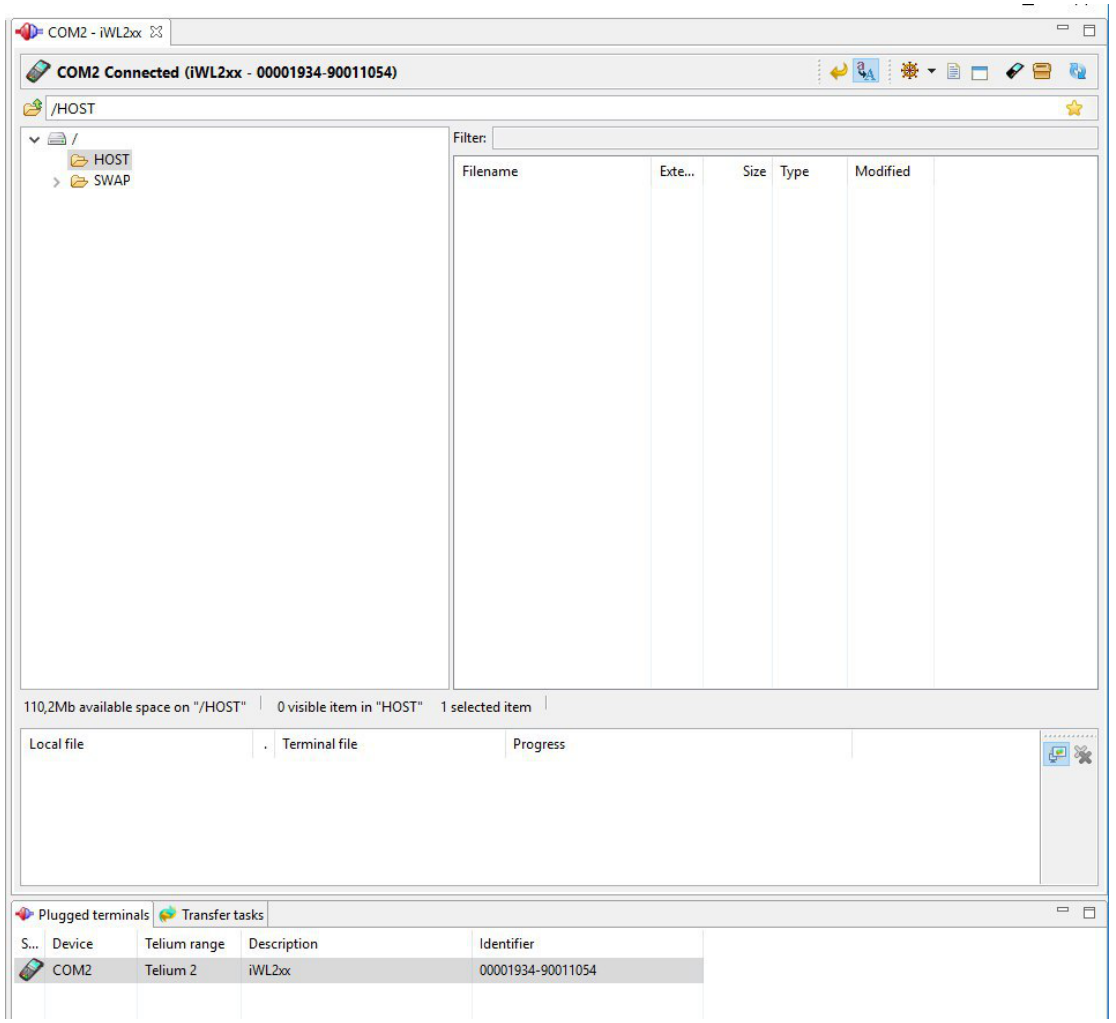
How to weaponize your Ingenico Telium2 POS

Ingenico Telium2 1st step

Just maintain it







Ingenico Telium2 1st step

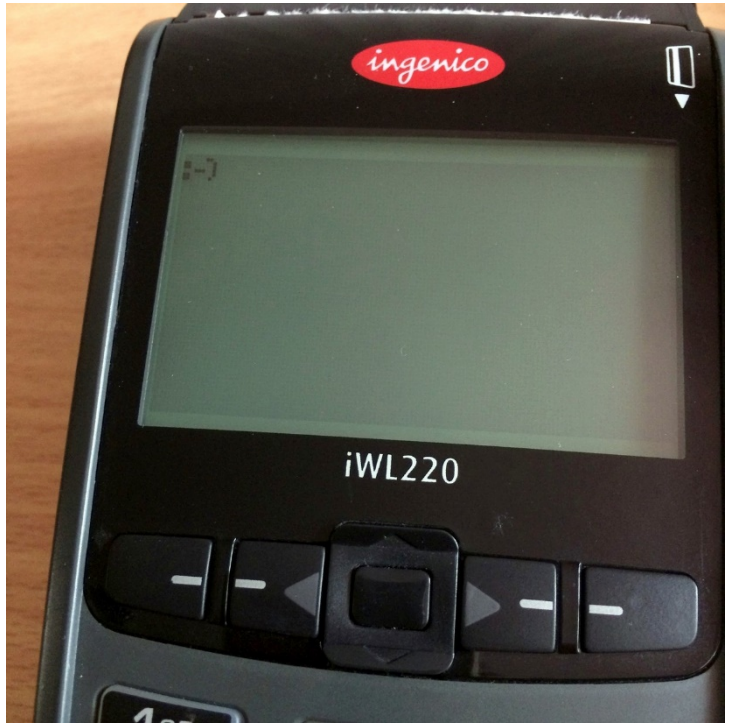
All vendors love hardcoded passwords

PPP connection in LLT mode:

- pppuser:123456

internal FTP service in LLT mode:

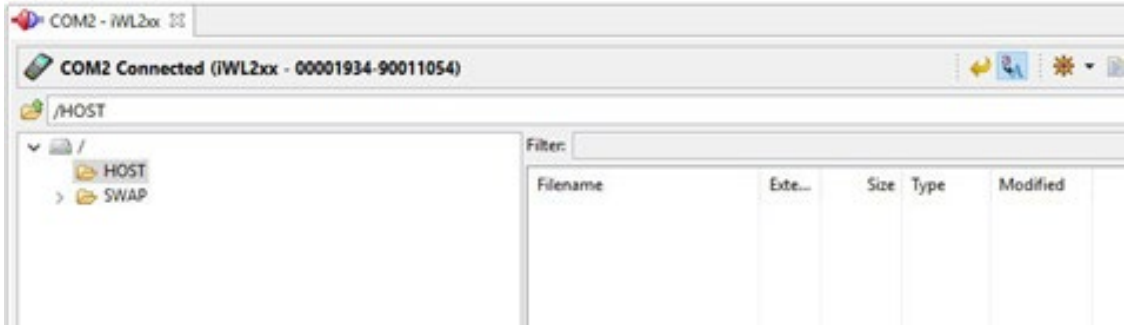
- ftpuser: 123456
- maint: 51966
- system: 31415926



Ingenico Telium2 2nd step

Magic NTPT3 protocol on TCP/6000 port.

- bypass file reading restrictions (i.e. “SYSTEM/SSL.CFG”)
- SOCKET_TASK Buffer overflow
- RemotePutFile command buffer overflow
- 0x26 command buffer overflow

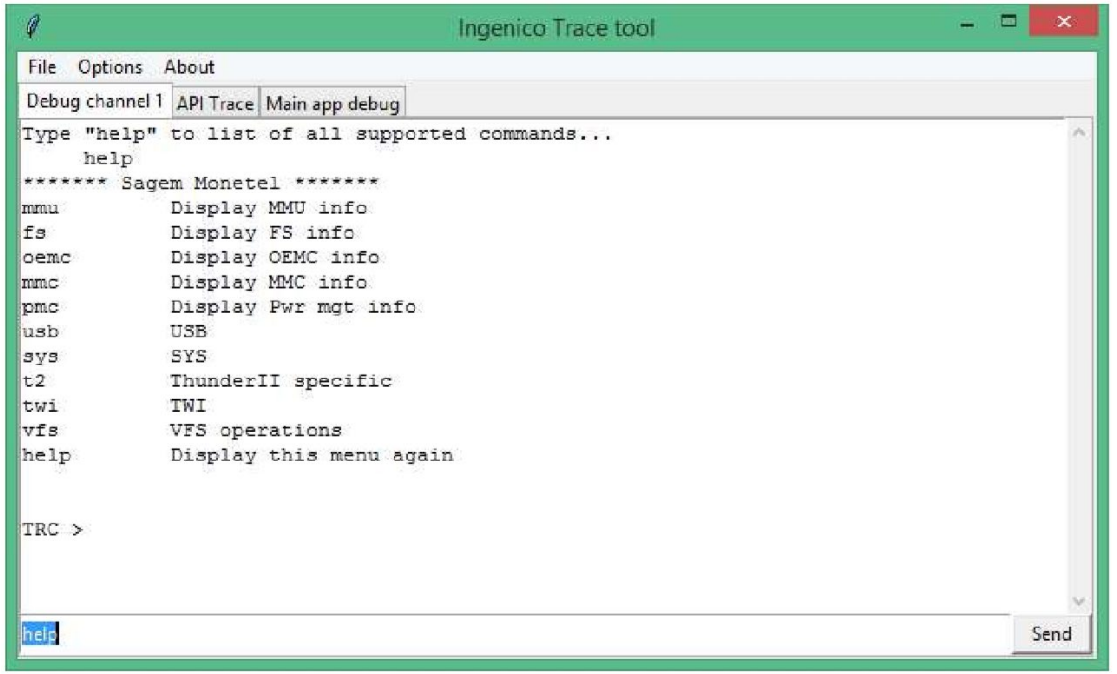


Ingenico Telium2 3rd step

Trace it

Some config magic:

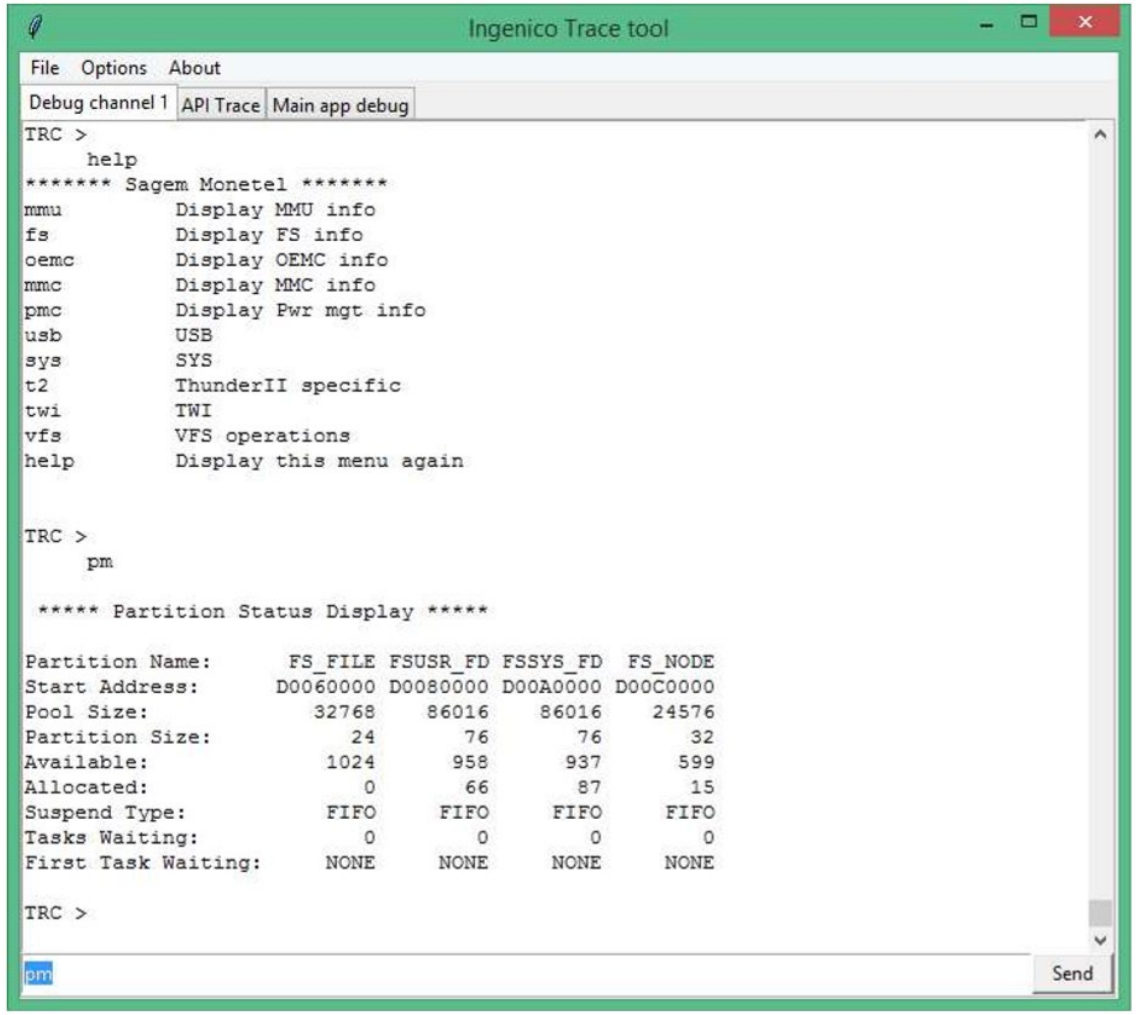
1. load the text file named “SYSTEM.CFG” with the following contents to the /SWAP/ directory:
   TRACE_DEV=5
   LDBG_DEV=0
2. Restart the terminal. TRACE mode on USB is now enabled. Command-line interface prompt is “TRC >”. To work in this mode, use the custom tool or “Trace.exe” tool also provided by Ingenico for terminal software developers working with Telium 1 and 2



Ingenico Telium2 3rd step

Trace it – hidden commands

ts - Task Status Display
tsd - Task Status Display (with dumps)
ms - Mailbox Status Display
qs - Queue Status Display
ps - Pipe Status Display
ss - Semaphore Status Display
es - Event Status Display
si - Signal Status Display
ti - Timer Status Display
pm - Partition Status Display
pmd - Partition Status Display (with dumps)
dm - Dynamic Memory Status Display
dmd - Dynamic Memory Status Display (with dumps)
hs - HISR Status Display

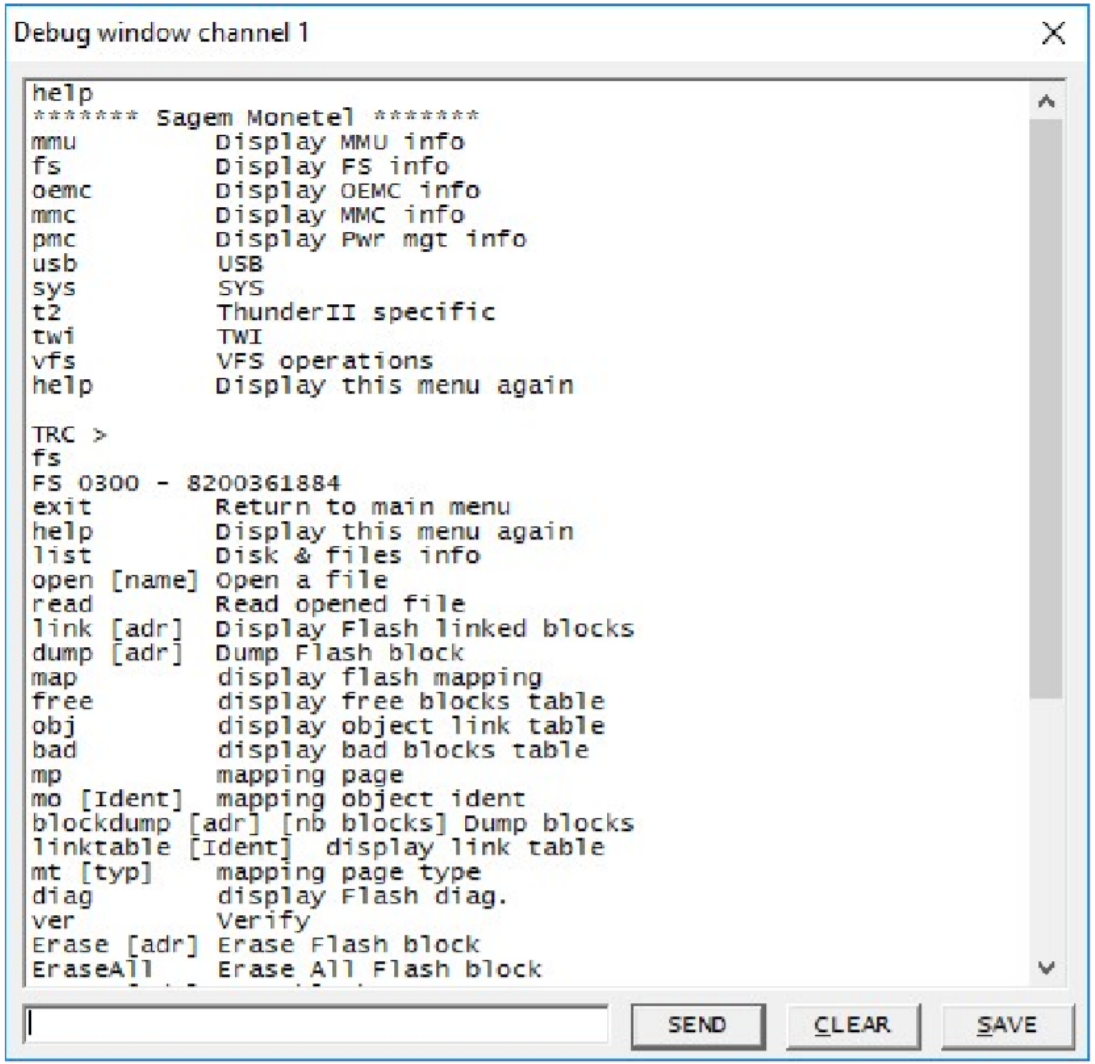


Ingenico Telium2 3rd step

Trace it – full control

Use the help command to list all available commands. The program functions allow:

- Allocating and deallocating memory
- Displaying the contents of all files on the terminal file system, including encryption keys
- Suspending and terminating processes



Ingenico Telium2 3rd step

Trace it – full control

1. Allocate memory space using the Alloc command available in the Debug window channel 1 window, in the mmu menu.
2. Write any malicious executable code in hexadecimal form using the sm command available in the main menu.
3. Suspend the task named PMC by using the hidden NU_Suspend_Task command.
4. Using the sm command, modify one of the return addresses for the PMC task so that it points to the memory space containing malicious code allocated by the attacker.
5. Resume the PMC task using the NU_Resume_Task command.

How to weaponize your VerixV POS

Verifone VerixV 1st step

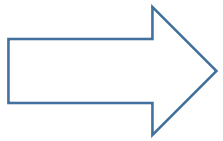
The password haven't changed from 80's

Full Download to a Terminal with no Existing Applications

When you turn on a terminal with no application loaded, it displays DOWNLOAD NEEDED, and a full download is required. Table 1 instructs you how to perform a full download.

Table 1 Perform a Full Download

1	Press F2+F4 to enter system mode *before* the application starts.
2	Enter the system mode password (manufacturer's default is 1 [ALPHA] [ALPHA] 66831[1]) and press the enter (↵) key.
3	Press the ↓ key twice to access the SYS MODE MENU 3 screen.
4	Press EDIT F3 to edit the CONFIG.SYS file
5	Press the backspace (←) key and enter the target GID for the download *or* press the enter (↵) key to select FILE GROUP _1, the default, GID1.
6	Reenter the system mode password (Z66831) if prompted, as shown in step 2. The terminal displays the CONFIG.SYS file.
7	Press the enter (↵) key to edit the CONFIG.SYS file. The terminal displays KEY.



https://www.databreaches.net/verifone-statement-on-default-password-z66831/

Verifone VerixV 2nd step

- What do they hide from you?

Internal filesystems:

"I:" – RAM drive;
"F:" –NAND Flash drive;
"N:" – NAND Flash drive Verix eVo OS extentions;
"B:" – Boot Block. Physical address 0x00010000;
“S:”, “T:”, “U:”, “V:”, and “W:” – some additional filesystem drives;

*drives "B:", “S:”, “T:”, “U:”, “V:”, and “W:” are reserved and written by Verix OS during production

Verifone VerixV 2nd step

- The gift from developers (secret player)

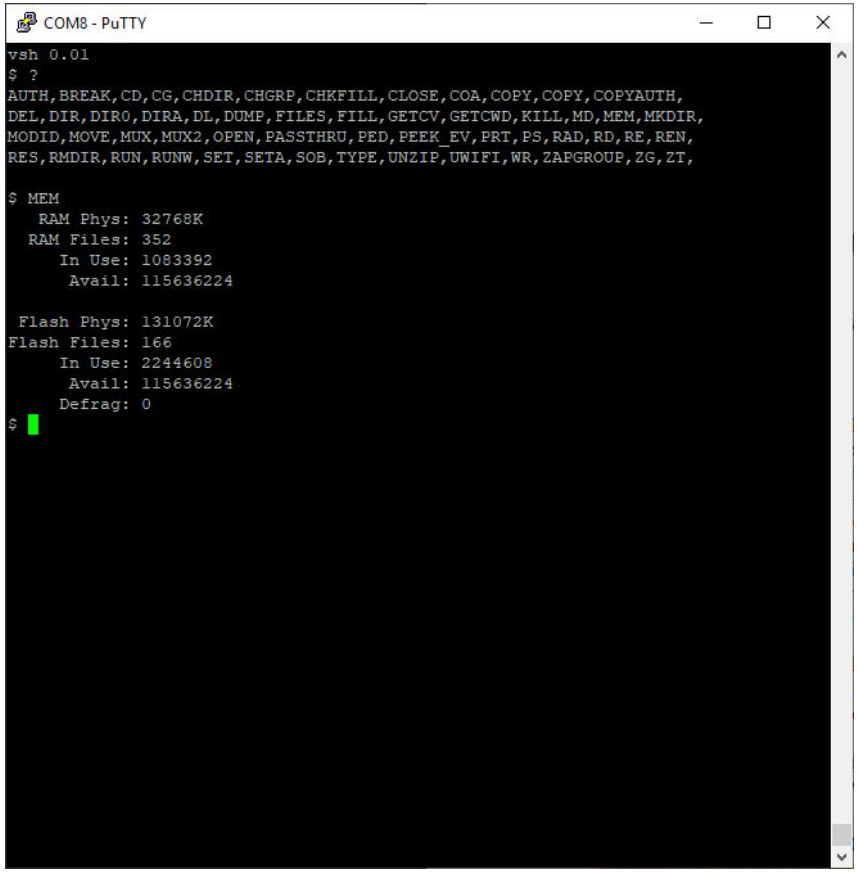
T:SHELL.OUT – ssshell

RUN/RUNW – run a process;
DUMP – hexdump format file read;
DIR/DIR0/DIRA – file list output command variations;
PS – process list;
MEM – memory usage;
RES – pinpad restart;
DL – terminal file load;
Etc…

*GO=T:SHELL.OUT
*ARG="/DEV/COM1"

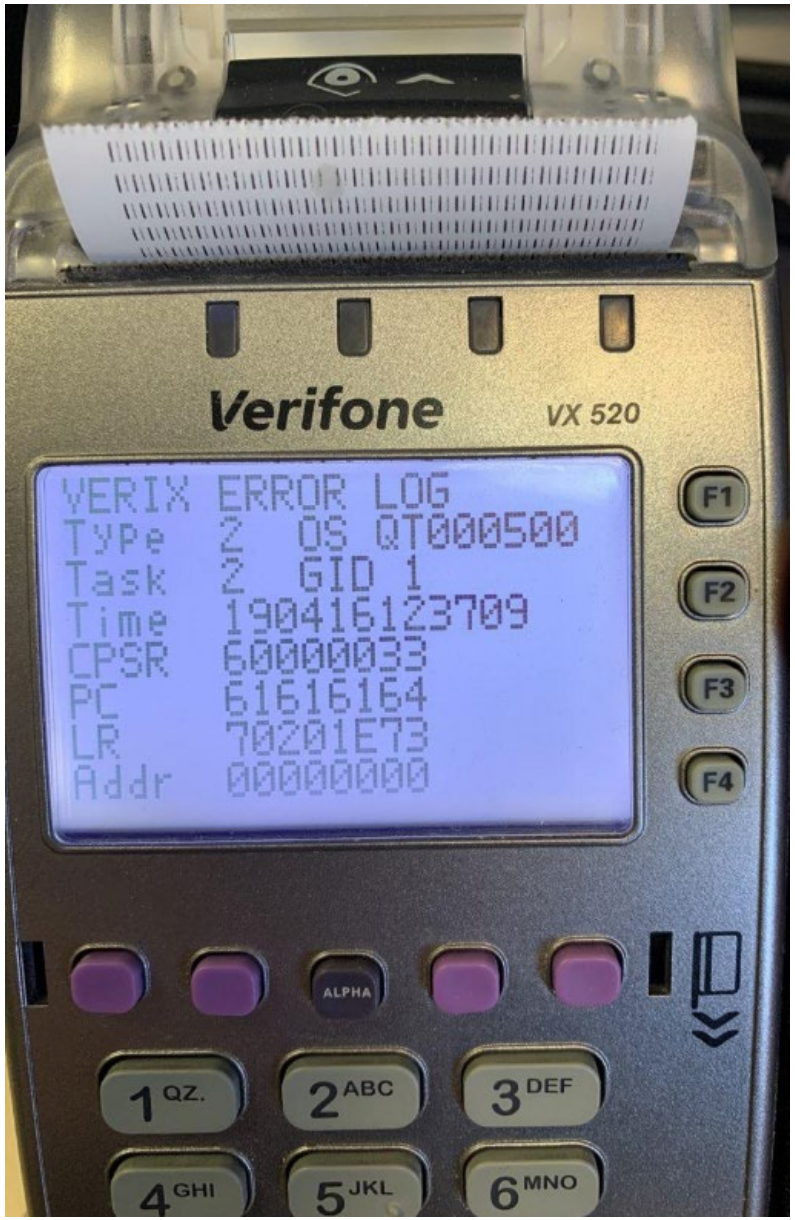
Verifone VerixV 2nd step

- The gift from developers (secret player)



Verifone VerixV 3rd step

- The bugs



Verifone VerixV 3rd step – final shot

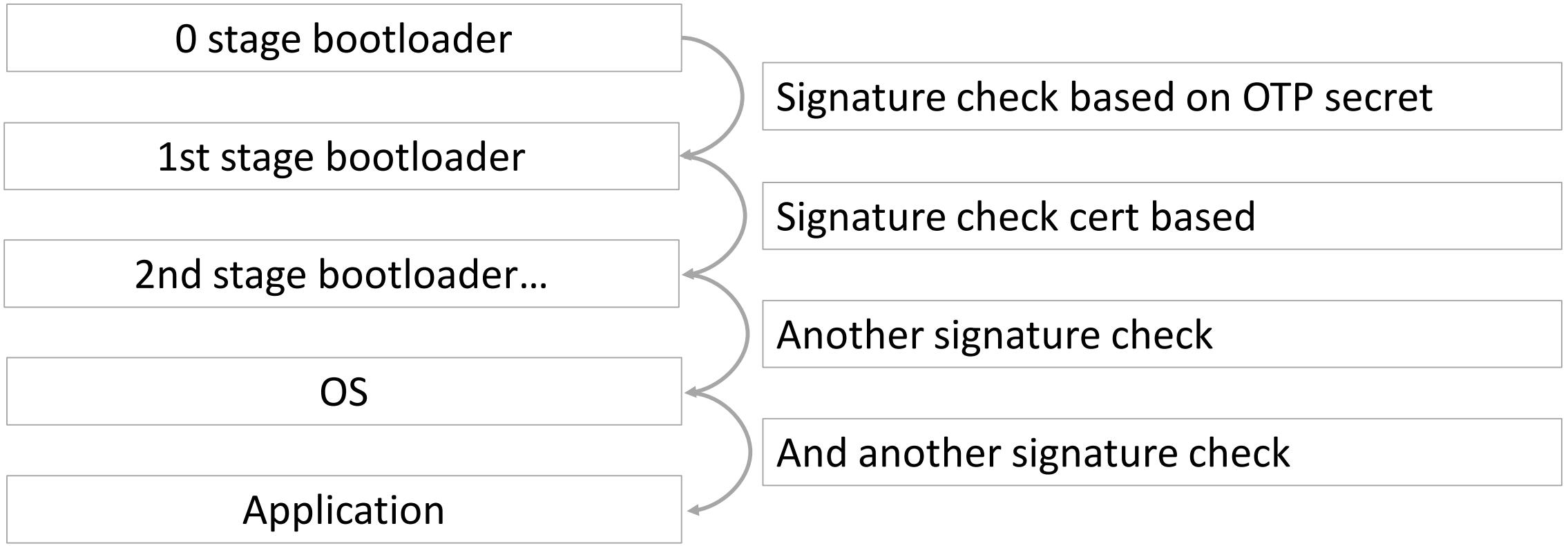
- The bug in syscall

```
71  char v71; // r0
72  int v72; // r0
73  int v73; // r1
74  bool v74; // zf
75  int v75; // r0
76  unsigned int v76; // r1
77  int v77; // r1
78  int v78; // r0
79  __int16 v79; // r0
80  __int16 v80; // r3
81  __int16 v81; // r2
82  __int16 v82; // r1
83  char sRandName[9]; // [sp+0h] [bp-150h]
84  unsigned __int8 ticks[4]; // [sp+Ch] [bp-144h]
85  int v85; // [sp+10h] [bp-140h]
86  char dst[32]; // [sp+14h] [bp-13Ch]
87  __int8 pc[32]; // [sp+34h] [bp-11Ch]
88  char *v88; // [sp+54h] [bp-FCh]
89  proc_meta *v89; // [sp+58h] [bp-F8h]
90  LIB_HEADER *v90[32]; // [sp+5Ch] [bp-F4h]
91  int proc_meta_start; // [sp+DCh] [bp-74h]
92  int v92; // [sp+E0h] [bp-70h]
93  int v93; // [sp+E4h] [bp-6Ch]
94  int v94; // [sp+E8h] [bp-68h]
95  char *v95; // [sp+ECh] [bp-64h]
96  int v96; // [sp+F0h] [bp-60h]
97  struc_30 a1; // [sp+F4h] [bp-5Ch]
98  int *v98; // [sp+118h] [bp-38h]
99  int *v99; // [sp+11Ch] [bp-34h]
100 int (__fastcall **pFuncs)(int, const unsigned __int8 *, int); // [sp+120h] [bp-30h]
101 const char *_fname; // [sp+124h] [bp-2Ch]
102 char *v102; // [sp+128h] [bp-28h]
103 int eeeeee; // [sp+12Ch] [bp-24h]
104
105 _fname = fname;
106 v102 = (char *)parms;
107 eeeeee = flags;
108 v92 = 1;
109 v3 = endswith(fname, " /");
110 v4 = v3 == 0;
111 v88 = v3;
112 if ( !v3 )
113   v3 = (char *)_fname;
114 if ( !v4 )
115 {
116   v5 = v3 - _fname;
117   v88 = v3 + 1;
118   SCHEDULR_memcpy(pc, (char *)_fname, v3 - _fname);
119   v3 = pc;
120   pc[v5] = 0;
```

```
-00000150 ; D/A/*   : change type (data/ascii/array)
-00000150 ; N       : rename
-00000150 ; U       : undefine
-00000150 ; Use data definition commands to create local variable
-00000150 ; Two special fields " r" and " s" represent return add
-00000150 ; Frame size: 150; Saved regs: 0; Purge: 0
-00000150 ;
-00000150
-00000150 sRandName       DCB 9 dup(?)
-00000147                 DCB ? ; undefined
-00000146                 DCB ? ; undefined
-00000145                 DCB ? ; undefined
-00000144 ticks           DCB 4 dup(?)
-00000140 var_140         DCD ?
-0000013C dst             DCB 32 dup(?)          ; string(C)
-0000011C pc              DCB 32 dup(?)
-000000FC var_FC          DCD ?                  ; offset
-000000F8 var_F8          DCD ?                  ; offset
-000000F4 var_F4          DCD 32 dup(?)          ; offset
-00000074 proc_meta_start DCD ?
-00000070 var_70          DCD ?
-0000006C var_6C          DCD ?
-00000068 var_68          DCD ?
-00000064 var_64          DCD ?
-00000060 var_60          DCD ?
-0000005C a1              struc_30 ?
-00000038 var_38          DCD ?
-00000034 var_34          DCD ?
-00000030 pFuncs          DCD ?
-0000002C _fname          DCD ?                  ; offset
-00000028 var_28          DCD ?                  ; offset
-00000024 eeeeee          DCD ?
-00000020
-00000020 ; end of stack variables
```

Verifone Verix - Special Move

- Ancient Secure boot



Verifone Verix - Special Move

- Ancient Secure boot

```
======================================================================
==== SBI V. 03_04 (Jan 13 2013 22:14:13)
==== SEARCHING USB STICK
==== USB NOT FOUND
==== LOADING FROM NAND
==== Read from nand 0x771e0 bytes in 63 mili seconds
==== Vx File Auth using PedGuard

==== Authenticated 0x771a0 bytes in 75 mili seconds
==== Loaded VX Module                          RAM at [ 40000000 .. 42000000 )

PF @ 0x40020E2C via:2 ct:2866 sts:52 last:1 raw:1 en:0 T=30789548
 KT=41D28031 CO=40269DB5 IS=40022E09
TS=1232832072

Pf @ 0x40020E2C via:2 ct:2867 sts:52 last:1 raw:1 en:0 T=4294834105
   KT=41D28031 CO=40269DB5 IS=40022E09 R43=0 R44=0



*************** VERIX initializing ************** 03/20/2017 QT000500
HEAP_MGR
System Reset value rDMU_RST_LAST = 0x1
UNCACHED: 200 KB at 49FCA000
DVIC MGR
Detected chip 58920400: using rev D FLASH controller
rNAND2_ONFI_STATUS  = 08000000
rNAND2_DEVICE_ID    = 2CA18015
rNAND2_DEVICE_ID_X  = 02000000
ROM size 128 MB
NF2_set_feature(1, 04000000)
NF2_get_feature(1)=04000000
Timing mode=4
rNAND2_ACC_CTL_CS1=C3040010
ECC type=4  supported
T1=11211115:  tWP=15   tWH=15   tRP=30  tREH=15  tCS=15  tCLH=15  tALH=15  tADL=75
T2=00000E43:  tWB=105  tWHR=60  tREAD=45
FLASH_SIZE_MB      = 128
BLOCKS             = 1024
CLUSTER_SIZE       = 2048
Drive(I:) has 240 files using   802 KB
Drive(F:) has  71 files using  1662 KB
Drive(S:) has  26 files using   348 KB
Drive(N:) has 326 files using  6676 KB
vfs: v2n[b=0 p=0 i=0]=10
verify file system: 0 errors
```

Verifone Verix - Special Move

- SBI BootLoader

```
1  int __cdecl main_like(int arg, int a2)
2  {
3    int v2; // r4
4    BOOL v4; // [sp+0h] [bp-10h]
5
6    v4 = periph_process() == 1;
7    if ( sub_189B64() != 0xB2D58B32 )
8    {
9  LABEL_4:
10     if ( !v4 )
11       goto LABEL_9;
12     goto LABEL_5;
13   }
14   if ( !v4 )
15   {
16     doXDL(&v4);
17     goto LABEL_4;
18   }
19 LABEL_5:
20   if ( script_load("PSSBI.SCR") )
21   {
22     unload_stuff();
23     print("\n====  SCRIPT ENDED");
24     print("\n *** SBI V. %s (%s %s) ***", "03_08", "Dec  9 2013", "11:08:39", v4);
25     print("\n *** PLEASE REMOVE USB STICK ***");
26     print("\n *** THE SYSTEM WILL RESTART IN 10 SECONDS ***\n");
27     v2 = BCM_Get_Timer1Value(1u);
28     while ( (unsigned int)(v2 - BCM_Get_Timer1Value(1u)) < 0x3938700 )
29       ;
30     reset_tgt(10);
31   }
32 LABEL_9:
33   unload_stuff();
34   JMP2NAND();
35   print("\n====  RESET TARGET");
36   reset_tgt(10);
37   return 0;
38 }
```

Verifone Verix - Special Move

- SBI BootLoader - arbitrary memory write

```
      case 'W':                                 // Download file
        v2 = -6;
        if ( dword_187A00 > 0 )
        {
          v10 = (int *)(this->paBufIn + 2);
          v11 = XDL_Recv__(this, (char *)v10, 2, 10, 1) == 2;
          while ( v11 )
          {
            v12 = *(unsigned __int8 *)v10;
            v13 = *((unsigned __int8 *)v10 + 1);
            v10 = &dword_1879F8;
            loaded_file.dataLen = (v12 << 8) + v13;
            if ( loaded_file.dataLen + 10 > this->paBufIn_size_0x400 )
              break;
            v14 = XDL_Recv__(this, this->paBufIn + 4, 4, 10, 1);
            v11 = v14 == 4;
            if ( v14 == 4 )
            {
              loaded_file.field_3A = this->paBufIn[7]
                                   + (this->paBufIn[4] << 24)
                                   + (this->paBufIn[5] << 16)
                                   + (this->paBufIn[6] << 8);
              if ( loaded_file.dataLen + 2 != XDL_Recv__(this, this->paBufIn + 8, loaded_file.dataLen + 2, 10, 1)
                || !sub_195EF6(this->paBufIn + 1, loaded_file.dataLen + 9) )
              {
                return -1;
              }
              BCM_URTx_write_char(this, 6u);
              if ( check_bootHeader(dword_187A00, this->paBufIn + 8, loaded_file.dataLen) >= 0 )
                goto LABEL_48;
              return v2;
            }
          }
        }
        break;
```

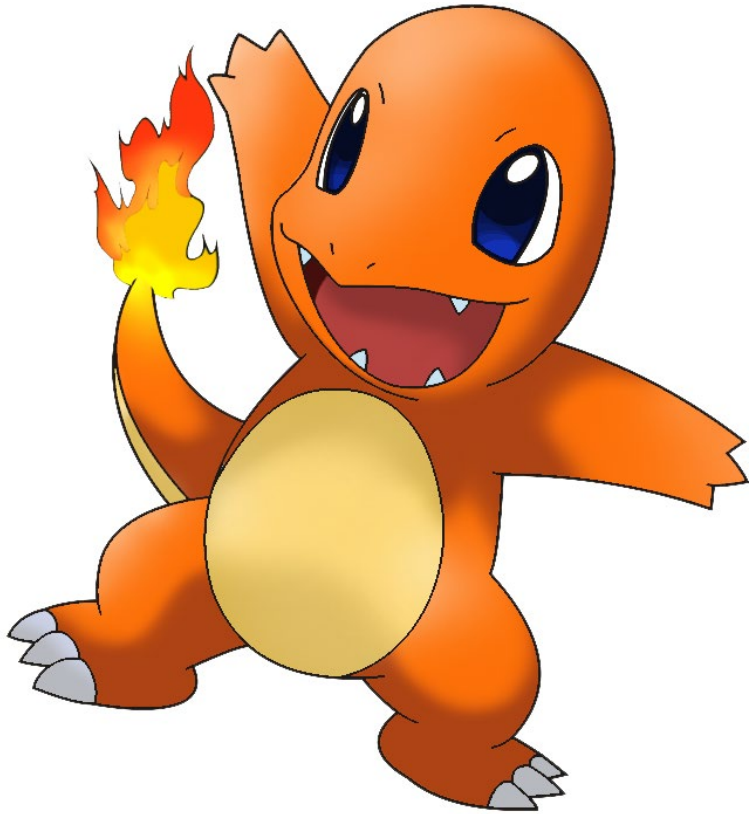
```
signed int __fastcall check_bootHeader(int a1, char *data, unsigned int len)
{
  unsigned int _len; // r4
  signed int v4; // r0
  void *v5; // r0
  unsigned int v6; // r2
  boot_hdr *v7; // r0
  struc_uploadedFiles *v9; // r0
  char *_data; // [sp+4h] [bp-1Ch]

  _data = data;
  _len = len;
  v4 = dword_1825E8;
  if ( dword_1825E8 < 0 )
  {
    v4 = loaded_file_already_contains(loaded_file.fname);
    dword_1825E8 = v4;
    if ( v4 < 0 )
      return -1;
  }
  if ( BOOT_loadAddr )
  {
    v9 = &dwFiles[v4];
    if ( !v9->loadAddr )
    {
      loaded_files_size += loaded_file.size;
      v9->loadAddr = BOOT_loadAddr;
    }
    memcpy((char *)(v9->loadAddr + XDL_file_ChunkOffset), _data, _len);
    XDL_file_ChunkOffset += _len;
    return 1;
  }
  v5 = dwFiles_alloc(XDL_file, XDL_file_ChunkOffset + _len);
  XDL_file = (boot_hdr *)v5;
  if ( !v5 )
    return -1;
  memcpy((char *)v5 + XDL_file_ChunkOffset, _data, _len);
  v6 = XDL_file_ChunkOffset + _len;
  XDL_file_ChunkOffset = v6;
  if ( v6 <= 0x40 )                             // sizeof(struct boot_hdr)
    return 1;
  if ( XDL_file->signature == 0xA19BC38F && XDL_file->type )// boot header magic
  {
    BOOT_loadAddr = XDL_file->load_addr;
    memcpy((char *)BOOT_loadAddr, (char *)XDL_file, v6);
    free(XDL_file);
    XDL_file = 0;
LABEL_12:
    dwFiles[dword_1825E8].loadAddr = BOOT_loadAddr;
    return 1;
  }
```

Verifone Verix - Special Move

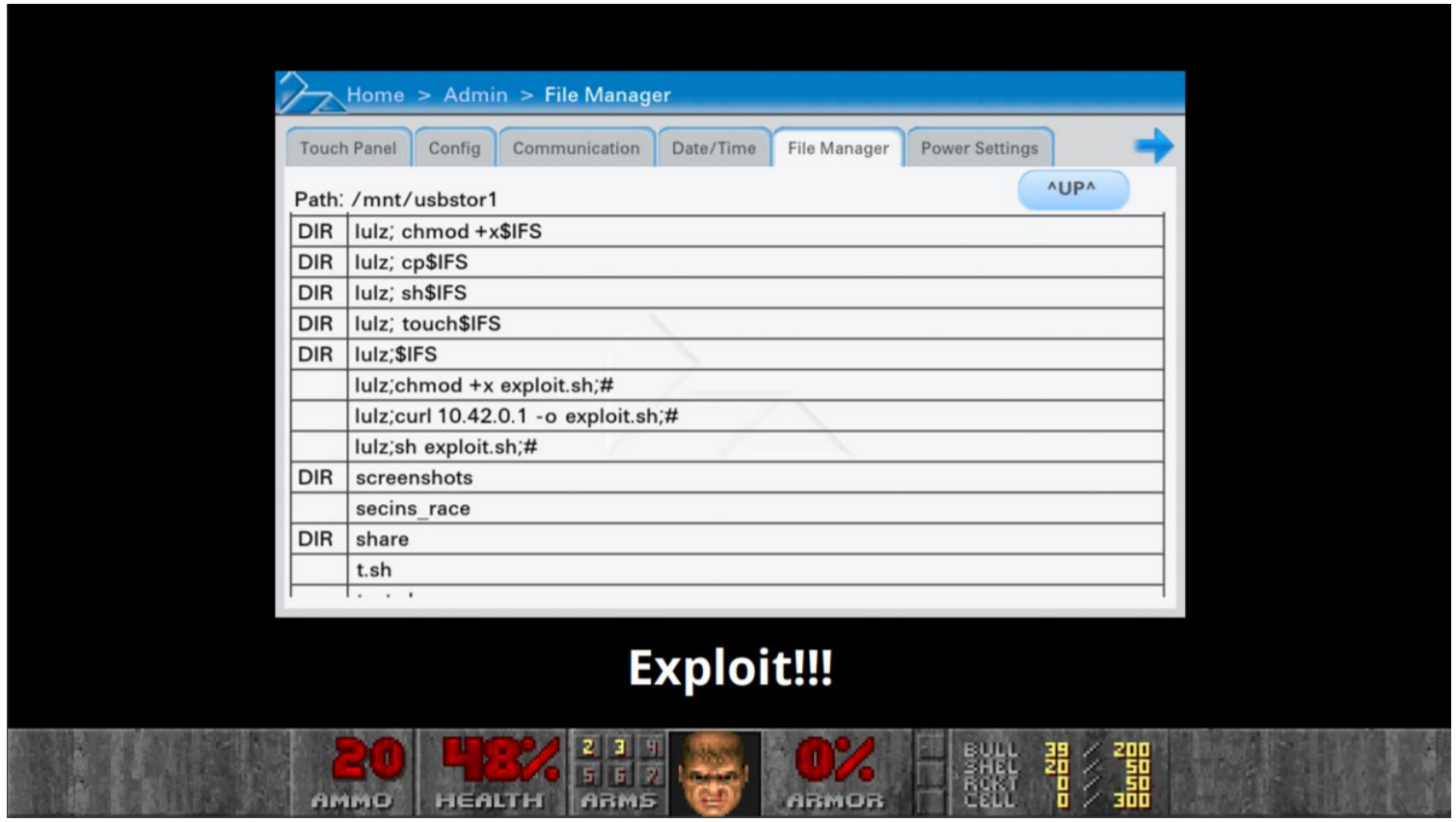
- Ancient Secure boot


```
COM8 - PuTTY

 *** DOWNLOADING IS FINISHED ***
 *** PLEASE PRESS ENTER TO RUN SCRIPT ***

==== Can't open PSSBI.SCR
==== Prompt Mode


=====================================================================

====>HELP
=====================================================================

====help - print this help
====quit - quit this prompt mode back to script
====All commands are NOT case sensitive
====File names are in (8.3) format no long file names
====Where hex number is required use this format 0x01ABCDEF
====LS     - <NAND> Show all root files and directories on USB Stick or on NAND
====LOAD   - <File name>  - Load file from the USB Stick into the RAM
====JMP    - <address in hex> - Jump into the previously loaded application
====JMPB   - <address in hex> - Jump into the previously loaded application and change to bigend
====DUMP   - <Page number> - Dump NAND page to Uart .
====ERASE  - <Start in blocks> <end in blocks>  - Erase NAND.
====DDRAM - <DDRAM option number>
====Config DDRAM <Option number >
====BURN  <File name> <start block (nand)  number in hex> - Burn file from the USB to the NAND
====BURNSBI <File name> - Burn sbi file from the USB to the NAND
====BURNP <File name> <start block><page index> - Burn file from the USB to the a specific page on NAND
====RESET <6 USB / 10 NAND > - Reset the terminal
====TRIBURN <File name> <start block (nand) number in hex> - Burn file from the USB to the NAND writing each block
  3 (3 is defined in project.def so it can be 2 and 1)  times
====MCFG <DDR Type> inits DDR get ncdl values and writes all of it to last page of each of the first 3 blocks
====RESET <6 USB / 10 NAND > - Reset the terminal
====NANDT     - Show NAND type
====BAD_BLOCKS  - List Bad NAND blocks
====BBLERASE - erase Scratch BBL
====INCLUDE <script file name>  - Run script
====DDRT - <loop counter>
====SVID - <write svid value (0..7 for production, or 8 with the DSA for development)>
====>quit
==== LOADING FROM NAND
==== Read from nand 0x715a7 bytes in 60 mili seconds
==== Vx File Auth using PedGuard

==== Authenticated 0x71567 bytes in 73 mili seconds
==== Loaded VX Module

Download Succeeded
Closing COM10
```

How to weaponize your Verifone MX



Verifone MX – DOOMed POS



https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20presentations/DEF%20CON%2025%20-%20trixr4skids-DOOMed-Point-of-Sale-Systems-UPDATED.pdf
https://www.nolanray.com/doomed-pos-systems

Verifone MX - Special Move

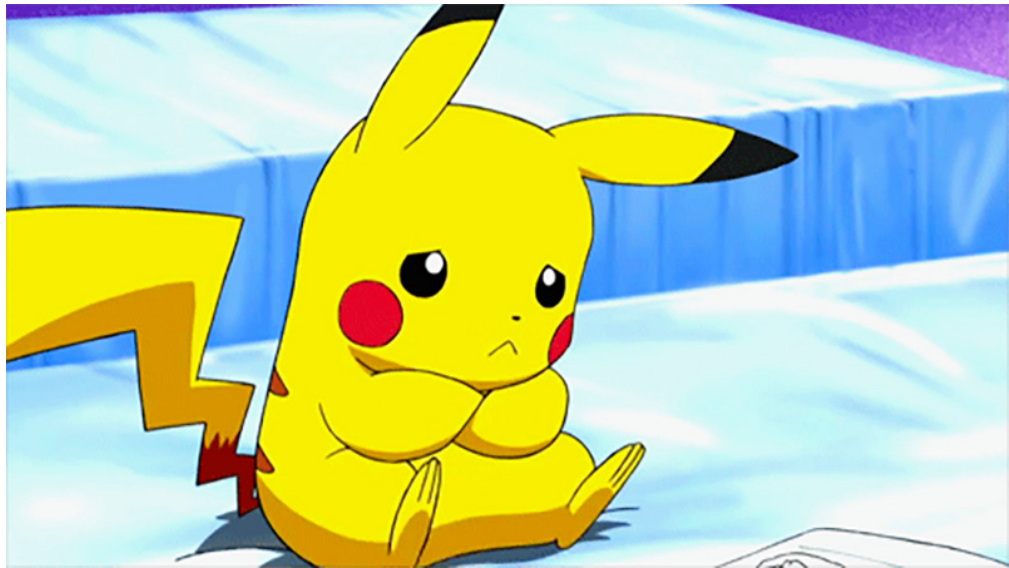
- Almost Secure boot


```
==========================================================================
==== SBI V. 03_08 (Dec  9 2013 11:08:39)
==== SEARCHING USB STICK
==== LOADING FROM NAND
==== Read from nand 0x21ba0 bytes in 24 mili seconds
==== vault File Auth using PedGuard

==== Authenticated 0x21b60 bytes in 40 mili seconds
==== Loaded Security Module
==== Read from nand 0x62200 bytes in 69 mili seconds
==== uboot File Auth using PedGuard

==== Authenticated 0x621c0 bytess in 57 mili seconds
[    0.000000] Linux version 2.6.31.14 (jenkins@rixlvbuild4) (gcc version 4.4.1 (Sourcery G++ 4.4-290) ) #1 PREEMPT Wed Apr 6
00:18:39 EEST 2016 armv6l GNU/Linux
[    0.000000] CPU: ARMv6-compatible processor [t117b365] revision 5 (ARMv6TEJ), cr=00c5387d
[    0.000000] CPU: VIPT aliasing data, VIPT aliasing instruction cache
[    0.000000] Machine: Verine VN210x Chip
[    0.000000] Memory policy: ECC disabled, Data cache writeback
[    0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 32512
[    0.000000] Kernel command line: root=/dev/ram0  ubi.mtd=2 console=tty0 console=ttyAMA0,115200nx kmemleak=off lpj=1994752
video_decoder mtdparts=vf21px_nand:384k(sbi),3712k(raw),-(system)
[    0.000000] kmemleak: Kernel memory leak detecr disabled
[    0.000000] PID hash table entries: 512 (order: 9, 2048 bytes)
[    0.000000] Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
[    0.000000] Ide-cache hash table entries: 8192 (order: 3, 32768 bytes)
[    0.000000] video_decoder found
[    0.000000] VdecHeapSize: vdec heapsize = 0x400000 bytes
[    0.000000] bootmemheap_setup: total heapsize = 0x400000 bytes = 1024 pages
[    0.000000] bootmemheap_setup: Allocated 1024 pages at 0x80507000.
[    0.000000] Memory: 128MB = 128MB total
```

POSWorld rule #1

If PCI Council doesn’t require it – we don’t need it

- Deleting firmware
- Wiping keys properly
- And more…



POSWorld rule #2

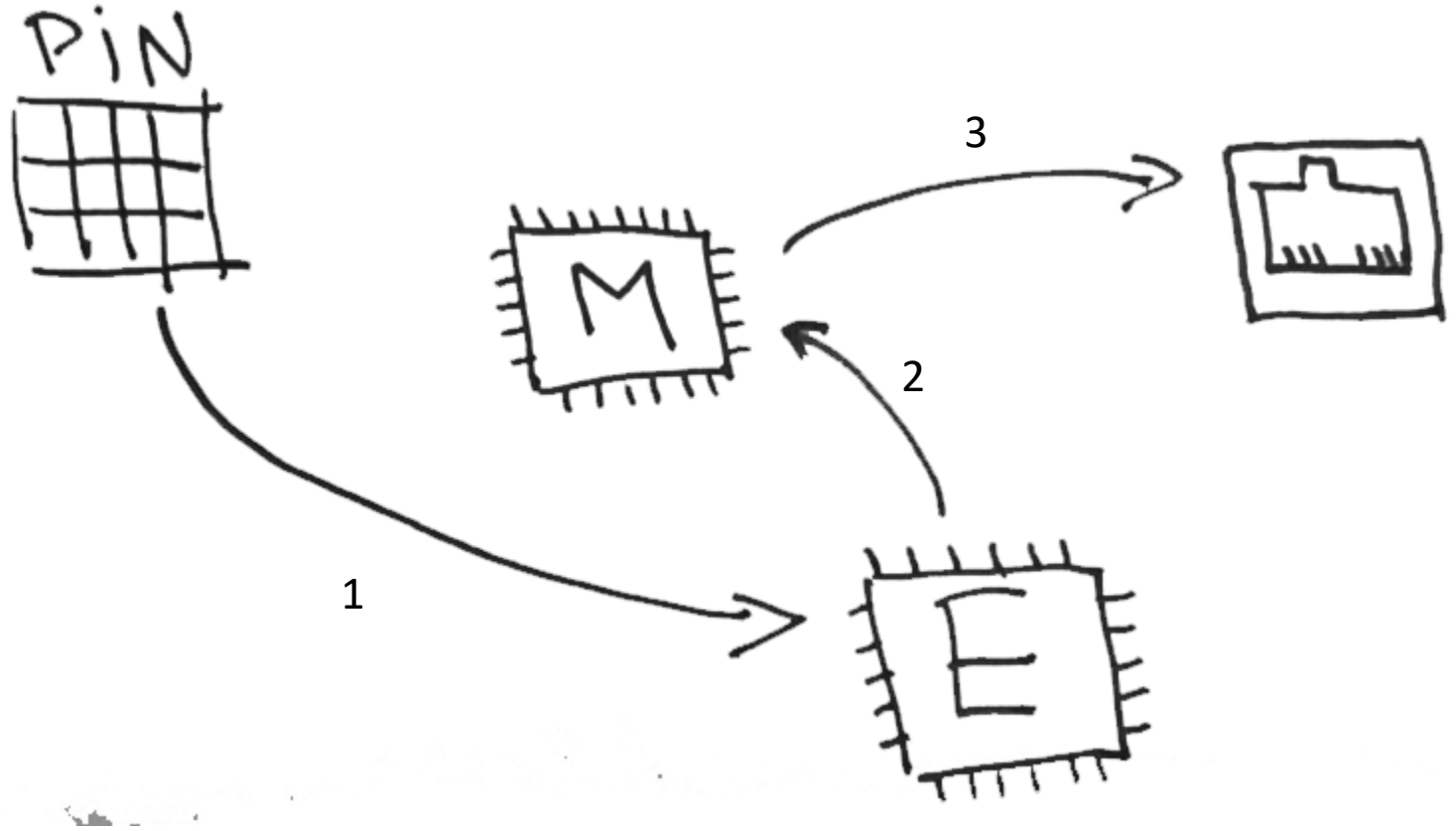
“Your proprietary hideness”

- Proprietary OS
- Proprietary outputs
- Proprietary protocols
- Special modes, killswitch combinations

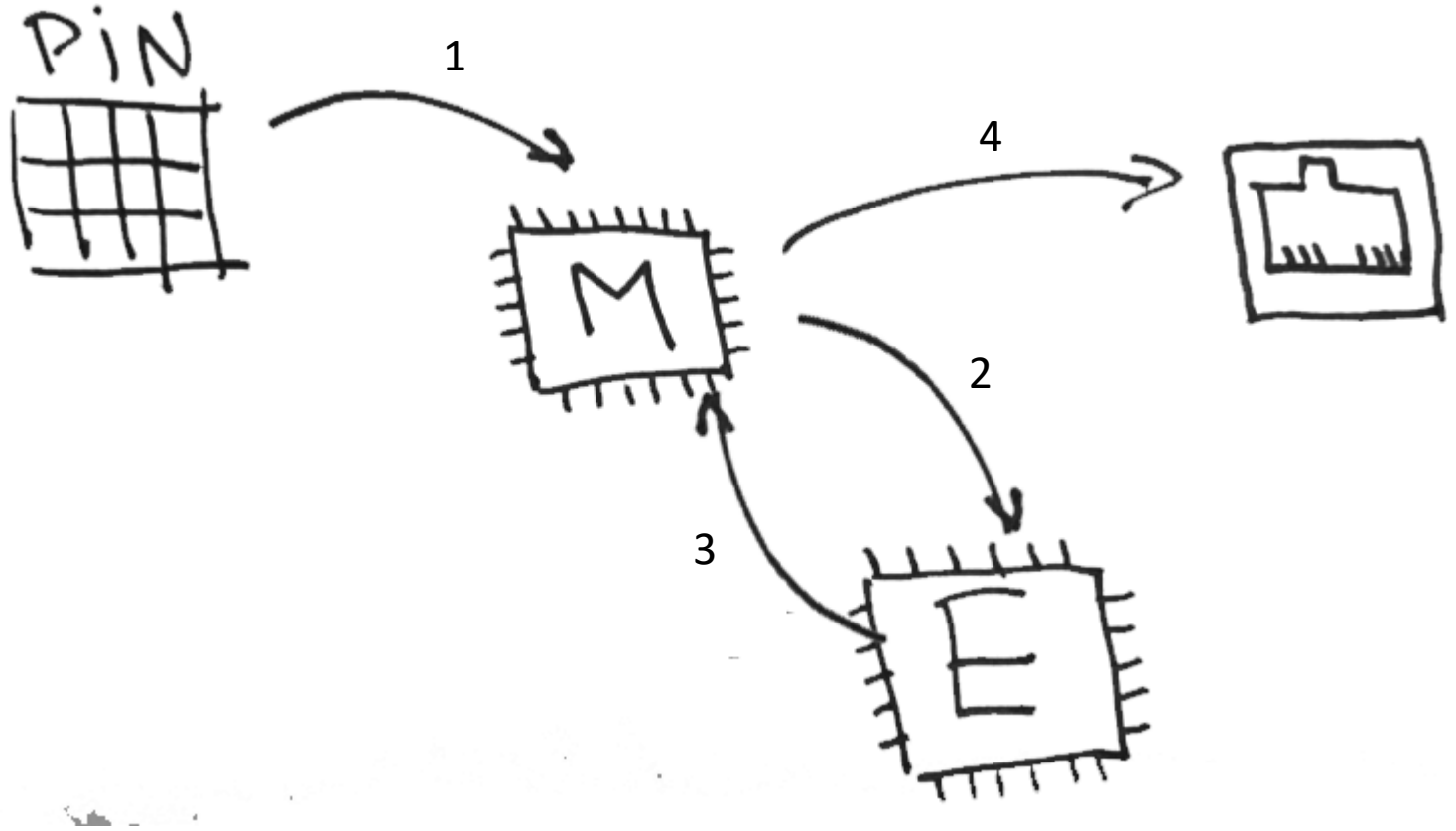
What can and can’t you do

	Verifone VX, MX series (no dedicated chip for cryptography)	Ingenico (dedicated chip for cryptography)
Send arbitrary packets	+	+
Clone cards	+	+
Clone terminals	+	-
Persistency	+	-

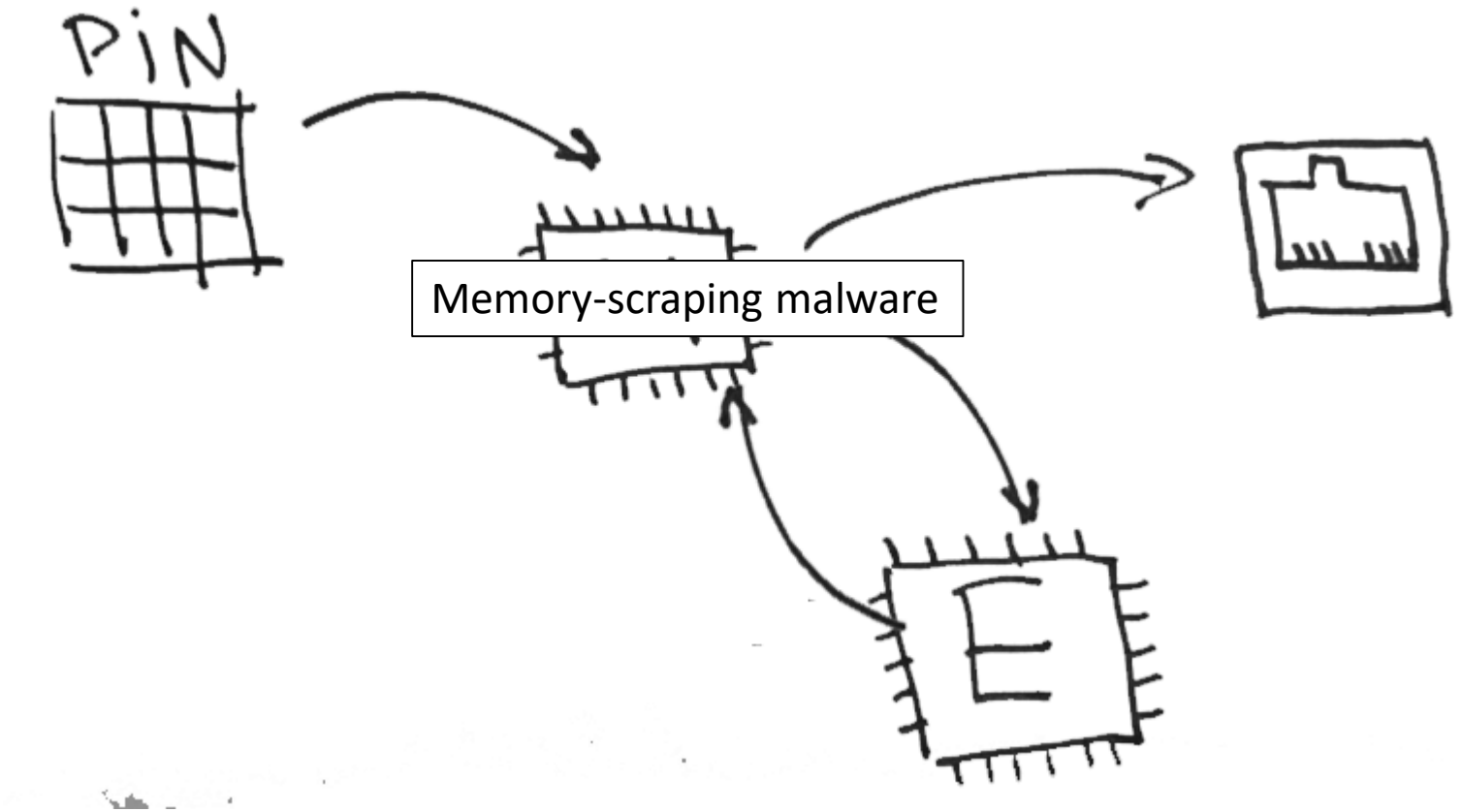
Cloning cards and PINs



Cloning cards and PINs



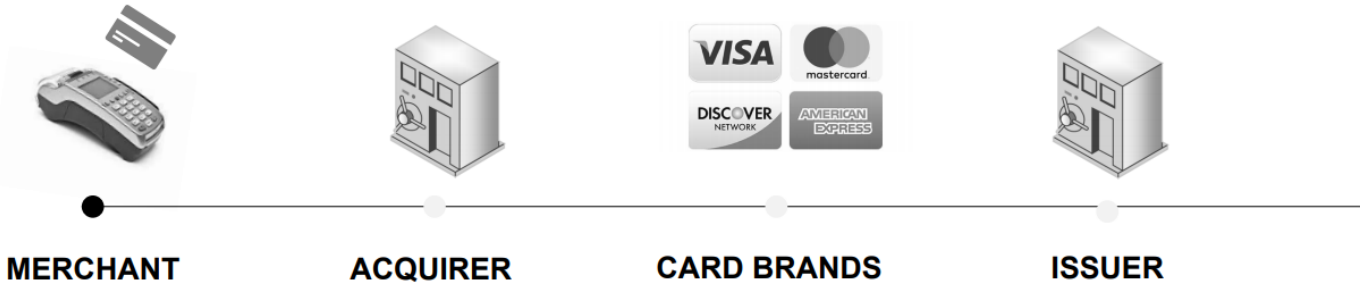
Cloning cards and PINs

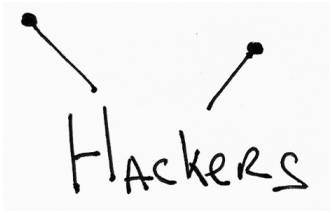


Cloning terminals

- Clone network encryption keys (located on the FS)
- PIN encryption key
- MAC key
- bbl_key – encrypted storage key
- KBPK – integrity control key

Weaponising terminals





Card Fraud in a PSD2 World: A Few Examples

https://www.cyberdlab.com/insights/card-fraud-in-a-psd2-world-a-few-example%D1%84s

TIMUR YUNUSOV
Head of Offensive Security Research

Research | Application Security

Ingenico (Telium2 OS)

- CVE-2018-17767 - Hardcoded PPP credentials
- CVE-2018-17771 - Hardcoded FTP credentials
- CVE-2018-17774 - Insecure NTPT3 protocol
- CVE-2018-17768 - Insecure TRACE protocol
- CVE-2018-17765 - Undeclared TRACE protocol commands
- CVE-2018-17766 - NTPT3 protocol - file reading restrictions bypass
- CVE-2018-17769 - Buffer overflow via the 0x26 command
- CVE-2018-17770 - Buffer overflow via the ‘RemotePutFile’ command
- CVE-2018-17772 - Arbitrary code execution via the TRACE protocol
- CVE-2018-17773 - Buffer overflow via SOCKET_TASK in the NTPT3 protocol

Verifone (MX900):

- CVE-2019-14711 - Race condition privilege escalation
- CVE-2019-14713 - Installation of unsigned packages.
- CVE-2019-14718 - Insecure Permissions
- CVE-2019-14719 - Multiple arbitrary command injections

Verifone (VerixV):

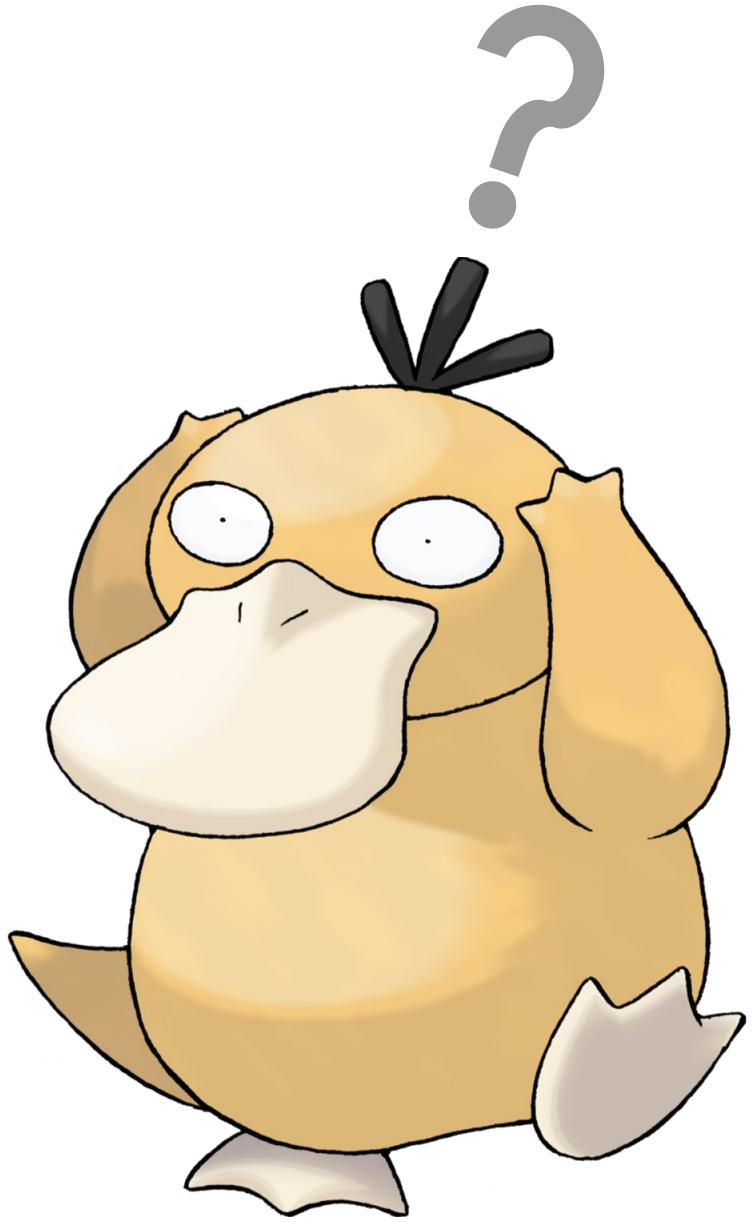
- CVE-2019-14712 - Integrity and origin control bypass
- CVE-2019-14717 - Buffer Overflow in Verix OS core
- CVE-2019-14716 - Undocumented physical access mode

Verifone (OS independent):

- CVE-2019-14715 - Undocumented physical access to the system

Open questions

- Is there more bugs in Ingenico?
- Is there more bugs in modern versions of POS?



Kudos

- Dmitry Sklyarov @_Dmit
- Egor Zaitsev @groke1105
- Vladimir Kononovich @KononovichVl
- Artem Ivachev @ivachyou
- Maxim Kozhevnikov

black hat EUROPE 2020

DECEMBER 9-10 BRIEFINGS

Questions?

@A1ex_S
@a66ot

Alexstennikov.re@gmail.com
Timur.Yunusov@cyberdlab.com

CYBER R&D LAB

POWERED BY EPAM | POSITIVE