

THE DNA OF DMA

"Anticheat is merely a suggestion"

Special Thanks

Reed (*ReadWriteError*)

For help dumping configs and helping me bypass guarded regions 40-60, project would've died otherwise.

Nick (*Oculus*)

Split cost of the board with me, testing and moral support.

Joe (*Joe*)

Writing the auto map creation scripts and advice.

Ulf (*Ulf Frisk*)

For being the guy to pioneer DMA attacks and write memprocFS / PCIlleech, all of my work is built on his initial efforts.



2600



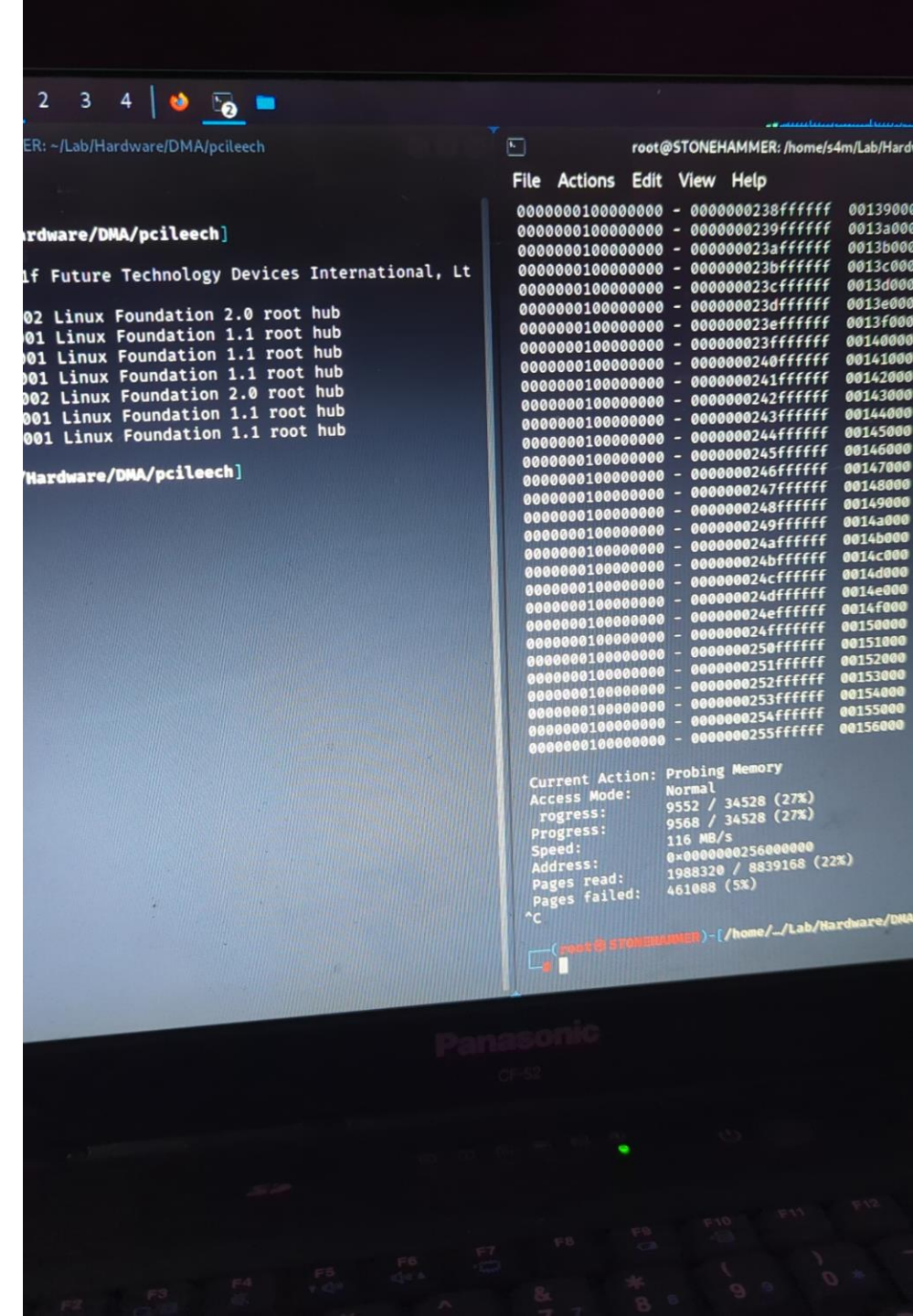
./whoami

2nd year student @ BU (Forensic Computing)

BUCSS president

Run a small server hosting business

I hack stuff..



A little background

I got my start in this field hacking games.

Instead of spending a few hundred hours getting good, decided to spend a few thousand hours learning C#

I got banned a lot...

I also learnt a lot.

The image features two large, overlapping circles. The circle on the left is a vibrant red, and the circle on the right is a dark charcoal gray. They overlap in the center, creating a shared space. The text 'Good question' is written in white within the red circle, and 'Why am I here?' is written in white within the gray circle.

Good question

Why am I here?

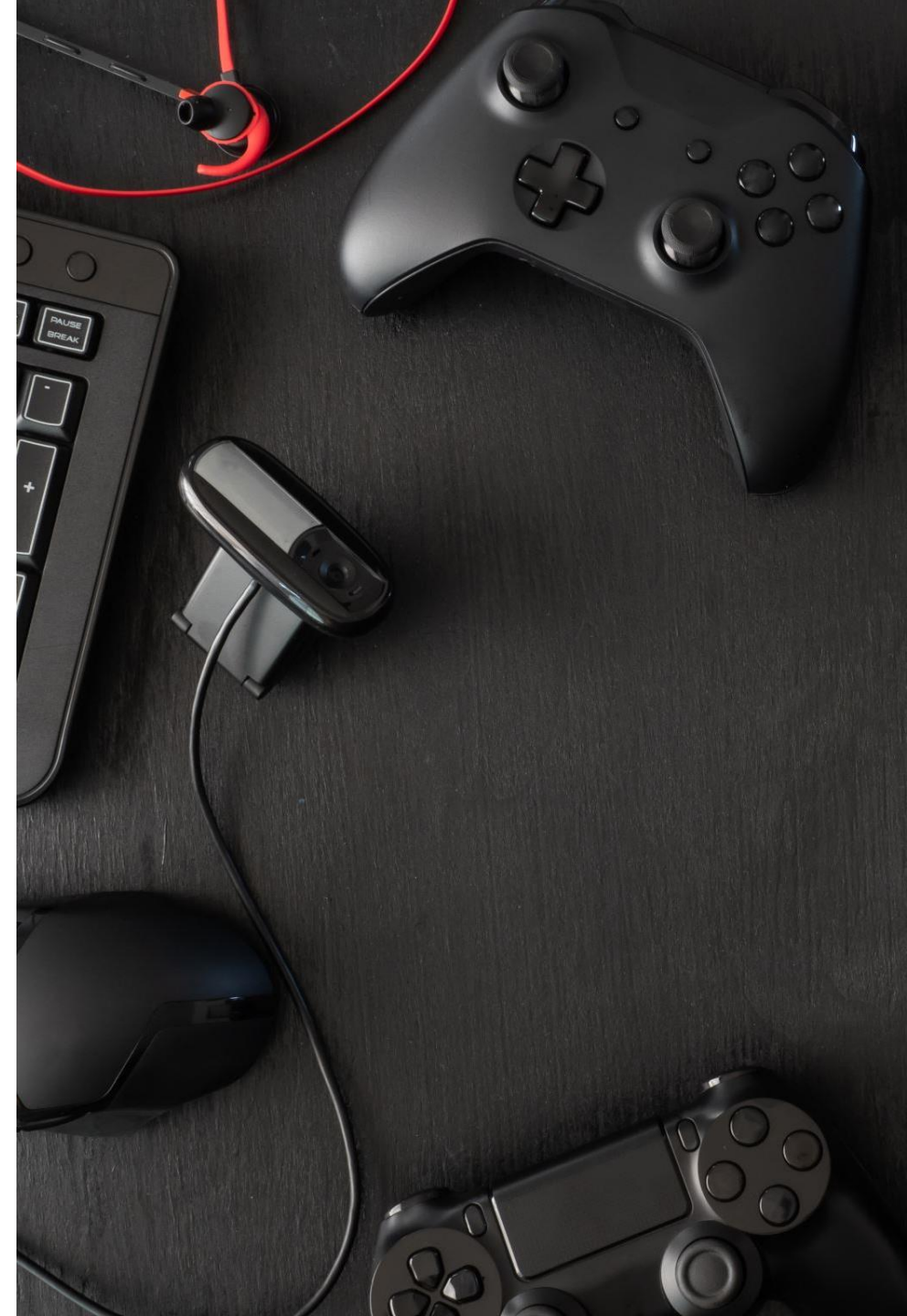
No really...

We'll go over DMA (Direct Memory Access)

See how to use DMA offensively

Learn how to bypass a billion dollar anticheat industry

The technologies potential usecases outside of games



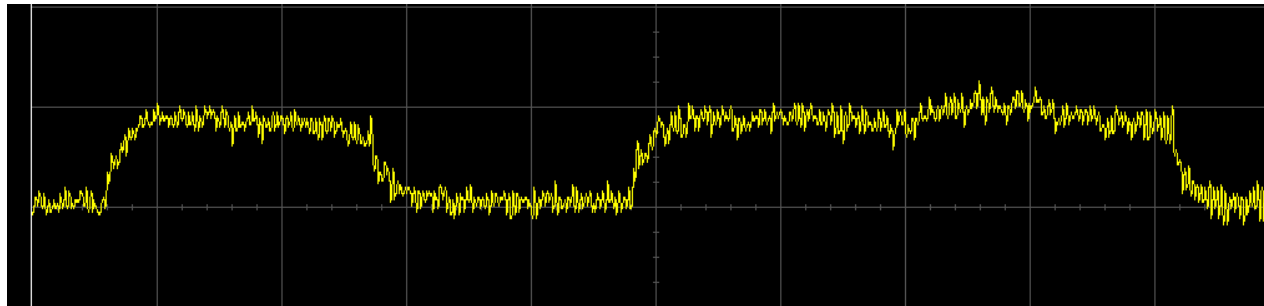
Why videogame hacking?

- Very secretive community, knowledge can be difficult to acquire.
- Generally considered taboo to talk about.
- The skills learnt hacking games can carry you for your entire career and are incredibly valuable / transferable.
- Let's break some games...

But first,
some definitions.

"Sidechannel attack" --help

- Gathering information from a system via the fundamental way a computer works, rather than a misconfiguration or exploit.
- Think knowing the distance a car has travelled from the wear on its tyres instead of hacking its GPS.



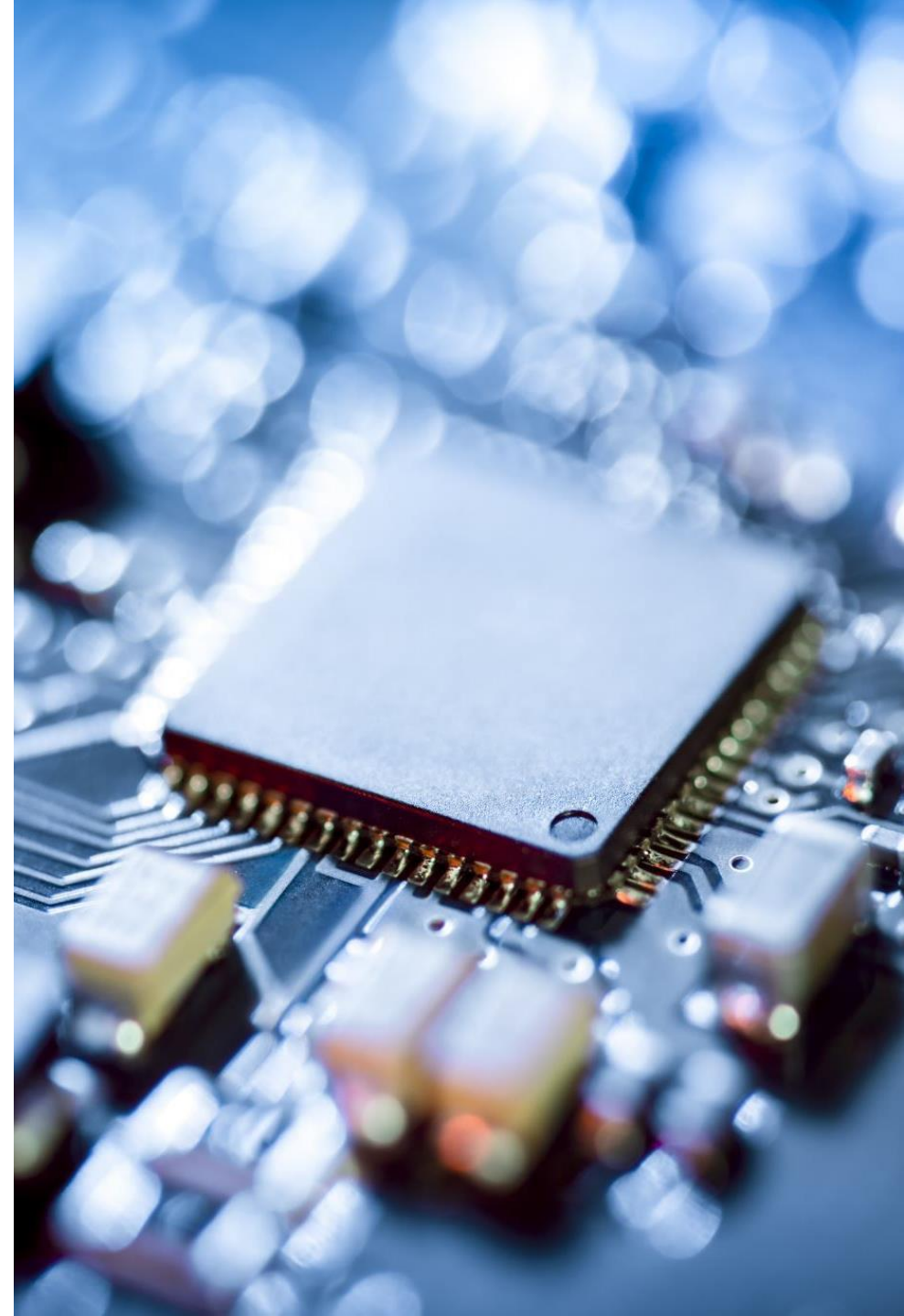
Decoding RSA key bits via CPU power draw,
(Peaks represent a 1, valleys a 0).

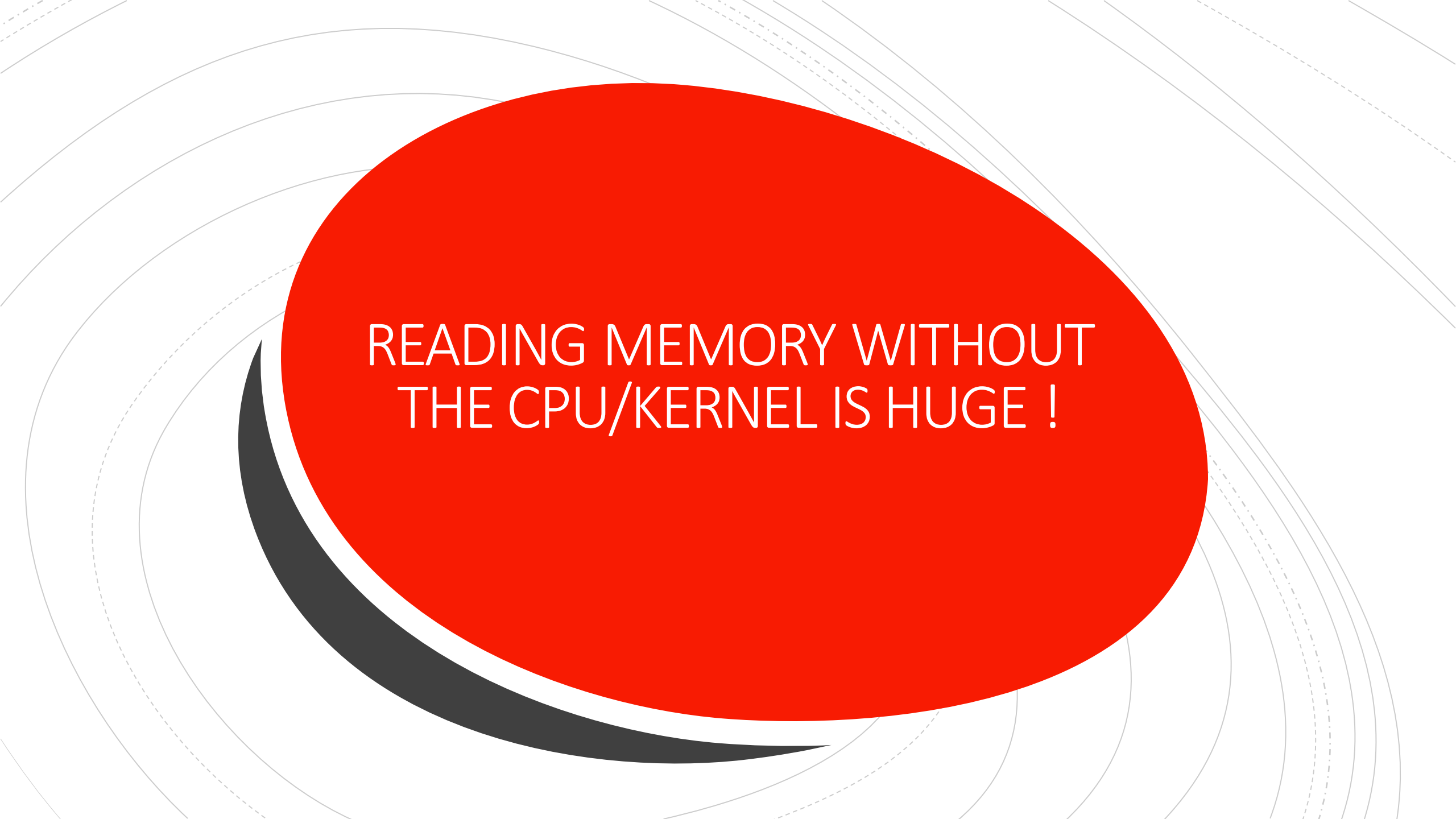
"DMA" --help

Stands for "Direct Memory Access".

Allows hardware to access memory independently of the CPU and Kernel.

Devices with DMA controllers are usually PCIE devices or thunderbolt (Can Also be Firewire).

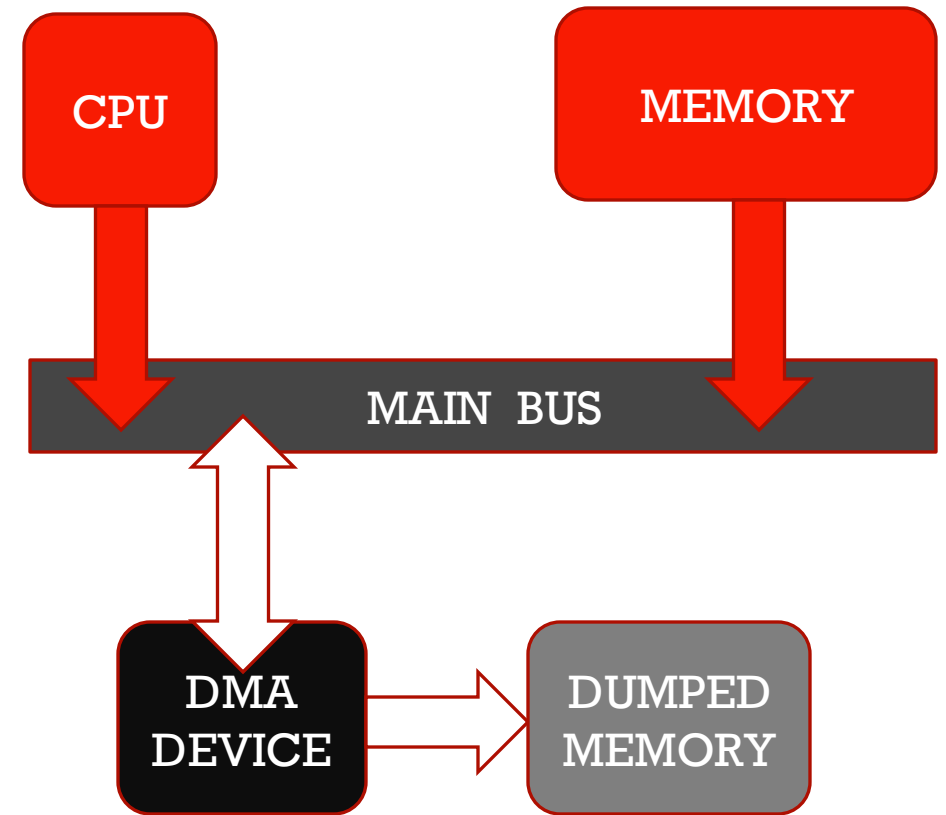


The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large, vibrant red speech bubble is positioned in the center-right of the frame. Inside this bubble, the text "READING MEMORY WITHOUT THE CPU/KERNEL IS HUGE !" is written in a clean, white, sans-serif font, centered both horizontally and vertically.

READING MEMORY WITHOUT
THE CPU/KERNEL IS HUGE !

Let's break this down

- No CPU, no kernel, no OS.
- Can freely sniff bus / memory.
- Fundamentally undetectable when reading memory.
- Can also freely write to memory.



Cool, now what?

Read memory live

Export keys or passwords

Remediate ransomware

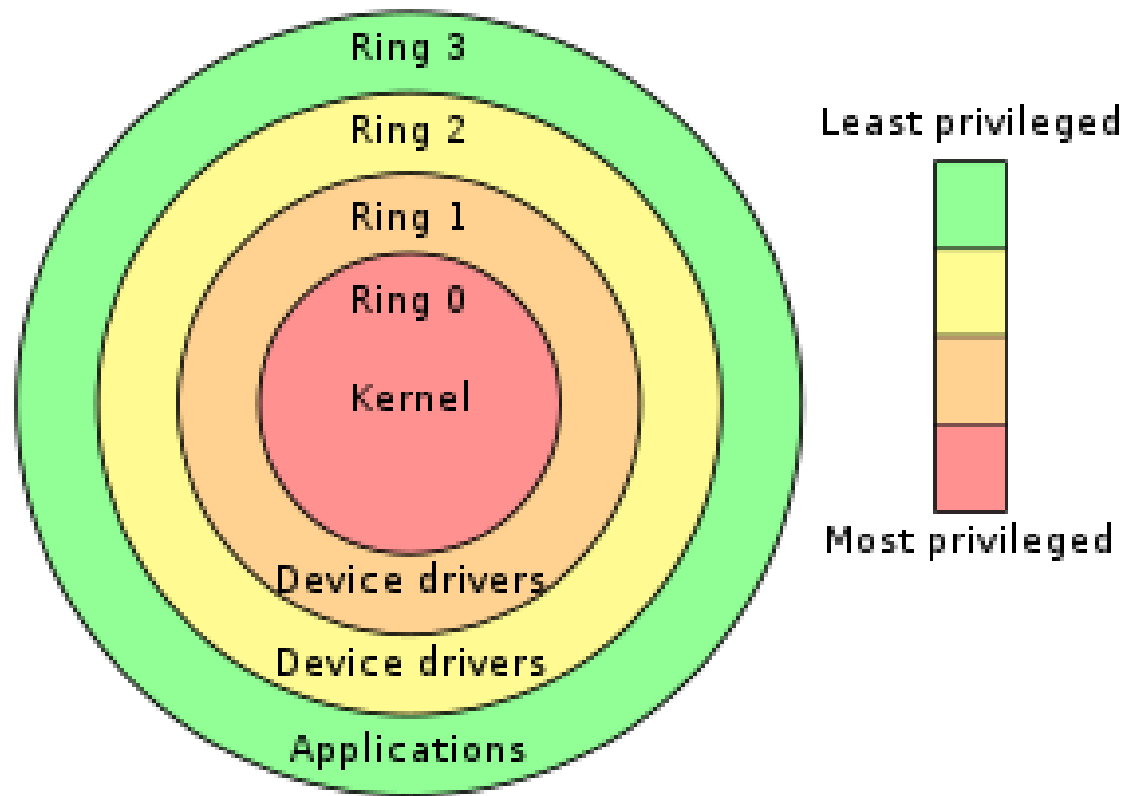
Break full disk encryption

Or if you're me, hack videogames...





HOW TO MAKE AN
ANTICHEAT DEVELOPER SWEAT:

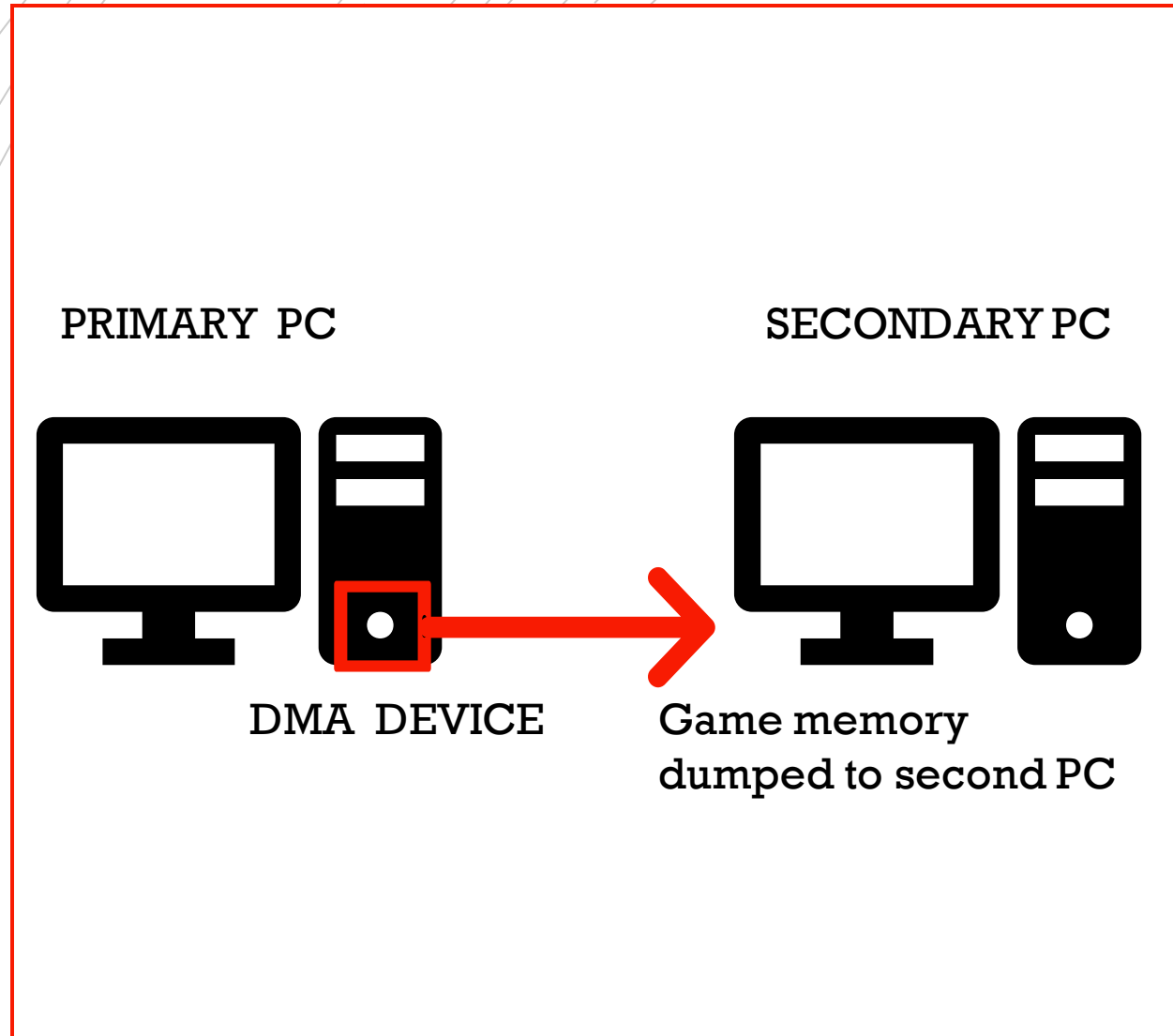


Anticheat?

- Modern videogames are protected by software called "anticheat" (ESEA, FACET, EAC, Battleye, etc) and is the biggest challenge for cheat/hack developers today.
- The most advanced are "kernel level" / ring0, and are very effective at detecting people hacking in their games.
- Kernel level however, is still bound by the CPU, so we can bypass it, we'll say our DMA device is "ring -1"
- For the following example, we'll be bypassing Battleye.

The setup

- For this example, our BE protected game is EFT (Escape from Tarkov), we'll need two PCs.
- The first PC is "clean" only playing our game of choice, with the anticheat installed (Game will not run otherwise).
- Second PC does not have game or anticheat installed, but receives game memory from first PC via the DMA card.
- Second PC will be the one to actively hack games memory from received memory dumps.





Shopping list

First we need hardware



We'll be using an M2/PCIE
Screamer

Cheap compared
to other FPGA /
DMA boards

Good I/O
speeds

Good
documentation

Firmware

Anticheat companies spend a lot of money detecting people doing mischief like this.

A stock card like this would be instantly detected due to;

Device **ID**

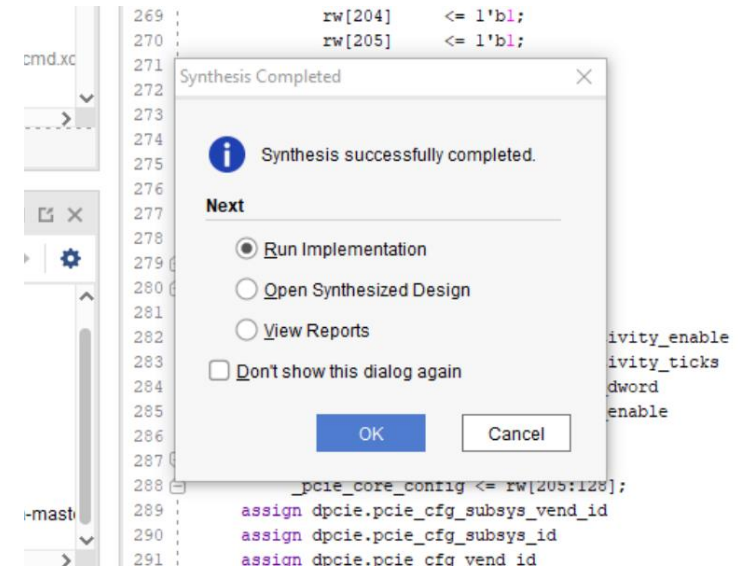
Vendor **ID**

Config

Shadow
config

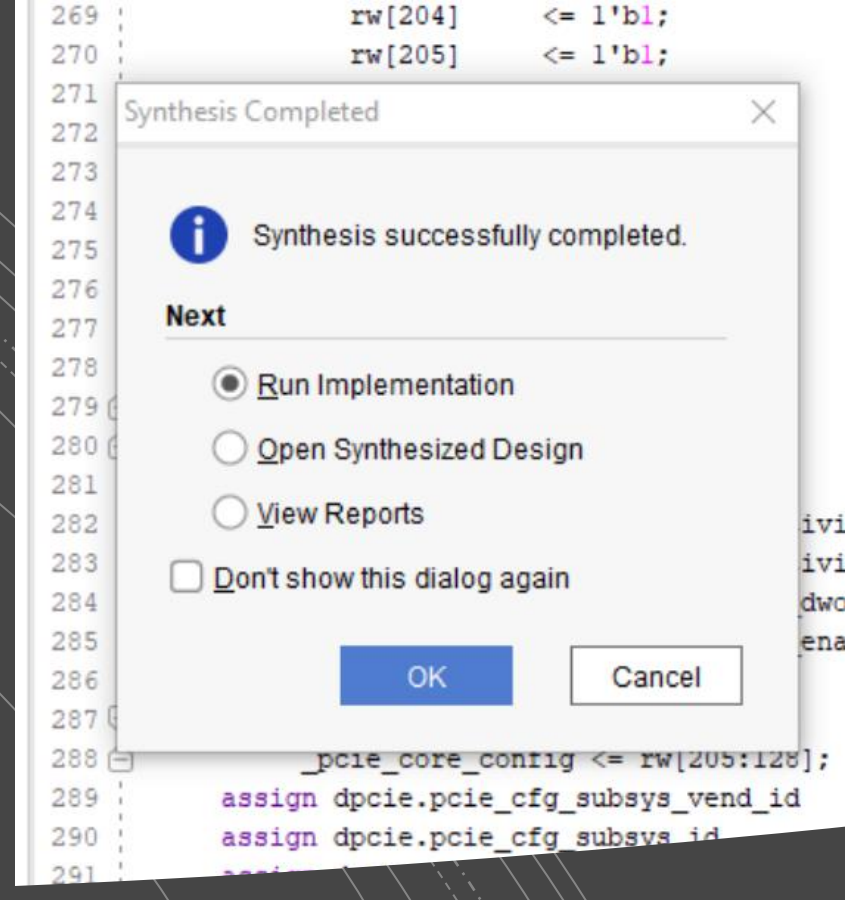
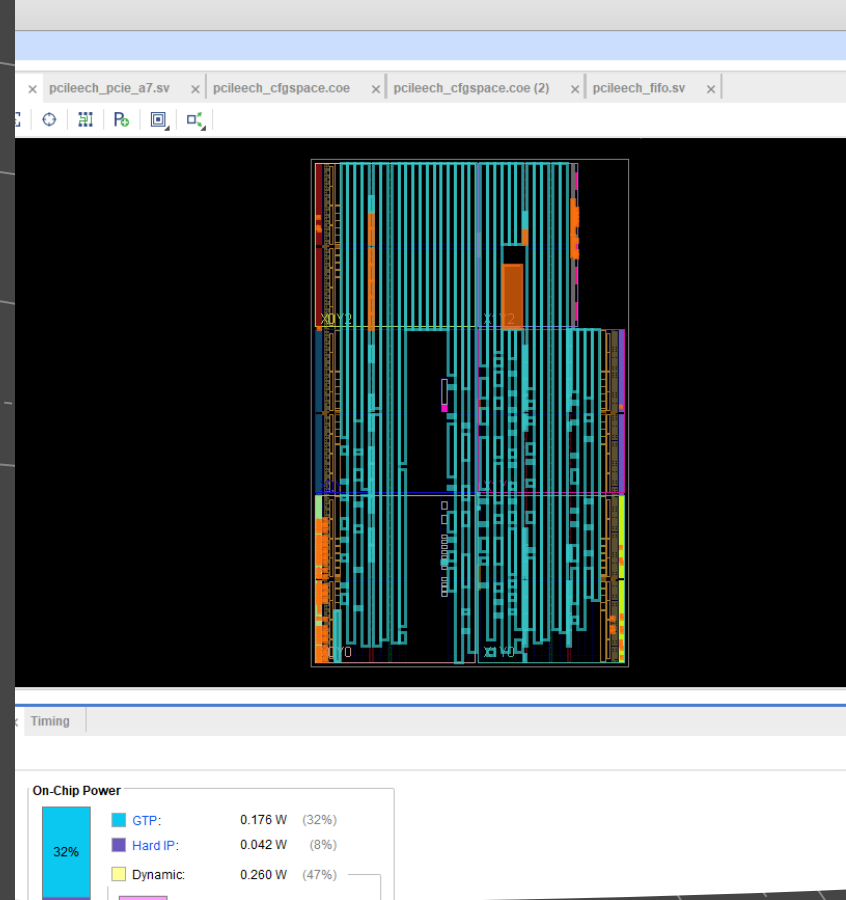
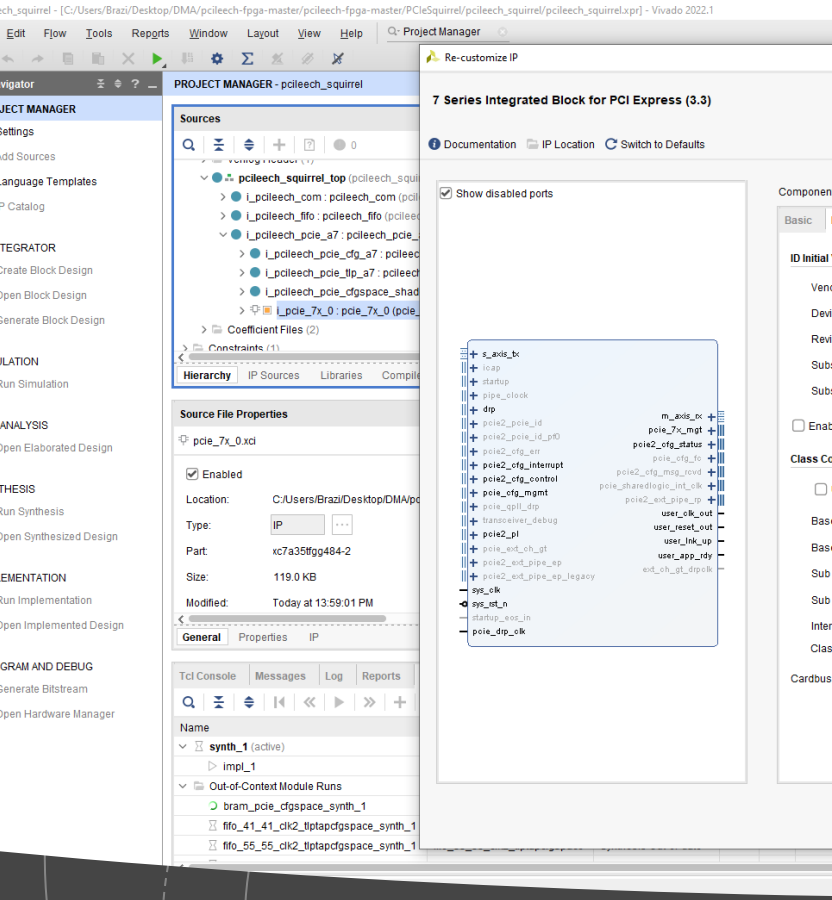
Master
Abort flag

Basic	IDs	BARs	Core Capabilities	Link Registers	Interrupts
ID Initial Values					
Vendor ID	10EE				Range: 0000..FFFF
Device ID	0666				Range: 0000..FFFF
Revision ID	02				Range: 00..FF
Subsystem Vendor ID	10EE				Range: 0000..FFFF
Subsystem ID	0007				Range: 0000..FFFF
<input type="checkbox"/> Enable ID Interface					





WE NEED CUSTOM FIRMWARE



But how?

- We'll "borrow" the stock firmware and modify it
- Spoof card to appear as something innocuous
- Firmware synthesis can take some time, and there is little documentation.


```
02:00.0 USB controller: Fresco Logic FL1100 USB 3.0 Host Controller (rev 10) (prog-if 30 [XHCI])  
Subsystem: Fresco Logic FL1100 USB 3.0 Host Controller  
Flags: bus master, fast devsel, latency 0, IRQ 17  
Memory at a3400000 (64-bit, non-prefetchable) [size=64K]  
Memory at a3411000 (64-bit, non-prefetchable) [size=4K]  
Memory at a3410000 (64-bit, non-prefetchable) [size=4K]  
Capabilities: <access denied>  
Kernel driver in use: xhci_hcd  
Kernel modules: xhci_pci
```

```
04:00.0 SD Host controller: Fresco Logic FL1100 USB 3.0 Host Controller (rev 02)  
Subsystem: Fresco Logic Device 0007  
Flags: fast devsel, IRQ 18  
Memory at a3300000 (32-bit, non-prefetchable) [size=16K]  
Capabilities: <access denied>  
Kernel modules: sdhci_pci
```

Firmware complete

- Disabled master abort, changed 40-60, etc
- Dumped config of a legitimate device
- Used legit config and variables for the DMA card
- DMA card now looks like a PCIE USB controller

Flashing

So you have the hardware and the firmware.

Now you need to combine (IE: openOCD).

Flashing the firmware onto the device is usually instant

If it isn't, you've bricked it (Say goodbye to £350.00).

```
Info : sector 0 took 100 ms
Info : sector 1 took 103 ms
Info : sector 2 took 104 ms
Info : sector 3 took 105 ms
Info : sector 4 took 105 ms
Info : sector 5 took 104 ms
Info : sector 6 took 101 ms
Info : sector 7 took 102 ms
Info : sector 8 took 114 ms
Info : sector 9 took 107 ms
Info : sector 10 took 122 ms
Info : sector 11 took 118 ms
Info : sector 12 took 106 ms
Info : sector 13 took 97 ms
Info : sector 14 took 122 ms
Info : sector 15 took 106 ms
Info : sector 16 took 101 ms
Info : sector 17 took 102 ms
Info : sector 18 took 99 ms
Info : sector 19 took 100 ms
Info : sector 20 took 103 ms
Info : sector 21 took 108 ms
Info : sector 22 took 89 ms
Info : sector 23 took 103 ms
Info : sector 24 took 130 ms
shutdown command invoked
```

```
(s4m@STONEHAMMER)-[~/Lab/Hardware/DMA/FLASHING]
$ echo "very nice"
very nice
```

```
(s4m@STONEHAMMER)-[~/Lab/Hardware/DMA/FLASHING]
$
```

Now what?

- Assuming the flash was successful, great, you have a working DMA setup.
- But that's all you have, working DMA can, by itself, it can only dump memory with PCILEECH.
- Still need to have something on the second PC to do the actual "hacking".

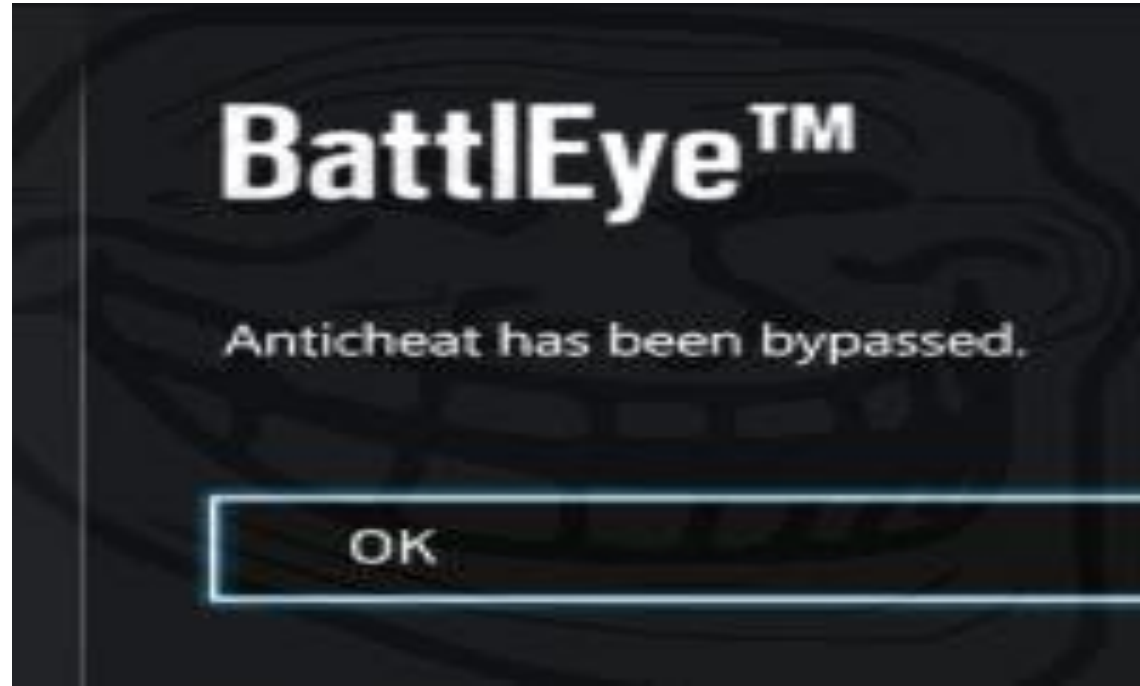
PCILEECH dumping memory of the first PC from a laptop.



The hack

- You have a working DMA setup, this on its own only dumps memory.
- Still need to make some kind of hack on the second PC, usually a radar with DMA setups.
- Either "borrow" existing public code and modify it to work with memprocfs / pcileech or write your own hack.
- Only detection vector was the DMA card, which we've already dealt with.
- If all goes well...

Congratulations,
you just bypassed
a billion dollar
industry



- Can now do anything that involves reading memory, sky is the limit.
- Only real mitigation for Anticheat companies is enforced IOMMU, which is not practical.
- Forbidden knowledge, use it wisely.

Taking it further...

- We're strictly reading memory right now.
- DMA hacks are usually restricted to radars on a 2nd monitor.
- EG: wallhacks / aimbot require writing to memory, which we want to avoid doing, as it would lead to possible detection.
- What if we could wallhack or aimbot without writing memory?
- HDMI injection could hold the answer (IE: netcatV2).
- Or spoofing HID inputs with an arduino for an aimbot.
- A little outside the scope of this talk, but food for thought...



Future applications

Proposal no.1




- Breaking full disk encryption
 - - Retrieve data from victim PC via DMA
 - - Bypass certain TPM based encryption
 - - Has been done before
 - - Currently only available to 3 letter agencies with incredibly expensive hardware
 - - Could achieve the same thing with the setup you just saw

Proposal no.2



- Immediate Ransomware Remediation
 - - Use DMA card as memory monitoring tool
 - - Constantly dump memory of monitored device to secondary / air gapped machine
 - - Retrieve keys used for encryption during attack
 - - Would only work for certain encryption schema
 - - Would be expensive to implement large scale

Thank you for
coming to my 
TED talk

FIN

