



PMFault: Faulting and Bricking Server CPUs through Management Interfaces

Or: A Modern Example of Halt and Catch Fire

Zitai Chen (Z.Chen@pgr.bham.ac.uk / zitaichen@outlook.com)

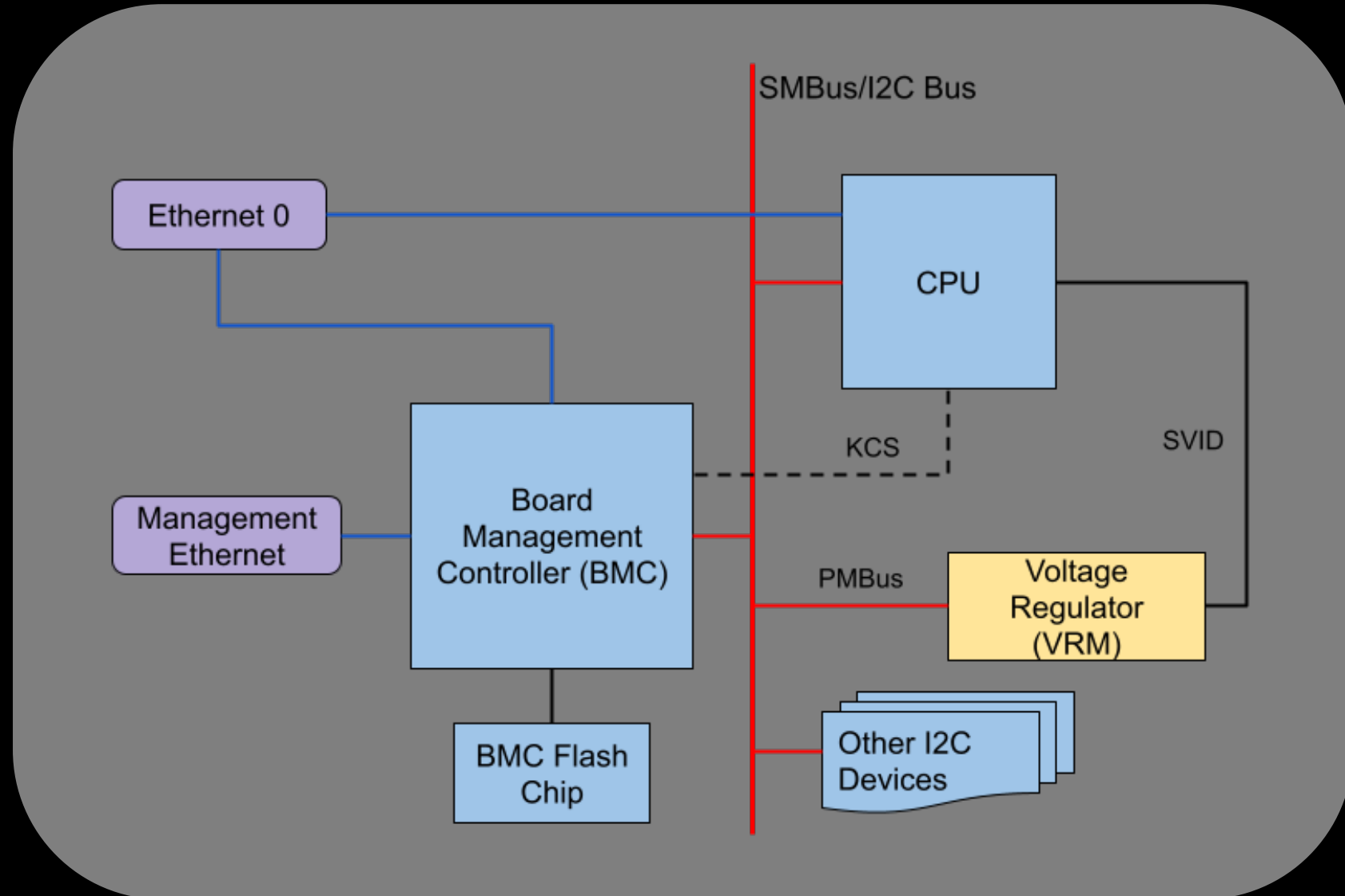
David Oswald (d.f.oswald@bham.ac.uk)

University of Birmingham

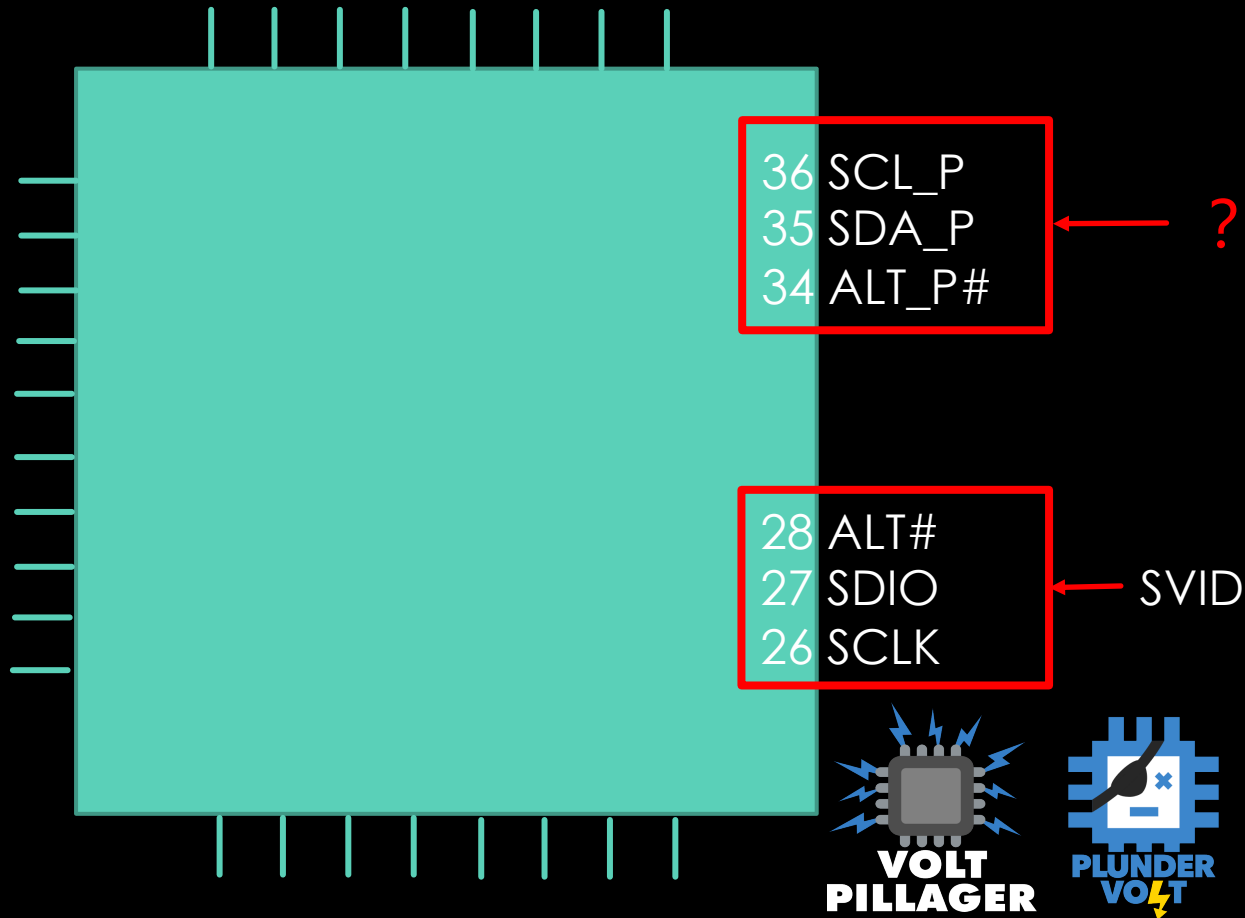
Adversary model

- Privileged software attacker, root on the host CPU.
 - Standard adversary model in the case of TEEs
 - Realistic in the case of overvolting to permanently destroy the CPU (ransomware)
- Do not require physical access (for additional hardware to be added to the system)

Connections on a server motherboard

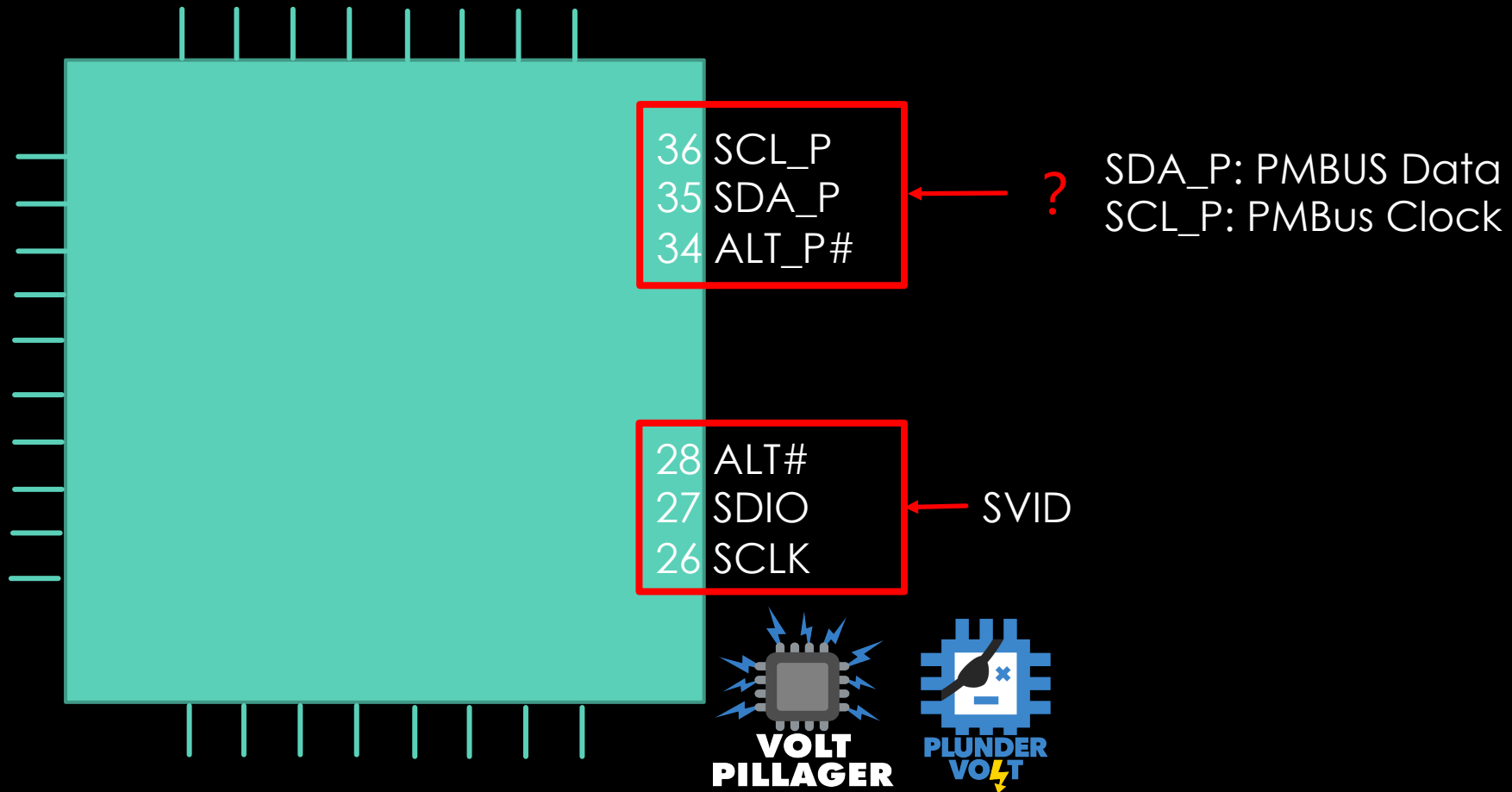


What is PMBus?



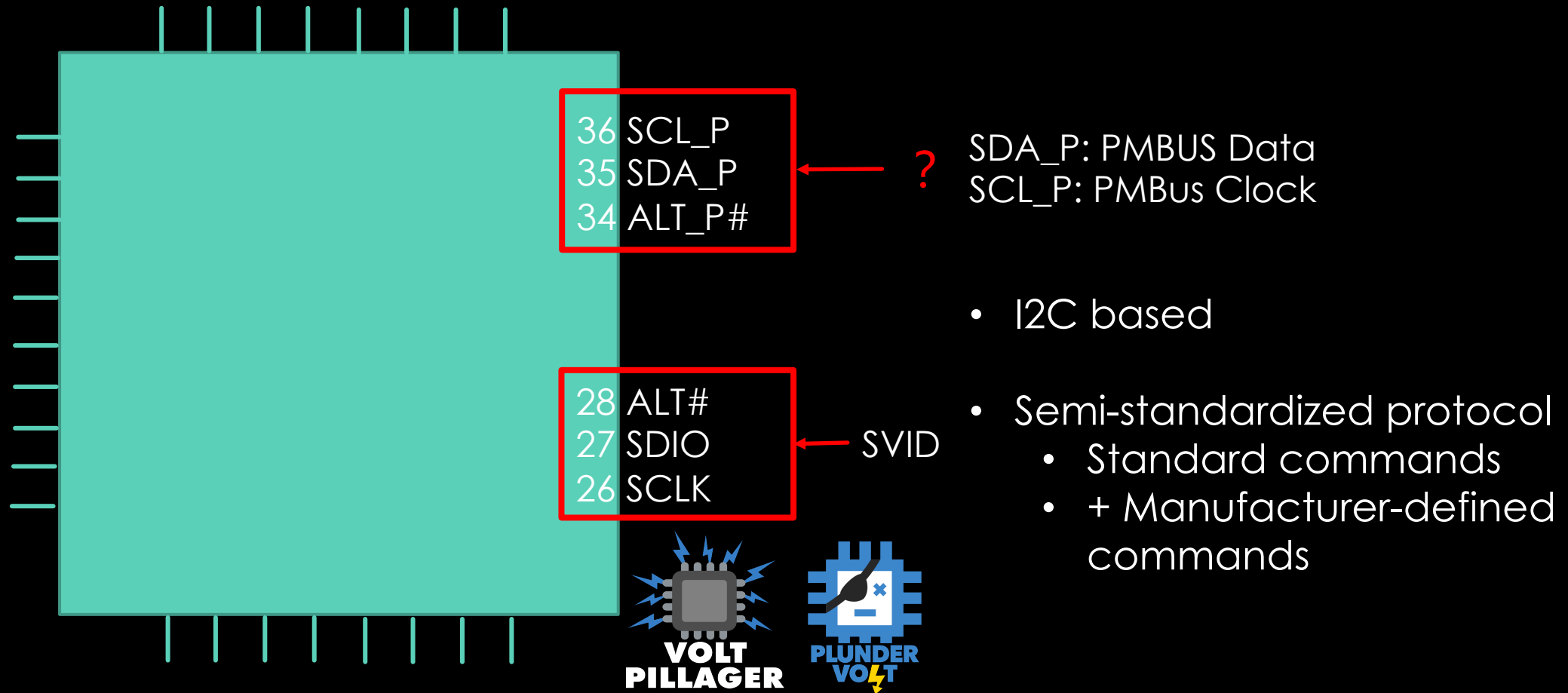
Ref: [MP2965 DataSheet](#) (Supermicro X11SSL-CF server motherboard uses MP2955)

What is PMBus?



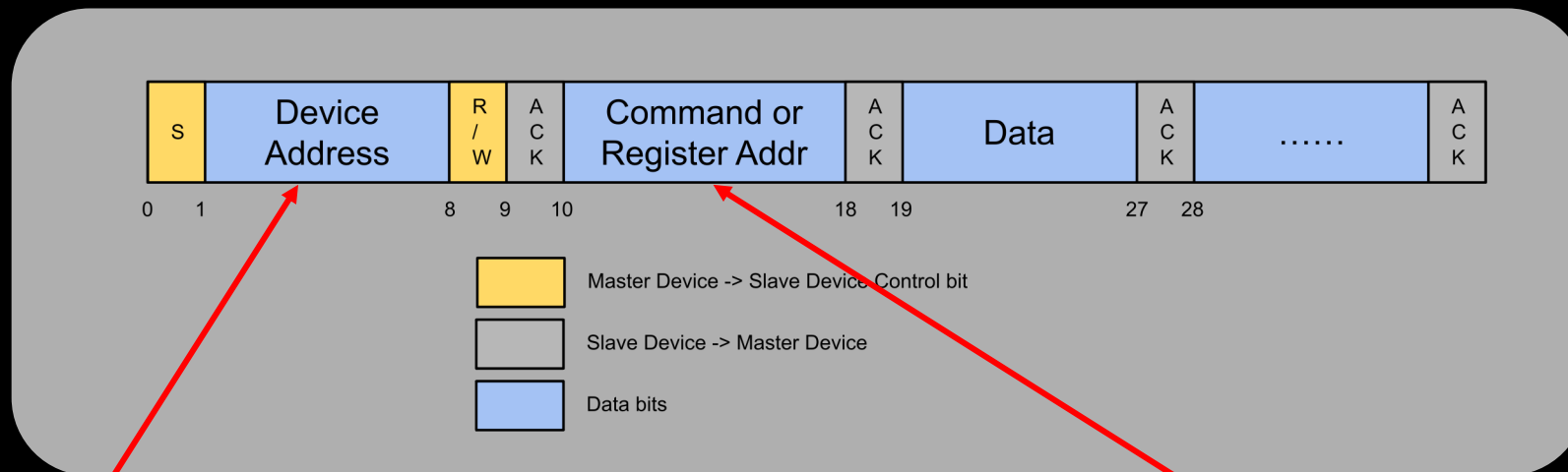
Ref: [MP2965 DataSheet](#) (Supermicro X11SSL-CF server motherboard uses MP2955)

What is PMBus?



Ref: [MP2965 DataSheet](#) (Supermicro X11SSL-CF server motherboard uses MP2955)

Packet structure



Each device is assigned a 7-bit address
What is the address for VRM?

From **PMBus Spec**
and **MP2965 VRM** datasheet

Experiment 0: From CPU? What is the VRM address?

```
~$ sudo modprobe i2c_i801
~$ sudo i2cdetect 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
[00-20]: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:      -- -- -- -- -- -- -- 37 -- -- -- -- -- -- --
40:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:      50 -- -- -- -- -- -- -- 58 -- -- -- -- -- --
60:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
~$ sudo i2cdetect 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:      -- -- -- -- -- -- -- 08 -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- 19 -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: 30 -- -- -- -- 35 36 -- -- -- -- -- -- -- -- --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- -- -- --
50: -- 51 -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

- 12 devices - Which one looks like VRM?

Experiment 0: From CPU? What is the VRM address?

```
~$ sudo modprobe i2c_i801
~$ sudo i2cdetect 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
[00-20]: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:      -- -- -- -- -- -- -- 37 -- -- -- -- -- -- --
40:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:      50 -- -- -- -- -- -- -- 58 -- -- -- -- -- --
60:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

~$ sudo i2cdetect 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:      -- -- -- -- -- -- -- 08 -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- 19 -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: 30 -- -- -- -- 35 36 -- -- -- -- -- -- -- -- --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- -- -- --
50: -- 51 -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

- 12 devices - Which one looks like VRM?
 - Response to common PMBus commands
 - The value returned make sense

Experiment 0: From CPU? What is the VRM address?

```
~$ sudo modprobe i2c_i801
~$ sudo i2cdetect 0
   0 1 2 3 4 5 6 7 8 9 a b c d e f
[00-20]: -- -- -- -- -- -- -- -- --
30:      -- -- -- -- -- -- 37 -- -- --
40:      -- -- -- -- -- -- -- -- --
50:      50 -- -- -- -- -- -- 58 -- --
60:      -- -- -- -- -- -- -- -- --
70:      -- -- -- -- -- -- -- -- --

~$ sudo i2cdetect 1
   0 1 2 3 4 5 6 7 8 9 a b c d e f
00:      -- -- -- -- -- -- 08 -- -- --
10: 10 -- -- -- -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- --
30: 30 -- -- -- -- 35 36 -- -- --
40: -- -- -- -- 44 -- -- -- -- --
50: -- 51 -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- --
```

- 12 devices - Which one looks like VRM?
 - Response to common PMBus commands
 - The value returned make sense

`READ_VOUT() < 0.55V`
`&& MFR_ADDR_PMBUS == ADDR`



Experiment 0: From CPU? What is the VRM address?

```
~$ sudo modprobe i2c_i801
~$ sudo i2cdetect 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
[00-20]: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:      -- -- -- -- -- -- -- 37 -- -- -- -- -- -- --
40:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:      50 -- -- -- -- -- -- 58 -- -- -- -- -- -- --
60:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:      -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

~$ sudo i2cdetect 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:      -- -- -- -- -- -- -- 08 -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- 19 -- -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: 30 -- -- -- -- 35 36 -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- 44 -- -- -- -- -- -- -- -- -- -- --
50: -- 51 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

- 12 devices - Which one looks like VRM?
 - Response to common PMBus commands
 - The value returned make sense

`READ_VOUT() < 0.55V`
`&& MFR_ADDR_PMBUS == ADDR`



- Next: Change the voltage!

Experiment 0: From CPU? Undervolt it!

With libi2c – library for sending commands on I2C bus

1. PMBus Override Mode -> REG_VOUT_OPERATION
2. Target Voltage -> REG_VOUT_COMMAND
3. SVID_OVERCLK2_EN (Bit 3) -> REG_MFR_VR_CONFIG

Experiment 0: From CPU? Undervolt it!

With libi2c – library for sending commands on I2C bus

1. PMBus Override Mode -> REG_VOUT_OPERATION
2. Target Voltage -> REG_VOUT_COMMAND
3. SVID_OVERCLK2_EN (Bit 3) -> REG_MFR_VR_CONFIG



Experiment 0: From CPU? Undervolt it!

With libi2c – library for sending commands on I2C bus

1. PMBus Override Mode -> REG_VOUT_OPERATION
2. Target Voltage -> REG_VOUT_COMMAND
3. SVID_OVERCLK2_EN (Bit 3) -> REG_MFR_VR_CONFIG



Stall... 🤪

Experiment 0: From CPU? Undervolt it!

With libi2c – library for sending commands on I2C bus

1. PMBus Override Mode -> REG_VOUT_OPERATION
2. Target Voltage -> REG_VOUT_COMMAND
3. SVID_OVERCLK2_EN (Bit 3) -> REG_MFR_VR_CONFIG

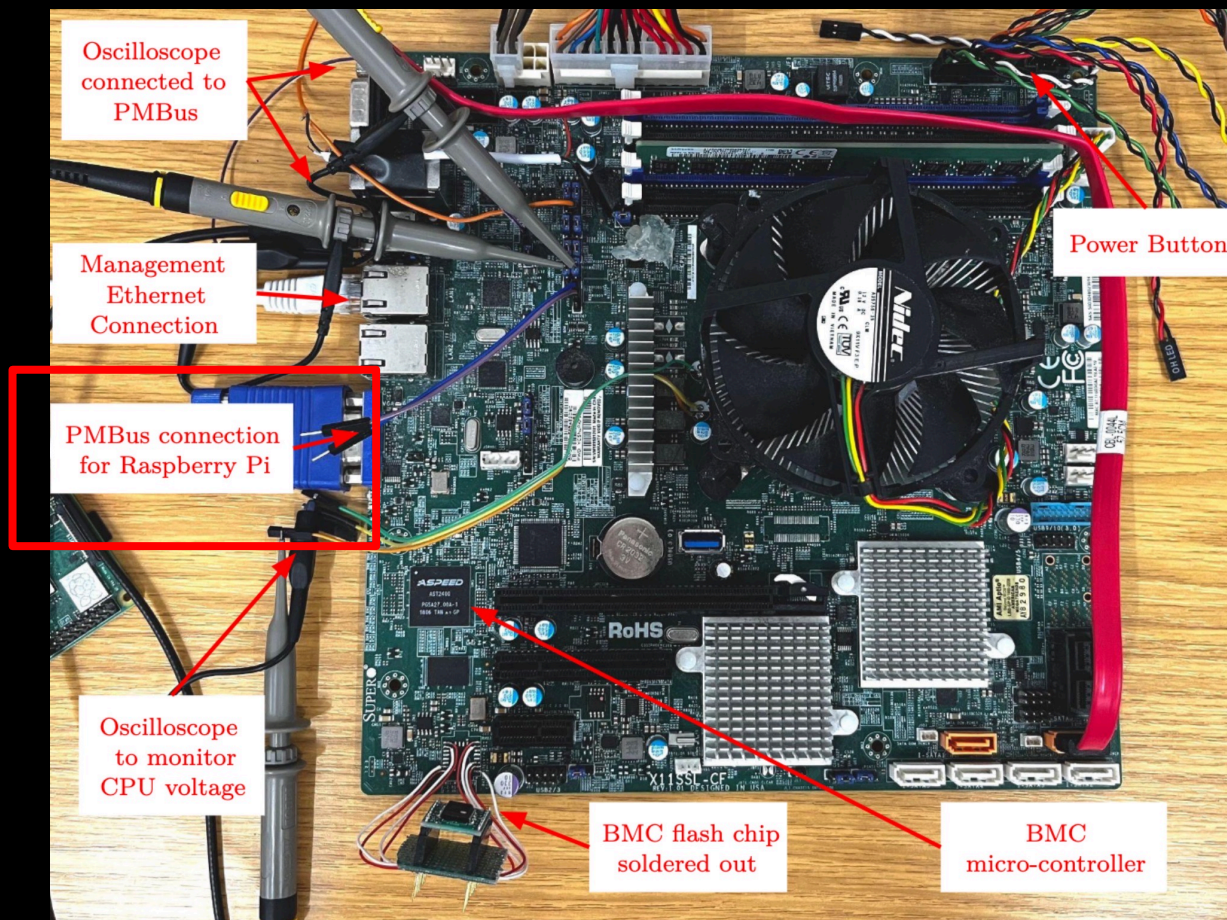


Stall... 🤖

At least... we know the address of the VRM now.

🤔 CPU crashed or recoverable?

Experiment 0.1: Try with "EXPENSIVE" equipment – Raspberry Pi



Luckily, we can use libi2c on RPi.
No changes in code needed.

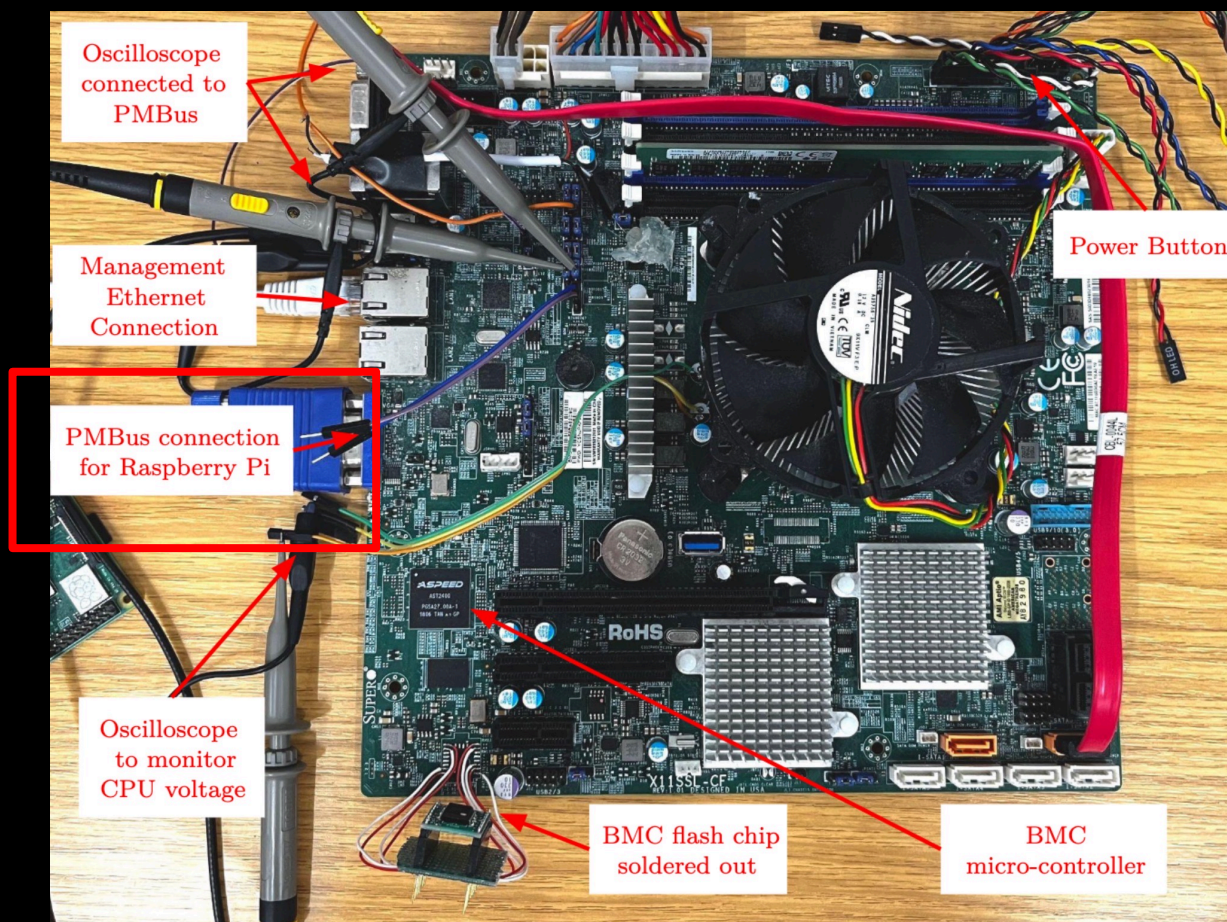
Send PMBus commands for undervolting

Voltage changed but stall...

CPU is running again!

Setting registers back

Experiment 0.1: Try with "EXPENSIVE" equipment – Raspberry Pi



Luckily, we can use libi2c on RPi.
No changes in code needed.

Send PMBus commands for undervolting

Voltage changed but stall...

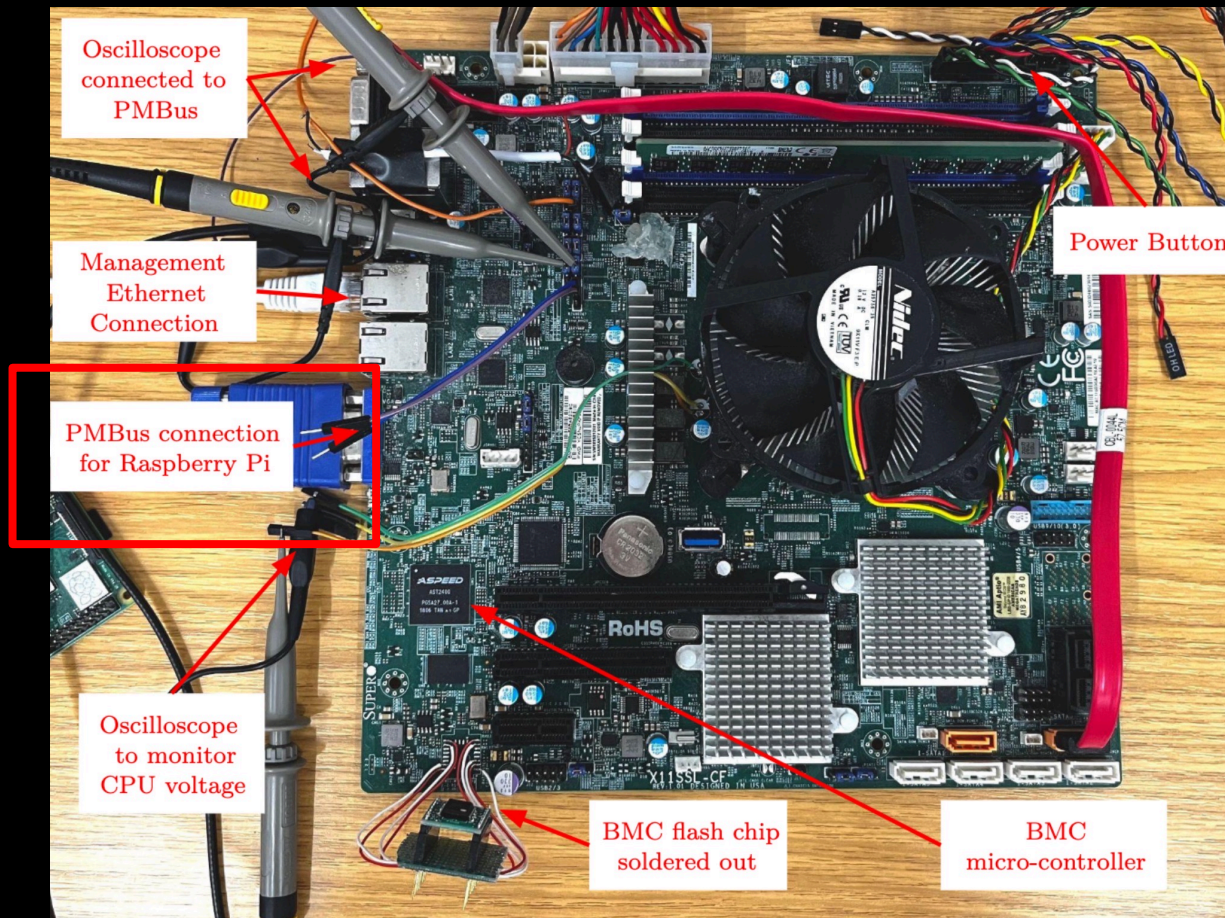
CPU is running again!

Setting registers back

Fault injection on CRT-RSA? Success!

Why 0.1 ?

Experiment 0.1: Try with "EXPENSIVE" equipment – Raspberry Pi



Luckily, we can use libi2c on RPi.
No changes in code needed.

Send PMBus commands for undervolting

Voltage changed but stall...

CPU is running again!

Setting registers back

Fault injection on CRT-RSA? Success!

Why 0.1 ?

-- Requires "Opening the box"



Power Button

Luckily, we can use libi2c on RPi. No changes in code needed.

Send PMBus commands for undervolting

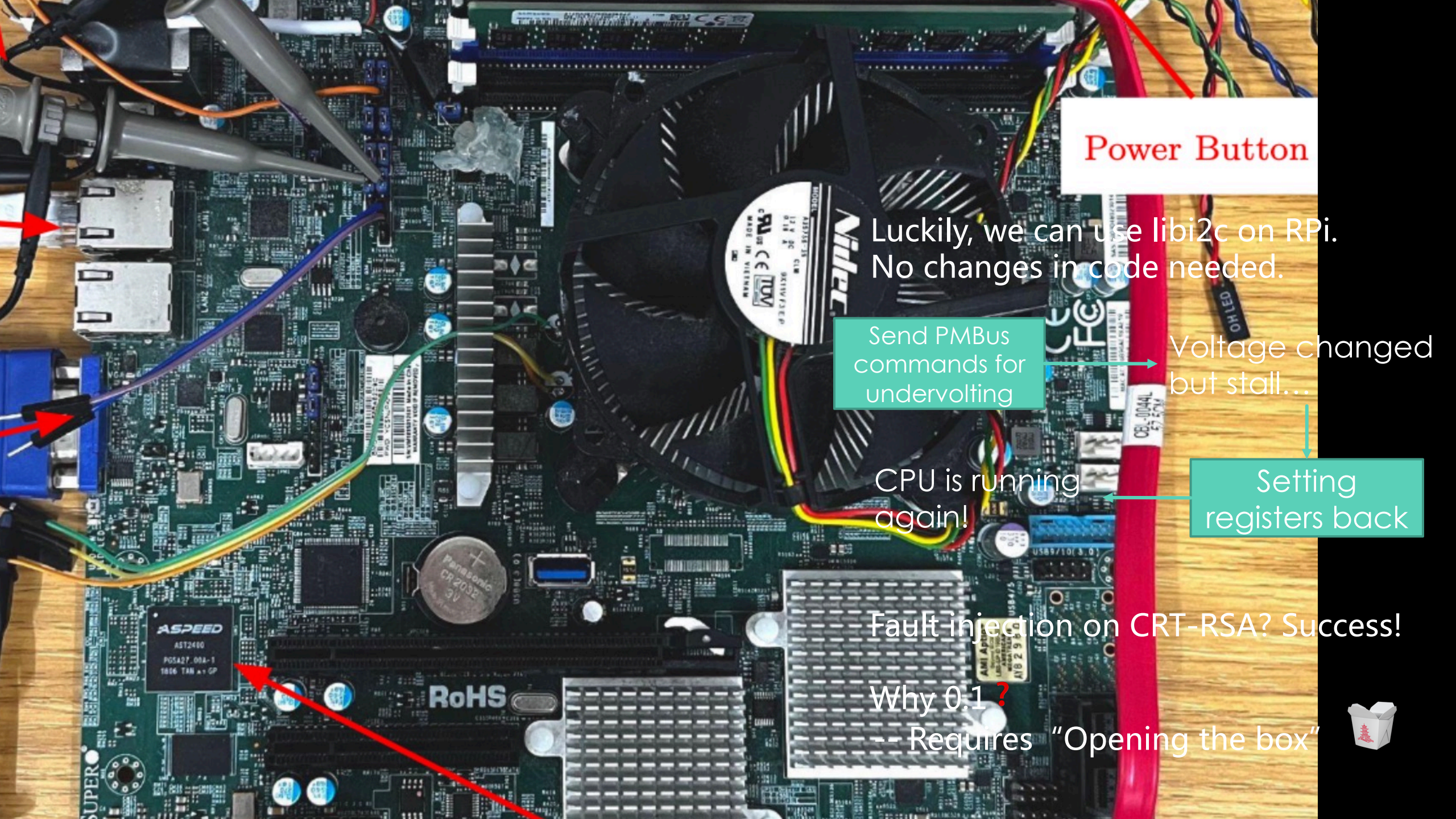
Voltage changed but stall...

CPU is running again!

Setting registers back

Fault injection on CRT-RSA? Success!

Why 0.1 ?
-- Requires "Opening the box"



Power Button

Luckily, we can use libi2c on RPi. No changes in code needed.

Send PMBus commands for undervolting

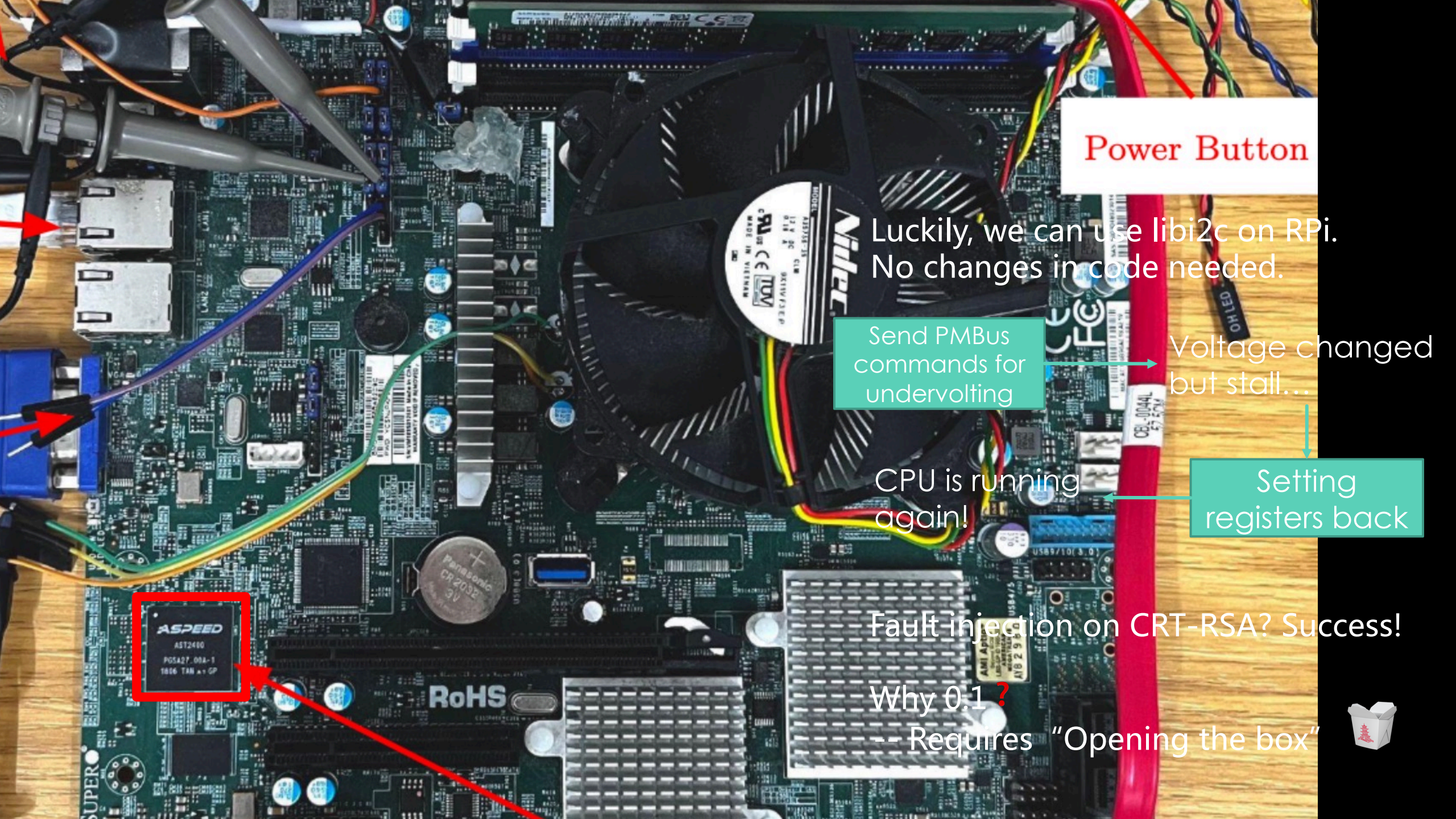
Voltage changed but stall...

CPU is running again!

Setting registers back

Fault injection on CRT-RSA? Success!

Why 0.1 ?
-- Requires "Opening the box"



Experiment 1: BMC

- How to run custom code on it or get SHELL?


Experiment 1: BMC

- How to run custom code on it or get SHELL?
 - 22 (SSH) -> gives "ATEN SMASH-CLP SystemManagement Shell" - limited commands available



Experiment 1: BMC

- How to run custom code on it or get SHELL?
 - 22 (SSH) -> gives "ATEN SMASH-CLP SystemManagement Shell" - limited commands available
 - Firmware reflashing?




Experiment 1: BMC

- How to run custom code on it or get SHELL?
 - 22 (SSH) -> gives "ATEN SMASH-CLP SystemManagement Shell" - limited commands available
 - Firmware reflashing?
 - Web Interface –  BMC password, diversified in Supermicro Servers.

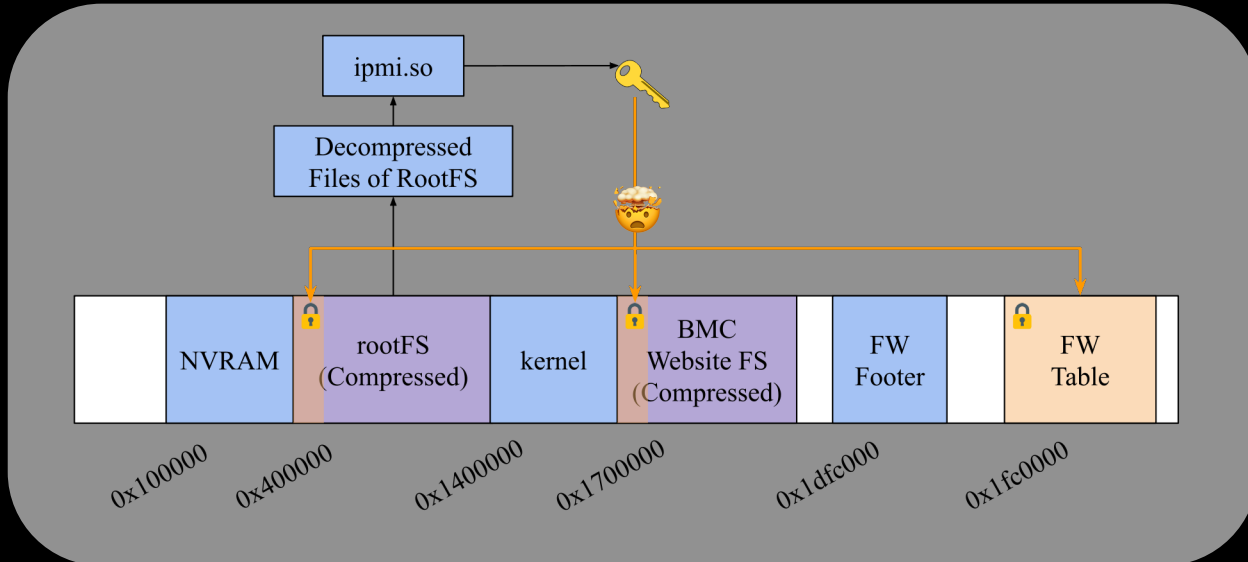
Experiment 1: BMC

- How to run custom code on it or get SHELL?
 - 22 (SSH) -> gives "ATEN SMASH-CLP SystemManagement Shell" - limited commands available
 - Firmware reflashing?
 - Web Interface –  BMC password, diversified in Supermicro Servers.
 - AlUpdate –  No password required.

Experiment 1: BMC

- How to run custom code on it or get SHELL?
 - 22 (SSH) -> gives "ATEN SMASH-CLP SystemManagement Shell" - limited commands available
 - Firmware reflashing?
 - Web Interface –  BMC password, diversified in Supermicro Servers.
 - AlUpdate –  No password required.
 - Firmware package is "encrypted" 

BMC Vulnerability – Firmware Upgrade



Firmware layout is mostly the same as described by Eclipsium![1]

- Write tool to decrypt, modify and repack firmware, based on
 - smcbmc [2] tool and ipmi_firmware_tools [3]
- Reverse-engineered the firmware
 - /SMASH/msh provides the shell
 - Replace it with shell script with content /bin/sh
- Re-flash via KCS with `AlUpdate`
- SSH and successfully get root shell !!!
 - PMBus - Implement libi2c by hand

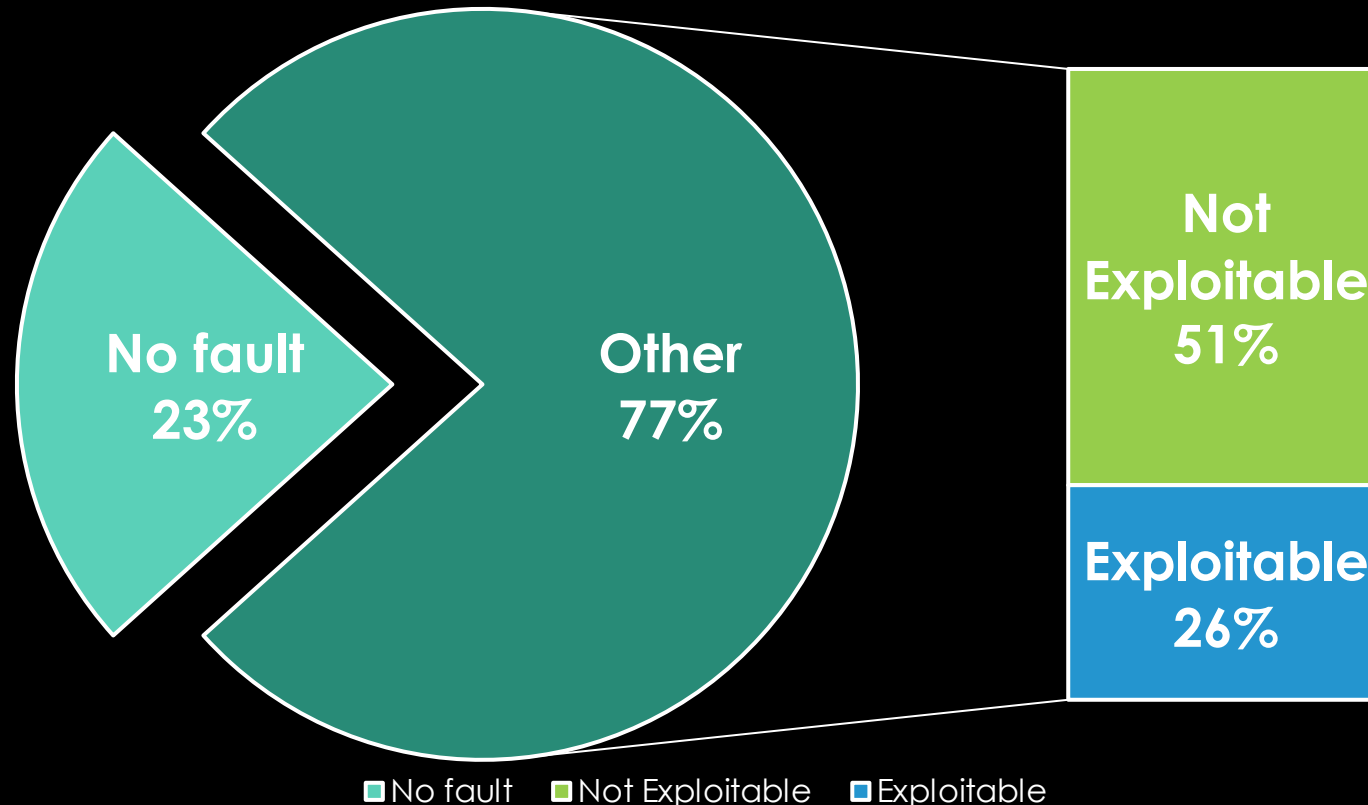
[1] **Insecure Firmware Updates in Server Management Systems**, Available at: <https://eclipsium.com/2018/09/06/insecure-firmware-updates-in-server-management-systems/>

[2] <https://github.com/c0d3z3r0/smcbmc>

[3] https://github.com/devicenull/ipmi_firmware_tools

Attack 1: Undervolting

- Fault injection on SGX WITHOUT physical access – Plundervolt revived! 🎉
- Stability test with CRT-RSA fault injection (in SGX):



253 tests in 545 mins, on average 9 mins for a useful fault

Things happen – server is broken

Things happen – server is broken

One day at 3:00AM 🌙

Things happen – server is broken

One day at 3:00AM 🌙

Why is my undervolting code not working!

Things happen – server is broken

One day at 3:00AM 🌙

Why is my undervolting code not working!

(😴 Dream coding 😴)

VID_STEP_SEL MFR_VR_CONFIG? **VR_CONFIG!!**

Things happen – server is broken

One day at 3:00AM 🌙

Why is my undervolting code not working!

(😴 Dream coding 😴)

VID_STEP_SEL MFR_VR_CONFIG? **VR_CONFIG!!**

Reset it to 0x00 try again™!!

Things happen – server is broken

One day at 3:00AM 🌙

Why is my undervolting code not working!

(😴 Dream coding 😴)

VID_STEP_SEL MFR_VR_CONFIG? **VR_CONFIG!!**

Reset it to 0x00 try again™!!



Attack 2: Overvolting



<https://youtu.be/hXuidPexanM?t=88>

Attack 2: Overvolting

VID_STEP_SEL MFR_VR_CONFIG
(p104 of [MP2965 Datasheet](#))

Bit 8: VID_STEP_SEL

1'b0: 10mV per VID step

1'b1: 5mV per VID step

With 10mV per VID step

Vcpu can be up to 3V!!! (CPU spec: 1.52V max)

We have BMC, maybe use ipmitool?

- `ipmitool i2c`
 - directly interact with I2C buses on the BMC
 - Via KCS: Need root on CPU, no need to login to BMC.
 - Via Ethernet: login required (password can be cleared with `ipmitool` via KCS)

We have BMC, maybe use ipmitool?

- `ipmitool i2c`
 - directly interact with I2C buses on the BMC
 - Via KCS: Need root on CPU, no need to login to BMC.
 - Via Ethernet: login required (password can be cleared with `ipmitool` via KCS)

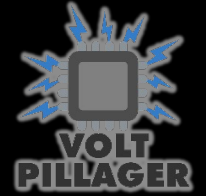


No need to reflash the firmware anymore, instead:

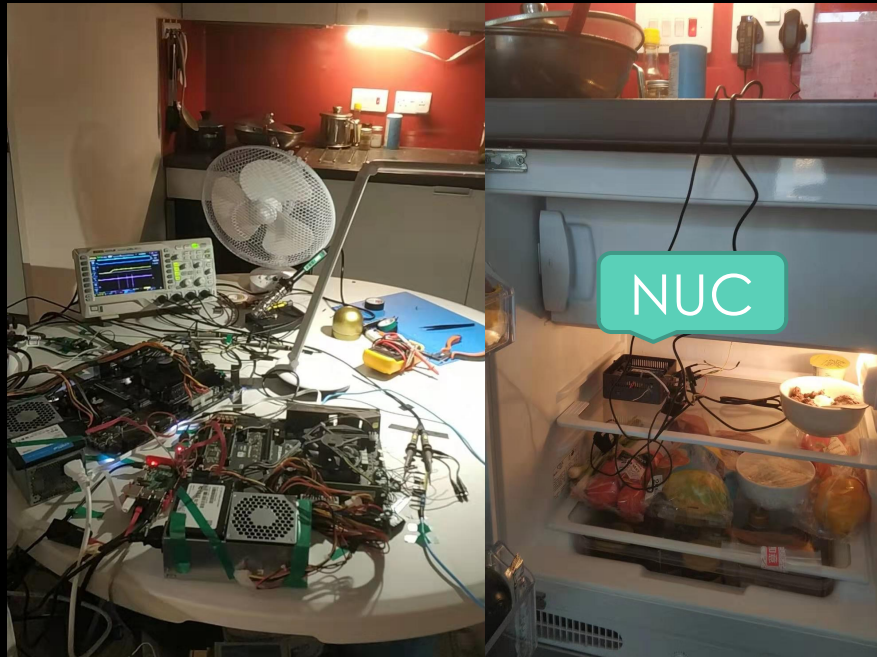
```
sudo ipmitool user set name
sudo ipmitool user set password
sudo ipmitool channel setaccess
```

```
ipmitool i2c (Via Ethernet)
```

I think this attack is nicer than the VoltPillager



I think this attack is nicer than the VoltPillager



I think this attack is nicer than the VoltPillager



I think this attack is nicer than the VoltPillager



Less messy

Attack via Ethernet

Tested on

- Supermicro X11SSL-CF - Vulnerable
- Supermicro X11SPG-TF and X11SSE-F
 - VRM reachable with default config, undervolting crashed the server
 - Didn't try overvolting as it was kindly provided by a friend
- Supermicro X12DPi-NT - NOT Vulnerable
- ASRock E3C246D4I-2 - Infinite boot loop – different PMBus command
- Responsible disclosed to Supermicro, see [security advisory](#)

Summary

- Think of a server as an embedded system
 - Vulnerability/functionality in one component --> rest of the system
 - Software + hardware
 - Plug-in devices
- SGX security
 - SGX attestation cannot measure BMC firmware

PMBusDetect Tool

```
$ sudo modprobe i2c_i801
$ sudo ./pmbusdetect -d /dev/i2c-1
Device 0x20          READ_TEMPERATURE success: 0019
!!!!!!!!!!!!!! Detected! Device addr: 20 !!!!!!!!!!!!!!!
Device 0x20          SVID_VENDOR_PRODUCT_ID success,
data: 2555 This device is likely to be a MPS VRM
# Save the page
Device 0x20 : 00      READ_PAGE success

Page: 00
Device 0x20 : 00      WRITE_PAGE success
Device 0x20 : 00      READ_VOUT success: 00D8

Page: 01
Device 0x20 : 01      WRITE_PAGE success
Device 0x20 : 01      READ_VOUT success: 0001
# Restore the page
Device 0x20 : 00      WRITE_PAGE success
```

Currently only tested with
ISL68137 and MP2955.

Contributions are welcome.

<https://github.com/zt-chen/PMFault>

Acknowledgements

- This research is partially funded by the Engineering and Physical Sciences Research Council (EPSRC) under grants EP/R012598/1, EP/R008000/1, and EP/V000454/1. The results feed into DsbDtech.
- We would also like to thank Supermicro for providing a X12DPi-NT6 server for further investigation of the issue.

Thank You!



GitHub Repo



PMFault Website