black hat®
EUROPE 2023
DECEMBER 4-7
EXCEL LONDON / UK

CloudVA

How we taught ChatGPT-4 to break Mbed TLS AES with side-channel attacks

Witold Waligóra

Who am I

MyreLabs, 2010, Founder&CEO

- Reverse engineering
- Embedded/IoT security
- Secure coding
- High-performance computing

CloudVA, 2021, Founder

- Side-channels as a service
- Side-channels as a regression test

Agenda

1. Intro
2. Why?
3. How?
4. Strong sides
5. Problems & workarounds
6. Demo!
7. Conclusions

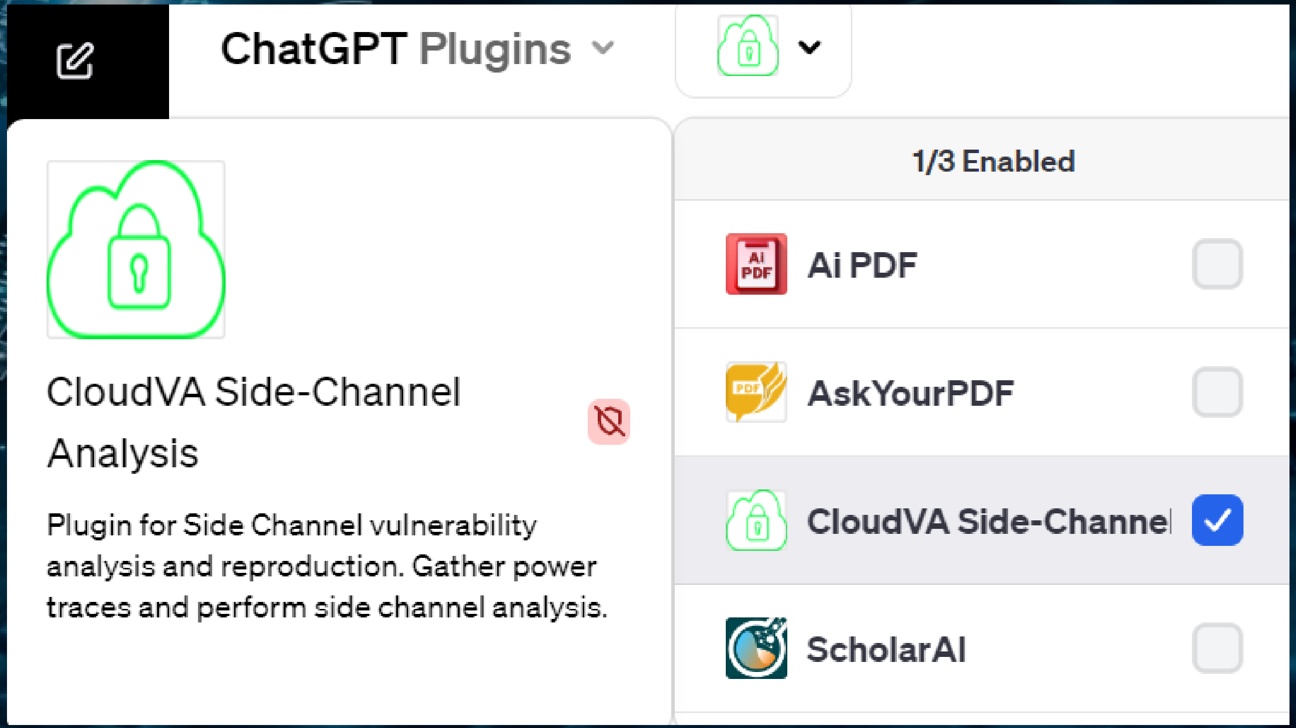
Intro: GPT-4

March 2023: Plugins

- GPT-4 can interface with RESTful APIs

August 2023: Analysis

- GPT-4 can run Python code



CloudVA Side-Channel Analysis

Plugin for Side Channel vulnerability analysis and reproduction. Gather power traces and perform side channel analysis.

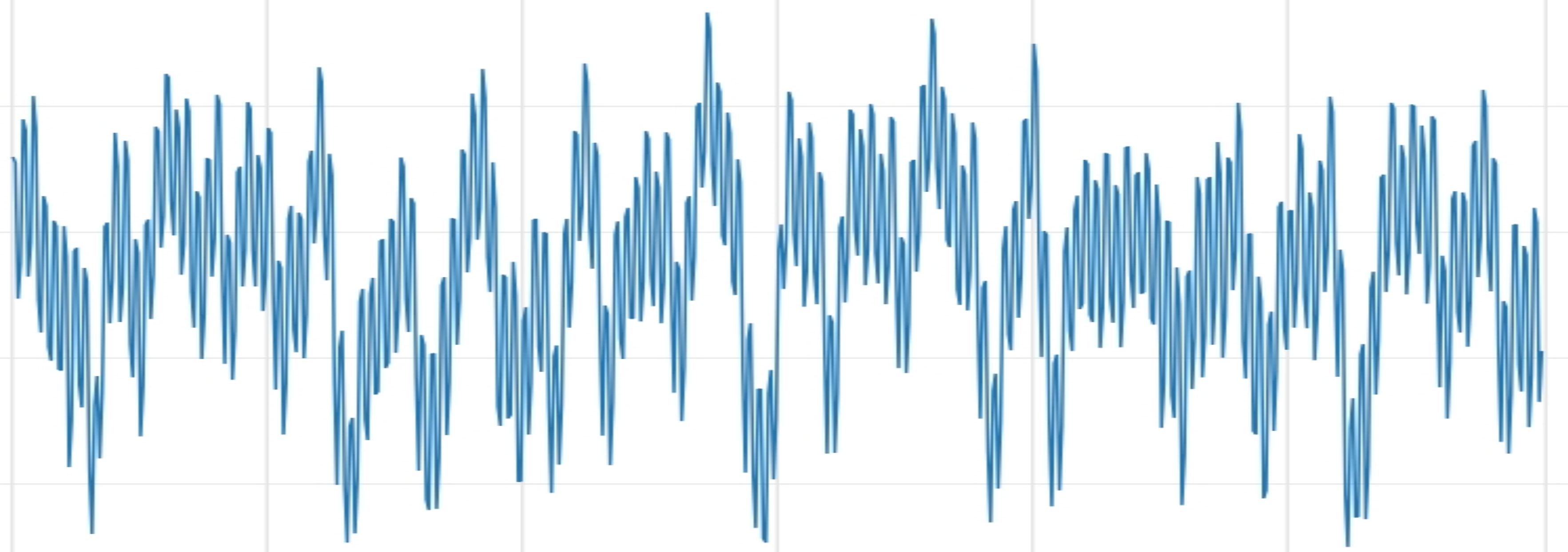
Intro: side-channels

Physical side-effects of computation

- **Time** taken
- **Power** drawn
- **EM** Emissions

Fault injections

- **Clock**
- **Power**
- **EM**

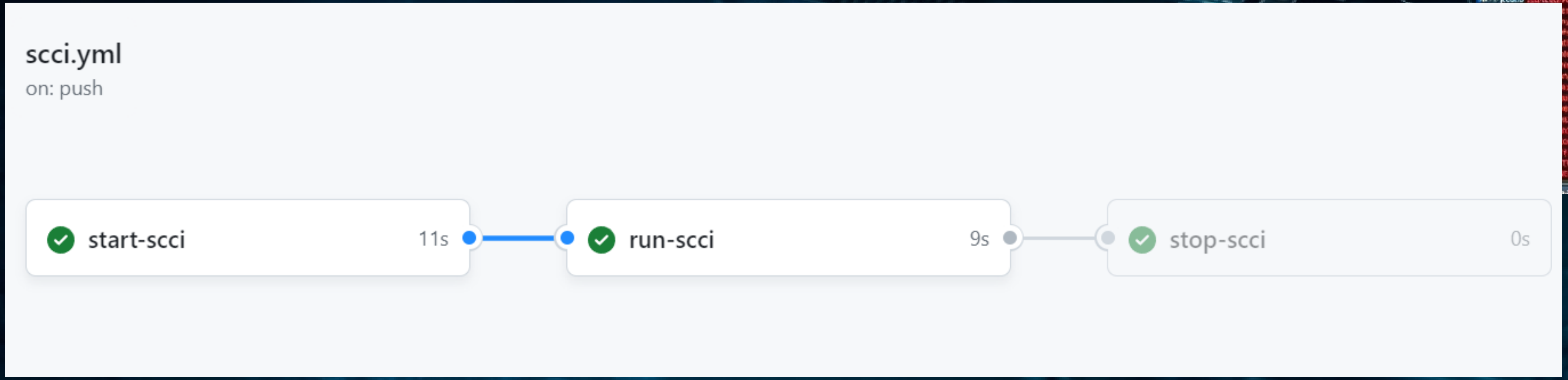


Intro: side-channels

Easier than you think

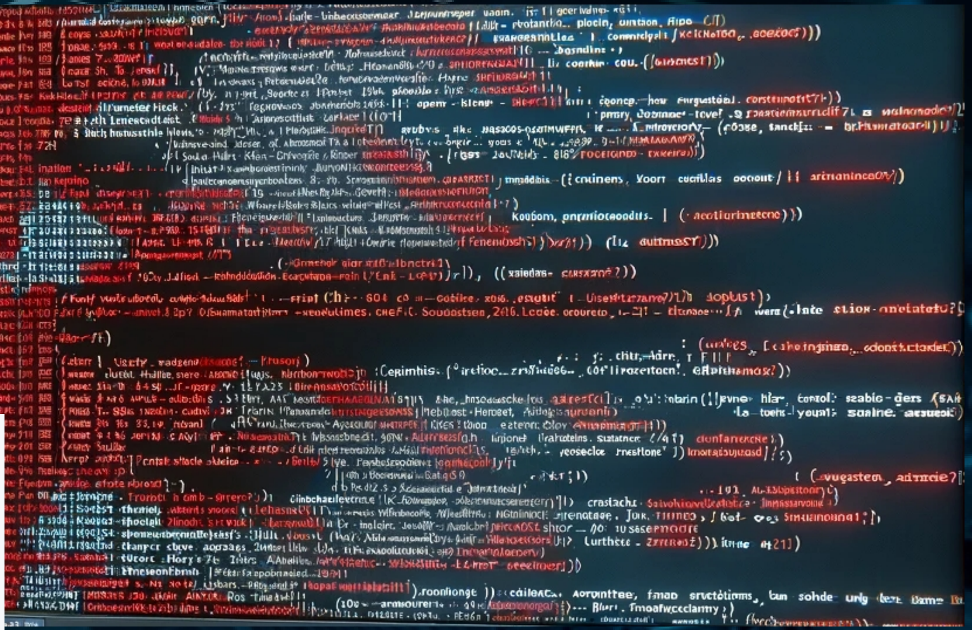
- High initial barrier of entry, but
- Everyone uses **the same** crypto
- Everyone uses **the same** hardware

Re-run the pipeline



scci.yml
on: push

Find a new zero-day



Why: side-channels

The sad state of IoT HWSec
- Non-secure element chips
- Low clocks
- No countermeasures
- Deployed in hostile environments

Cost of attack: a laptop + $100 PicoScope
“Clever teenager” threat level

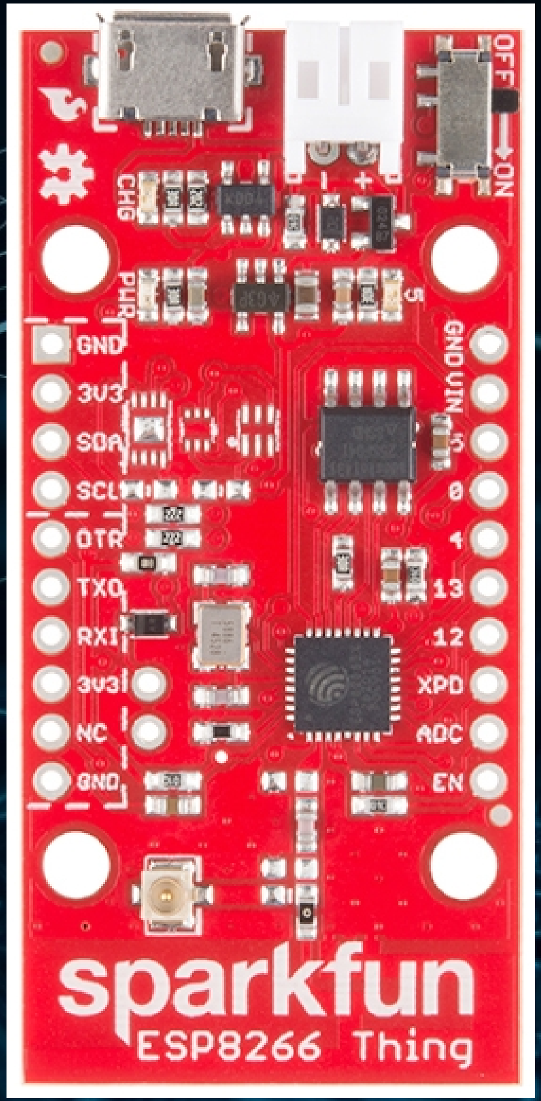
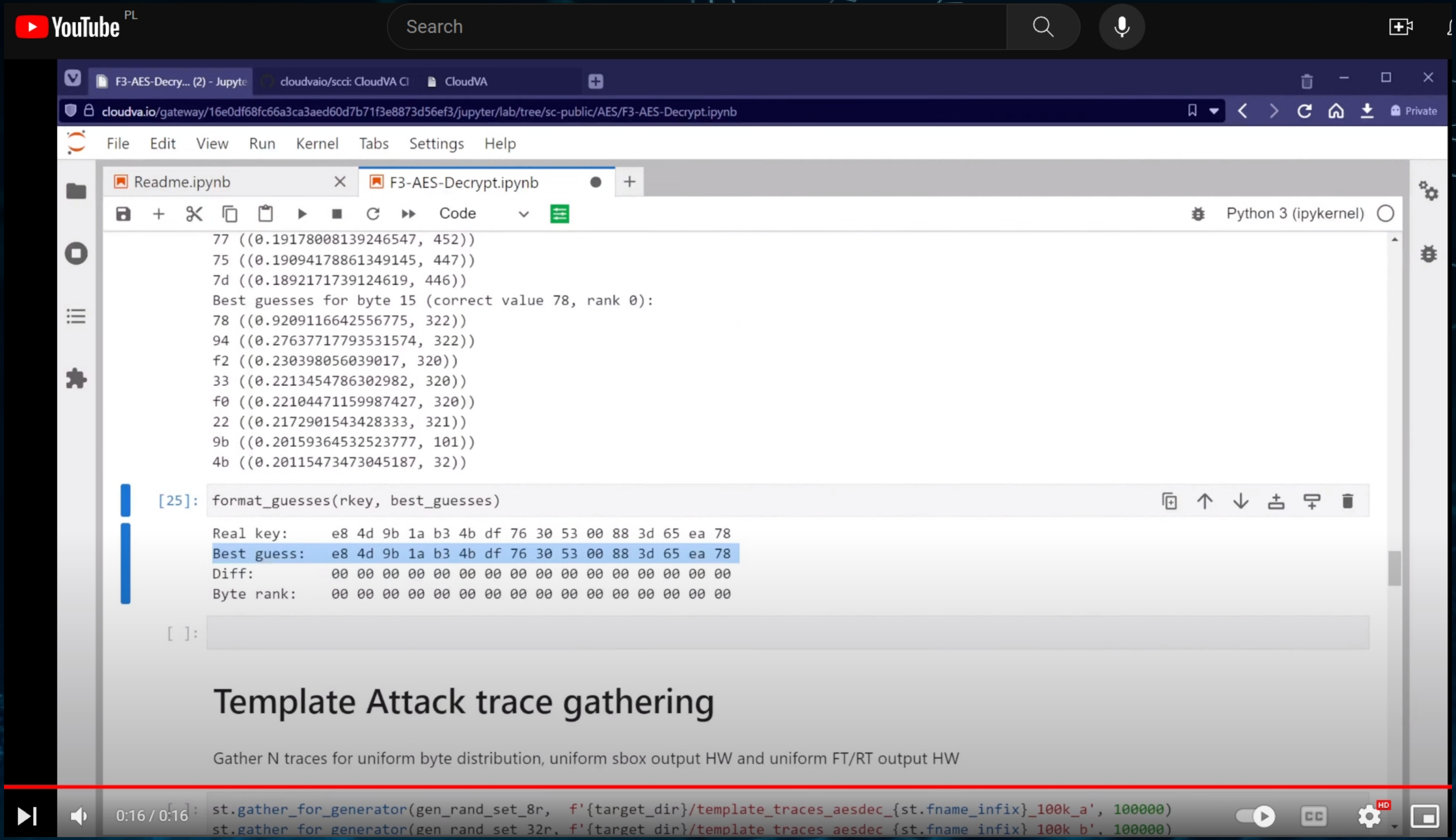


Image by SparkFun Electronics, CC BY 2.0

STM32F3, mbedTLS AES Decrypt

https://youtube.com/watch?v=CCeK_S3ED4A



YouTube
Search
F3-AES-Decry... (2) - Jupyte
cloudvaio/scci: CloudVA Cl
CloudVA
cloudva.io/gateway/16e0df68fc66a3ca3aed60d7b71f3e8873d56ef3/jupyter/lab/tree/sc-public/AES/F3-AES-Decrypt.ipynb
Private
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Readme.ipynb
F3-AES-Decrypt.ipynb
Code
Python 3 (ipykernel)
0:16 / 0:16

```
77 ((0.19178008139246547, 452))
75 ((0.19094178861349145, 447))
7d ((0.1892171739124619, 446))
Best guesses for byte 15 (correct value 78, rank 0):
78 ((0.9209116642556775, 322))
94 ((0.27637717793531574, 322))
f2 ((0.230398056039017, 320))
33 ((0.2213454786302982, 320))
f0 ((0.22104471159987427, 320))
22 ((0.2172901543428333, 321))
9b ((0.20159364532523777, 101))
4b ((0.20115473473045187, 32))
```

```
[25]: format_guesses(rkey, best_guesses)
```

```
Real key:     e8 4d 9b 1a b3 4b df 76 30 53 00 88 3d 65 ea 78
Best guess:   e8 4d 9b 1a b3 4b df 76 30 53 00 88 3d 65 ea 78
Diff:         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Byte rank:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
[ ]:
```

Template Attack trace gathering

Gather N traces for uniform byte distribution, uniform sbox output HW and uniform FT/RT output HW

```
st.gather_for_generator(gen_rand_set_8r,  f'{target_dir}/template_traces_aesdec_{st.fname_infix}_100k_a', 100000)
st.gather_for_generator(gen_rand_set_32r, f'{target_dir}/template_traces_aesdec_{st.fname_infix}_100k_b', 100000)
```

Why: GPT-4

Exploring the limits

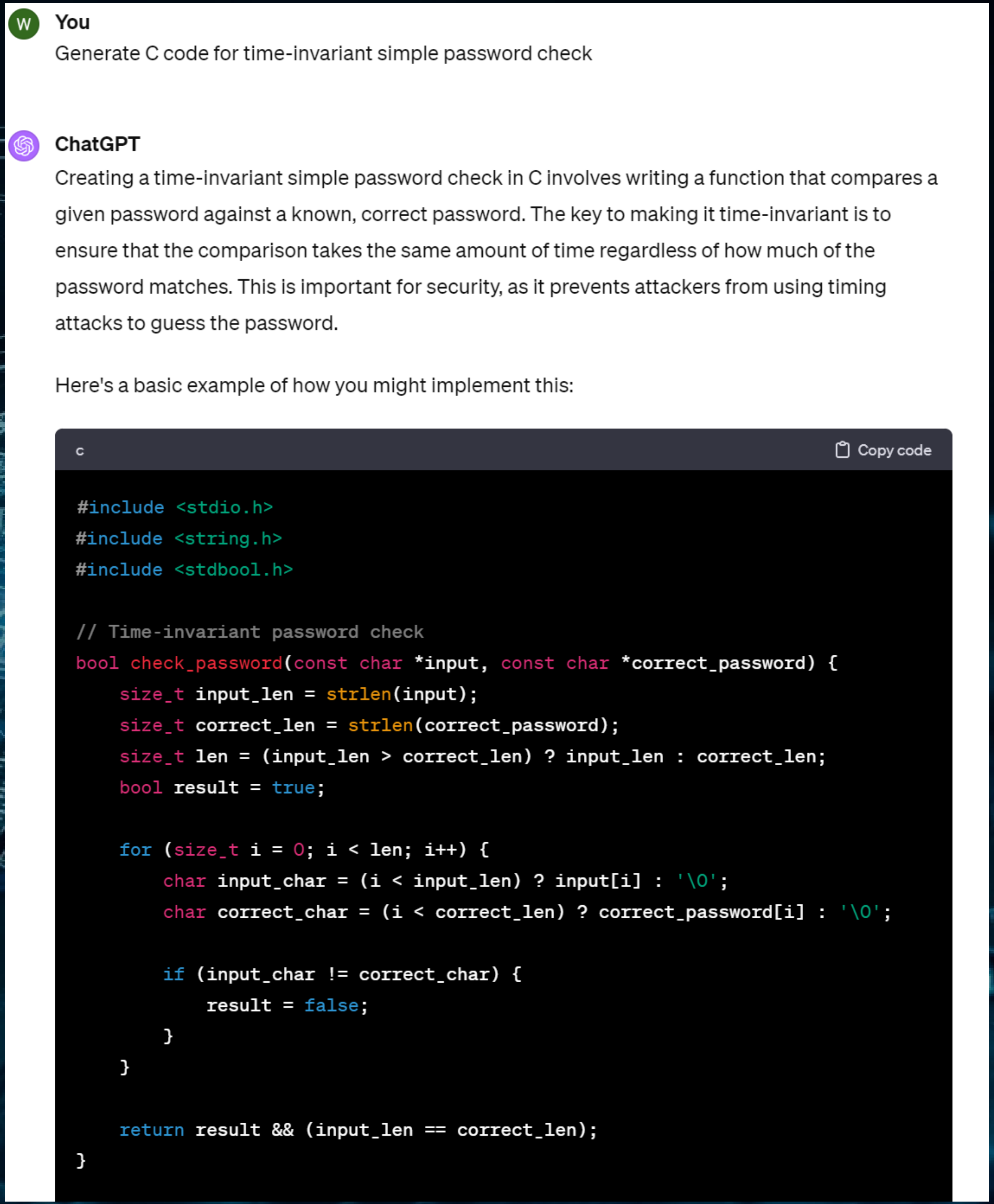
- GPT-4 can do time-invariant code
- What else can it do?

Low-cost defense

- Semi-automatic countermeasures

Auto-training

1. Generate code
2. Evaluate leakage
3. Learn
4. Repeat



You

Generate C code for time-invariant simple password check

ChatGPT

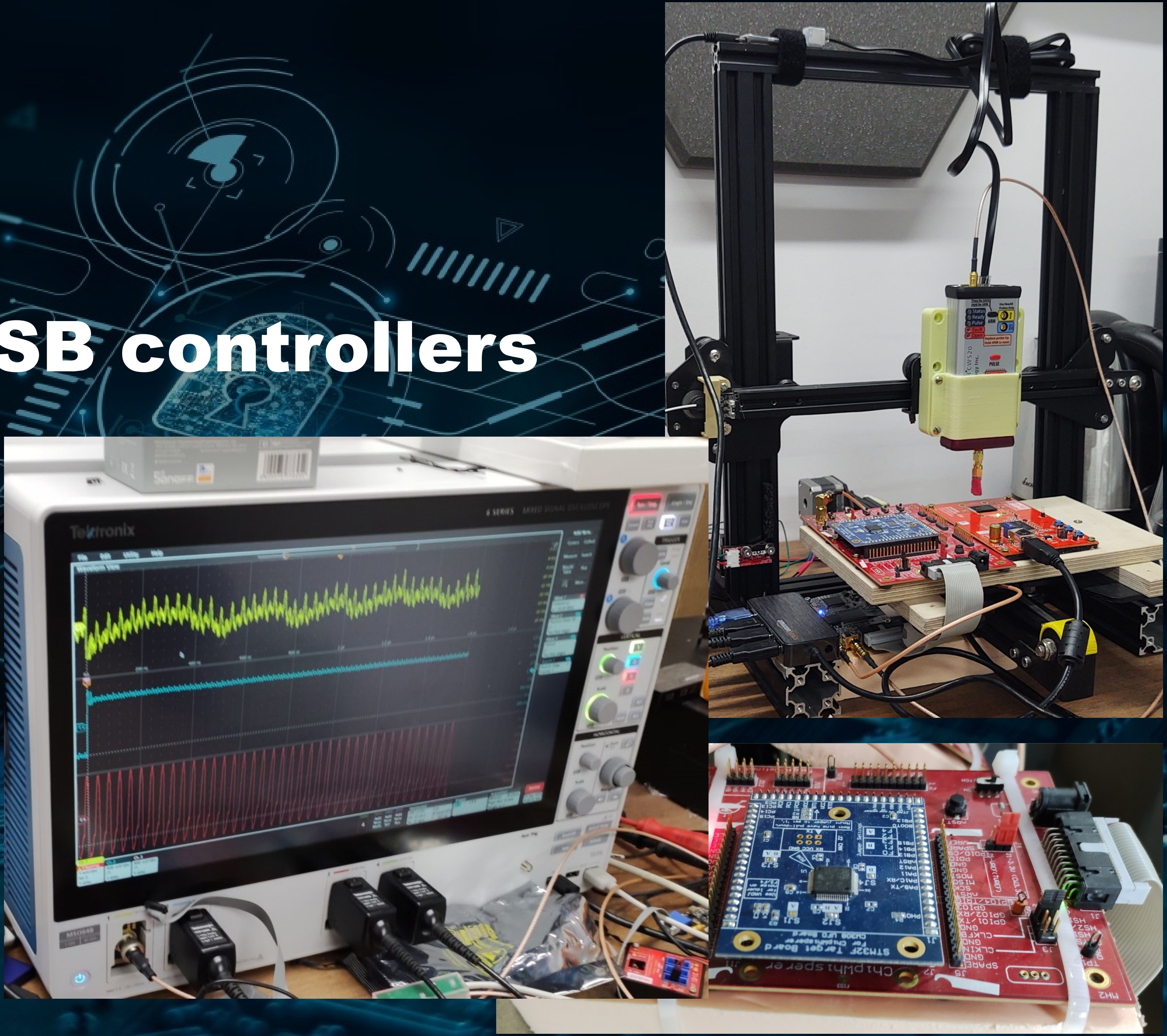
Creating a time-invariant simple password check in C involves writing a function that compares a given password against a known, correct password. The key to making it time-invariant is to ensure that the comparison takes the same amount of time regardless of how much of the password matches. This is important for security, as it prevents attackers from using timing attacks to guess the password.

Here's a basic example of how you might implement this:

```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

// Time-invariant password check
bool check_password(const char *input, const char *correct_password) {
    size_t input_len = strlen(input);
    size_t correct_len = strlen(correct_password);
    size_t len = (input_len > correct_len) ? input_len : correct_len;
    bool result = true;

    for (size_t i = 0; i < len; i++) {
        char input_char = (i < input_len) ? input[i] : '\0';
        char correct_char = (i < correct_len) ? correct_password[i] : '\0';

        if (input_char != correct_char) {
            result = false;
        }
    }

    return result && (input_len == correct_len);
}
```

How: **Hardware**

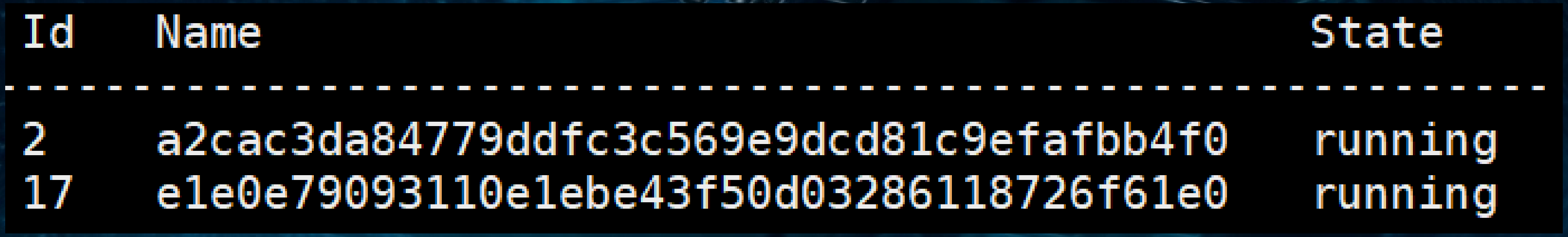
- PCIe **passthrough** USB controllers
- PPPS **USB** hubs
- **ChipWhisperers**
- ChipShouters
- **Oscilloscopes**
- Protocol analyzers
- Debuggers



How: Software

Standard virtualization stack

- Linux
- QEMU KVM
- Libvirt
- RESTful API



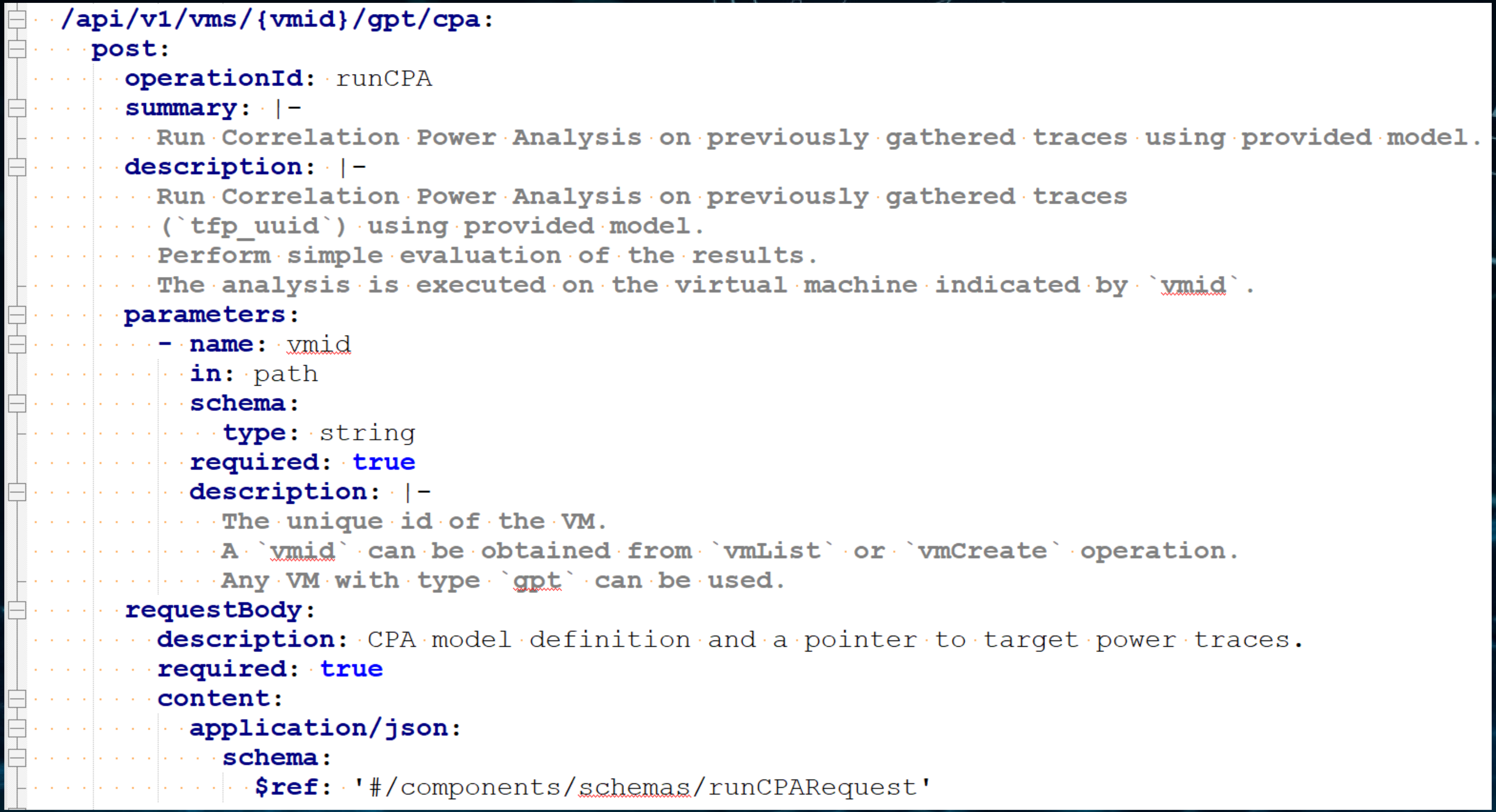
```
Id   Name                                           State
---------------------------------------------------------
2    a2cac3da84779ddfc3c569e9dcd81c9efafbb4f0       running
17   e1e0e79093110e1ebe43f50d03286118726f61e0       running
```

How: GPT-4 **plugin**

OpenAPI specification

- **When** to use your API
- **What** and how to call
- **How** to interpret the results
- Codegen **instructions**
- Usage **examples**

GPT-4 plugin



```
/api/v1/vms/{vmid}/gpt/cpa:
  post:
    operationId: runCPA
    summary: |-
      Run Correlation Power Analysis on previously gathered traces using provided model.
    description: |-
      Run Correlation Power Analysis on previously gathered traces
      (`tfp_uuid`) using provided model.
      Perform simple evaluation of the results.
      The analysis is executed on the virtual machine indicated by `vmid` .
    parameters:
      - name: vmid
        in: path
        schema:
          type: string
        required: true
        description: |-
          The unique id of the VM.
          A `vmid` can be obtained from `vmList` or `vmCreate` operation.
          Any VM with type `gpt` can be used.
    requestBody:
      description: CPA model definition and a pointer to target power traces.
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/runCPARequest'
```

How: GPT-4 plugin

Details matter

“Do not generate 'main' function”
“All symbols except 'entrypoint' should be static.”
“The file may use one of the provided cryptographic libraries (mbedtls, wolfssl)”
“Function `hw(x)` computes hamming weight of x for bytes and integers”

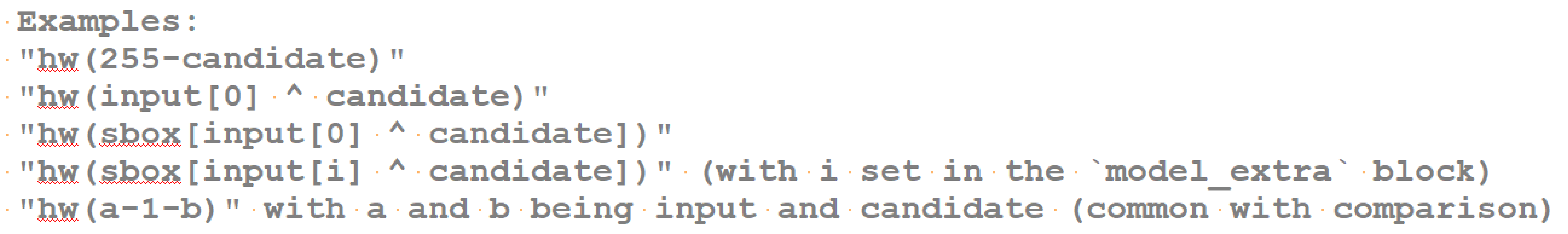
Our CPA endpoint specification is 187 LoC
https://cloudva.io/.well-known/ai-plugin.json

How: GPT-4 plugin

Examples matter more than descriptions

- Token-denser
- Provide context
- Show usage, defaults, conventions

Annotated examples are awesome!



```
Examples:
"hw(255-candidate)"
"hw(input[0] ^ candidate)"
"hw(sbox[input[0] ^ candidate])"
"hw(sbox[input[i] ^ candidate])" (with i set in the `model_extra` block)
"hw(a-1-b)" with a and b being input and candidate (common with comparison)
```

Strong sides: GPT-4

Boilerplate

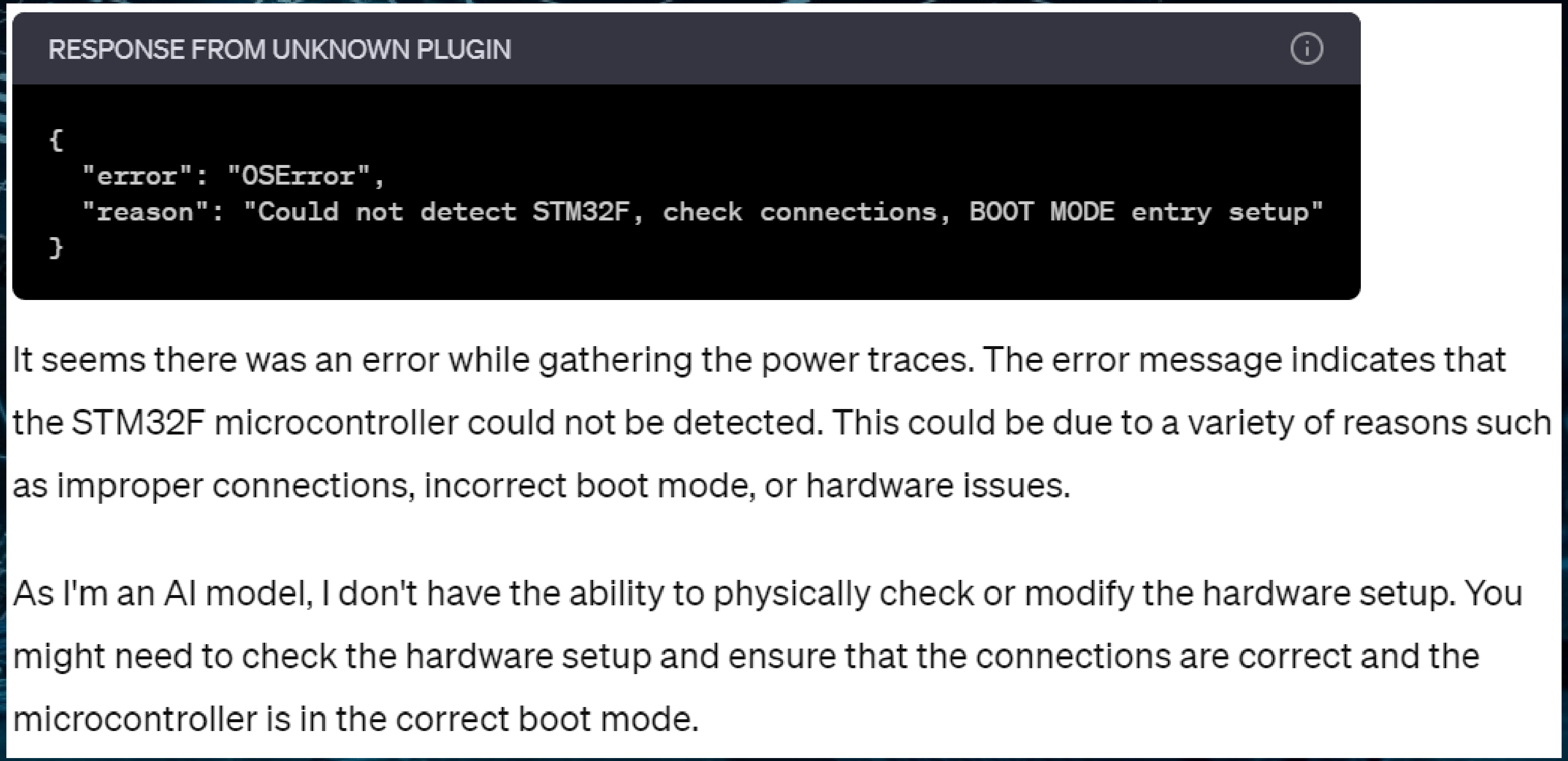
- Knows APIs better than I do
- Writes faster than I can

Translations

- English to code
- Code to code
- Scientific formula to code

Self-correcting

- C/C++ build issues
- Python imports
- It’s was right about a loose cable once



RESPONSE FROM UNKNOWN PLUGIN

```
{
  "error": "OSError",
  "reason": "Could not detect STM32F, check connections, BOOT MODE entry setup"
}
```

It seems there was an error while gathering the power traces. The error message indicates that the STM32F microcontroller could not be detected. This could be due to a variety of reasons such as improper connections, incorrect boot mode, or hardware issues.

As I'm an AI model, I don't have the ability to physically check or modify the hardware setup. You might need to check the hardware setup and ensure that the connections are correct and the microcontroller is in the correct boot mode.

Problems

- Context size
- Hallucinations
- BLOB handling
- Timeouts
- Arithmetics
- Metrics interpretation

Problem: Context size

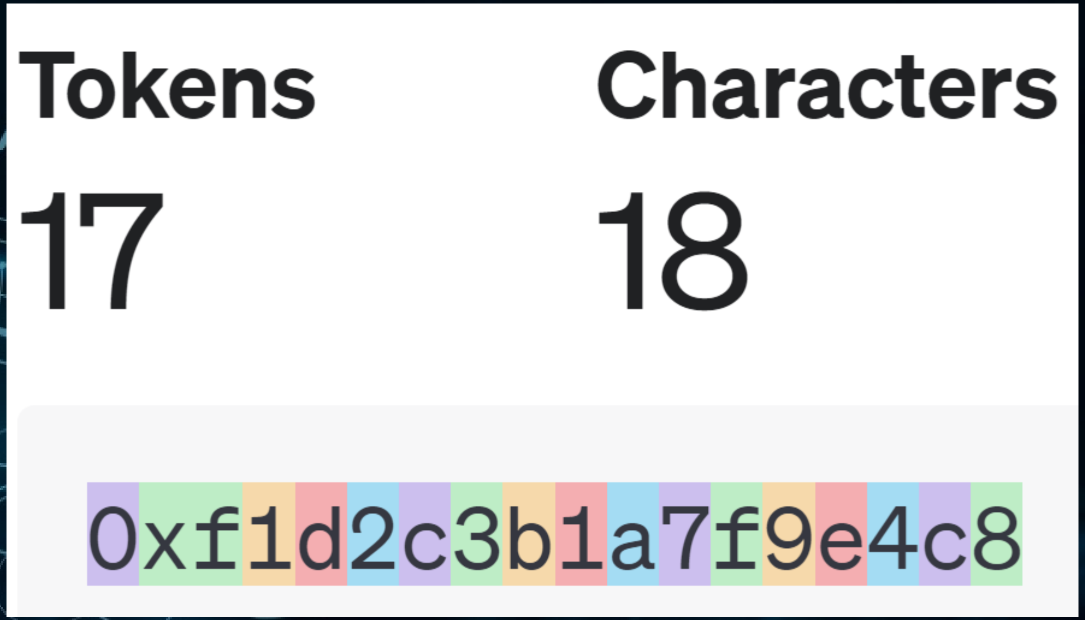
Up to 20 tokens/64-bit element
256 elements = 4K+ tokens

April 2023:

- Unable to produce complete AES S-Box
- Unable to produce a 200 element random array

November 2023:

- Both problems resolved
- But still unable to process large pieces of code

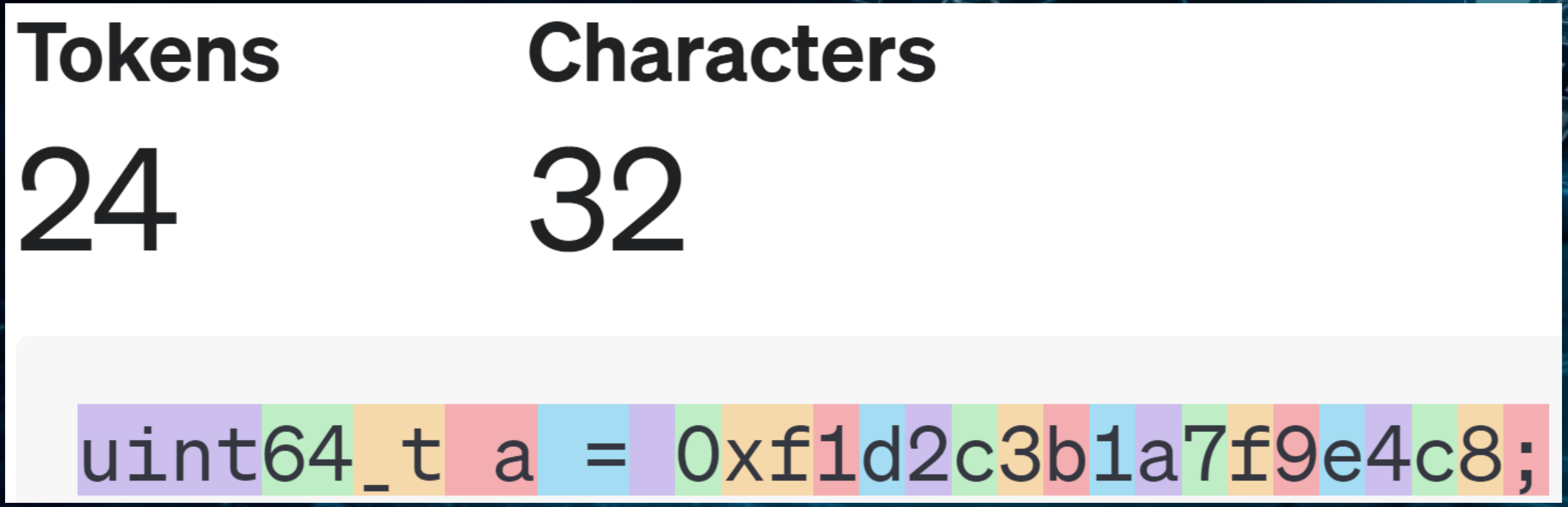


Tokens	Characters
17	18

`0xf1d2c3b1a7f9e4c8`

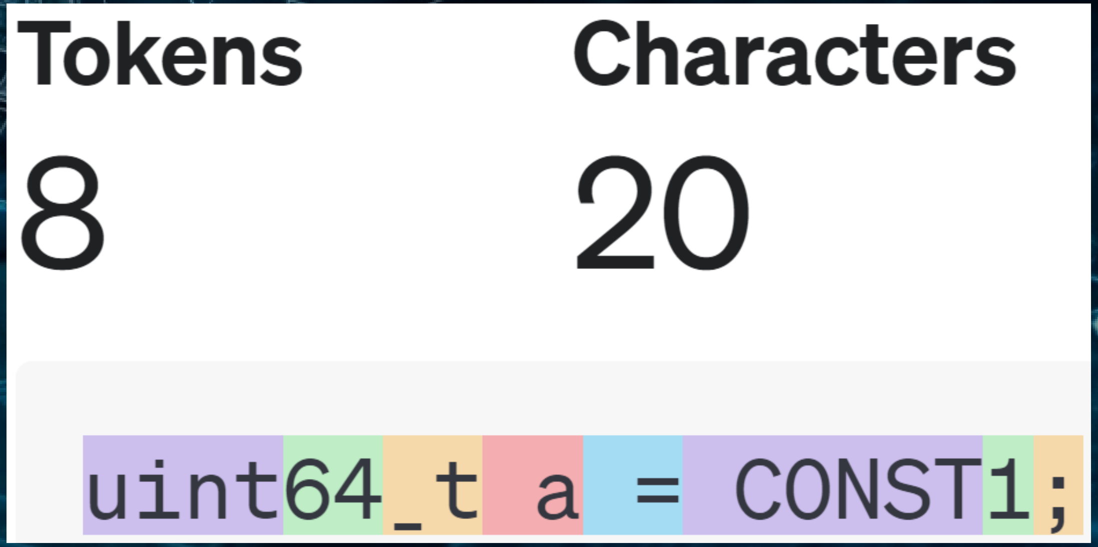
Code token size optimization

- Process only what you **need**
- Move what you can to **separate** files and headers
- Provide **clear** naming so model can infer from **examples**
- #define constants away



Tokens 24 Characters 32

```
uint64_t a = 0xf1d2c3b1a7f9e4c8;
```



Tokens 8 Characters 20

```
uint64_t a = CONST1;
```

ID token optimization

	Sample	Entropy	Tok
UUID	**550e8400-e29b-41d4-a716-446655440000**	**128 bits**	**18**
XKCD1	**Tr0ub4dor&3**	**~28 bits**	**8**
XKCD2	**CorrectHorseBatteryStaple**	**~44 bits**	**7**
TOK4	**HeavyCostWithinLanguage**	**64 bits?**	**4**

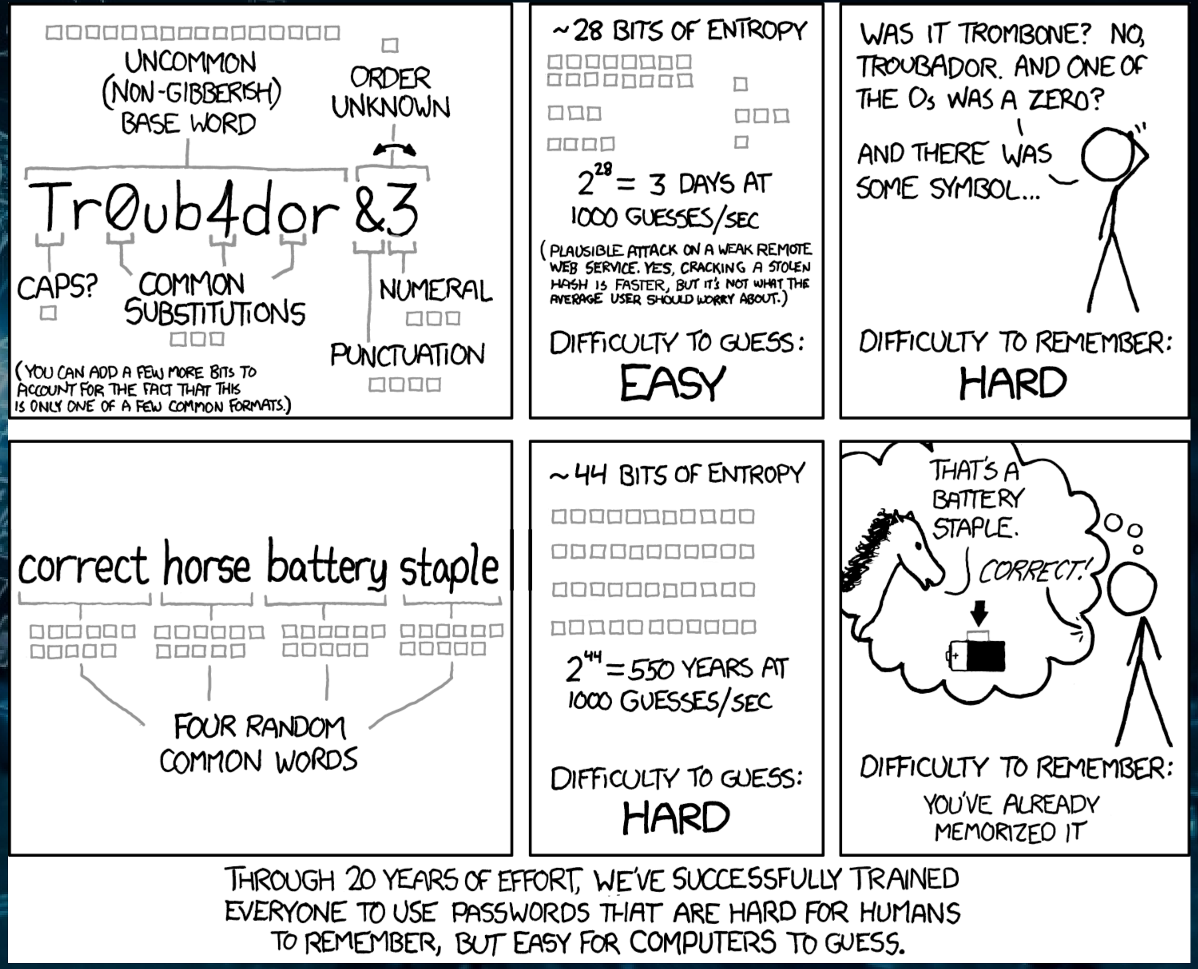
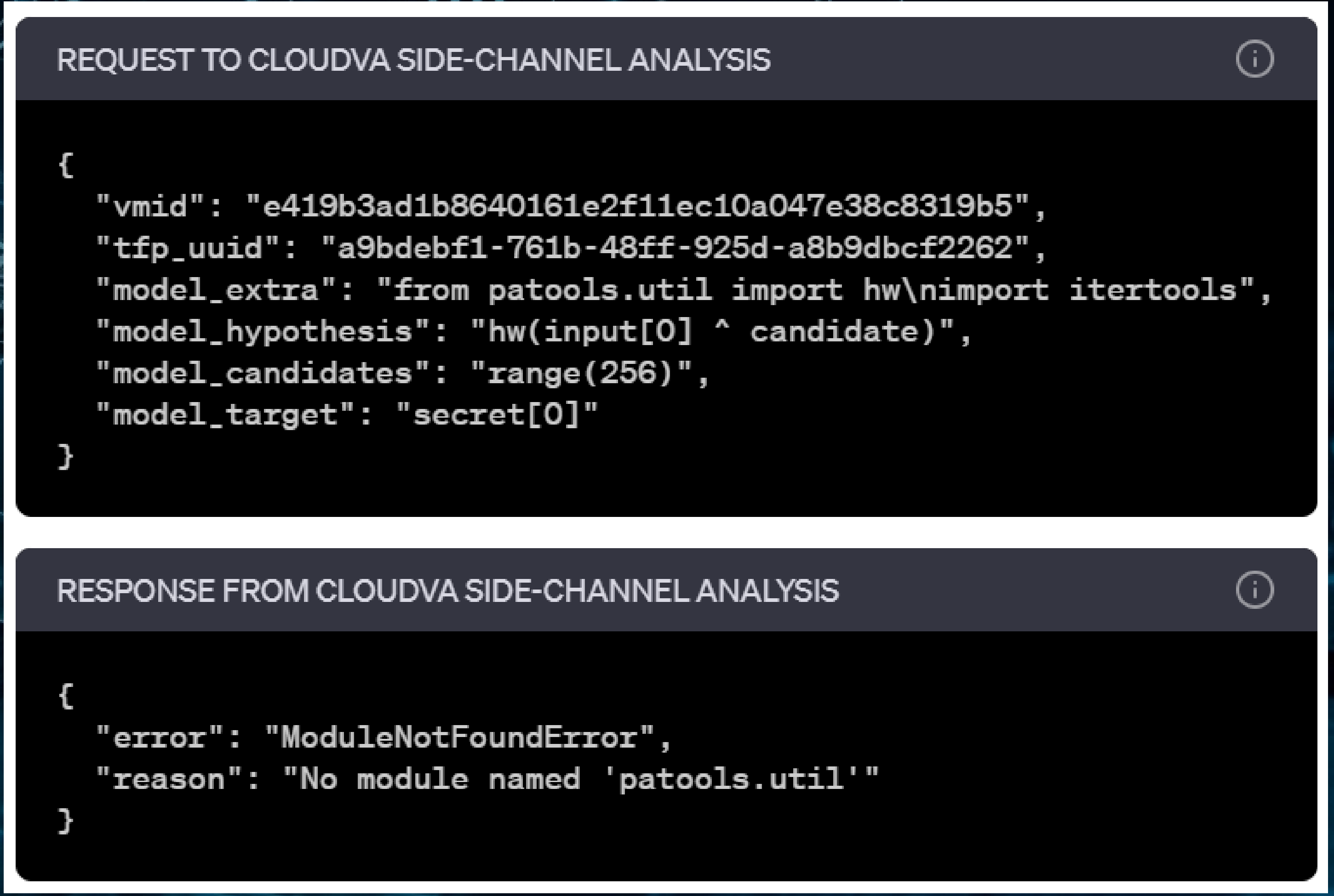

UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Image by XKCD, CC BY 2.5

Problem: Hallucinations

- Calling nonexistent functions
- Importing imaginary modules
- Making up libraries
- Ignoring your instructions
- Mixing up versions



REQUEST TO CLOUDVA SIDE-CHANNEL ANALYSIS

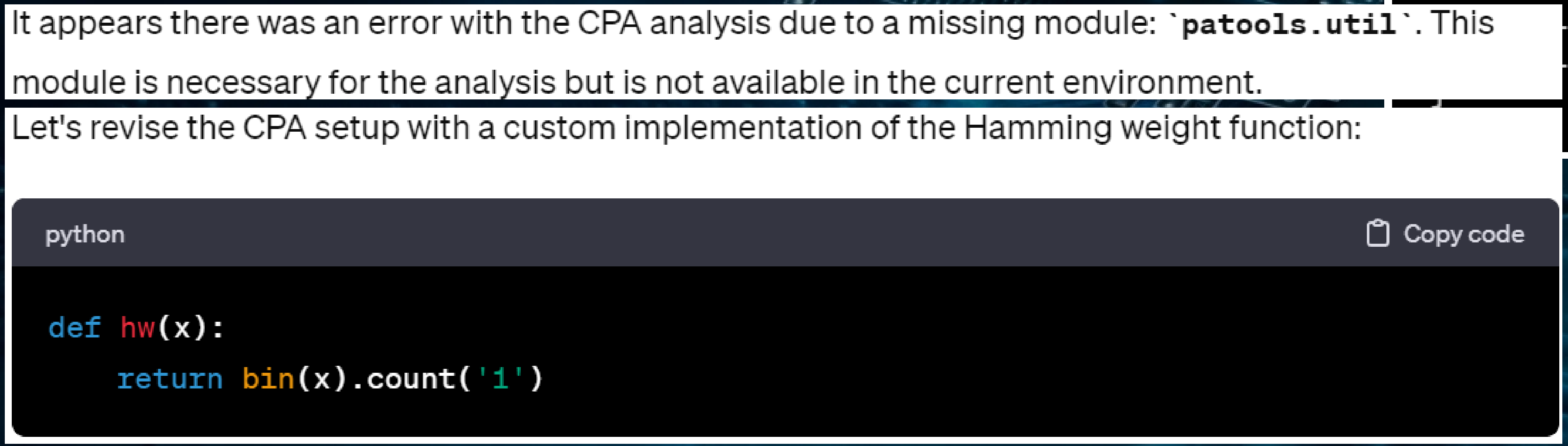
```
{
  "vmid": "e419b3ad1b8640161e2f11ec10a047e38c8319b5",
  "tfp_uuid": "a9bdebf1-761b-48ff-925d-a8b9dbcf2262",
  "model_extra": "from patools.util import hw\nimport itertools",
  "model_hypothesis": "hw(input[0] ^ candidate)",
  "model_candidates": "range(256)",
  "model_target": "secret[0]"
}
```

RESPONSE FROM CLOUDVA SIDE-CHANNEL ANALYSIS

```
{
  "error": "ModuleNotFoundError",
  "reason": "No module named 'patools.util'"
}
```

Dealing with hallucinations

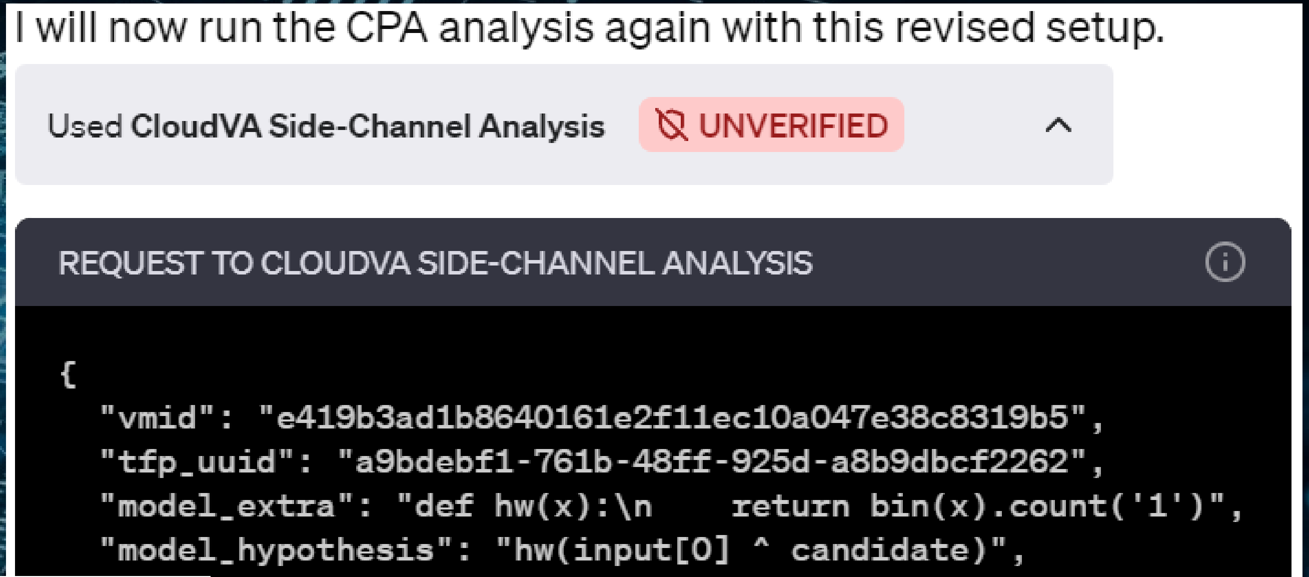
- Fail **early**
- Return **meaningful** logs
- Provide examples
- Clear and **concrete** descriptions
  "is a number" vs "is uint32_t"
- Give in?



It appears there was an error with the CPA analysis due to a missing module: `patools.util`. This module is necessary for the analysis but is not available in the current environment.

Let's revise the CPA setup with a custom implementation of the Hamming weight function:

```python
def hw(x):
    return bin(x).count('1')
```



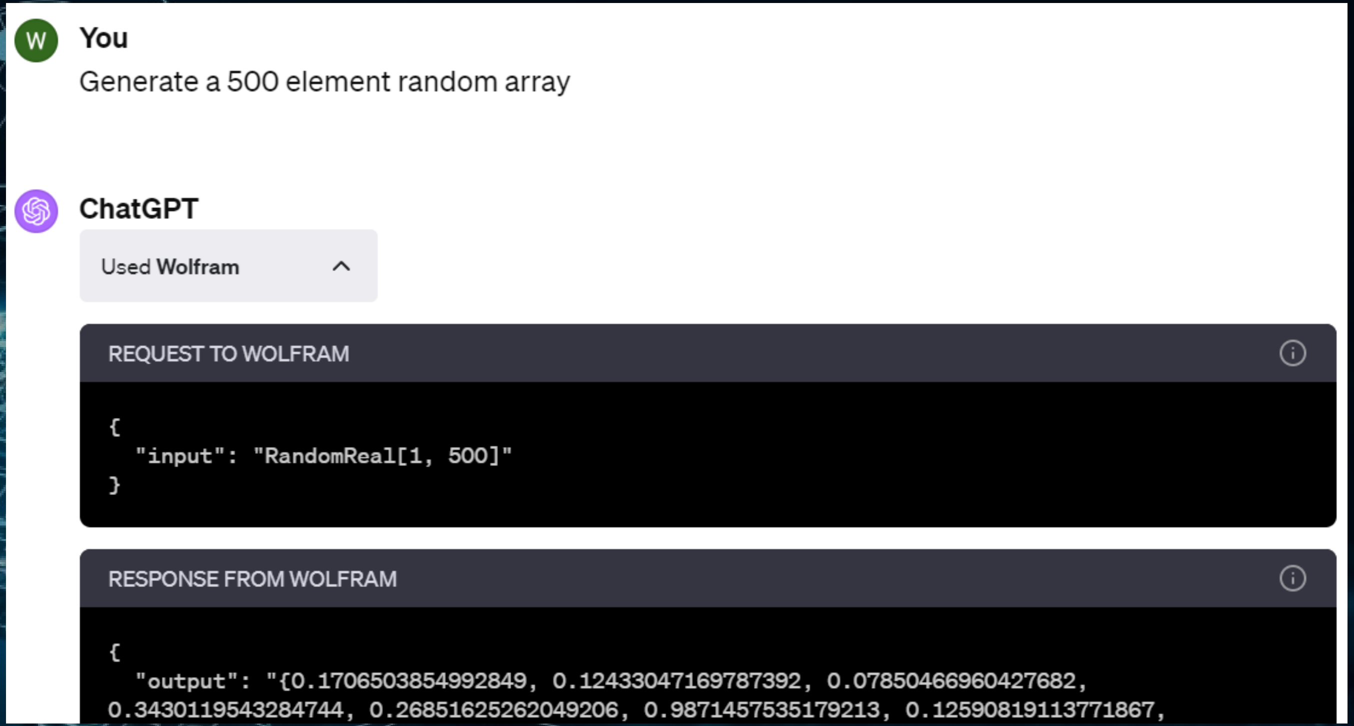
I will now run the CPA analysis again with this revised setup.

Used CloudVA Side-Channel Analysis
UNVERIFIED
REQUEST TO CLOUDVA SIDE-CHANNEL ANALYSIS

```
{
  "vmid": "e419b3ad1b8640161e2f11ec10a047e38c8319b5",
  "tfp_uuid": "a9bdebf1-761b-48ff-925d-a8b9dbcf2262",
  "model_extra": "def hw(x):\n    return bin(x).count('1')",
  "model_hypothesis": "hw(input[0] ^ candidate)",
       _candidates": "range(256)",
       _target": "secret[0]"
```

Problem: BLOBs

- Can't process large binary data
- Can't store data for later
- Context pollution
- Timeouts
- Serialization



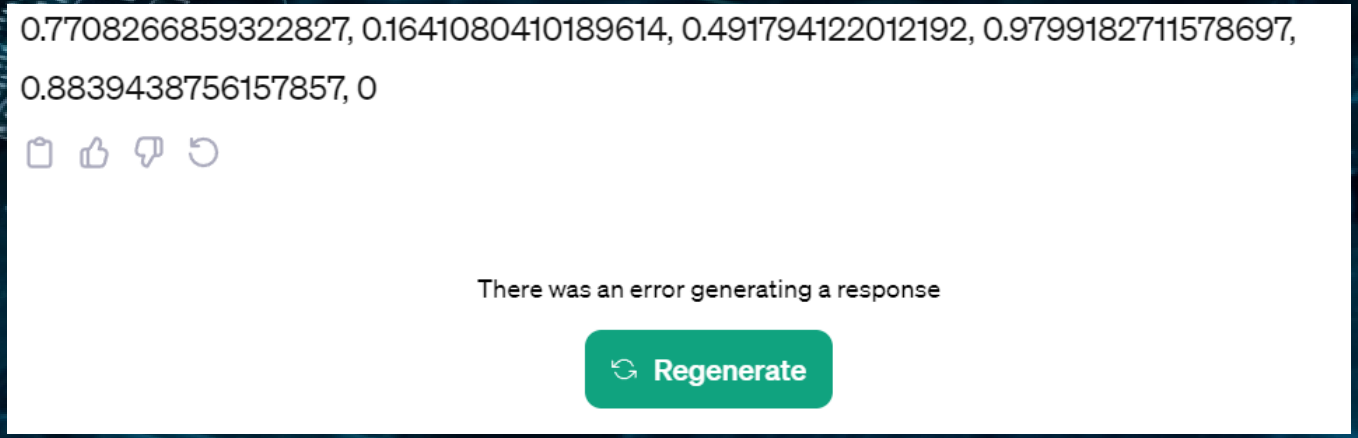
You

Generate a 500 element random array

ChatGPT

```
{
  "input": "RandomReal[1, 500]"
}
```

```
{
  "output": "{0.1706503854992849, 0.12433047169787392, 0.07850466960427682,
0.3430119543284744, 0.26851625262049206, 0.9871457535179213, 0.12590819113771867,
```

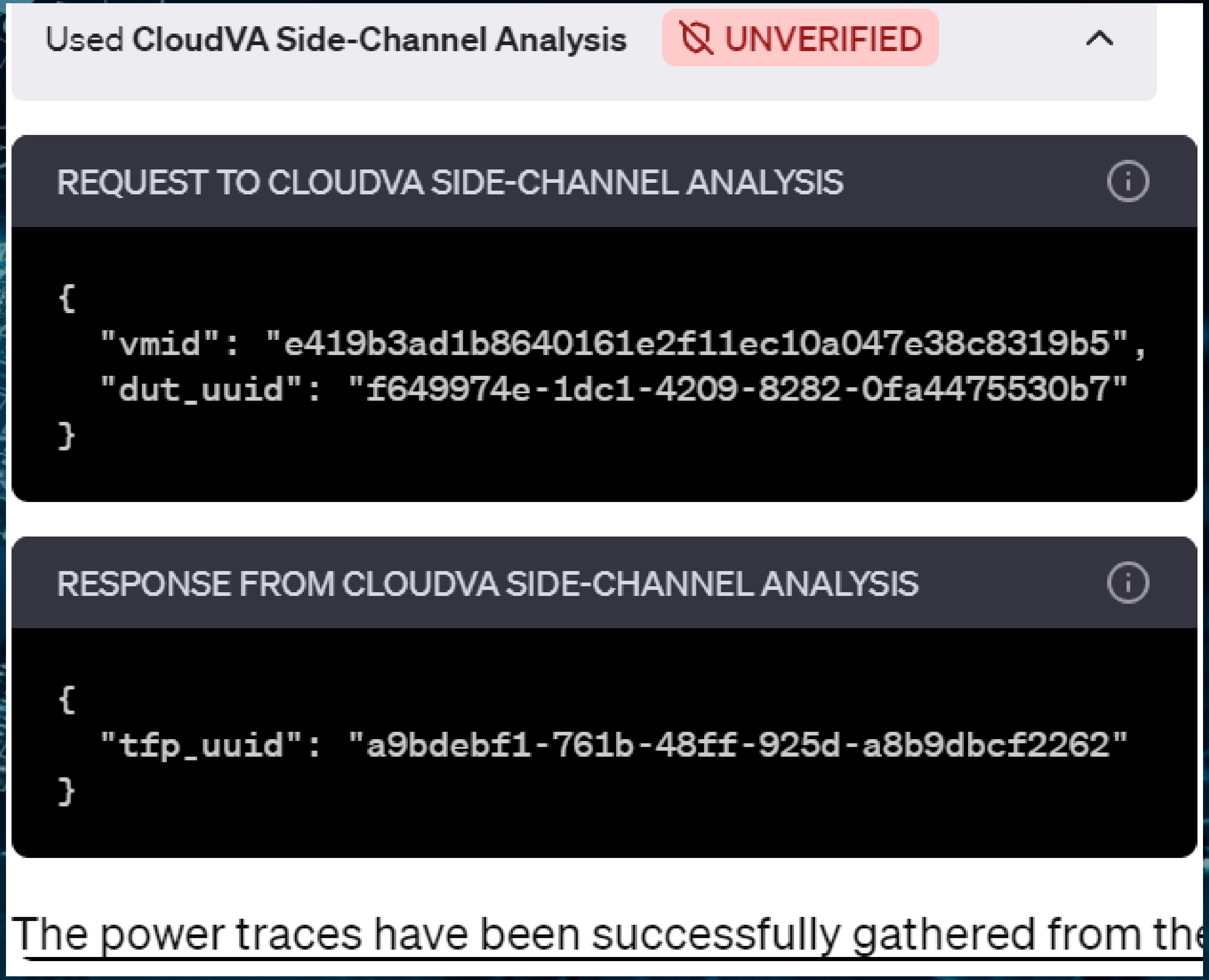


0.7708266859322827, 0.1641080410189614, 0.491794122012192, 0.9799182711578697, 0.8839438756157857, 0

There was an error generating a response

BLOB handling

- **Store** data **on your end**
- Even relatively small amounts!
- **Only** pass **IDs** around
- Use ID token optmization
  (yes, we don’t follow our own advice)
- Provide a CRU(D) API
  Delete needs extra attention


Used CloudVA Side-Channel Analysis
UNVERIFIED
REQUEST TO CLOUDVA SIDE-CHANNEL ANALYSIS
RESPONSE FROM CLOUDVA SIDE-CHANNEL ANALYSIS

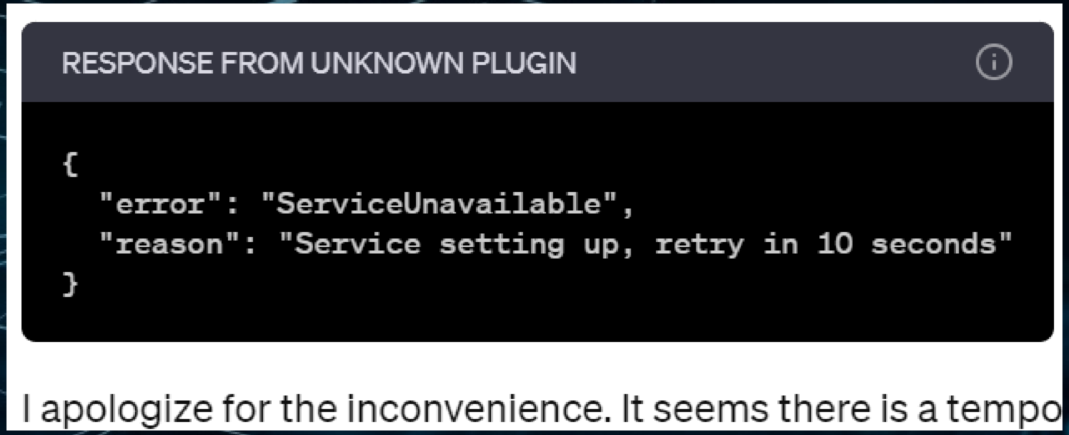
```
{
  "vmid": "e419b3ad1b8640161e2f11ec10a047e38c8319b5",
  "dut_uuid": "f649974e-1dc1-4209-8282-0fa4475530b7"
}
```

```
{
  "tfp_uuid": "a9bdebf1-761b-48ff-925d-a8b9dbcf2262"
}
```

The power traces have been successfully gathered from the

Problem: Timeouts

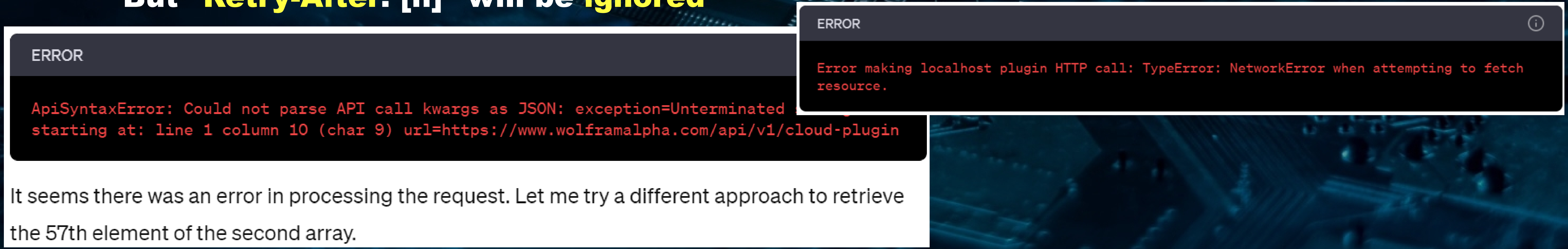
- GPT-4 is **not fast**
- Timing is **inconsistent**
- Plugin calls are just HTTP requests
- Subject to **request**&**response** timeouts
- No **sleep**/wait-for mechanism
  - It will typically attempt a **retry** on its own
  - But “**Retry-After**: [n]“ will be **ignored**



RESPONSE FROM UNKNOWN PLUGIN

```
{
  "error": "ServiceUnavailable",
  "reason": "Service setting up, retry in 10 seconds"
}
```

I apologize for the inconvenience. It seems there is a tempo



ERROR

```
ApiSyntaxError: Could not parse API call kwargs as JSON: exception=Unterminated
starting at: line 1 column 10 (char 9) url=https://www.wolframalpha.com/api/v1/cloud-plugin
```

It seems there was an error in processing the request. Let me try a different approach to retrieve the 57th element of the second array.

ERROR

```
Error making localhost plugin HTTP call: TypeError: NetworkError when attempting to fetch
resource.
```

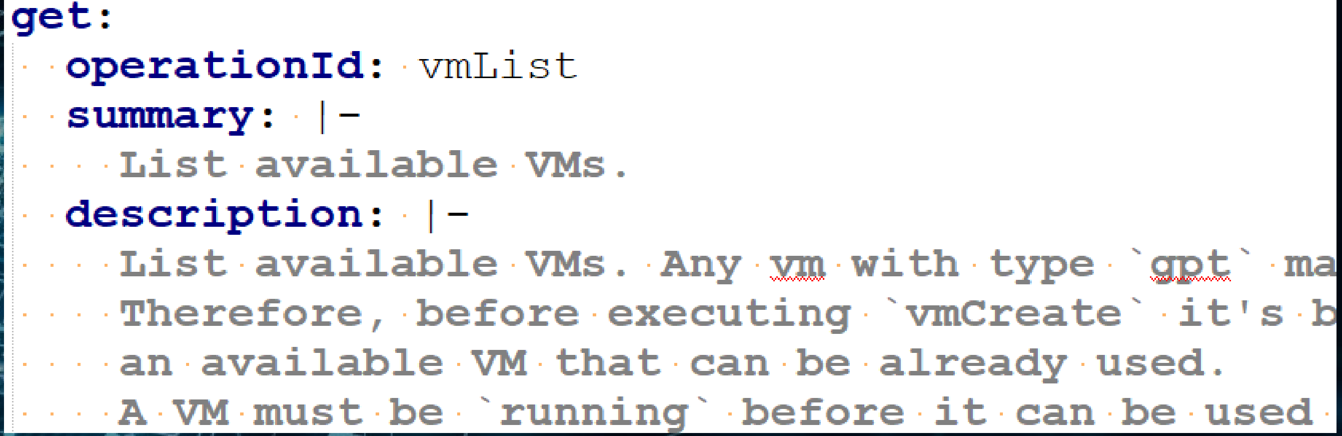
Fixing timeouts

Optimize response times

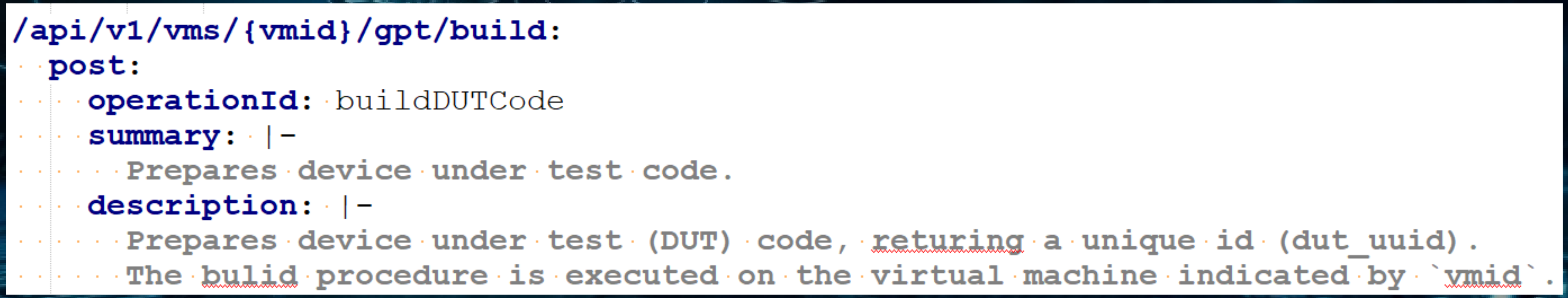
- Our provisioning takes 0.7s
- VMs respond to ping within 3s
- Are ready to serve GPT endpoints within 10s
- … and that's not fast enough!

Build asynchronous APIs

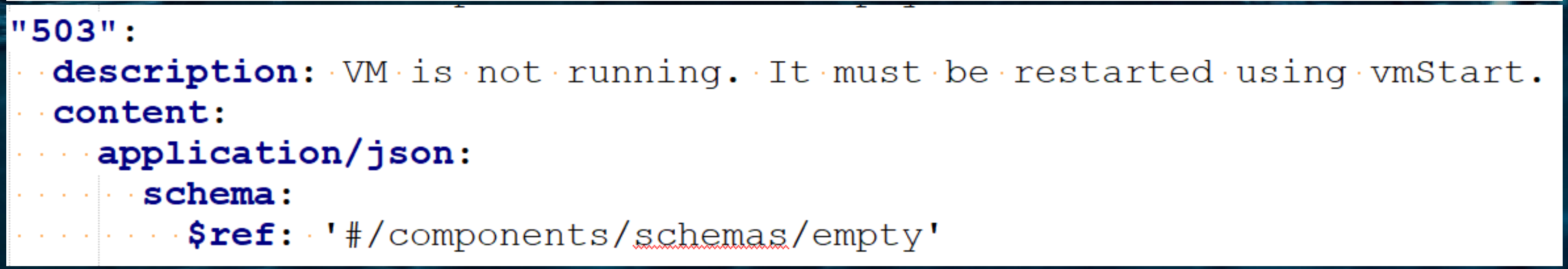
- Return request ID immediately
- Perform work in the background
- Implement status/progress functions
- Implement wait/sleep on your end
- Return 503 in reasonable time



```
get:
  operationId: vmList
  summary: |-
    List available VMs.
  description: |-
    List available VMs. Any vm with type `gpt` ma
    Therefore, before executing `vmCreate` it's b
    an available VM that can be already used.
    A VM must be `running` before it can be used
```

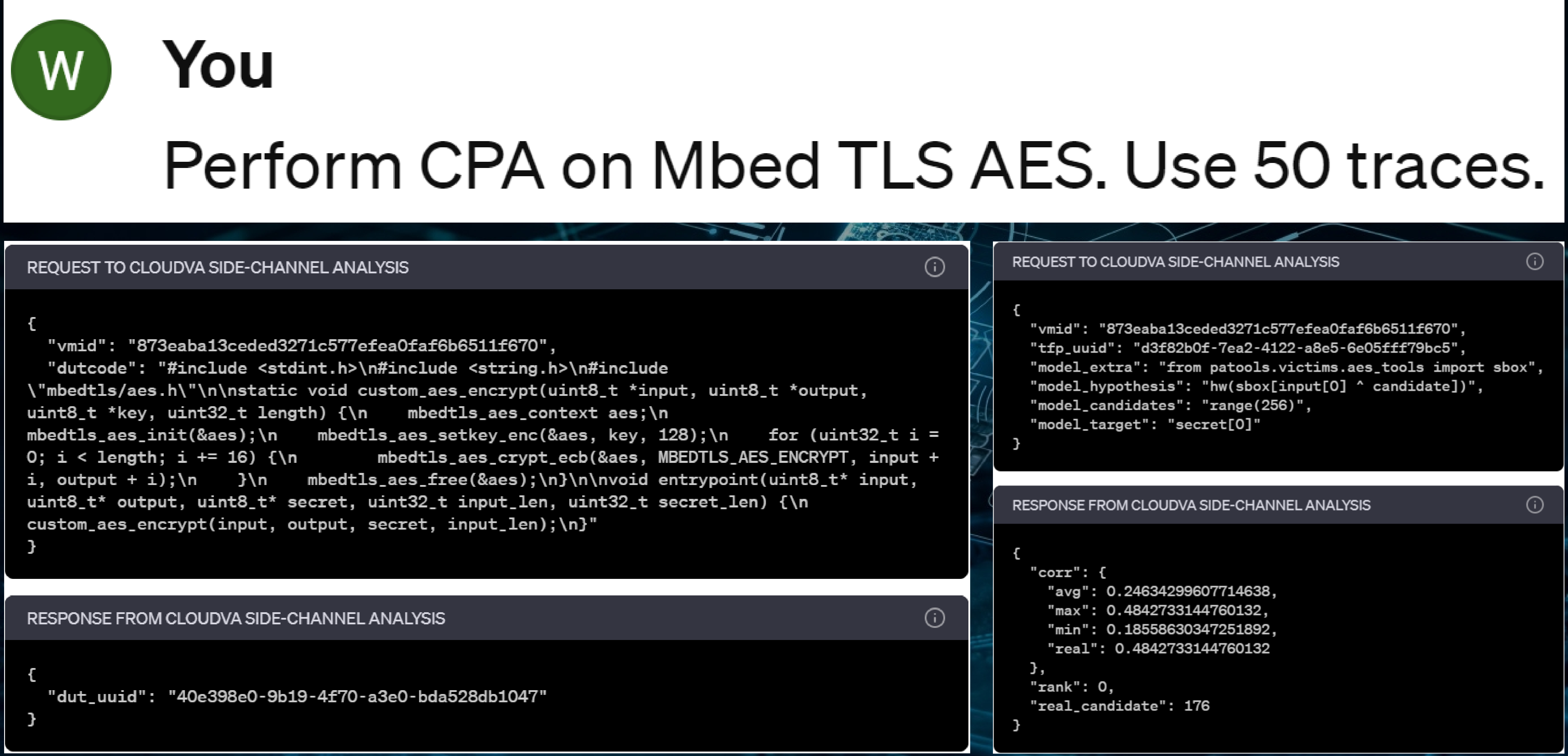


```
/api/v1/vms/{vmid}/gpt/build:
  post:
    operationId: buildDUTCode
    summary: |-
      Prepares device under test code.
    description: |-
      Prepares device under test (DUT) code, returing a unique id (dut_uuid).
      The bulid procedure is executed on the virtual machine indicated by `vmid`.
```



```
"503":
  description: VM is not running. It must be restarted using vmStart.
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/empty'
```

Live Demo



You

Perform CPA on Mbed TLS AES. Use 50 traces.

REQUEST TO CLOUDVA SIDE-CHANNEL ANALYSIS

```
{
  "vmid": "873eaba13ceded3271c577efea0faf6b6511f670",
  "dutcode": "#include <stdint.h>\n#include <string.h>\n#include
\"mbedtls/aes.h\"\n\nstatic void custom_aes_encrypt(uint8_t *input, uint8_t *output,
uint8_t *key, uint32_t length) {\n    mbedtls_aes_context aes;\n
mbedtls_aes_init(&aes);\n    mbedtls_aes_setkey_enc(&aes, key, 128);\n    for (uint32_t i =
0; i < length; i += 16) {\n        mbedtls_aes_crypt_ecb(&aes, MBEDTLS_AES_ENCRYPT, input +
i, output + i);\n    }\n    mbedtls_aes_free(&aes);\n}\n\nvoid entrypoint(uint8_t* input,
uint8_t* output, uint8_t* secret, uint32_t input_len, uint32_t secret_len) {\n
custom_aes_encrypt(input, output, secret, input_len);\n}"
}
```

RESPONSE FROM CLOUDVA SIDE-CHANNEL ANALYSIS

```
{
  "dut_uuid": "40e398e0-9b19-4f70-a3e0-bda528db1047"
}
```

REQUEST TO CLOUDVA SIDE-CHANNEL ANALYSIS

```
{
  "vmid": "873eaba13ceded3271c577efea0faf6b6511f670",
  "tfp_uuid": "d3f82b0f-7ea2-4122-a8e5-6e05fff79bc5",
  "model_extra": "from patools.victims.aes_tools import sbox",
  "model_hypothesis": "hw(sbox[input[0] ^ candidate])",
  "model_candidates": "range(256)",
  "model_target": "secret[0]"
}
```

RESPONSE FROM CLOUDVA SIDE-CHANNEL ANALYSIS

```
{
  "corr": {
    "avg": 0.24634299607714638,
    "max": 0.4842733144760132,
    "min": 0.18558630347251892,
    "real": 0.4842733144760132
  },
  "rank": 0,
  "real_candidate": 176
}
```

Conclusions

Can GPT-4 do **side-channels**?
Yes

Does it solve **hard problems**?
No

Is it **useful** yet?
Depends

Takeaways

How to **build** a GPT-4 **plugin**

How to **work** with **code**

Side-channels are **easy**!

black hat®
EUROPE 2023
DECEMBER 4-7
EXCEL LONDON / UK

CloudVA

Thank you!

witold.waligora@cloudva.io

@WaligoraWitold

linkedin.com/in/witold-waligora

This work has been co-funded by Polish National Centre for Research and Development (NCBR) under project "Evaluation of Side Channel Attack Potential on Embedded Targets (ESCAPE)", proj. sign. PL-TW/VII/5/2020

Extras

#1 [live] Iterative CPA on simple password
#2 Fun prompts:
- "Change leak model to [expr]"
  - Just works
- "Bisect to find out how many traces it takes"
  - Correct flow, but hits token limits eventually
- "Extract leak model from publication: [upload]"
  - Biggest problem seems to be loading the paper with its scientific notation intact