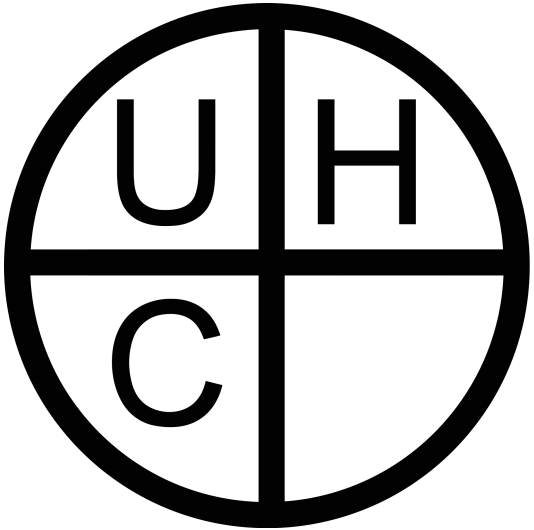
# The Underhanded Crypto Contest



# In case you haven’t heard of us...

- Like the underhanded C contest, but for crypto.
  - Every year since 2014.
  - Invent-your-own-vulnerability playground.
  - Crypto is hard.
- There’s now an Underhanded Rust contest, too.
  - Check them out: https://underhanded.rs.

# The team

- Taylor Hornby
  - Security Researcher & Consultant.
  - Focused on secure usable APIs and side-channel attacks.
  - On vacation from the infosec world studying quantum computing.
- Adam Caudill
  - Security Researcher & Consultant
  - Focused on Crypto & Secure Communications

# Crypto & Privacy Village

Thanks to Whitney & team!

# Looking back to the past

- Open contests do better than focused contests
- Plagiarism is a problem
- Good descriptions are important
- Spreading the word is hard

# This year

- Prizes: 15 ZEC from Zcash, $500 from NCC Group.
- Judge: JP Aumasson

The winner is...

# The winners

First Place: JP Smith & Will Song

Second Place: Neville Longbottom

# The winner: JP Smith and Will Song

- There’s a post-quantum algorithm called SIDH (Supersingular isogeny Diffie-Hellman key exchange).
- SIDH needs something called *supersingular* curves.
- The entry is a curve generator, implementing Broker’s algorithm.
- This is possibly the first backdoor that’s only exploitable if you have a quantum computer!

# The code...

```
# ...

broker = [
  (lambda x: x == 2,
    lambda x: [0,0,1,0,0]),
  (lambda x: x % 4 == 3,
    lambda x: [0,0,0,-1,0]),
  (lambda x: smallestCongPrime(x) % 4 != 3,
    lambda x: getCurve(x)),
  (lambda x: True,
    lambda x: [0,0,0,0,-1])]

def coeffs(x):
  for func in [lambda x: c[1](x) if c[0](x) else False for c in broker]:
    if func(x): return func(x)

# ...
```

```
broker = [
  (lambda x: x == 2,
    lambda x: [0,0,1,0,0]),
  (lambda x: x % 4 == 3,
    lambda x: [0,0,0,-1,0]),
  (lambda x: smallestCongPrime(x) % 4 != 3,
    lambda x: getCurve(x)),
  (lambda x: True,
    lambda x: [0,0,0,0,-1])]

x = [lambda x: c[1](x) if c[0](x) else False for c in broker]
for e in x:
    print(e(0))
```

Output:

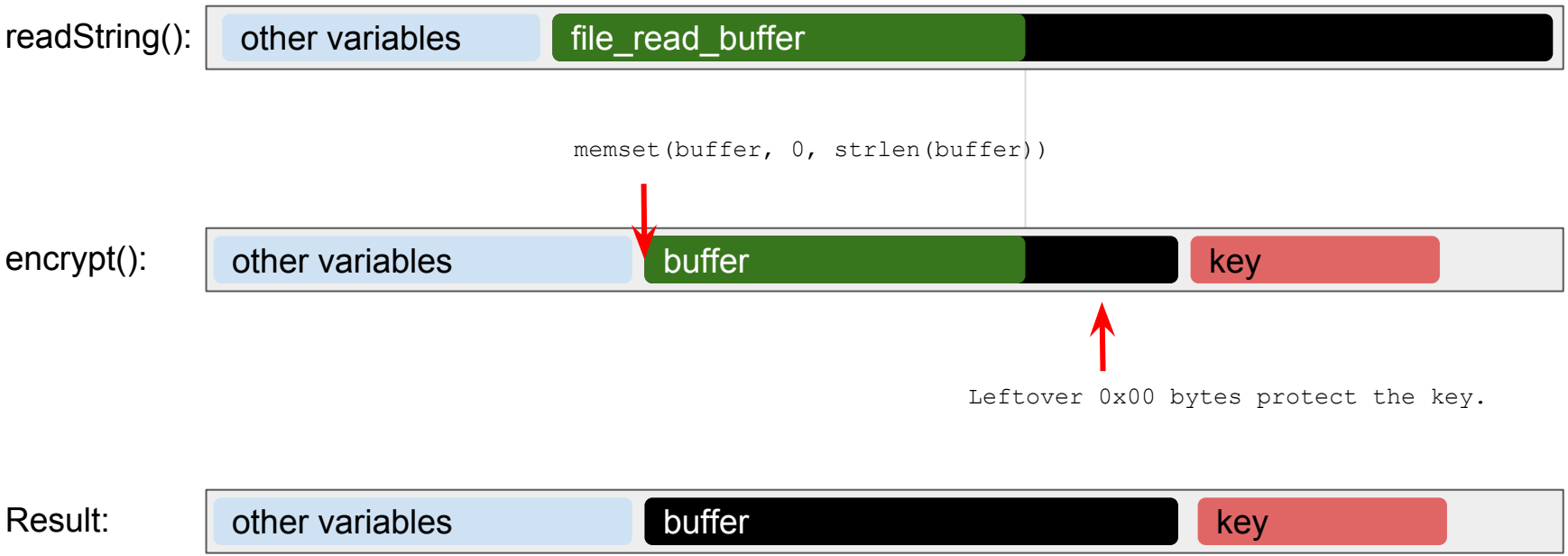
```
$ python test.py
[0, 0, 0, 0, -1]
[0, 0, 0, 0, -1]
[0, 0, 0, 0, -1]
[0, 0, 0, 0, -1]
```

# Second place: Neville Longbottom

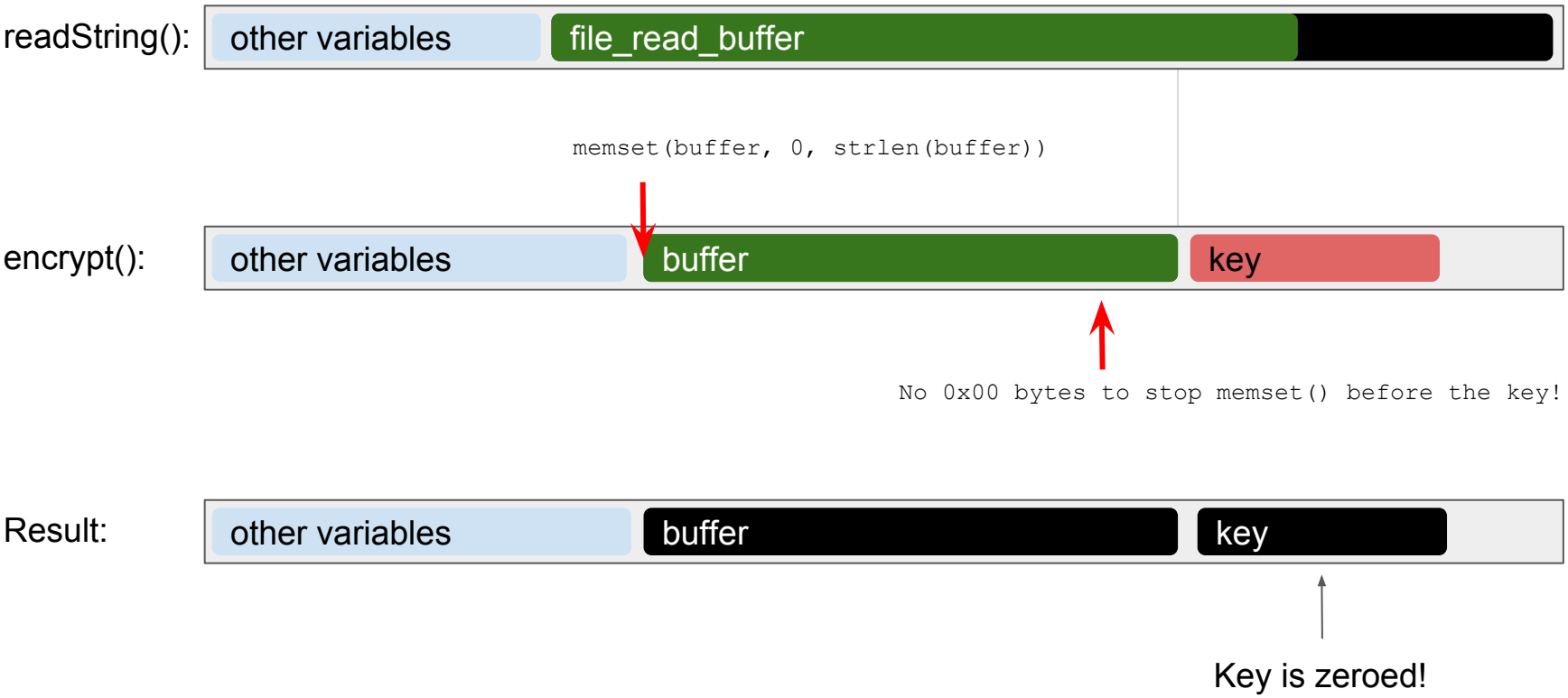
- It’s a tool for encrypting files, written in C.
- It uses AES in CBC mode.
- It comes with a test suite that compares its output against OpenSSL.
- So it should be good, right?

- It uses the fact that, in C, the same stack memory gets reused by different functions, to encrypt short messages properly and long messages with an all-zero key.

# Bug Inactive



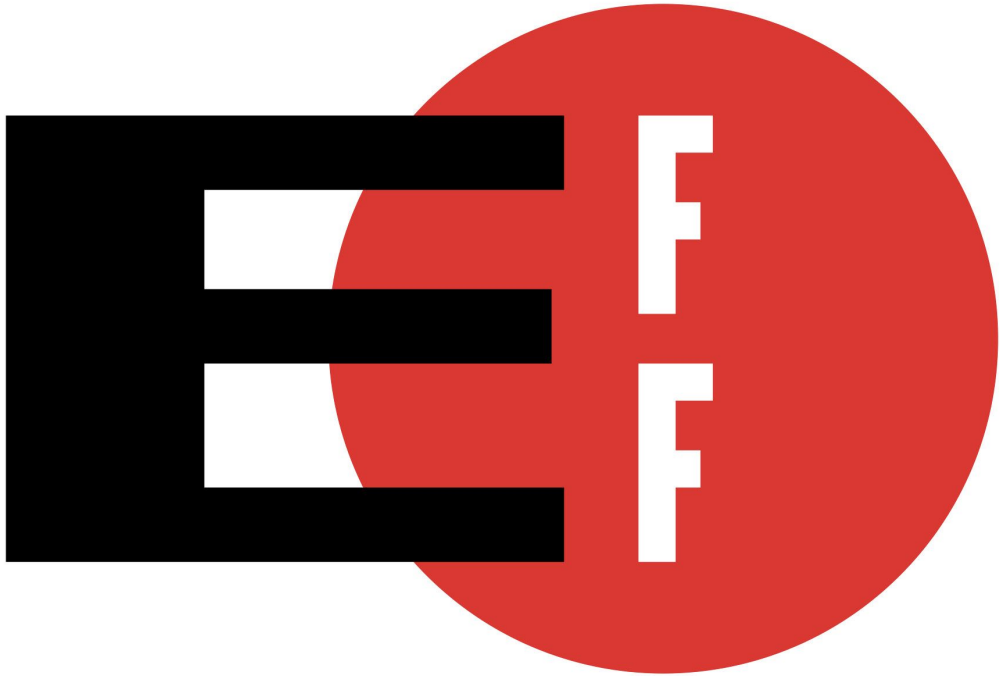
# Bug Active


readString():
other variables
file_read_buffer
memset(buffer, 0, strlen(buffer))
encrypt():
other variables
buffer
key
No 0x00 bytes to stop memset() before the key!
Result:
other variables
buffer
key
Key is zeroed!

# The backdoor...

- Seems to work fine for most input ranges.
- You can compare its output to test vectors (short ones).
- You can compare its output against OpenSSL (short messages).
- But all long messages can be decrypted with the all-zero key.

- The decrypt function has the same bug, so it decrypts its own ciphertexts correctly (except for a short range of sizes).



# All the contestants

- Average Security Guy — XOR-shuffle bug zeroes CSPRNG seed.
- Ella Rose — Good crypto stored inside Python's pickle.
- Joseph Birr-Pixton — A developer-unfriendly API in OpenSSL.
- JP Smith & Will Song — Backdoored SIDH curve generator.
- Neville Longbottom — Memory safety bug, big ciphertexts decryptable.
- Sc00bz — Storing a fast password verifier in the salt.
- Sc00bz (again) — Web application in E2E-encrypted chat app.
- TheStig — Detectable steganography, stego inception.

https://github.com/UnderhandedCrypto/entries

# Next year

- Announcing details in January 2018
- Larger prizes
- More challenges

# Thank you!

nccgroup