CTF ~~write~~ talk-up

- Hackvent 2017 - Day 15: Unsafe Gallery
- 34C3 Junior CTF – crypto: kim (easy)

HV17 - Day 15: Unsafe Gallery

- The List of all Users of the Unsafe Gallery was leaked (See account list).
- With this list the URL to each gallery can be constructed. E.g. you find Danny's gallery here.
- http://challenges.hackvent.hacking-lab.com:3958/gallery/bncqYuhdQVey9omKA6tAFi4rep1FDRtD4H8ftWiw
- Now find the flag in Thumper's gallery.

Gallery

screenshot missing

Gallery

- bncqYuhdQVey9omKA6tAFi4rep1FDRtD4H8ftWiw
- looks like base64

```
$ echo bncqYuhdQVey9omKA6tAFi4rep1FDRtD4H8ftWiw | base64 -d | xxd
00000000: 6e77 2a62 e85d 4157 b2f6 898a 03ab 4016  nw*b.]AW......@.
00000010: 2e2b 7a9d 450d 1b43 e07f 1fb5 68b0       .+z.E..C....h.
```

- 30 bytes / 240 bits of data
- Might be a hash, but which one and what‘s the input?

account list

- CSV file with 43799 entries
- Header:

  id,prename,name,address,zip,city,email,crmId,
  memberType,pictureCount,galleryCount,mbUsed,
  logCorrelationId,advertisingId,state
- 82 Dannys, 70 of them with state „active“
- Only two with pictureCount == 15
- Let‘s create a dictionary

dictionary

```
$ grep Danny accounts.csv | grep active |
grep ',15,' | tr ',' '\n' | sort -u > dict
```

- **dict contains 25 lines**
- **So we have the possible input but still no idea what hash function was used.**
- **MDXfind to the rescue!**

MDXfind

- https://www.hashes.org/mdxfind.php
- not (yet) open source
- supports 537 different hashing algorithms
- can check them ALL at once
- partial matching, reversed hashes
- CPU only (still faster than hashcat in some usecases)
- 32/64bit, x86, ARM[678], POWER8
- Linux, FreeBSD, MacOSX, AIX, Windows

crack the hash

- **I‘m not aware of a 240 bit hash function**
- **might be truncated or two truncated hashes**

```
$ cat hashes

6e772a62e85d4157b2f6898a03ab40162e2b7a9d450d1b43e07f1fb568b0

6e772a62e85d4157b2f6898a03ab4016

6e772a62e85d4157b2f6898a03ab40

162e2b7a9d450d1b43e07f1fb568b0
```

crack the hash

```
$ mdxfind -h ALL -h '!user,!salt,!drupal,!md5x'
-f hashes dict
```

```
SHA256x01 6e772a62e85d4157b2f6898a03ab40:Danny.Dixon@sunflower.org
```

figuring out the rest

- `$ echo -n Danny.Dixon@sunflower.org | sha256sum`

  `6e772a62e85d4157b2f6898a03ab40162e2b7a9d7e143f91b43e07ffc7ed5a2c`

  `6e772a62e85d4157b2f6898a03ab40162e2b7a9d450d1b43e07f1fb568b0`
- **dafuq?**
- **It becomes obvious when using base64 encoding**

  `bncqYuhdQVey9omKA6tAFi4rep1+FD+RtD4H/8ftWiw=`

  `bncqYuhdQVey9omKA6tAFi4rep1 FD RtD4H 8ftWiw`

putting it together

```
import sys
import hashlib

for line in sys.stdin:
  line = line.rstrip()
  hash = hashlib.sha256(line).digest().encode('base64')
  print hash.replace('/', '').replace('+', '').replace('=', '')
```

putting it together

```
URL=http://challenges.hackvent.hackinglab.com:3958/gallery
for i in `grep Thumper accounts.csv | cut -d',' -f7 | ./solve.py`;
  do curl -s $URL/$i | grep 'HV17-';
done
```

flag: HV17-el2S-0Td5-XcFi-6Wjg-J5aB

Questions?

34C3 Junior CTF – crypto: kim (easy)

Check this out!!!!!!!!!!\x80\x00....

Update: Source

source code

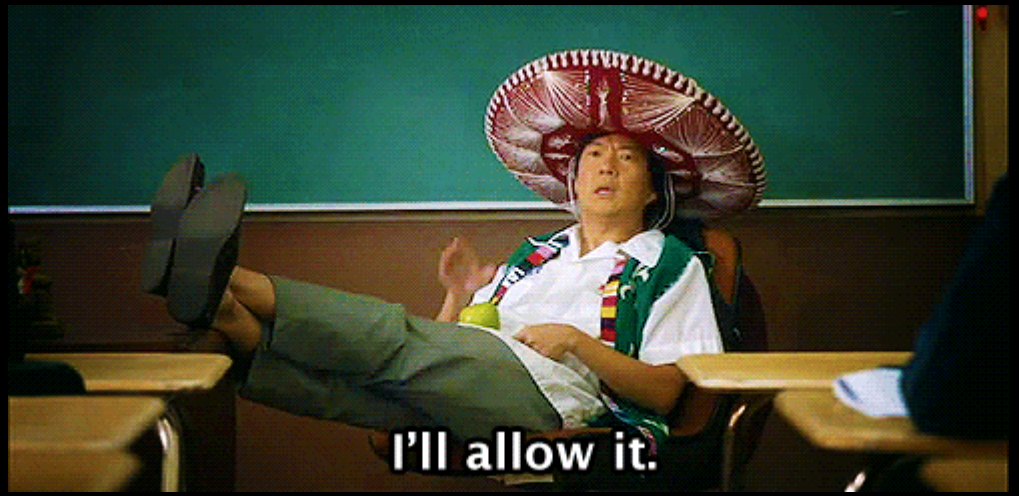
```
def mac(msg):
  return hashlib.sha1(SECRET + msg).hexdigest()
@route('/files/<umac>/')
def download(umac):
  delim = msg = ''
  for k,v in request.query.allitems():
    msg += delim + k + '=' + v
    delim = '&'
  if mac(msg) == umac:
    return static_file(request.query.f, root='./files')
  else:
    return redirect('/files/' + mac('f=dont.gif') + '/?f=dont.gif')
```

files

- http://localhost:8888/files/
- sample.gif
- dont.gif
- flag

sample.gif

- http://localhost:8888/files/952bb2a215b032abe27d24296be099dc3334755c/?f=sample.gif



what we know

- SHA1(SECRET + 'f=sample.gif')
- 952bb2a215b032abe27d24296be099dc3334755c
- my solution?
- crack the SECRET
- again using MDXfind
- algorithm: SHA1PASSSALT
- PASS == SECRET
- SALT == f=sample.gif

cracking the SECRET

- `$ mdxfind -f hash -s salt -h ^SHA1PASSSALT$ rockyou.txt`
- `952bb2a215b032abe27d24296be099dc3334755c:f=sample.gif:somethingstupid`
- **SECRET == somethingstupid**

get the flag

```
$ echo -n somethingstupidf=flag | sha1sum
20db4f5f09af7b30e85d464cc743ef7a1eae24b8
$ curl -s http://localhost:8888/files/20db4f5f09af7b30e85d464cc743ef7a1eae24b8/?f=flag
```

34C3_a11_yOu_ne3d_is_puMp_and_dump

Questions?

can anyone spot the problem?

- What if the secret is not something stupid?
- Might still be cracked by using brute force
- But unlikely in a timely fashion
- Intended solution:
  hash length extension attack (not covered in this talk)