CMSC389R

Cryptography I

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

homework questions

Homework 7?

Extra credit homework?

cryptography

cryptography

There’s more than just Caesar cipher?

cryptography

- Science behind securing digital information
  - Authentication
  - Data integrity
  - Message secrecy
  - Access control

history

- Secret codes to protect information
  - Substitution
    - Caesar, ROT13
  - Transposition
    - Rail Fence, Route Cipher
- Historically used for military purposes
- “Art form” - little scientific rigor

modern cryptography

- Mathematically rigorous
  - Formal definitions
  - Provability
  - Well-defined assumptions

uh, math?

- Out of scope for this course
- Take CMSC456, MATH456, or CMSC489R/ENEE459E

hashing

- One-way cryptographic function
- Input -> hashing algorithm -> output
  - Small input change -> large output change
  - Unique* output for every unique input

*collisions

common hashing algorithms

- MD5 - very common, insecure
  - Brute-forceable, suffers from collisions
- SHA1 - also common, recently declared insecure
  - Google & CWI *SHAttered* https://shattered.io/static/shattered.pdf
  - Used by git for commit hashes https://gist.github.com/masak/2415865
- SHA256 - common, not shown to be feasibly insecure
- SHA512 - more bits more secure?

hashing on linux

- MD5: md5sum <text or file>
- SHA1: sha1sum <text or file>
- SHA256: sha256sum <text or file>
- SHA512: sha512sum <text or file>

Common syntax = easy to hash!

**NOTE: use echo -ne to NOT hash a trailing newline in your input, and to interpret escape sequences (e.g. \t as tab)

```
echo -ne “Hello World!” | md5sum
```

hashing in python

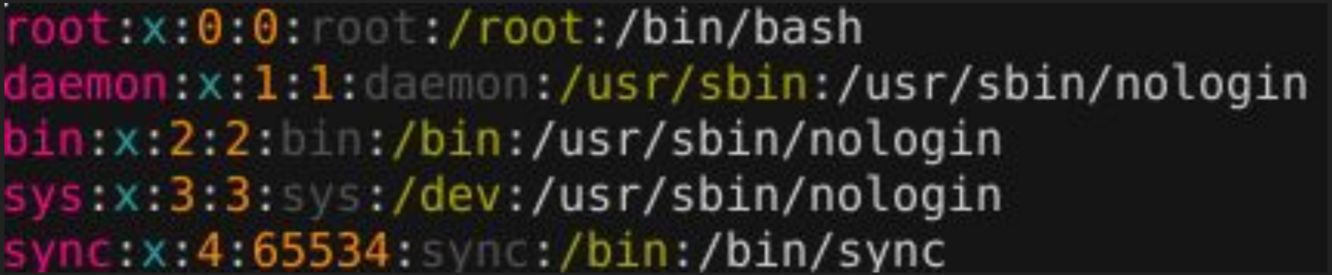
- Python’s hashlib
- Provides hashing capabilities for MD5, SHA1, SHA224, SHA256, SHA512, and more

```
>>> import hashlib

>>> h = hashlib.md5(“Hello CMSC389R!”)

>>> print h.hexdigest()

a0d708f132eda63ce4b0d11b60ec701c
```

where to find hashes?

- OS passwords
  - Linux: /etc/shadow
  - Windows: C:/Windows/System32/config
    - Locked while OS is booted, loaded into RAM
    - Tools to recover these hashes
- Leaked databases
- From OSINT or Forensics discoveries

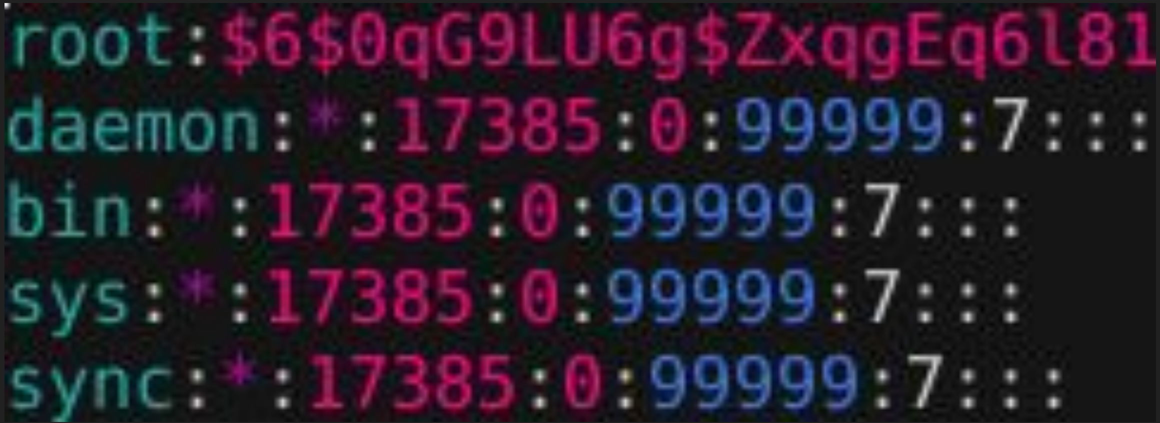
/etc/passwd



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

- login ID, password, user ID, group ID, username, home folder, command interpreter
- If password is ‘x’ then it is located in /etc/shadow

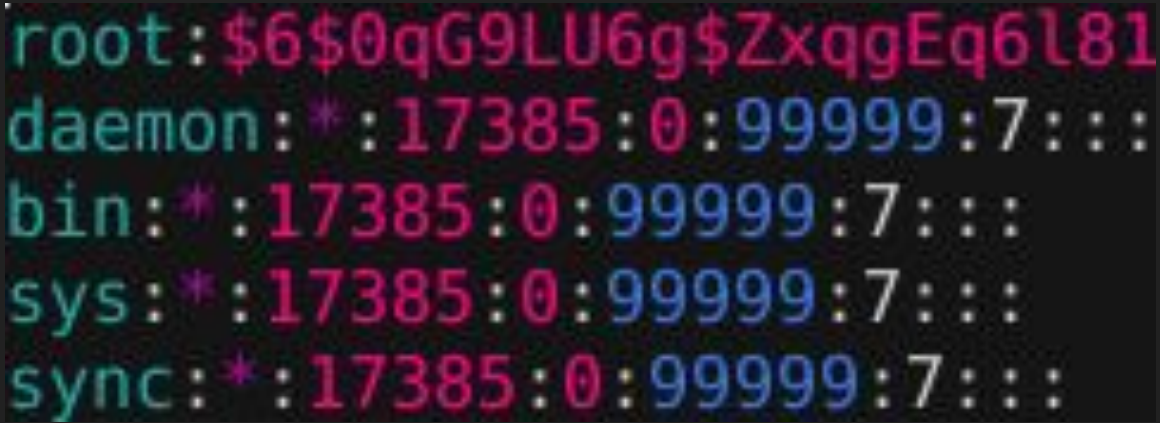
/etc/shadow



```
root:$6$0qG9LU6g$ZxqgEq6l81
daemon:*:17385:0:99999:7:::
bin:*:17385:0:99999:7:::
sys:*:17385:0:99999:7:::
sync:*:17385:0:99999:7:::
```

- 1. login name, 2. encrypted password (crypt(3)), 3. date of last pass change, 4. min pass age, 5. max pass age, 6. pass warning period, 7. pass inactivity period, 8. account expiration, 9. reserved

/etc/shadow



```
root:$6$0qG9LU6g$ZxqgEq6l81
daemon:*:17385:0:99999:7:::
bin:*:17385:0:99999:7:::
sys:*:17385:0:99999:7:::
sync:*:17385:0:99999:7:::
```

- ‘*’ or ‘!’ for password means account can’t be logged in normally w/ password
- password formatted as $id$salt$encrypted

hash recovery

I thought it was one way?

hash recovery

brute forcing

- Define message space
  - All lowercase letters, letters + numbers, symbols, emojis?
- Enumerate each possible combination of items in your message space
  - aaaa, aaab, ..., zzzz
- Hash each enumeration and test against target hash

mask attacks

- Smarter brute force
- Say you know how the message was formatted...
  - 3 lowercase letters then 5 numbers
- Enumerate all combinations of your more-specific message space
  - aaa00000, aaa00001, ..., zzz99999
- Hash each enumeration and test against target hash

dictionary attacks

- Say you know the password was used from a leak...
  - i.e. SecLists
- Download password list
- Hash each password from list and test against target hash

hybrid attacks

- Brute force + dictionary
- Say the user’s password was found in a leak, and they only changed it slightly...
    - e.g. blink182 -> blink182!
- Download password list
- Enumerate potential modifications and append/prepend to password from list
- Hash each string and test against target hash

password recovery tools

- John the Ripper - cracks password hashes for UNIX and Windows
- Hashcat - cross-platform general hash cracker, better GPU support

john the ripper

- Syntax: john <flags> <password-files>
- Flags:
  - -wordlist:LIST_FILE
  - -incremental:MODE
- May need to “unshadow” /etc/passwd file
  - i.e. if passwords are ‘x’ and not hashes
  - unshadow /etc/passwd /etc/shadow
- Manpage kinda sucks, http://www.openwall.com/john/doc/

hashcat

- Syntax: hashcat <flags> <file-with-hashes>
- Flags:
  - -m <hash-type-code>
    - 0 = MD5, 100 = SHA1, 1400 = SHA256
  - -a <attack-mode-code>
    - 0 = dictionary, 3 = bruteforce, etc
  - -1 <custom-password-mask>
  - And many more... use the manpage!

homework #7

has been posted.

Let us know if you have any questions!

This assignment has X parts.

It is due by 4/5 at 11:59PM.