Security Tools Lab 1

Module 1 – Password & Hashing

Schedule

Week	STL1 (Mon)	Lab	STL2 (Fri)	Lab
1	15-May	-	19-May	-
2	22-May	1	26-May	1
3	29-May	2	2-Jun	*2
4	5-Jun	3	9-Jun	3
5	12-Jun	4	16-Jun	4
6	19-Jun	5	23-Jun	5
7	26-Jun	6	30-Jun	6
8	3-Jul	7	7-Jul	7
9	10-Jul	8	14-Jul	8
10	17-Jul	9	21-Jul	9
11	24-Jul	Project	28-Jul	Project
12	31-Jul		4-Aug	
13	7-Aug		11-Aug	
14	14-Aug		18-Aug	
15	21 Aug - Project Submission			

*Due to PH, recorded lecture will be released online

- **Classes start in Week 2**
- **Vesak Day PH class will be released online**
- **9 Labs (54%)**
  - 6 marks each for a total of 54 marks
  - Hand-out on Tue, Hand-in on Wed 2359 in 8 days
- **1 Project (44%)**
  - 3-4 choices (Individual or Group options)
  - Week 11 to Week 15 (4 weeks)
  - Hand-out 24-Jul (STL1 and STL2)
  - Hand-in 21-Aug (STL1 and STL2)
- **Term Feedback (2%)**

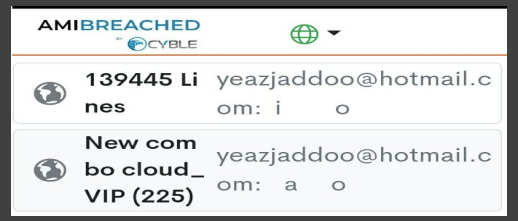
Passwords

- How long does it take to brute force ‘Password1’?

Assume 100 billion guesses per second

Passwords

- Password examples
  - password1234 – 23 bit quality
  - j6gxezn0e11q – 58 bit quality
  - rJovntNRz3an – 65 bit quality
  -):(xR{8V5vb/Z&78!Uoa – 124 bit quality
  - Bit quality is the log-base-2 of the sample space
  - http://www.fon.hum.uva.nl/rob/PasswordStrength.html
  - https://www.grc.com/haystack.htm
- How passwords are stored on the server?



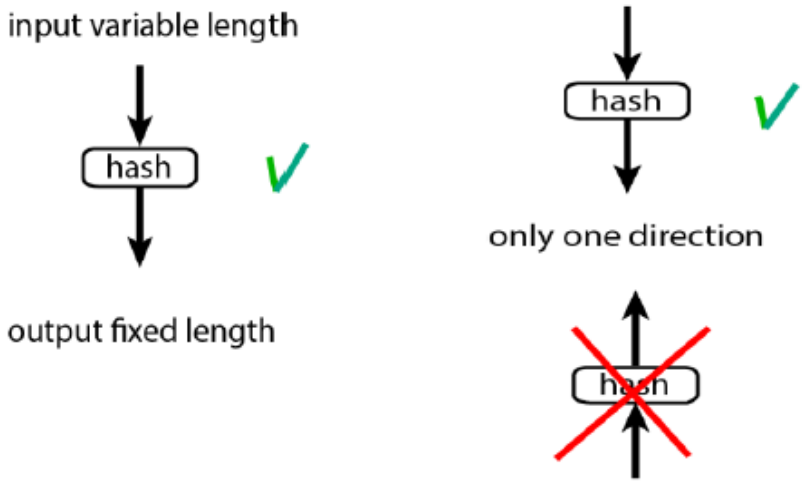
- Does password complexity actually help?
- Are there better options?

Bits	Strength
0-64	Very weak
64-80	Weak
80-112	Moderate
112-128	Strong
≥ 128	Very strong

```
C:\Users\MSSD>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018,
Type "help", "copyright", "credits" or "licer
>>> import math
>>> math.log2(36**12)
62.039100017307746
>>>
```

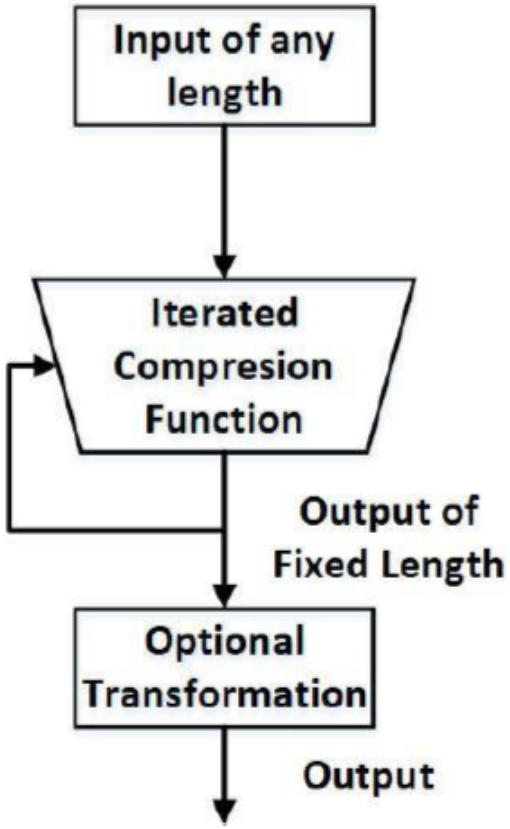
Name	Surname	Login	Password
Edwin	Franco	edwinf	rJovntNRz3an
Yeaz	Jaddoo	yeazj	password1234

Hash function	Digest length	Secure?
MD2	128 bits	No
MD4	128 bits	No
MD5	128 bits	No
SHA-1	160 bits	No
SHA-256	256 bits	Yes



Hashes

- Hash is a function that :
  - Accepts input of any length
  - Produces output of fixed length
  - Is deterministic
- Additional properties
  - H(x) is relatively easy to compute
  - Pre-image resistance (1-way function)
    - For a given y is computationally infeasible to find x, such that H(x)=y
  - Collision-free
    - It is computationally infeasible to find such x and x', where x <> x', that H(x) = H(x')



Hashes

- Input is split into blocks of fixed length
- Each block is then passed to the compress function along with output from last iteration
- Repeats until all blocks are processed
- Outputs hash value of fixed length

Name	Surname	Login	Password
Edwin	Franco	edwinf	8257e9022bb55b5a8d80427e1d434712
Yeaz	Jaddoo	yeazj	bdc87b9c894da5168059e00ebffb9077

Hashes – Salting

- Prevention of guessing attack
- Stored without salting

Name	Surname	Login	Password
Yeaz	Jaddoo	yeazj	bdc87b9c894da5168059e00ebffb9077
Edwin	Franco	edwinf	bdc87b9c894da5168059e00ebffb9077

- Salt randomizes the hash

Name	Surname	Login	Password	Salt
Yeaz	Jaddoo	yeazj	2e3a83257957bffd0dbaea89d61684dc	1
Edwin	Franco	edwinf	7e37cdf7329e00a28b613cc817589c37	e

Hashes – Not a solution for everything!

Is our password safe if it is hashed on the server?

There is still plenty of possibilities how to steal your password:

- Online guessing attack–Attacker simply tries to guess the password
- Social engineering/phishing – Unwittingly reveals the password to the attacker
- Eavesdropping – Attacker intercepts the password from the network traffic
- Malware – Captures the passwords and sends it to the attacker (keyloggers)

Defenses

- Use encrypted connection to server
- Use 2FA
- Deploy security software like AV

MD5 (Message Digest)

First published in 1992

Digest Length (Output) – 128bit

Block size – 512bit

Still popular

- **Eatigo** breach (Oct 2018) – 2.8M accounts, unsalted MD5 hashes
- **Creative** breach (May 2018) – 500k accounts, salted MD5 hashes
- **123RF** breach (Mar 2020) – 8M accounts, salted MD5 hashes

Suffers from extensive vulnerabilities

Can be cracked by bruteforce attack, or with use of Rainbow Tables

Attacks on Hashes

Brute Force

- The most simple and straight forward attack
- Calculates a hash on every attempt => Slow

Look-up Table

- Table of precomputed hashes
- Fastest, but requires a lot of storage resources

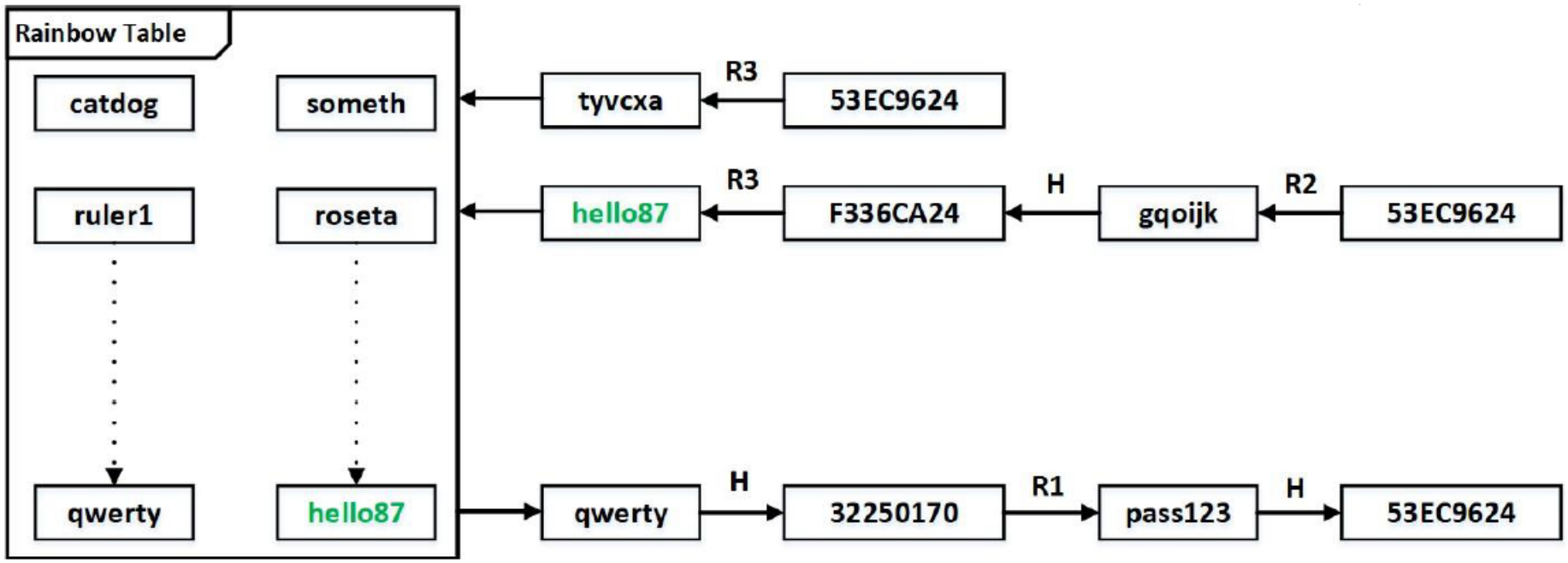
Rainbow Table

- Trade-off between time and space complexity
- Difficult if salting is used

Rainbow Table

- Invented by Phillippe Oechslin in 2003
- M. Hellman had an initial idea in 1980
- Precomputed table for reversing hash functions
- Usually used in recovering a password up to a certain length consisting of a limited set of characters
- Uses reduction functions R_x that maps hash values back into values in a finite set of passwords P
- Reduction function is NOT actually an inverse of the hash function
- Table contains precomputed hash chains (e.g., length of chain n = 2)
  - Table stores only start and end points of chain

Rainbow Table



- Input hash is 53EC9624
- Hash is step by step reduced by chain 1,2…,n reduction functions, while the result is compared to all end points in the table (as only start and end is stored)
- If match found (hello87), then the end point is reduced by chain of reduction functions until plaintext is found (pass123)

Hashcat

- CPU or GPU-based
- Multi threaded
- Multi-Algorithm (MD5, SHA1, Whirlpool, RipeMD)
- Multi-OS (Linux, Windows, MAC)
- All attack modes can be extended by rules
  - Bruteforce
  - Dictionary
  - Mask
    - Char sets (?l = abcdefghijklmnopqrstuvwxyz, ?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ, ?d = 0123456789
    - password?d?d?d?d (password0000 - password9999)
  - Permutation
    - Dictionary attack with all permutations of itself
  - Hybrid