Password Cracking, Approaches and Tools

Overview

1. Probabilistic Password Cracking using Context-Free Grammars

2. John the Ripper and its 3 approaches: brute force, dictionary attack and rainbow tables

3. Demonstrating John the Ripper and password best practices

Paper

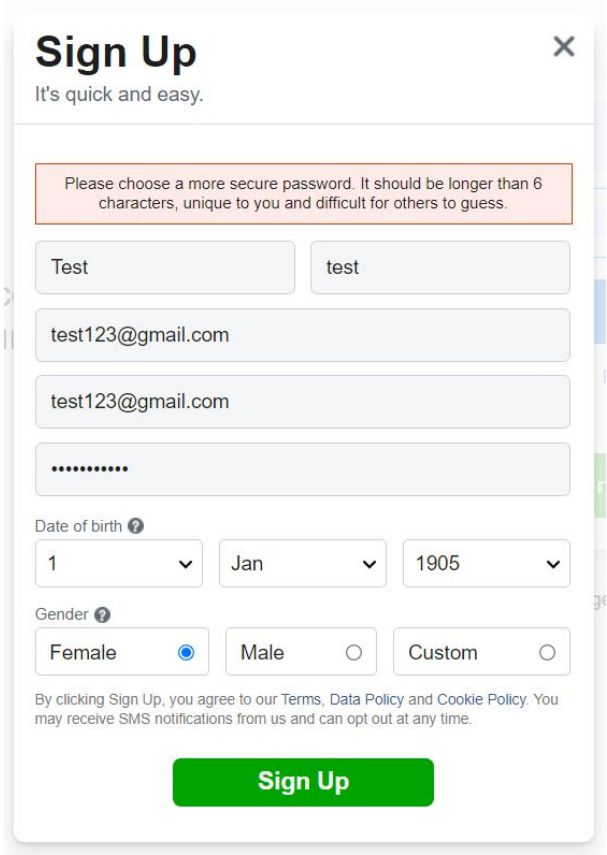
Password Cracking Using Probabilistic Context-Free Grammars

2009 30th IEEE Symposium on Security and Privacy
Matt Weir, Sudhir Aggarwal, Breno de Medeiros, Bill Glodek

Background

- Researchers wanted to extend the traditional dictionary attack
- Not in dictionary = never guessed
- Password creation policies mean regular words are less likely to be the final password choice

  e.g. **`password`** → **`password123!`**



New Approach

- Analyse leaked password structures to generate a probabilistic context-free grammar
- Use grammar to generate popular password structures
- Faster than brute-force
- 28% to 129% more passwords cracked than John the Ripper

Probabilistic Password Cracking Process

1. Derive base structures
2. Calculate probabilities
3. Derive pre-terminal structures
4. Generate terminal structures
5. Attempt password guesses

Type	Symbols	Example String
Alpha*	abcdefghijklmnopqrstuvwxyz	cat
Digit	0123456789	1300655506
Special	!@#$%^&*()-_=+[]{};':",./<>?	-_-

* can differ based on language & difference in case not considered

1password^

D_1 L_8 S_1

Derive base structures

1password^	$D_1\ L_8\ S_1$	0.6
4wordpass$	$D_1\ L_8\ S_1$	
4alicebob^	$D_1\ L_8\ S_1$	
login1##	$L_5\ D_1\ S_2$	0.4
qwert5!!	$L_5\ D_1\ S_2$	

1 2 3 4 5

Calculate probabilities

$D_1\ L_8\ S_1$	0.6
$L_5\ D_1\ S_2$	0.4

1password^	$D_1\ L_8\ S_1$
4wordpass$	$D_1\ L_8\ S_1$
4alicebob^	$D_1\ L_8\ S_1$
login1##	$L_5\ D_1\ S_2$
qwert5!!	$L_5\ D_1\ S_2$

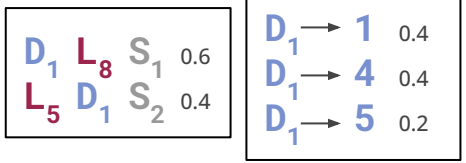
$D_1 \rightarrow 1$ } 0.4

$\rightarrow 4$ } 0.4

$\rightarrow 5$ } 0.2

1 2 3 4 5

Calculate probabilities



1password^	$D_1 L_8 S_1$
4wordpass$	$D_1 L_8 S_1$
4alicebob^	$D_1 L_8 S_1$
login1##	$L_5 D_1 S_2$
qwert5!!	$L_5 D_1 S_2$

$S_1 \rightarrow$ ^ } 0.66

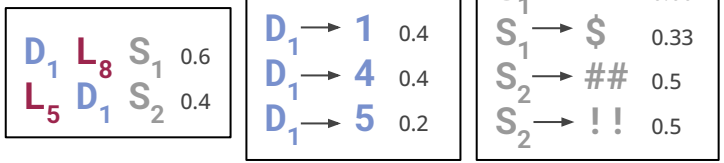
$S_1 \rightarrow$ $ } 0.33

$S_2 \rightarrow$ ## } 0.5

$S_2 \rightarrow$! ! } 0.5

1 2 3 4 5

Calculate probabilities



Construct a context-free grammar (CFG) with these!

$D_1\ L_8\ S_1$ 0.6

$L_5\ D_1\ S_2$ 0.4

$D_1 \rightarrow 1$ 0.4

$D_1 \rightarrow 4$ 0.4

$D_1 \rightarrow 5$ 0.2

$S_1 \rightarrow$ ^ 0.66

$S_1 \rightarrow$ $ 0.33

$S_2 \rightarrow$ ## 0.5

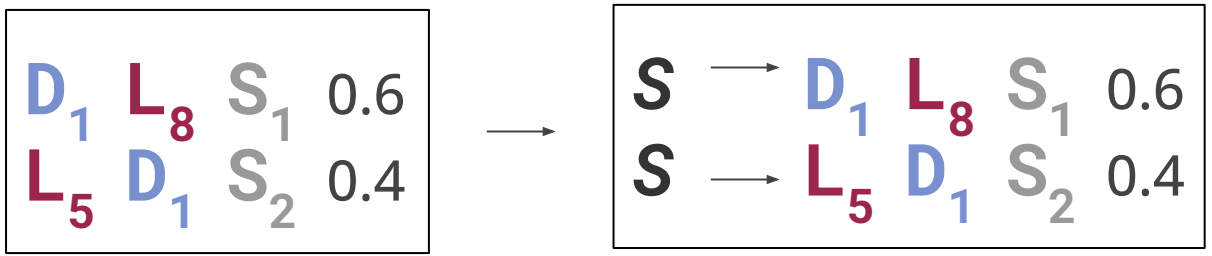
$S_2 \rightarrow$! ! 0.5

1 2 3 4 5

Derive pre-terminal structures

Context Free Grammars (CFG)

- A CFGs rules tell you how to derive a sentence of that CFG
- Expansion rules follow the form:
  α → β
- Probabilistic CFGs just have probabilities associated with rules

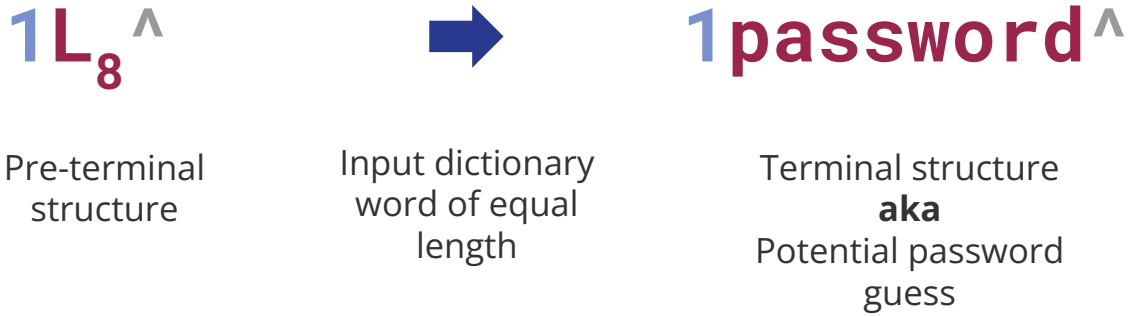


LHS	RHS	Probability	
S	D_1 L_8 S_1	0.6	*
S	L_5 D_1 S_2	0.4	
D_1	1	0.4	*
D_1	4	0.4	
D_1	5	0.2	
S_1	^	0.66	*
S_1	$	0.33	
S_2	##	0.5	
S_2	! !	0.5	

Pre-terminal structure: $1L_8$^

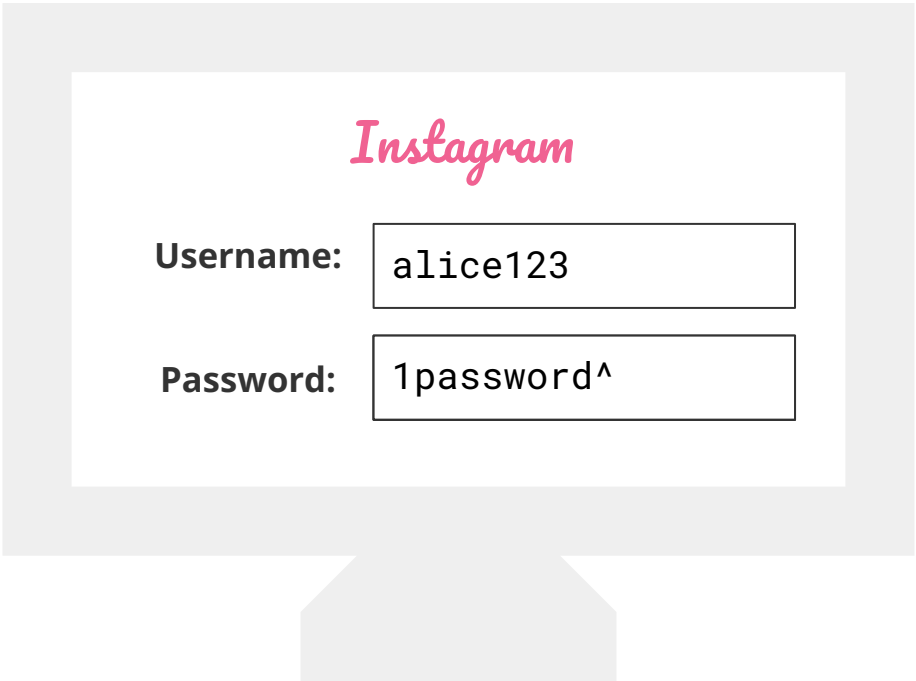
Probability: 0.6 * 0.4 * 0.4 = 0.096

Derive pre-terminal structures


1L8^
1password^
Pre-terminal structure
Input dictionary word of equal length
Terminal structure
aka
Potential password guess

```
1password^
1wordpass^
1alicebob^
1testtest^
```

Terminal structures



1 2 3 4 5

Attempt password guesses

John The Ripper

- A very popular open source password cracking tool
- Released in 1996
- Initially built for UNIX- based systems.
- Used to test password strength.
- Crack passwords using brute force and dictionary attack.
- Runs on all major operating systems
  Eg: Windows, MacOS, LINUX

Working Of John The Ripper

Makes use of three password cracking approaches to correctly guess the password :

- Dictionary Attacks
- Brute Force Attack
- Rainbow Tables

Brute Force Attack

- Passwords are stored as hashes, not in plain text.
- Leakage of database shouldn’t leak passwords
- Different hash methods
  Eg:
  User1: password123 -> 482c811da5d5b4bc6d497ffa98491e38
  John: john1991 -> 05ca5a76e59202112d04469fad75ab2d
- Constant trial and error of every possible combination of password
- Online and Offline brute force attack

Online and Offline Brute Force Attacks

- Online Brute Force Attack

1. Trying the login process
2. Possibility of accounts to lockout
3. Time consuming and slow process

- Offline Brute Force Attack

1. Done using list of password produced hashes
2. Random passwords are hashed and matched against the list
3. Large requirement of computational resources

Dictionary Attacks

- Using common words found in the dictionary to brute force
- Start with the most common passwords
  Eg: password, 123456, qwerty
- Targets weak passwords containing common words
- Passwords can also be a combination of these words, numbers and special characters

Rainbow Tables

User	Password
Stephen	auhsoJ
Lisa	hsifdrowS
James	1010NO1Z
Harry	sinocarD tupaC
Sarah	auhsoJ

User	Password Hash
Stephen	39e717cd3f5c4be78d97090c69f4e655
Lisa	f567c40623df407ba980bfad6dff5982
James	711f1f88006a48859616c3a5cbcc0377
Harry	fb74376102a049b9a7c5529784763c53
Sarah	39e717cd3f5c4be78d97090c69f4e655

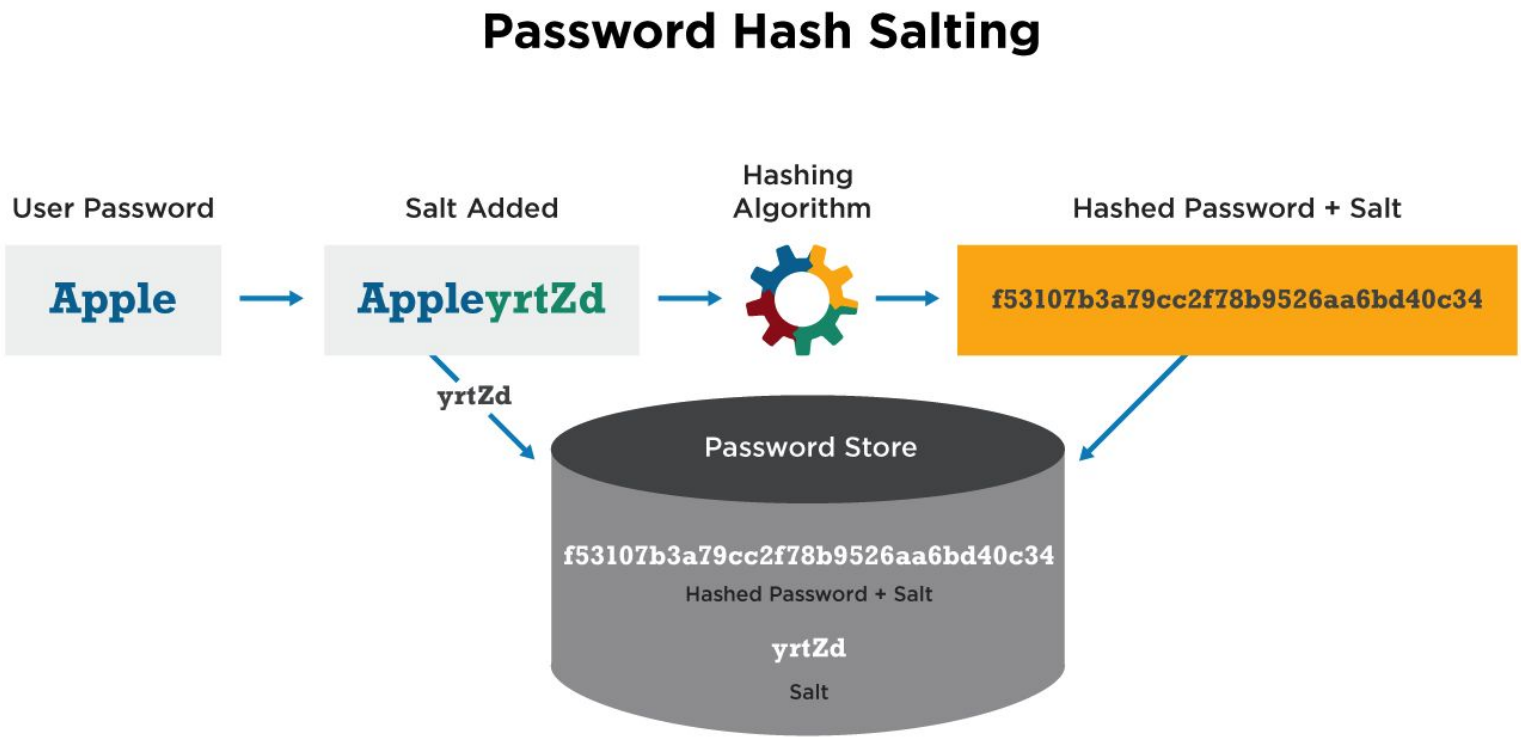
User	Random Salt	Password Hash
Stephen	06917d7ed65c466fa180a6fb62313ab9	b65578786e544b6da70c3a9856cdb750
Lisa	51f2e43105164729bb46e7f20091adf8	2964e639aa7d457c8ec0358756cbffd9
James	fea659115b7541479c1f956a59f7ad2f	dd9e4cd20f134dda87f6ac771c48616f
Harry	30ebf72072134f1bb40faa8949db6e85	204767673a8d4fa9a7542ebc3eceb3a2
Sarah	711f51082ea84d949f6e3efecf29f270	e3afb27d59a34782b6b4baa0c37e2958

Figure 1. Password and Hash Tables

Rainbow Tables

- Table consisting of all used passwords and their hashes
- Compare the hash with the value in the table to determine the password
- Increased speed in password cracking
- Tables would require large storage space
- Require different rainbow tables for different hashing process
  Eg: MD5, LANMAN, SHA-1, SHA -2

Salting


Password Hash Salting
User Password
Apple
Salt Added
AppleyrtZd
yrtZd
Hashing Algorithm
Hashed Password + Salt
f53107b3a79cc2f78b9526aa6bd40c34
Password Store
f53107b3a79cc2f78b9526aa6bd40c34
Hashed Password + Salt
yrtZd
Salt

How Websites store password?

- Passwords are first encrypted and then stored on database.
- Hashes like MD5, SHA2 are used.

Original Password	After Hashing(MD5crypt)
ronaldo189	`$1$6akX.Sto$1U8Lan06dV3P96eLJSKxF.`
passw0rd	`$1$6akX.Sto$G5edk/Z0dBwjxs4AYOHFC1`

Demonstration - John The Ripper

Safe Practices

- Create Long, Range, unique passphrase
- Implement multi-factor authentication
- Securely storing passwords

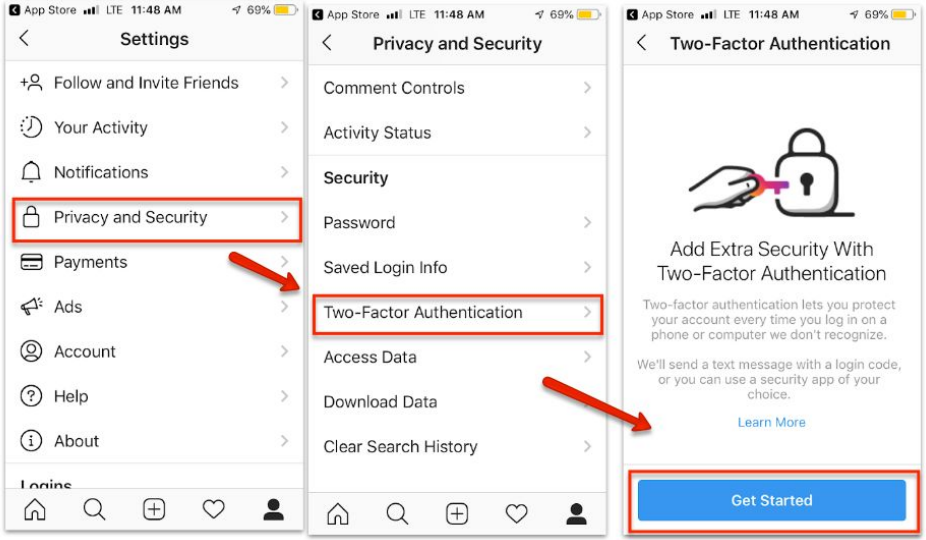
Create Long, Range and Unique Passphrase

- Don’t make passwords using dictionary words, use a combination of alphabets, numbers and symbols.
- Do not add details like birthday dates and pet names into passwords.

Implement Multi-Factor Authentication

- Authentication method that requires the user to provide two or more verification factors.
- Login requires more than a username and password.
- Used by banks, shopping sites, google.

Multi-factor Authentication



Securely Storing Passwords

- Use Salts and hashes algorithms
- Salting - password is appended to a password before hashing.
- Don’t use spam links as they can get your system hacked.

References

Weir, Matt, et al. "Password cracking using probabilistic context-free grammars." Security and Privacy, 2009 30th IEEE Symposium on. IEEE, 2009. http://ieeexplore.ieee.org/document/5207658/

the Guardian. 2022. *Passwords and hacking: the jargon of hashing, salting and SHA-2 explained*. [online] Available at: <https://www.theguardian.com/technology/2016/dec/15/passwords-hacking-hashing-salting-sha-2#:~:text=When%20a%20password%20has%20been,key%2C%20using%20a%20set%20algorithm.> [Accessed 12 May 2022].

Sharma, A. (2022). *John the Ripper explained: An essential password cracker for your hacker toolkit*. CSO Online. Retrieved 12 May 2022, from https://www.csoonline.com/article/3564153/john-the-ripper-explained-an-essential-password-cracker-for-your-hacker-toolkit.html.

Other Resources

More About CFGs