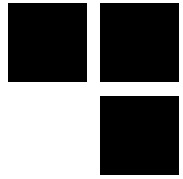
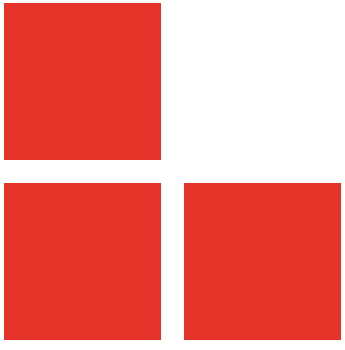
SYNACKTIV
DIGITAL SECURITY

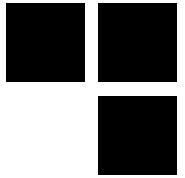


Password crackers

simon.marechal@synacktiv.com

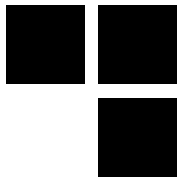
2023/2024





Password hashing 101

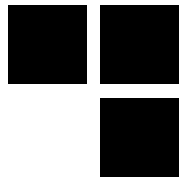
Level 0, plain text storage



`user:password`

- original, most obvious implementation
- hugely problematic
- still in use today :(

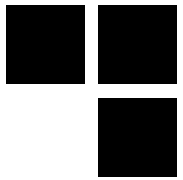
Level 1, cryptographic hash function



`user:8a9d093f14f8701df17732b2bb182c74`

- for example, windows NT hash uses MD4
- several problems:
  - identical passwords
  - speed
  - for Windows, the hash is the password :(
- hash the prompted password, compare it with the stored password

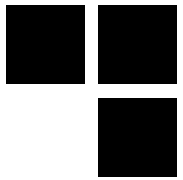
Level 2, salts



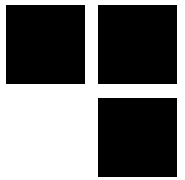
`user:$1$sHK38GAs$tSmLS3viggL8sHAouDXx2.`

- password is mixed with a, known value, hopefully distinct for all passwords
- get the user's salt, hash the prompted password with it, compare it with the stored password

Level 3, think about crackers

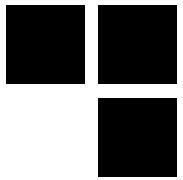


- specialized password storage functions
  - often very tunable
  - memory-hard, GPU-hard functions
- examples:
  - *scrypt*
  - *argon2*



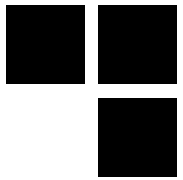
Password cracking 101

What's the idea?



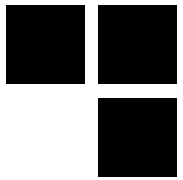
- pick a plausible plain text
- hash it as you would hash the password
- compare the output, if it is identical, you found the password (or a collision)

Techniques



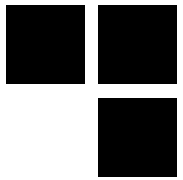
- the faster you hash, the more passwords you will crack
  - optimized implementation
  - vector instructions for parallel hashing
  - GPUs!
  - bitslicing
- you can also have better strategies for picking candidates

Brute force



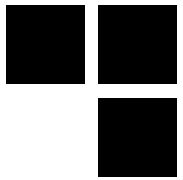
- try a, then b, then c, ...
- simple and efficient

Dictionaries



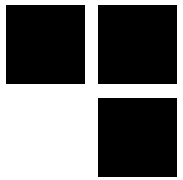
- most efficient password picking method
- mangling/mutation rules
  - `password -> p4ssw0rd123`
- might not be sufficient

Probabilistic approaches

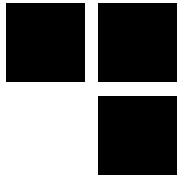


- passwords are not chosen randomly, so it might be possible to model them
- balance between generation speed, quality, parallelization

Other consideration

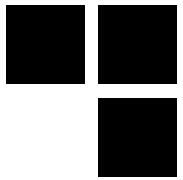


- password comparison must scale
- what about rainbow tables?



Task

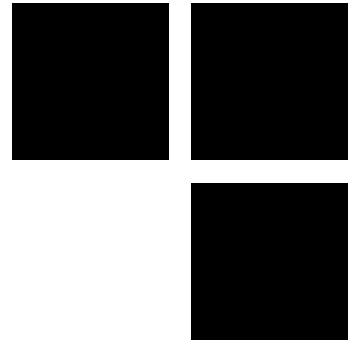
Simple password cracker



task: write a simple md4 hash cracker

- brute force password generator
- single target hash
- start with an unoptimized implementation
  - optimize it a bit
  - then write a vectorized version
  - then a GPU version

QUESTIONS?



Thank you for your attention

SYNACKTIV
DIGITAL SECURITY