

PRINCE

modern password guessing algorithm

Why do we need a new attack-mode?

FUTURE OF PASSWORD HASHES

Future of modern password hashes

Feature

- High iteration count
- Salted
- Memory-intensive
- Configurable parameters
- Anti-Parallelization
- ...

Effect

- Slow
- Rainbow-Tables resistance
- GPU resistance
- Slow
- Slow

Algorithms used for password hashing, by performance*

Name	Speed
NTLM, MD5, SHA1-512, Raw-Hashes	1 BH/s - 10 BH/s
Custom (Salt): VBull, IPB, MyBB	100 MH/s - 1 BH/s
DEScript	10 MH/s - 100 Mh/s
MD5crypt	1 MH - 10 MH/s
TrueCrypt, WPA/WPA2 (PBKDF2)	100kH/s - 1 MH/s
SHA512crypt, Bcrypt (Linux/Unix)	10kH/s - 100 kH/s
Custom (Iteration): Office, PDF, OSX	1kH/s - 10 kH/s
Script (RAM intensive): Android 4.4+ FDE	< 1 kH/s

* Performance oclHashcat v1.32
Single GPU
Default settings for configurable algorithms

Effects of modern password hashes

- Obsolete attack-modes:
 - Brute-Force-attack
 - Rainbow-Tables

So, what can the attacker do?

REMAINING ATTACK VECTORS

Remaining attack vectors

- Hardware (FPGA/ASIC)
- Extract keys from RAM
- Efficiency

Remaining attack vectors

- Hardware (FPGA/ASIC)
- Extract keys from RAM
- Efficiency
- Easier to cool
- Lower power consumption
- Easier to cluster
- Clustering only linear
- Expensive development
- Unflexible?

Remaining attack vectors

- Hardware (FPGA/ASIC)
- Extract keys from RAM
- Efficiency
- Highest chance of success
- Requires physical access to the System
- System must run

Remaining attack vectors

- Hardware (FPGA/ASIC)
- Extract keys from RAM
- Efficiency
- Exploit human weakness:
 - Psychology aspects
 - Password reuse
 - Pattern
- Limited keyspace
- Using rules:
 - Limited pattern
 - Takes time to develop

Features and advantages compared to previous attack modes

PRINCE ATTACK

Advantages over other Attack-Modes

- Simple to use, by design
- Smooth transition
- Makes use of unused optimizations:
 - Time works for attacker
 - Personal aspects

Advantages over other Attack-Modes

- Simple to use, by design
- Smooth transition
- Makes use of unused optimizations:
 - Time works for attacker
 - Personal aspects
- No monitoring required
- No extension required
- No syntax required

Advantages over other Attack-Modes

- Simple to use, by design
- Smooth transition
- Makes use of unused optimizations:
 - Time works for attacker
 - Personal aspects
- Primary goal of the algorithm
- Starts with highest efficiency
 - Wordlist
 - Hybrid
 - Keyboard walks / Passphrases
 - Brute-Force + Markov
- Not a scripted batch

Advantages over other Attack-Modes

- Simple to use, by design
- Smooth transition
- Makes use of unused optimizations:
 - Time works for attacker
 - Personal aspects
- Does not run out of (good) wordlists
 - Time-consuming monitoring
- Does not need ideas
 - Time-consuming extension

Advantages over other Attack-Modes

- Simple to use, by design
- Smooth transition
- Makes use of unused optimizations:
 - Time works for attacker
 - Personal aspects
- Personal Aspects
 - Religion
 - Political wing
 - Red car
- Not hobbies, friends, dates, ...
 - Already covered with Wordlist-Attack
 - Common knowledge not to use them

Algorithm details

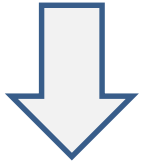
PRINCE ATTACK

PRINCE-attack

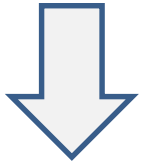
- PRobability
- INfinite
- Chained
- Elements

Attack basic components

- Element



- Chain



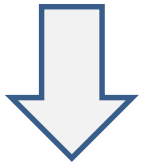
- Keyspace

Attack basic components

- Element



- Chain



- Keyspace

- Smallest entity
- An unmodified line (word) of your wordlist
- No splitting / modification of the line
- Sorted by their length into element database

Element example

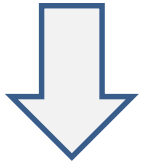
- 123456  • Table: 6
- password  • Table: 8
- 1  • Table: 1
- qwerty  • Table: 6
- ... • ...

Attack basic components

- Element



- Chain



- Keyspace

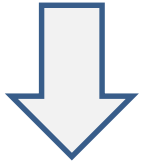
- Sum of all elements lengths in a chain = chain output length
- Fixed output length
- Best view on this is in reverse order, eg. a chain of length 8 can not hold an element of length 9

Chains example, general

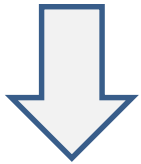
- Chains of output length 8 consists of the elements
 - 8
 - 2 + 6
 - 3 + 5
 - ...
 - 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1
- Number of chains per length = $2^{(\text{length} - 1)}$

Attack basic components

- Element



- Chain



- Keyspace

- Number of candidates that is getting produced, per chain
- Different for each chain
- The product of the count of the elements which build the chain


Element example (rockyou)

- length 1: 45
- length 2: 335
- length 3: 2461
- length 4: 17899
- ...

Keyspaces of chains of length 4 (rockyou)

Chain	Elements	Keyspace
4	17,899	17,899
1 + 1 + 1 + 1	45 * 45 * 45 * 45	4,100,625
1 + 1 + 2	45 * 45 * 335	678,375
1 + 2 + 1	45 * 335 * 45	678,375
1 + 3	45 * 335	15,075
2 + 1 + 1	335 * 45 * 45	678,375
2 + 2	335 * 335	112,225
3 + 1	335 * 45	15,075

Keyspaces of chains of length 4 (rockyou)

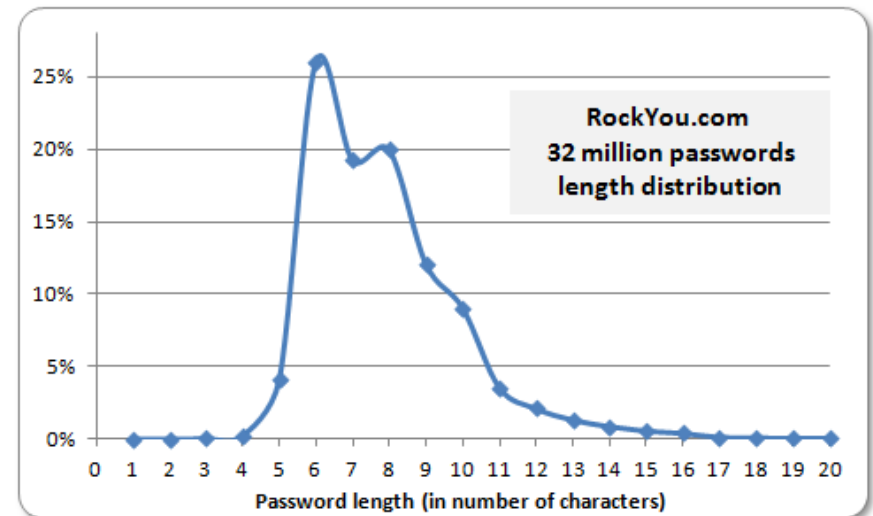
Chain	Elements	Keyspace 
3 + 1	335 * 45	15,075
1 + 3	45 * 335	15,075
4	17,899	17,899
2 + 2	335 * 335	112,225
2 + 1 + 1	335 * 45 * 45	678,375
1 + 2 + 1	45 * 335 * 45	678,375
1 + 1 + 2	45 * 45 * 335	678,375
1 + 1 + 1 + 1	45 * 45 * 45 * 45	4,100,625

Keyspace selection, general

- Sorting by lowest keyspaces creates the floating effect inside the prince attack-mode:
 - Wordlist
 - Hybrid
 - Keyboard walks / Passphrases
 - Brute-Force + Markov

Candidate output length selection

- The Algorithm has to chose the order of the output length for candidates
- Word-length distribution in a wordlist is a known structure →
- The algorithm recreates its own stats from the input wordlist



<http://blog.erratasec.com/>

Personal aspects

- To make use of this feature, you need a specific wordlist
 - Use a tool like wordhound to compile such a wordlist (grabs data from URL, twitter, reddit, etc)
- Cookbook phase:
 - Decide yourself if you want to use the raw list or
 - Preprocess the wordlist with some rules applied
 - Mix in like top 10k from rockyou
 - Mix in some single chars for late BF

Problems of the attack

- Elements in the wordlist requires all lengths
- Chain-count for long outputs
- Generated dupes

Problems of the attack

- Elements in the wordlist requires all lengths
- Chain-count for long outputs
- Generated dupes
- For calculation length distribution

Problems of the attack

- Elements in the wordlist requires all lengths
- Chain-count for long outputs
- Generated dupes
- Can be suppressed with divisor parameter

Problems of the attack

- Elements in the wordlist requires all lengths
- Chain-count for long outputs
- **Generated dupes**

Princeprocessor internal

- Load words from wordlist
- Store words in memory
- Generate element chains for each password length
 - Reject chains that does include an element which points to a non-existing password length
- Sort chained-elements by keySPACE of the chain
- Iterate through keySPACE (mainloop)
 - Select the next chain of that password length
 - Generate password with chain
 - Print

Usage

PRINCE ATTACK

How to use it from users view

- Download princeprocessor
- Choose an input wordlist which could be:
 - One of your favourite wordlist (rockyou, etc...)
 - Target-specific optimized wordlist
- Pipe princeprocessor to your cracker
 - `./pp64 < wordlist.txt | ./oclHashcat hash.txt`

How to use it from users view

- Optionally
 - Choose password min / max length
 - Choose character classes to pass / filter
 - Choose start / stop range -> Distributed
 - Choose minimum element length
 - Choose output file, otherwise written to STDOUT

LIVE DEMO 1

- Wordlist
 - Top 100k of rockyou.txt
- Hashlist
 - Public leak „stratfor“, 822k raw MD5 hashes
- Preparation
 - Removing raw dictionary hits first

LIVE DEMO 2

- Wordlist
 - Generated by scraping stratfor site
- Hashlist
 - Public leak „stratfor“, 822k raw MD5 hashes
- Preparation
 - Removing raw dictionary hits first

Download from: <https://hashcat.net/tools/princeprocessor/>

- Linux
- Windows
- OSX

PRINCEPROCESSOR V0.10 RELEASE

Email: jens.steube@gmail.com

IRC: freenode #hashcat

THANKS! QUESTIONS?