Use of Probabilistic Context-Free Grammars in Password Cracking

Radim Janča

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno

ijanca@fit.vutbr.cz

Presentation Overview

- Motivation
- Password cracking
- Probabilistic Context-Free Grammars
- PCFG in Password Cracking
- Conclusion

Motivation

- Human-memorable passwords remain a common form of access control to data and computational resources.
- Legitimate restoration of forgotten/lost password
- Illegal attack on legitimate systems
- If the most efficient attack is indeed publicly known, then at least legitimate system operators will not underestimate the risk of password compromise.
- Systems that allow users to choose their own passwords are typically vulnerable to space-reduction attacks that can break passwords considerably more easily than through a brute-force attack

Password cracking

- Attacker/administrator has access to password hashes
  - Brute force attack using rainbow tables (precomputed hashes)
  - Dictionary attack
  - Dictionary attack + word mangling rules
  - Brute force attack
- Attacker/administrator has access to salted password hashes $hash(salt + password)$
  - Dictionary attack
  - Dictionary attack + word mangling rules
  - Brute force attack

Dictionary attack with word mangling rules

- Users typically don't use unmodified elements from dictionaries (password policies).
- Users typically modify words to be recalled eassily with some word mangling rules.
  - adding symbols/digits to words
  - combining words
  - ...
- Ideally we would like to get sorted set of passwords ordered from the highest probability to the lowest.
- How to decide which rules are most probable?

Dictionary attack with word mangling rules

- Application of wordmangling rule on dictionary words multiplies the number of possible passwords.
- Combining multiple word mangling rules results in exponential growth of final database.
- Choosing the word order and word-mangling rule is crucial.
- Learning the probability of rules from real world passwords.
- Information can be modeled with probabilistic context free grammar (PCFG).

Probabilistic Context-Free Grammars

- Probabilistic Context-Free Grammars G is a quintuple:

$$G = (N, T, R, S, P)$$

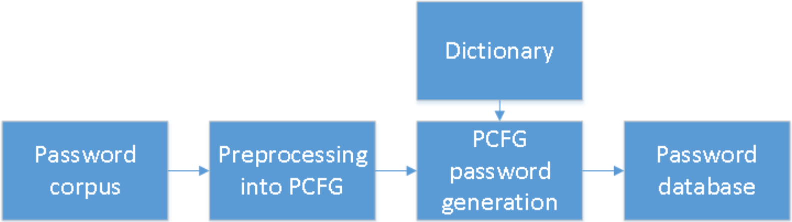
- N - finite set of nonterminal symbols
- T - finite set of terminal symbols
- R - finite set of production rules of the form:

$$A \to x$$

  where $A \in N$ and $x \in (N \cup T)^*$
- S - start symbol, $S \in N$
- P - set of probabilities p on production rules, where for each $A \in N$ and all rules $(A \to x) \in R$:

$$\sum p(A \to x) = 1$$

Generation of password database



- Password corpus - collection of passwords, typically leaked database of passwords
- Preprocessing - transformation from passwords corpus into PCFG
- Password generation from PCFG and chosen dictionary
- Password database - list of generated password sorted with descending probability of its occurrence

Preprocessing

- We define:
  L_n - alpha string
  D_n - digit string
  S_n - special string
- $L_n \in \{a, b, c, d, e, f \ldots, z\}^*$, $|L_n| = n$ and $n \in \mathbb{N}^+$
- $D_n \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^*$, $|D_n| = n$ and $n \in \mathbb{N}^+$
- $S_n \in \{!, @, \#, \$, \%, \&, \ldots\}^*$, $|S_n| = n$ and $n \in \mathbb{N}^+$

Preprocessing

- For each password we derive its *base form* $\in \{L_n, D_n, S_n\}^*$.
- For example password !Pa$$word53 derives into $S_1 L_2 S2 L_4 D_2$.
- We compute frequency of occurence from password corpus (traning set) with respect to n for each
  - *base form*
  - *digit string* D_n
  - *special string* S_n
- Probability of L_n *alpha strings* is not learned from training set, since corpus of words possibly used by users is much larger.

PCFG construction

- We generate PCFG G
  $G = (N, T, P, S, R)$
  $N = \{L_n, D_n, S_n\} \cup \{S\}$ (n is based on training set)
  $T = \{a, b, c, ..., z\} \cup \{0, 1, 2, .., 9, \} \cup \{!, @, \#, \$, \%, \&, ...\}$
- Generation of production rules from starting symbol S to *base form*
- Generation of production rules from symbols L_n, D_n, S_n to terminal strings
- Production rules from L_n are separately from dictionary.

PCFG example

- Example of PCFG rules R and their probabilities P:

Rule	Probability
$S \rightarrow D_1 L_6 D_1$	0.8
$S \rightarrow S_1 L_6 D_1$	0.2
$D_1 \rightarrow 3$	0.5
$D_1 \rightarrow 7$	0.3
$D_1 \rightarrow 8$	0.2
$S_1 \rightarrow !$	0.8
$S_1 \rightarrow \$$	0.2
$L_6 \rightarrow ?$	?
...	

- In PCFGs probability p of generated terminal string is computed as sum of all probabilities of all rules used.

$$S \xrightarrow{0.3} S_1 L_6 D_1 \xrightarrow{0.8} !L_6 D_1 \xrightarrow{0.5} !L_6 3 \xrightarrow{0.1} !letter3$$

$!letter3$ is *terminal string* with assigned probability p

$$p(!letter3) = (0.3 * 0.8 * 0.5 * 0.1)$$

$$p(!letter3) = 0.012$$

Dictionary-derived rules

- Rules for L_n are created as follows:

$$L_n \rightarrow \mathit{dictionary\ word}, \text{ where } |\mathit{dictionary\ word}| = n$$

- Probabilities of these rules are not gathered from training dataset.
- Probabilities of these rules can be assigned in multiple ways:
  - *Pre-terminal probability order*
  - *Terminal probability order*
  - ...

Pre-terminal probability order

- *Pre-terminal probability order* - probability p of derived password is equal to the probability of the sentence containing only L_n nonterminal and terminal symbols.
- This can be viewed as assigning probability equal to 1 to all rules $L_n \rightarrow$ *dictionary word* rules.

$$S \xrightarrow{0.3} S_1 L_6 D_1 \xrightarrow{0.8} !L_6 D_1 \xrightarrow{0.5} !L_6 3 \xrightarrow{1} !letter3$$

$$p(!letter3) = 0.12$$

Terminal probability order

- *Terminal probability order* – probability p of derived password is based on how many dictionary words of length n are present in dictionary.

$$p(L_n \rightarrow dictionary\ \ word) = \frac{1}{|x|},$$

$$\text{where } x = \{i | i \in dictionary \text{and } |i| = n\}$$

- For example, if we have 10 words of length 6 in our dictionary, we would get:

$$S \xrightarrow{0.3} S_1 L_6 D_1 \xrightarrow{0.8} !L_6 D_1 \xrightarrow{0.5} !L_6 3 \xrightarrow{0.1} !letter3$$

$$p(!letter3) = 0.012$$

Password generation

- Passwords need to be generated with decreasing probability
- Generation of all possible passwords can be huge (TB)
- Online algorithm (we want to end when password is found)
- Priority queue

- Nonterminals in *base form* have index based on position from the left.
  example: index of L_6 in $D_1 L_6 D_1$ is 1
- For each *base form* we find *pre-terminal* form with highest probability
- These rows are put into *priority queue* based on probability with *pivot* set to 1

Base form	Pre-terminal	Probability	Pivot (index)
$D_1 L_6 D_1$	$3L_63$	0.175	0
$S_1 L_6 D_1$	$!L_63$	0.12	0

Password generation

- In the next step top entry of queue is popped
- Next *pre-terminal* structures are generated by substituting variables in the popped base structure by values with next highest probability
- Only one nonterminal is replaced to create each new candidate
- Only nonterminals with index equal or higher than *pivot*
- Index of nonterminal in *base form* is stored as *pivot*

Password generation

Initial state

Base form	Pre-terminal	Probability	Pivot (index)
$D_1 L_6 D_1$	$3L_6 3$	0.175	0
$S_1 L_6 D_1$	$!L_6 3$	0.12	0

State after top row of queue is popped

Base form	Pre-terminal	Probability	Pivot (index)
$S_1 L_6 D_1$	$!L_6 3$	0.12	0
$D_1 L_6 D_1$	$7L_6 3$	0.105	0
$D_1 L_6 D_1$	$3L_6 7$	0.105	2

Conclusion

- PCFG can be used as viable option for improving dictionary attacks
- Proposed method can be targeted to specified field
- Method is can be updated to accurately map actual password practices
- Method can be further improved with addition of other type of word-mangling rules and strategies

References

- Weir, M.; Aggarwal, S.; de Medeiros, B.; Glodek, B., "Password Cracking Using Probabilistic Context-Free Grammars," Security and Privacy, 2009 30th IEEE Symposium on , vol., no., pp.391,405, 17-20 May 2009 doi: 10.1109/SP.2009.8
- Jelinek, Frederick, John D. Lafferty, and Robert L. Mercer. Basic methods of probabilistic context free grammars. Springer Berlin Heidelberg, 1992.

Thank you for your attention.