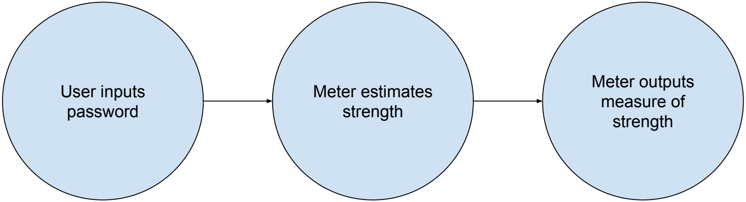
Password Strength Meters: Implementations and Effectiveness

Dalton Gusaas

Computer Science Senior Seminar
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

December 5, 2015

Password Strength Meters



- Goals:
  - Provide accurate strength measure
  - Help create strong, usable passwords
  - Improve security in a given system
- A password is strong if it is hard to guess

Outline

Shannon Entropy

- A method to determine the randomness of a variable based upon knowledge contained in the rest of the message
- Originally applied to a 27 character alphabet (a-z and space)
- Measured in bits

$$H(x) = -\sum_{i=0}^{n} P(x_i) log_2 P(x_i)$$

Example 1

Using the random string of characters *mom*:

$$\begin{aligned} H(x) &= -\sum_{i=0}^{n} P(x_i) log_2 P(x_i) \\ &= -[(0.33 * log_2 0.33) + (0.67 * log_2 0.67)] \\ &= -[(-0.53) + (-0.39)] \\ &= 0.92 \end{aligned}$$

The total entropy of *mom* is 2.76 bits

Example 2

Using the English word *mom*:

$$H(x) = -\sum_{i=0}^{n} P(x_i) log_2 P(x_i)$$
$$= -[(1 * log_2 1)]$$
$$= -[(1 * 0)]$$
$$= 0$$

The total entropy of *mom* is 0 bits.

Considerations

- Avoid overestimation of entropy (Example 1 vs Example 2)
- Need to know distribution of characters
- Entropy of English words and phrases is low due to patterns and non-uniform usage of letters

Guessing Entropy

- Guessing entropy is the approximate amount of work required to guess a given password
- Often used measure the strength of a password

Minimum Entropy

- Minimum entropy is the measurement of the amount of work needed to guess the easiest password in a system
- In an optimal attack the minimum entropy will usually be the amount of work needed to access a system

Attack Methods

Optimal strategy is to guess weaker, more probable passwords first and stronger, less probable passwords later

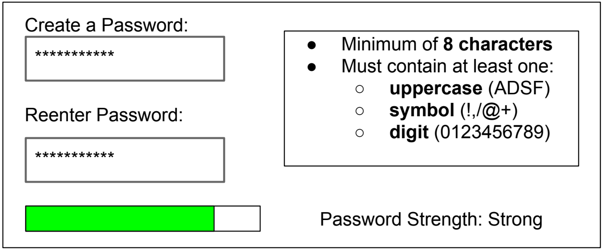
- Brute-force attack: guess every single character combination
- Dictionary attack: guess passwords based off of a collection of common passwords
- Probabilistic attack: combination of the above

RockYou Password Leak

- RockYou builds games for social media sites
- 32 million leaked passwords
- Passwords reflect different password creation policies
- Provided training data for attackers and defenders

Rule-Based Meters

- Usage of rules to shape user-created passwords
- Implemented by many websites and companies (Microsoft, Google, University of MN-Morris, etc.)
- Based on the NIST SP-800-63-1



NIST SP-800-63-1

- Authentication guideline for public and private organizations
- Shannon entropy used as a rough estimate of guessing entropy
- Translated Shannon's work on a 27 character language to the 95 ASCII printable characters

NIST Entropy

Daltong!u

Using the above as an example password...

NIST Entropy

Daltong!u

- 4 bits for *D*

The entropy of the first character is 4 bits.

NIST Entropy

Daltong!u

- 4 bits for *D*
- 14 bits for *altong!*

The entropy of the next 7 characters is 2 bits per character.

NIST Entropy

Dalton<u>g!u</u>

- 4 bits for *D*
- 14 bits for *altong!*
- 1.5 bits for *u*

The 9th through the 20th character have 1.5 bits per character

NIST Entropy

Dalton<u>g</u>!u

- 4 bits for *D*
- 14 bits for *altong!*
- 1.5 bits for *u*
- 6 bits for *D* and *!*

A bonus of 6 bits of entropy is assigned for a composition rule that requires both upper case and non-alphabetic characters.

Levels of Security

- Even with rough estimate of entropy, limiting guesses would provide enough security
- Level 1: 1 in 1024 (or 0.097%)
- Level 2: 1 in 16,384 (or 0.0061%)

Levels of Security

$$\text{Level 1: Number of Allowed Guesses} = 2^{H(x)} * 2^{-10}$$

$$\text{Level 2: Number of Allowed Guesses} = 2^{H(x)} * 2^{-14}$$

Weaknesses

Shannon entropy $\neq$ guessing entropy

Value	7+ Chars	8+ Chars	9+ Chars	10+ Chars
NIST Entropy	16	18	19.5	21
Level 1 # of Guesses	64	256	724	2048
% Cracked Using Guesses allowed by Level 1	3.21%	6.04%	7.19%	7.12%
Acceptable Level 1 Failure Rate	0.097%	0.097%	0.097%	0.097%
Level 2 # of Guesses	4	16	45	128
% Cracked Using Guesses Allowed by Level 2	0.98%	2.19%	2.92%	2.63%
Acceptable Level 2 Failure Rate	0.0061%	0.0061%	0.0061%	0.0061%

Weaknesses

Does not take in to account user behavior

String: U=Uppercase, L=Lowercase	Probability
UUUUUUU	53.56%
ULLLLLL	35.69%
ULLLULL	1.05%
LLLLLLL	1.03%
ULLLLLU	0.90%
ULLULLL	0.85%
ULULULU	0.68%
LLLLLLU	0.62%
UULLLLL	0.61%
UUULLLL	0.59%

Weaknesses

Does not take in to account user behavior

Special Character	Probability
.	17.81%
_	14.72%
!	11.34%
-	10.25%
<space>	8.72%
@	7.19%
*	6.54%
#	3.92%
/	3.01%
&	1.84%

Summary

- Shannon entropy $\neq$ guessing entropy, as levels of security do not actually work
- Does not take into account the distribution of user-created passwords, which is non-uniform
- Rule-based meters are based on a flawed metric

Adaptive Password Strength Meter

- Adaptive Password Strength Meter (APSM)
- Uses *n*-grams models to measure password strength
- Trained on leaked passwords to provide accurate measure of password strength

N-gram

- An *n*-gram is a sequence of *n* consecutive items taken from a larger sequence
- Using the string *dalton*
  - The 1-grams are: *d*, *a*, *l*, *t*, *o*, *n*
  - The 2-grams are: *da*, *al*, *lt*, *to*, *on*
  - The 3-grams are: *dal*, *alt*, *lto*, *ton*,

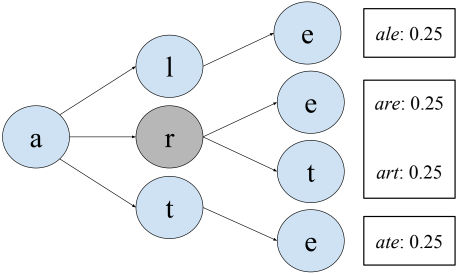
    ...
  - The 6-gram is: *dalton*

N-gram Model

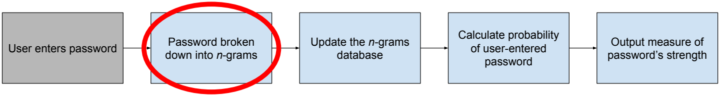
- An *n*-gram model is a sequence of *n*-grams that can be used to determine the probability of the next occurring *n*-gram based on the probability distribution

N-gram Model

- Language made up of the 1-grams: *a*, *e*, *l*, *t*, and *r*
- 1-grams can combine into 3-grams that are also English words
- What is the most probable 1-gram to follow *a*?



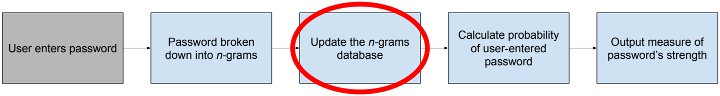
Password to *n*-grams



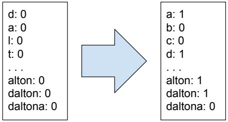
- This is done for each *n*-gram from 1 to the password's length
  - The 1-grams are: *d*, *a*, *l*, *t*, *o*, *n*
  - The 2-grams are: *da*, *al*, *lt*, *to*, *on*
  - The 3-grams are: *dal*, *alt*, *lto*, *ton*,

    ...
  - The 6-gram is: *dalton*

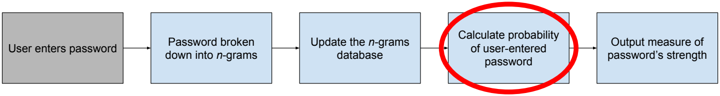
Update *N*-grams Database



- For each *n*-gram in a password the the corresonding count in the database is incremented

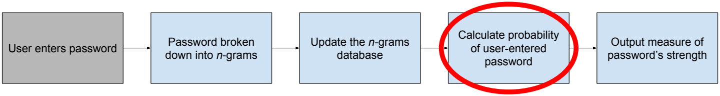


Calculating Password Probability



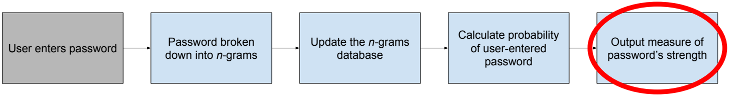
- The probability of a password is the product of the probabilities of its *n*-grams

Calculating Password Probability



- $P(password) = P(p)P(a|p)P(s|pa)P(s|pas)...P(d|passwor)$
- $P(s|pa) = \frac{count(pas)}{count(pa)}$
- Castelluccia et al. calculated $P(password) = 0.0016$, the actual frequency in the RockYou database was 0.0018

APSM Output



- Outputs a number from 0 to 1 based on the probability of the password

Analyzer Modifier for Passwords

- Analyzer Modifier for Passwords (AMP)
- Uses probabilistic context-free grammars (PCFGs) to estimate password strength

Context-Free Grammar

A context-free grammar is a 4-tuple (V, Σ, R, S) where:

- V is a finite set called the variables,
- Σ is a finite set, disjoint from V, called the terminals,
- R is a finite set of rules, with each rule being a variable and a string of variables and terminals
- $S \in V$ is the start variable.

Context-Free Grammar

The grammar, G, is:
$(\{A, B, C\}, \{x, y, z\}, \{\text{set of 3 rules}\}, S = A)$

A => xBz
B => A | xCz
C => y

Example derivation:
$A \Rightarrow xBz \Rightarrow xxCzz \Rightarrow xxyzz$

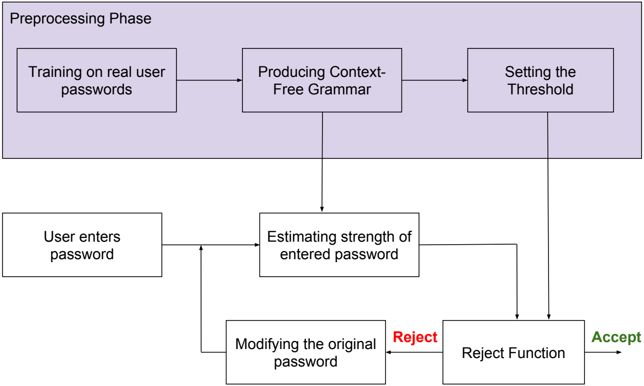
Probabilistic Context-Free Grammar

- A PCFG is the same as a CFG but with an added set P
- Add 100% chance that $B \Rightarrow A$
- Only derivations of form $x^n z^n$ can occur

Example derivation:

$A \Rightarrow xBz \Rightarrow xxAzz \Rightarrow xxxzzz$

How the AMP Works



Producing the PCFG

- The grammar is composed of components and base structures
- Components are the different types of characters
- Base structures are a collection of components
- Probabilities are based off of frequencies of components and base structures in training data

Producing the PCFG

Components:
L stands for lowercase alphabet letters
M stands for uppercase alphabet letters
D stands for digits
S stands for symbols

Base structures:
$Password123! \Rightarrow M_1 L_7 D_3 S_1$
$\#sRK!jB7ilZ \Rightarrow S_1 L_1 M_2 S_1 L_1 M_1 D_1 L_2 M_1$

Setting the Threshold

- The threshold value is a given probability based off of the guessing entropy
- Calculated using the PCFG, provides a conservative lower bound

Estimating Password Strength

- Like the APSM, the strength of a password is equal to its probability based of off training data
- $P(Dalton123!) = P(M_1 L_5 D_3 S_1) P(M_1) P(L_5) P(D_3) P(S_1)$
- If the probability is above the the threshold value the password is rejected

Distance Function

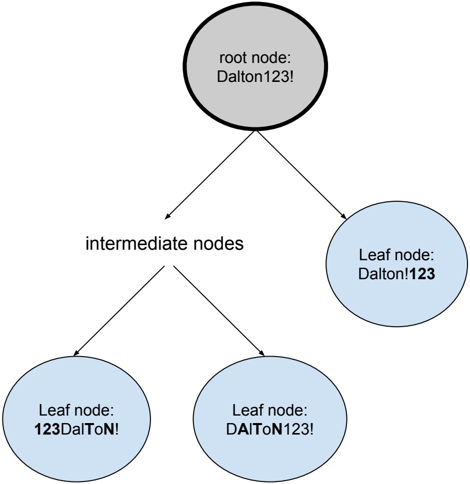
Operations on base structures:

- Insertion: $L_3S_9 = L_3D_3S_9$
- Deletion: $M_3S_9 = S_9$
- Transposition: $L_3D_3S_9 = D_3S_9L_3$

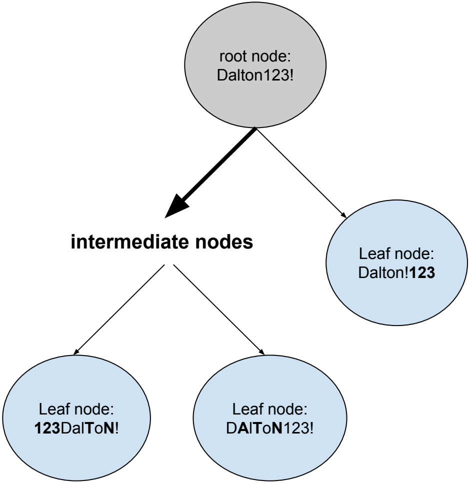
Operations on components:

- Insertion: *DJG* = *DJHG*
- Deletion: *#$%#* = *#%#*
- Substitution: *1111* = *1211*
- Case: *password* = *Password*

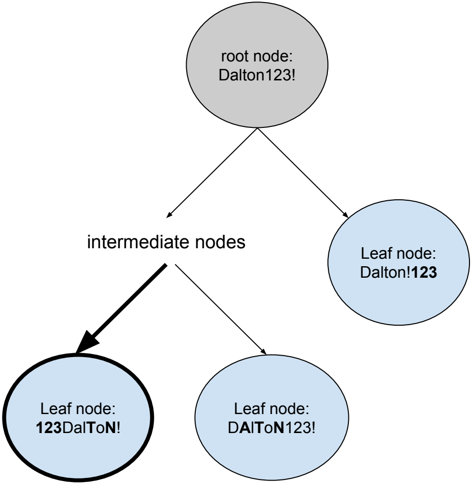
Modifying Passwords 1



Modifying Passwords 2



Modifying Passwords 3



Discussion

- Rule-based meters potentially give attackers advantage
- Rule-based meters fairly easy to implement
- Where is it ok to use rule-based meters?

AMP vs APSM

- Both evenly distribute passwords across available key-space
- Neither provide attackers with any significant advantage
- Fairly resistant to attack methods
- AMP modifies weak user-created passwords

Conclusion

- Of the three discussed meters, the AMP best accomplishes the goals of a password strength meter
- If password modification could be implemented by the APSM, it would be at about the same level as the AMP

Questions?