Adaptive Password Strength Meters From Markov Models

Claude Castelluccia, Markus Durmuth, Daniele Perito

Feb 7 2012

Password creation

- Randomly generated passwords
  - Your password is
    k$Hgw8*lp@
- User chosen passwords
  - Usually have low entropy

Better Solution:
Users choose password, reject weak passwords

Password	Occurence
	0.9%
	0.25%
	0.24%
_	0.19%
	0.16%
	0.10%
	0.07%
	0.07%
	0.06%
	0.05%

[Numbers from the RockYou password list]

Ad-hoc password checkers

Common password meters give scores based on:

- Use at least one non-alpha character
- Length
- Special chars

NIST checker:

- the entropy of the first character is taken to be 4 bits
- the entropy of the next 7 characters are 2 bits per character
- Bonus for capital or special, etc

Ad-hoc password checkers are too simple to gauge password strength

RockYou	w/ Policy
123456	abc123
12345	princess1
123456789	blink182
password	angel1
iloveyou	123abc
princess	iloveyou2
1234567	babygirl1
rockyou	iloveyou1
12345678	jesus1
abc123	monkey1

Ad-hoc password checkers: instance

- Usually ad-hoc checkers give a score in [0,4]
  - Example: Too weak (reject), weak, medium, strong, very strong
- Studied Google, Microsoft and NIST
- Example, Microsoft password checker
  - 'Very strong' score is given to passwords with
    - > 14 characters
    - span 3 character sets
    - Pass blacklist dictionary check
    - Not too close to a blacklisted word

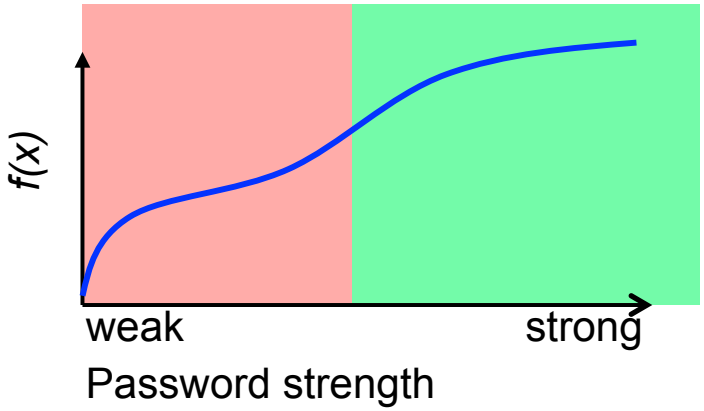
Is all this necessary?

Considerations

- Scoring strong passwords as weak affects usability
  - Random [a-z]{11} password ‘ynazwuaewfv’ is scored as weak by Microsoft password checker [1]
  - ‘ynazwuaewfv’ has 58 bits of entropy > DES key
- Scoring weak passwords as strong affects security
  - Microsoft rates P@ssw0rd as strong!

[1] microsoft.com/security/pc-security/password-checker.aspx

“Optimal” password checkers



$f(x) = -log(P(x))$

P(x) varies with sites

Solution:
Compute P(x) from the current password database

RockYou	MySpace	PhpBB	Singles.org
123456	password1	123456	123456
12345	abc123	password	jesus
123456789	password	phpbb	password
password	iloveyou1	qwerty	love
iloveyou	iloveyou2	12345	12345678
princess	fuckyou1	letmein	christ
1234567	myspace1	12345678	jesus1
rockyou	soccer1	1234	princess
12345678	iloveyou	test	blessed
abc123	iloveyou!	123	sunshine

Estimating Password Probabilities with Markov Models

- Markov models are used in speech recognition to estimate the probability of the next token in a sequence
- For example given ‘th’, what’s the next character?
  - ‘e’
  - ‘q’
- Estimation only taking into account the last k tokens, e.g. 2
- The Markov assumption allows us to learn those conditional probabilities from a suitable corpus
- Previous work has analyzed Markov models in relation to passwords [1][2]

[1] Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff. Arvind Narayanan, Vitaly Shmatikov

[2] Measuring Password Strength: An Empirical Analysis. Matteo Dell'Amico, Pietro Michiardi, Yves Roudier

Estimating Password Probabilities with Markov Models

- For passwords:
  - $P(c_1, \dots, c_n) = \prod_{i=k}^{n} P(c_i | c_{i-k+1}, \dots, c_{i-1})$
- Estimate the conditional probabilities from frequencies
  - $P(c_i | c_{i-k+1}, \dots, c_{i-1}) = \frac{count(c_{i-k+1}, \dots, c_{i-1}, c_i)}{count(c_{i-k+1}, \dots, c_i)}$
- For example
  - $P(w|pass) = \frac{count(passw)}{count(pass*)} = \frac{97963}{114218} = 0.86$

n-gram database

aaaaa	17988
aaaab	340
aaaac	303
...	
…	
passa	1129
passb	225
…	
passw	97963
…	
…	
zzzzz	0

We need to store the n-gram database!

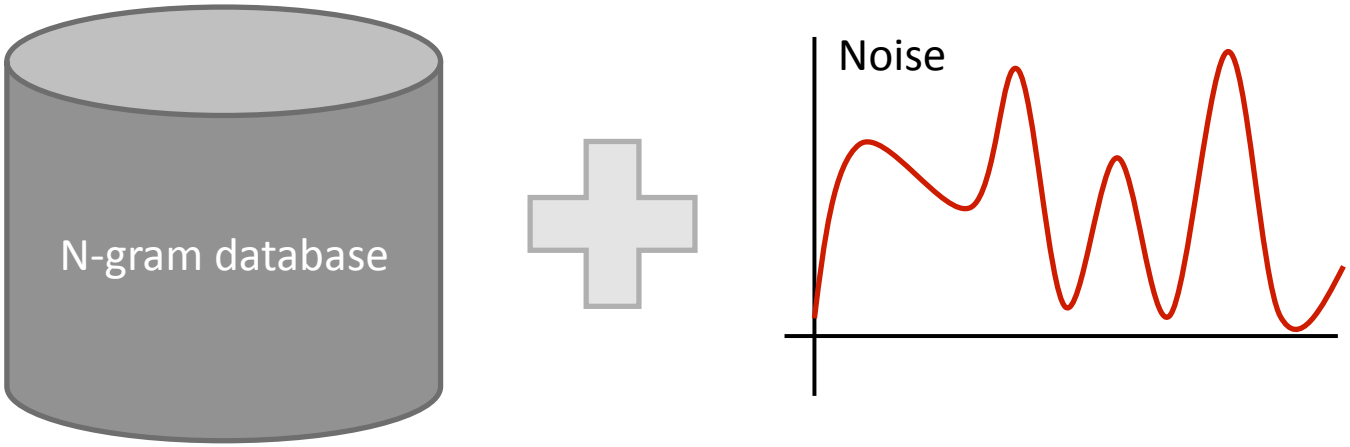
Security considerations

- In case of a breach the n-gram database will be revealed
- The attacker could use the n-gram database to reconstruct passwords
- Strong passwords are especially vulnerable
- Take password:
  - **k$Hgw8*lp@**
  - Each n-gram is likely to appear only once
  - The attacker could chain them together and reconstruct the password

n-gram database

gw8*l	1
…	
k$Hgw	1
...	
…	
Hgw8*	1
…	
w8*lp	1
…	
$Hgw8	1

Solution



Two questions:

- Security
  - Adding a carefully chosen amount of noise prevents leaking ‘too many’ bits (proof in the paper)
- Accuracy
  - How much does the strength estimation degrade when noise is added?

Dataset

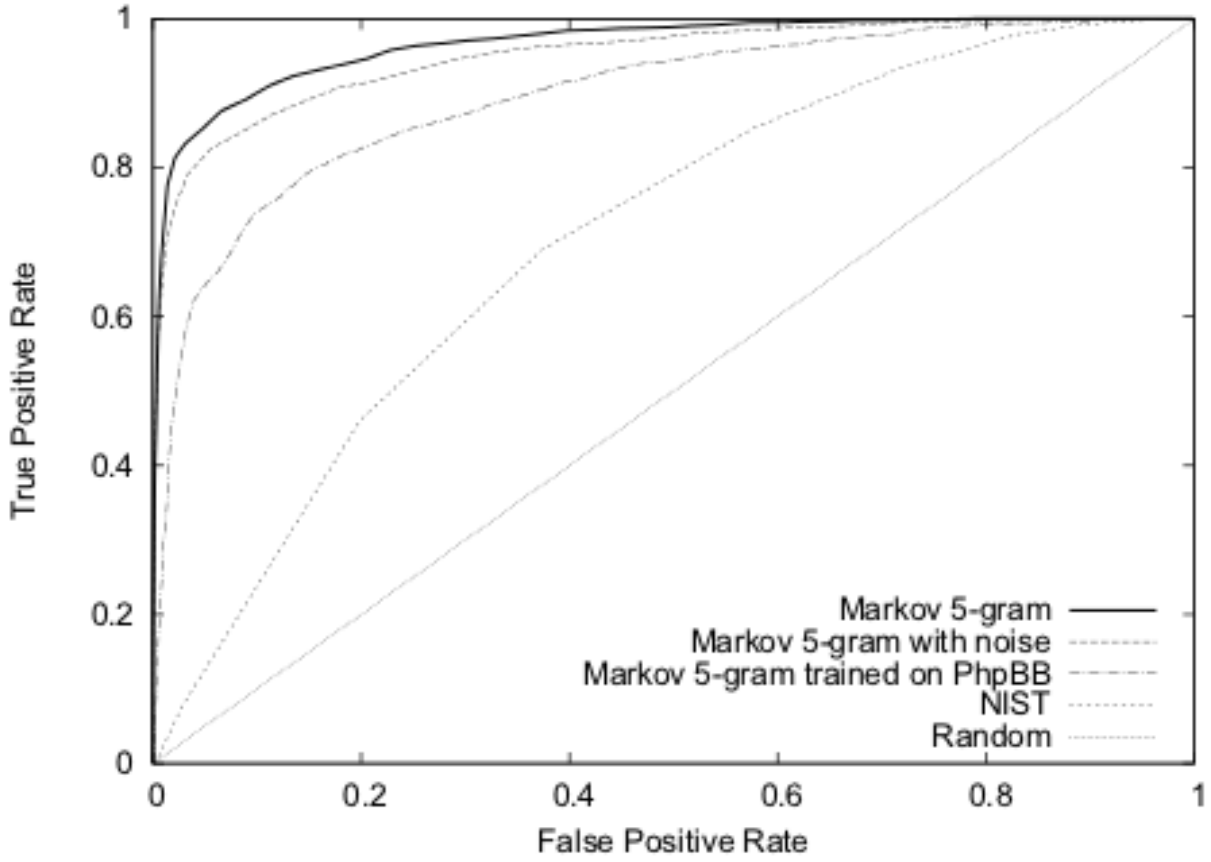
- **32.6 million** passwords leaked from RockYou in 2009
- **37 thousand** passwords leaked from MySpace
- **184 thousand** passwords leaked from PhpBB
- **8437** passwords leaked from religious website FaithWriters

Datasets were divided and used for training and testing

Experiment #1

- Find strong and week password in the RockYou dataset
  - Threshold probability $p = 2^{-20}$
  - Build ground truth (weak, strong labels) from empirical frequencies
- See how well we can classify strong and weak password
  - Measure precision and recall

Results


True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
Markov 5-gram
Markov 5-gram with noise
Markov 5-gram trained on PhpBB
NIST
Random

Experiment #2

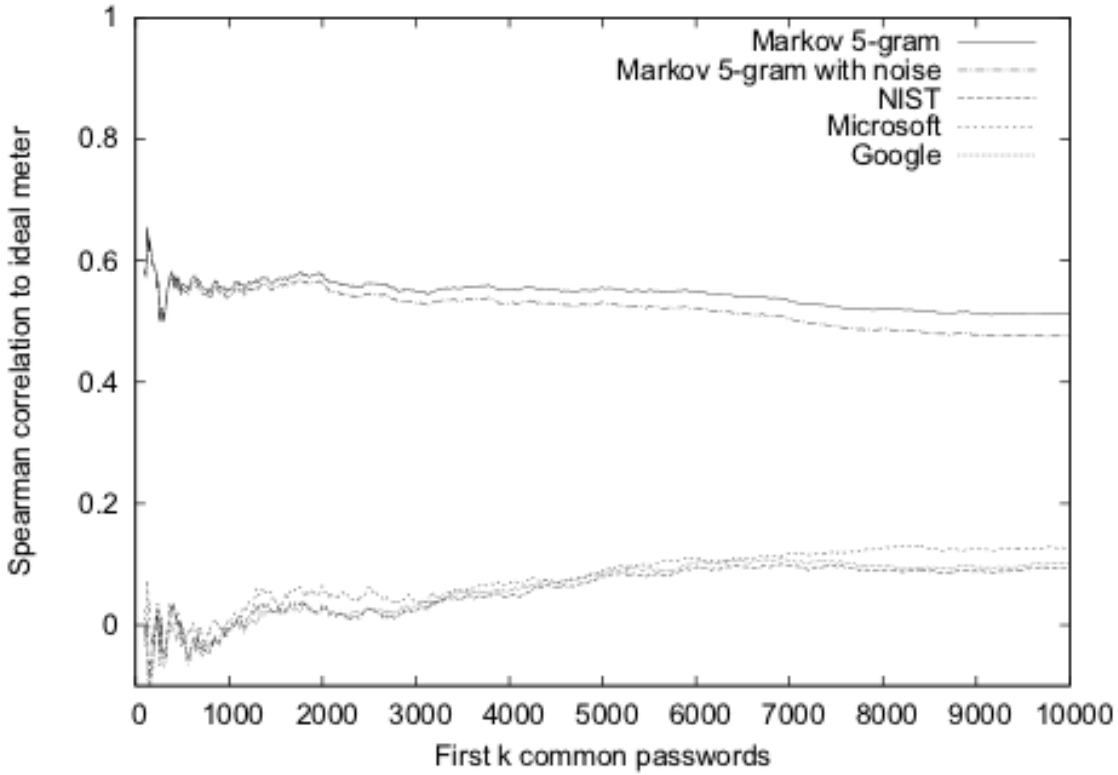
Note

- Measuring password strength is the complementary of finding the attack order
- The best order is from most frequent passwords to least frequent

Recipe

- Measure the frequency of the passwords in the RockYou password set
- See how well password checkers follow this ground truth order

Results



(a) Spearman correlation coefficient against the ideal password meter.

Conclusions

- Ad-hoc password checkers do not work
- Need adaptive password strength meters that use current password knowledge
- Password meters should be based on the best performing password crackers
  - We used Markov Models
  - Other approaches might be work as well

Thanks for your time

Questions?