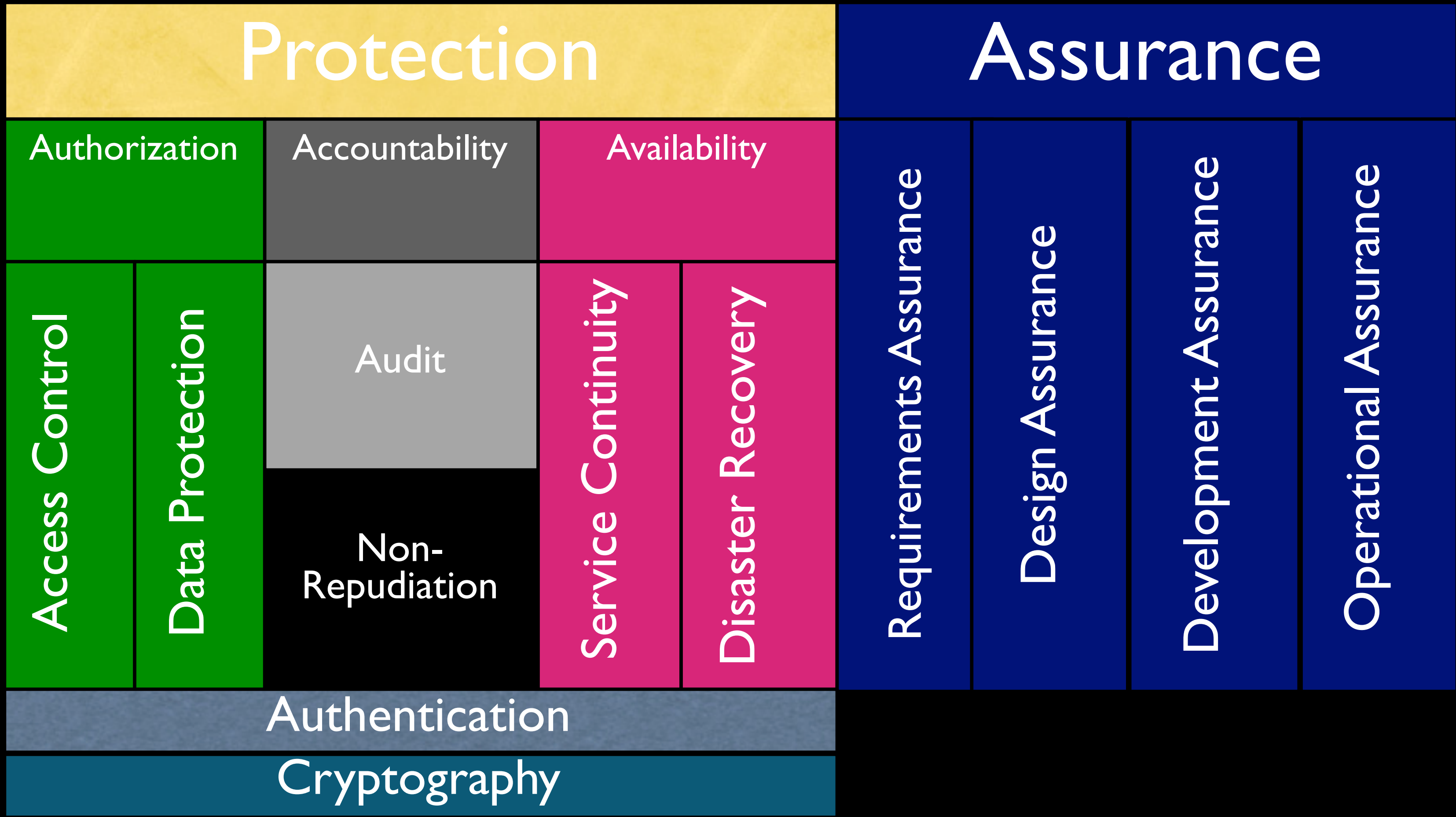
Authentication

where we are


Protection
Assurance
Authorization
Accountability
Availability
Access Control
Data Protection
Audit
Non-
Repudiation
Service Continuity
Disaster Recovery
Requirements Assurance
Design Assurance
Development Assurance
Operational Assurance
Authentication
Cryptography

What is Authentication?

- Real-world and computer world examples?
- What is a result of authentication?
- What are the means for in the digital world?

Basics and Terminology

definition

authentication is binding of identity to subject

- Identity is that of external entity
- Subject is computer entity
- Subject a.k.a. principal

What Authentication Factors are used?

- What you know
- What you have
- What you are

Password-based Authentication

What's Password?

- Lots of things act as passwords!
  - PIN
  - Social security number
  - Mother's maiden name
  - Date of birth
  - Name of your pet, etc.

illustration

- Monty Python and the Holy Grail (1h18m)

Keys vs Passwords

Crypto keys

- Suppose key is 64 bits
- Then 2^{64} keys
- Choose key at random
- Then attacker must try about 2^{63} keys

Passwords

- Suppose passwords are 8 characters, and 256 different characters
- Entropy is $\log_2(b^n)$
- Then $256^8 = 2^{64}$ pwds
- Users do not select passwords at random
- Attacker has far less than 2^{63} pwds to try (**dictionary attack**)

Why not Crypto Keys?

"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations.

(They are also large, expensive to maintain, difficult to manage, and they pollute the environment.

It is astonishing that these devices continue to be manufactured and deployed.

But they are sufficiently pervasive that we must design our protocols around their limitations.)"

Charlie Kaufman, Radia Perlman, Mike Speciner
in "Network Security: Private Communication in a Public World"

Why Passwords?

- Why is "something you know" more popular than "something you have" and "something you are"?
- **Cost**: passwords are free
- **Convenience**: easier for SA to reset password than to issue new smartcard

Good and Bad Passwords

bad passwords?

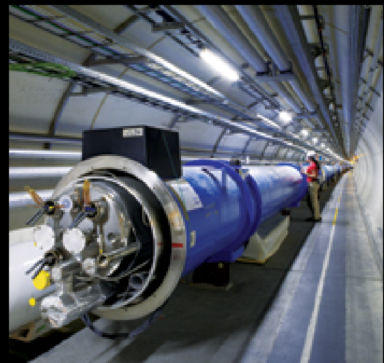
- frank
- Fido
- password
- 4444
- Pikachu
- 102560
- AustinStamp
- samfox

good passwords?

- jflej,43j-EmmL+y
- 09864376537263
- P0kem0N
- FSa7Yago
- 0nceuP0nAt1m8
- PokeGCTall150

European Organization for Nuclear Research (CERN)

source:ebaumsworld.com



source:ebaumsworld.com

source: gallery.hd.org

Samanta Fox

http://www.youtube.com/watch?v=-WrVrIZ4p4Q

300K leaked hotmail passwords

The top 10 passwords:

1. 123456
2. 123456789
3. alejandra
4. 111111
5. alberto
6. tequiero
7. alejandro
8. 12345678
9. 1234567
10. estrella

- The longest password was 30 chars long: lafaroleratropezooooooooooooo.
- The shortest password was 1 char long :)

source: http://www.acunetix.com/blog/websecuritynews/statistics-from-10000-leaked-hotmail-passwords/

Attacks on Passwords

- Attacker could…
  - Target one particular account
  - Target any account on system
  - Target any account on any system
  - Attempt denial of service (DoS) attack
- Common attack path
  - Outsider → normal user → administrator
  - May only require **one** weak password!

How to Store Passwords in the System?

- Store as cleartext
  - If password file compromised, all passwords revealed
- Encipher file
  - Need to have decipherment, encipherment keys in memory
- Store one-way hash of password

Password File

- Bad idea to store passwords in a file
- But need a way to verify passwords
- Cryptographic solution: hash the passwords
  - Store $y = \mathrm{hash}(\mathrm{password})$
  - Can verify entered password by hashing
  - If attacker obtains password file, he does not obtain passwords
  - But attacker with password file can guess x and check whether $y = \mathrm{hash}(x)$
  - If so, attacker has found password!

Dictionary Attack

- “online” or “offline”
- Attacker pre-computes hash(x) for all x in a dictionary of common passwords --- Rainbow Table
- Suppose attacker gets access to password file containing hashed passwords
  - Attacker only needs to compare hashes to his pre-computed dictionary
  - Same attack will work each time
- Can we prevent this attack? Or at least make attacker’s job more difficult?

Password File

- Store hashed passwords
- Better to hash with salt
- Given password, choose random s, compute

$$y = \text{hash}(\text{password}, s)$$

  and store the pair (s,y) in the password file

- Note: The salt s is not secret
- Easy to verify password
- Attacker must recompute dictionary hashes for each user — lots more work!

Assumptions for Password Cracking

- Passwords are 8 chars, 128 choices per character

  Then $128^8 = 2^{56}$ possible passwords

- Attacker has dictionary of 2^{20} common pwds
- Probability of 1/4 that a pwd is in dictionary
- Work is measured by number of hashes

Password Cracking

I. Finding single password without dictionary
- Must try $2^{56}/2 = 2^{55}$ on average
- Just like exhaustive key search

II. Finding single password with dictionary
- Expected work is about

$$1/4\,(2^{19}) + 3/4\,(2^{55}) = 2^{54.6}$$

- But in practice, try all in dictionary and quit if not found — work is at most 2^{20} and probability of success is $1/4$

III. password cracking without dictionary

- there is a password file with 2^{10} pwds
- goal: Find any of 1024 passwords in file

Without dictionary:

- assume all 2^{10} passwords are distinct
- need 2^{55} comparisons before expect to find password
- if no salt, each hash computation gives 2^{10} comparisons ⇒

  the expected work (number of hashes) is

  $2^{55}/2^{10} = 2^{45}$

- if salt is used, expected work is
  2^{55} since each comparison requires a new hash computation

IV. password cracking with a dictionary

- Find any of 1024 passwords in file
- With dictionary
  - Probability at least one password is in dictionary is

    $1 - (3/4)^{1024} = 1$
  - We ignore case where no password is in dictionary
  - If no salt, work is about

    $2^{19}/2^{10} = 2^9$
  - If salt, expected work is less than 2^{22}
  - Note: If no salt, we can precompute all dictionary hashes and amortize the work (Rainbow Tables)

Other Password Issues

- Too many passwords to remember
  - Results in password reuse
  - Why is this a problem?
- Failure to change default passwords
- Social engineering
- Error logs may contain “almost” passwords
- Bugs, keystroke logging, spyware, etc.

users and passwords

over 0.5 M passwords

- The average user has 6.5 passwords, each of which is shared across 3.9 different sites.
- Each user has about 25 accounts that require passwords, and types an average of 8 passwords per day.
- Users choose passwords with an average bitstrength 40.54 bits.
- The overwhelming majority of users choose passwords that contain lower case letters only (i.e., no uppercase, digits, or special characters) unless forced to do otherwise.
- 0.4% of users type passwords (on an annualized basis) at verified phishing sites.
- At least 1.5% of Yahoo users forget their passwords each month.

source: Florencio, D. and Herley, C. "**A large-scale study of web password habits**," In Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 657-666. DOI= http://doi.acm.org/10.1145/1242572.1242661

example


Multi Expresso
Bradesco
Tarifas de
Serviços Bancários


Bradesco
Tarifas de
Serviços Bancários
Pessoa Física

the camera transmits photos wirelessly up to 200 meters

the camera has its own battery and transmission antena

Shoulder surfing of ATM in Brazil

http://www.snopes.com/fraud/atm/atmcamera.asp

the bottom line

- Password cracking is too easy!
  - One weak password may break security
  - Users choose bad passwords
  - Social engineering attacks, etc.
- The bad guy has all of the advantages
- All of the math favors bad guys
- Passwords are a big security problem

how to improve password-based systems?

Against off-line password guessing

- Random selection
- Pronounceable passwords
    - przbqxdfl, zxrptglfn
    - helgoret, juttelon
- User selection of passwords
- Proactive password checking for "goodness"
- Password aging
- Against guessing many accounts
    - Salting

Against on-line password guessing

- (exponential) Back-off
- Disconnection
- Disabling
- Jailing

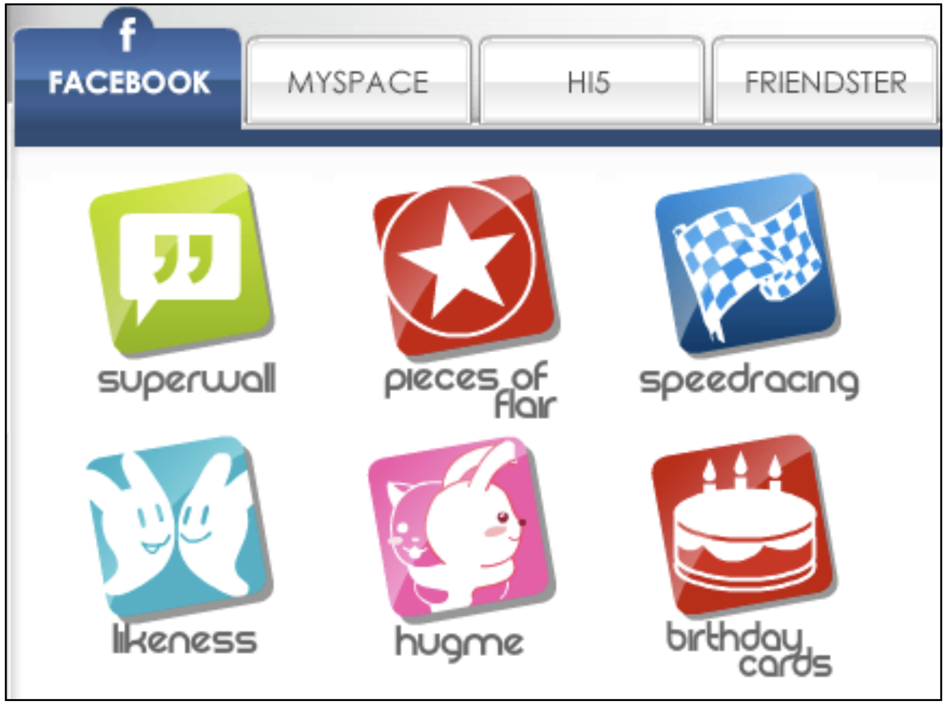
Case Study

From:
Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords

Matt Weir - Florida State University
Sudhir Aggarwal - Florida State University
Michael Collins - Redjack LLC
Henry Stern - Cisco Ironport Systems

Presented at Computer and Communications Security (CCS) Conference
October 2010

The RockYou List



- Provided widgets for most of the major social networking sites
- Hacked in November 2009
- Over 32 million plaintext passwords were released

The PhpBB List

- Development site for the popular phpbBB bulletin board software
- Hacked in January 2009
- Over 259k unsalted MD5 hashed passwords, and another 83k salted passwords

And Many Others:

Singles.org

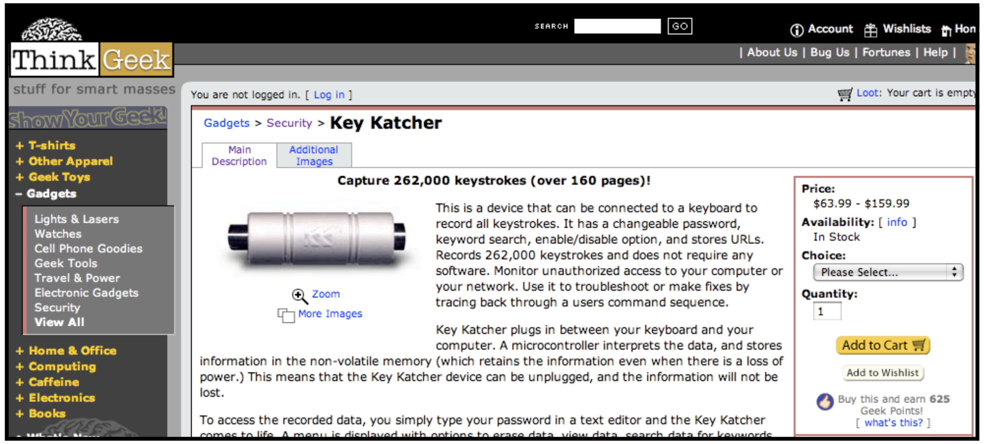
FaithWriters

NeoPets

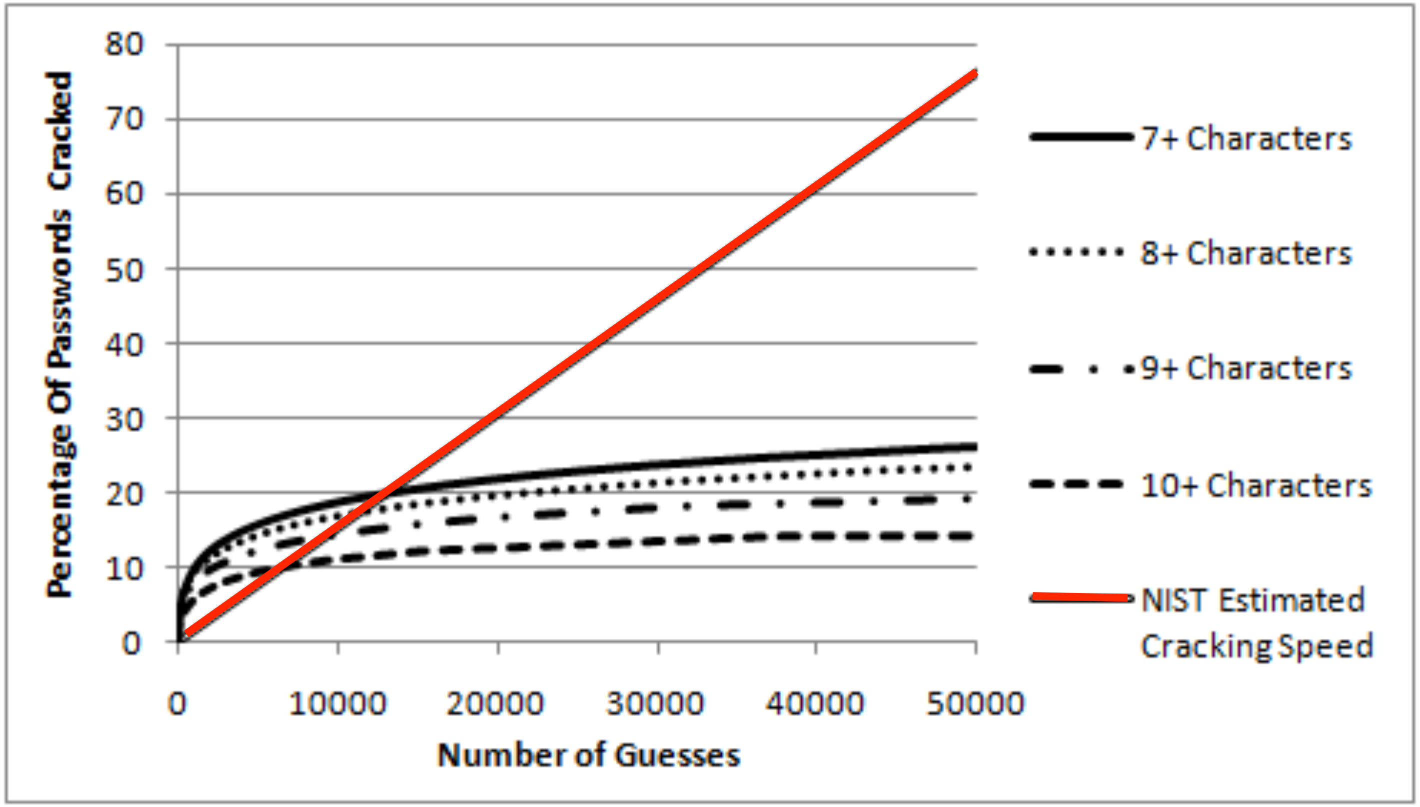
MySpace

Full Disclosure:

- Password strength rarely matters in an online attack
- More common attacks take advantage of:
    1. Password reuse
    2. Malware
    3. Phishing attacks



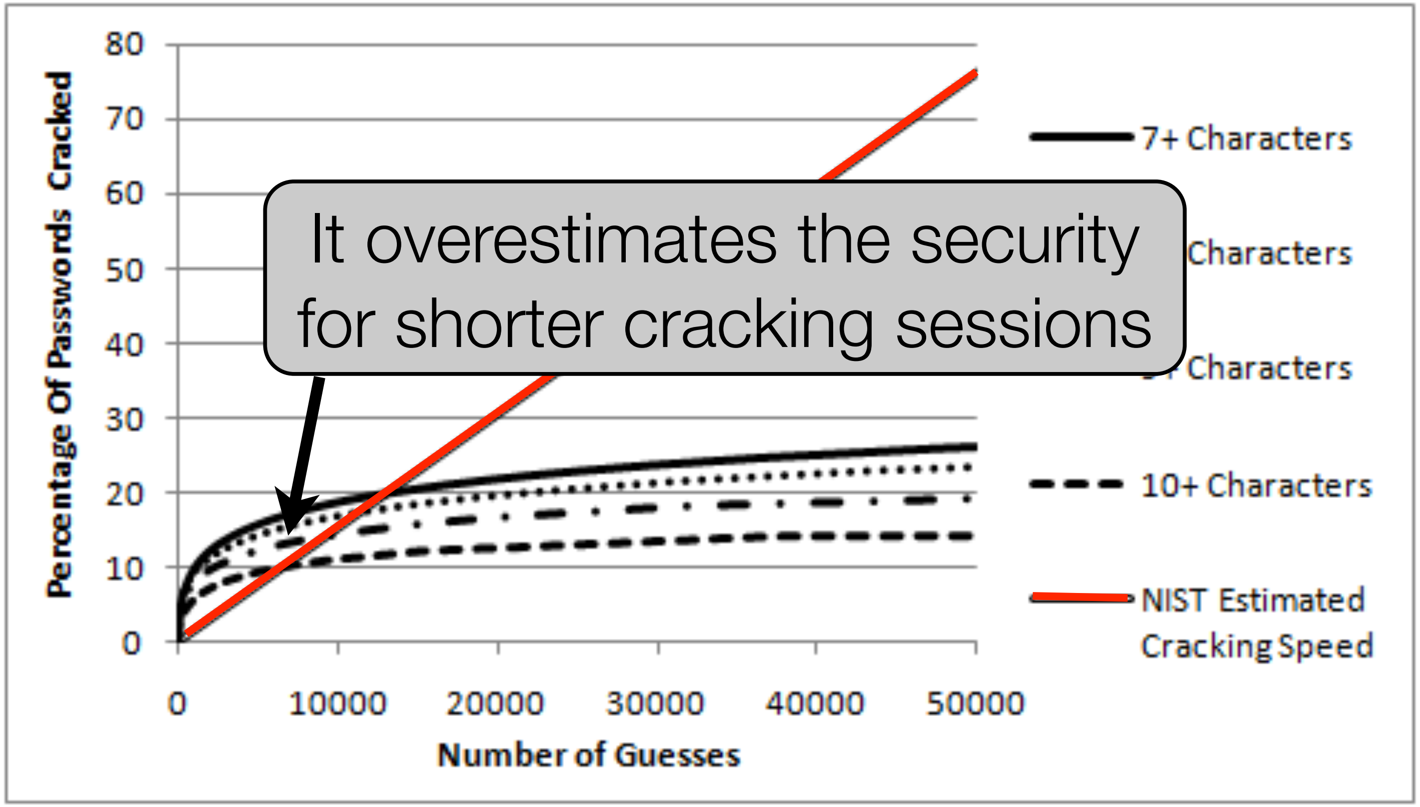
Where this Breaks Down



A 7 Character Min Length Policy's Entropy = 16

Chance_of_success(y) = Number_of_guesses(x)/2^{16}

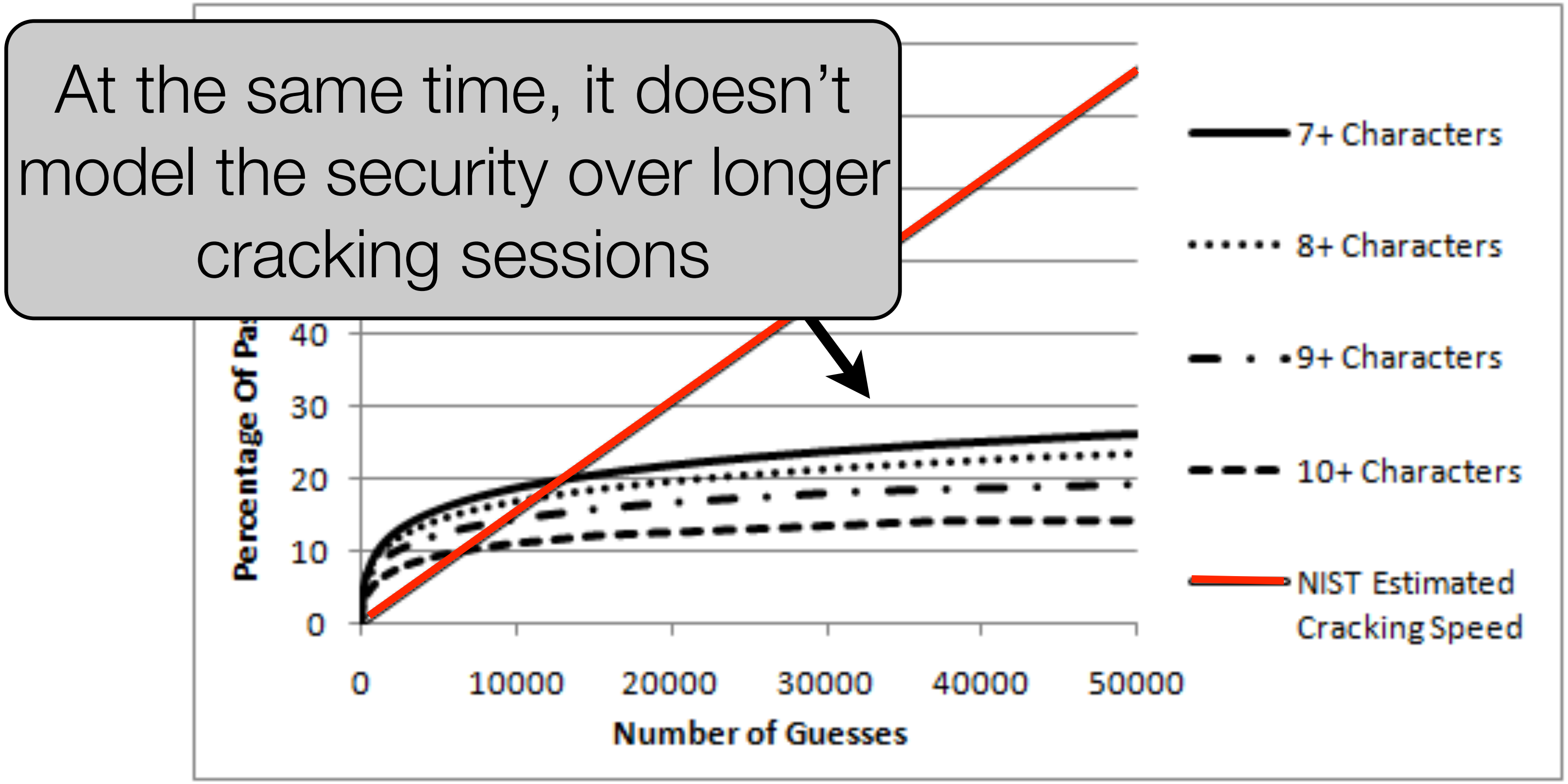
Where this Breaks Down



A 7 Character Min Length Policy's Entropy = 16

$$\textbf{Chance\_of\_success(y) = Number\_of\_guesses(x)/}2^{16}$$

Where this Breaks Down



A 7 Character Min Length Policy's Entropy = 2^{16}

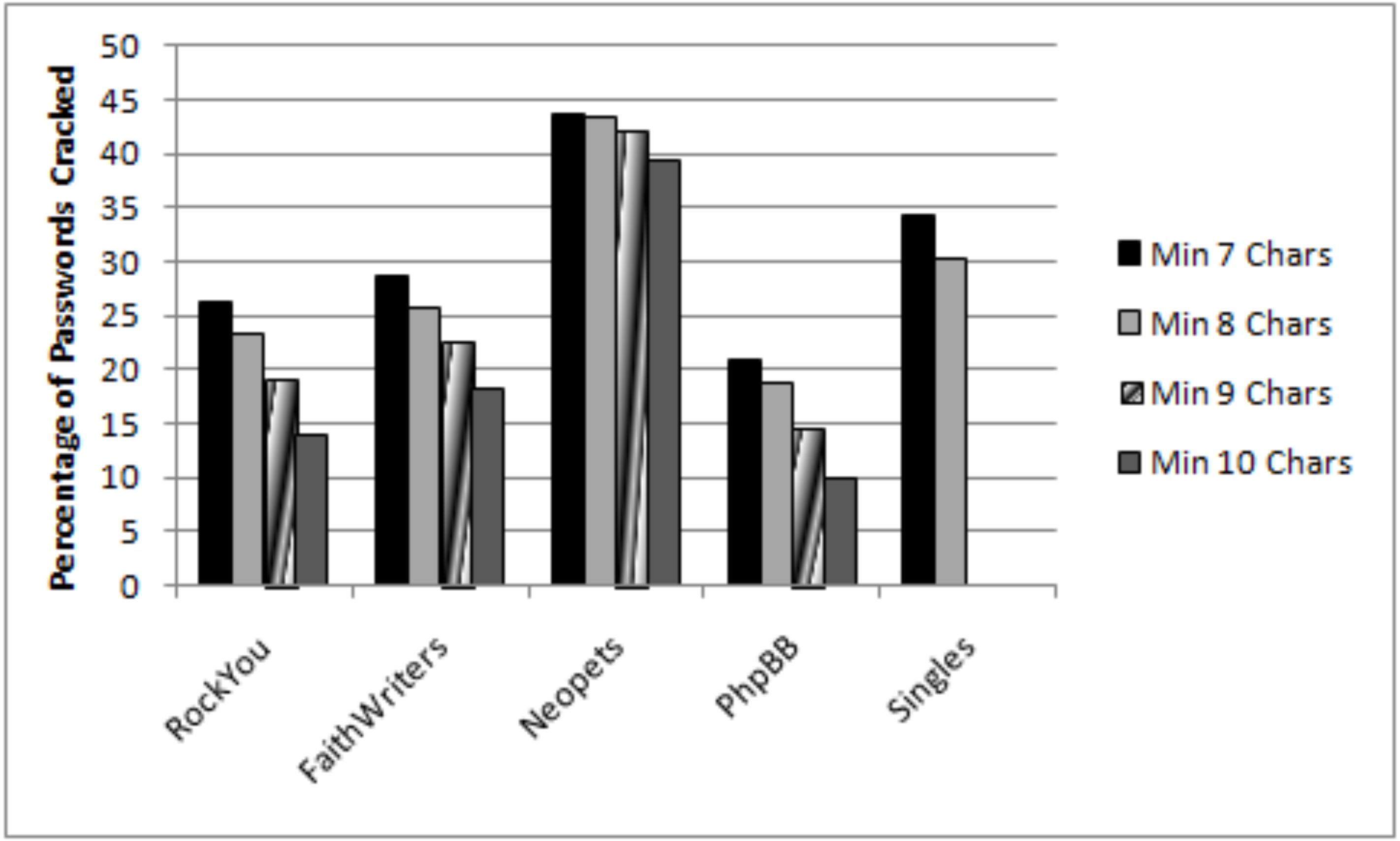
Chance_of_success(y) = Number_of_guesses(x)/2^{16}

What this all Means:

Shannon Entropy != Guessing Entropy

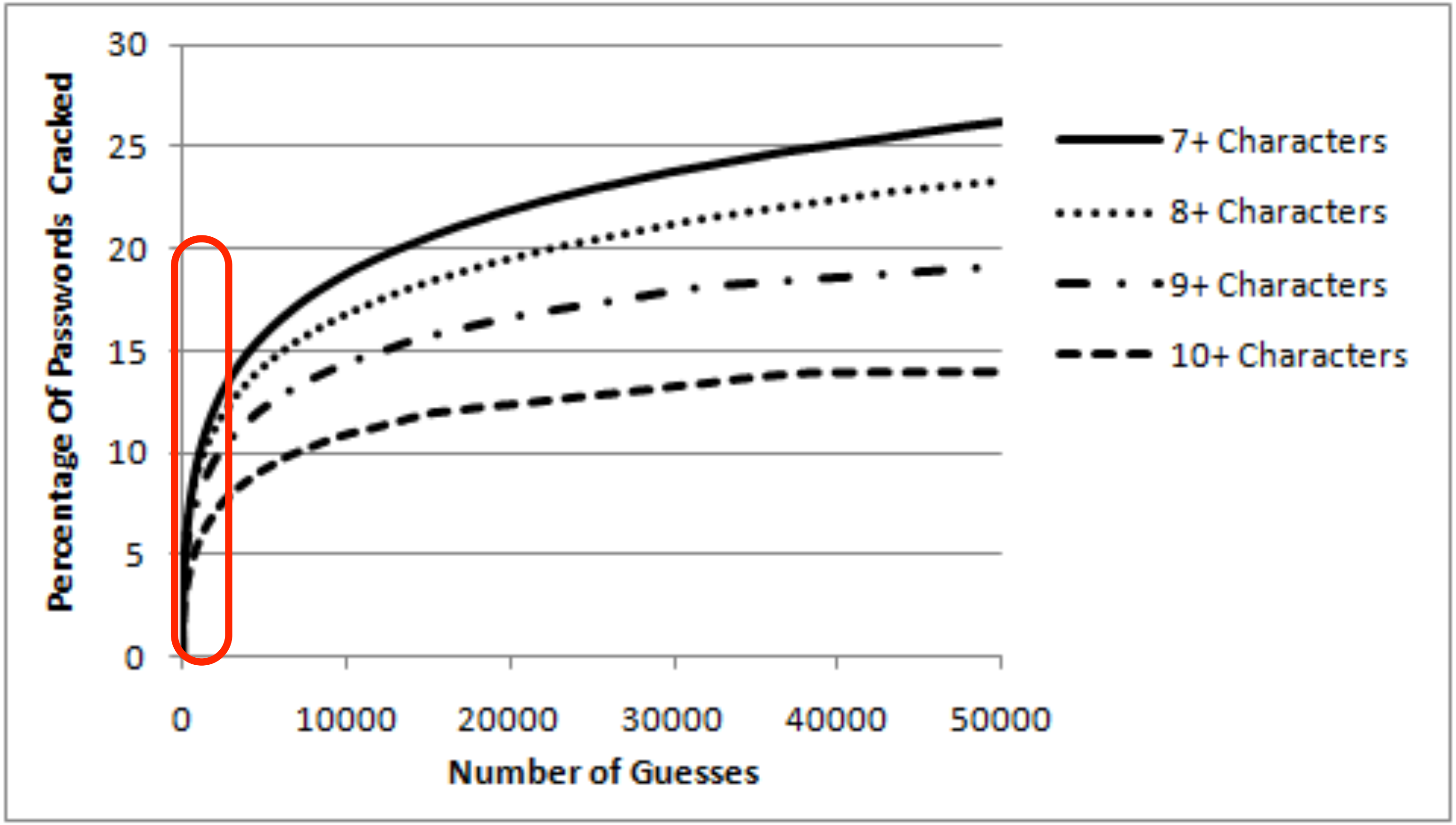
Password entropy as defined in NIST 800-63 is not a useful measurement for the defender

The Effect of Minimum Password Length:

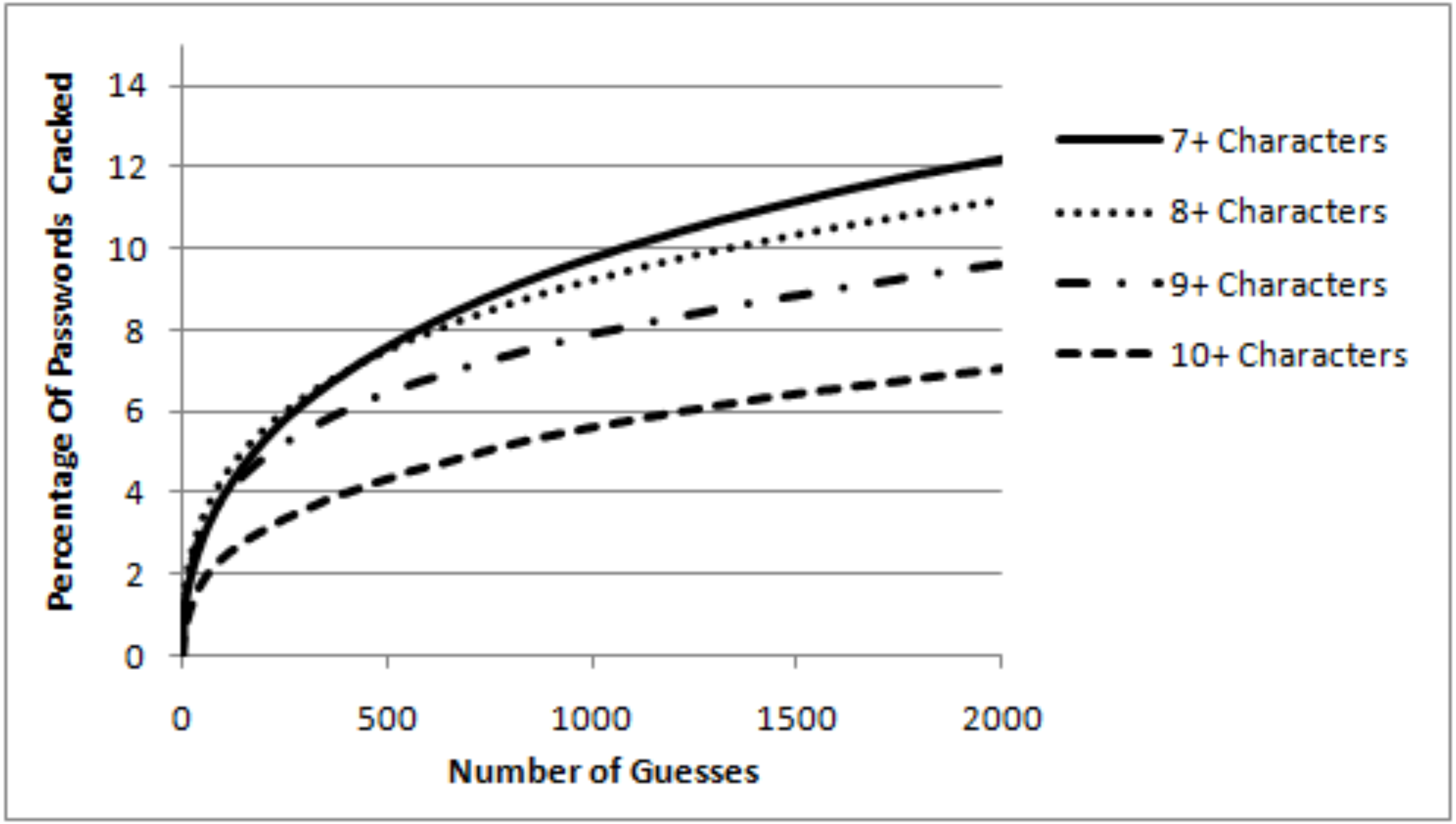


Password Cracking Session of 50k Guesses

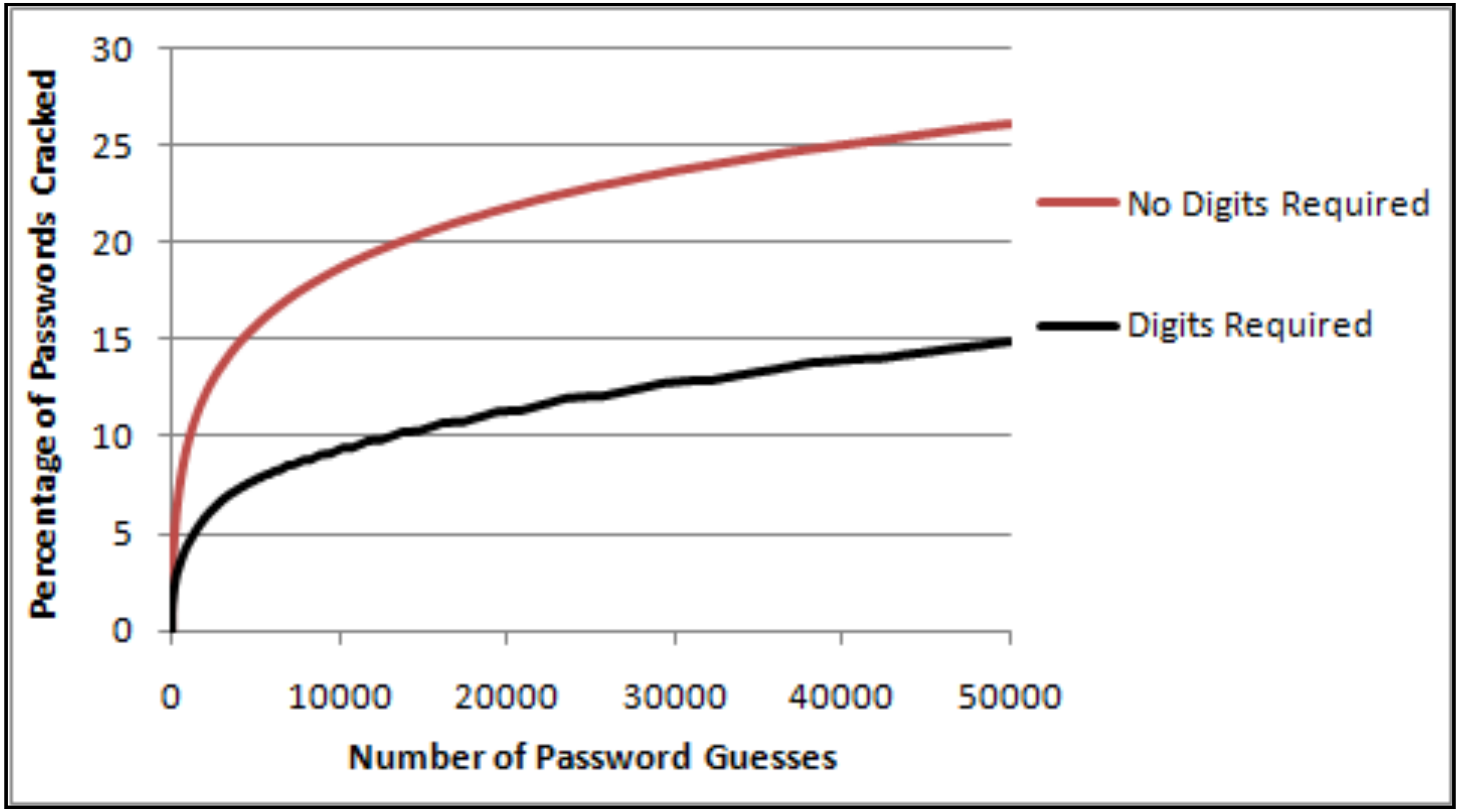
Effect of Password Length



An Even Shorter Cracking Session:



The Effect of Requiring Digits



How Digits were Used:

Location	Example	Percentage
After	password123	64.28%
All Digits	1234567	20.51%
Other	passw0rd, pass123word, p1a2ssword...	9.24%
Before	123password	5.95%

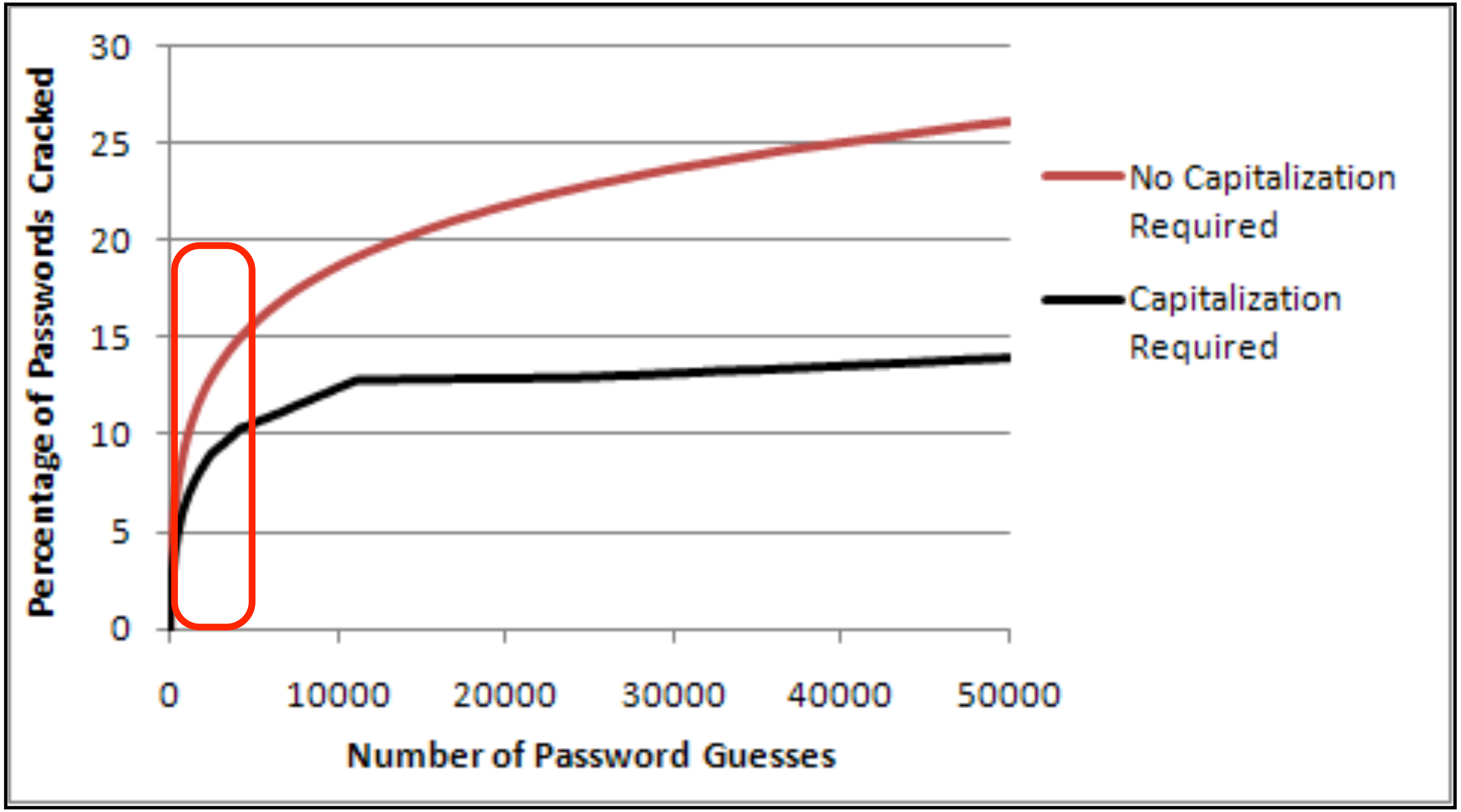
*Taken from 7+ character long passwords that contained at least one digit

Top 10 Digits From the RockYou Training List

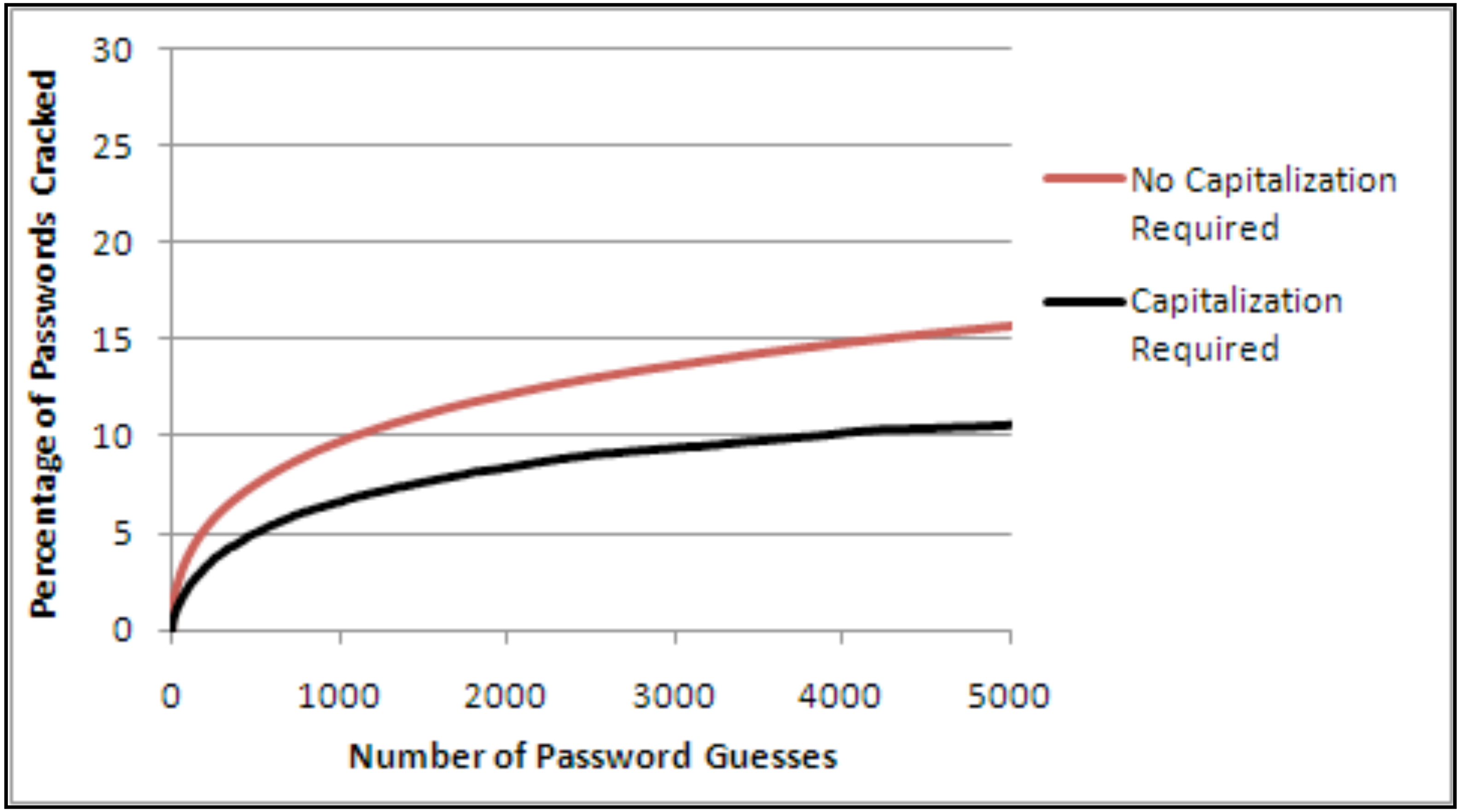
Rank	Digit	Percentage
#1	1	10.98%
#2	2	2.79%
#3	123	2.29%
#4	4	2.10%
#5	3	2.02%
#6	123456	1.74%
#7	12	1.49%
#8	7	1.20%
#9	13	1.07%
#10	5	1.04%

26.72% of All Digits

When Uppercase Characters are Required



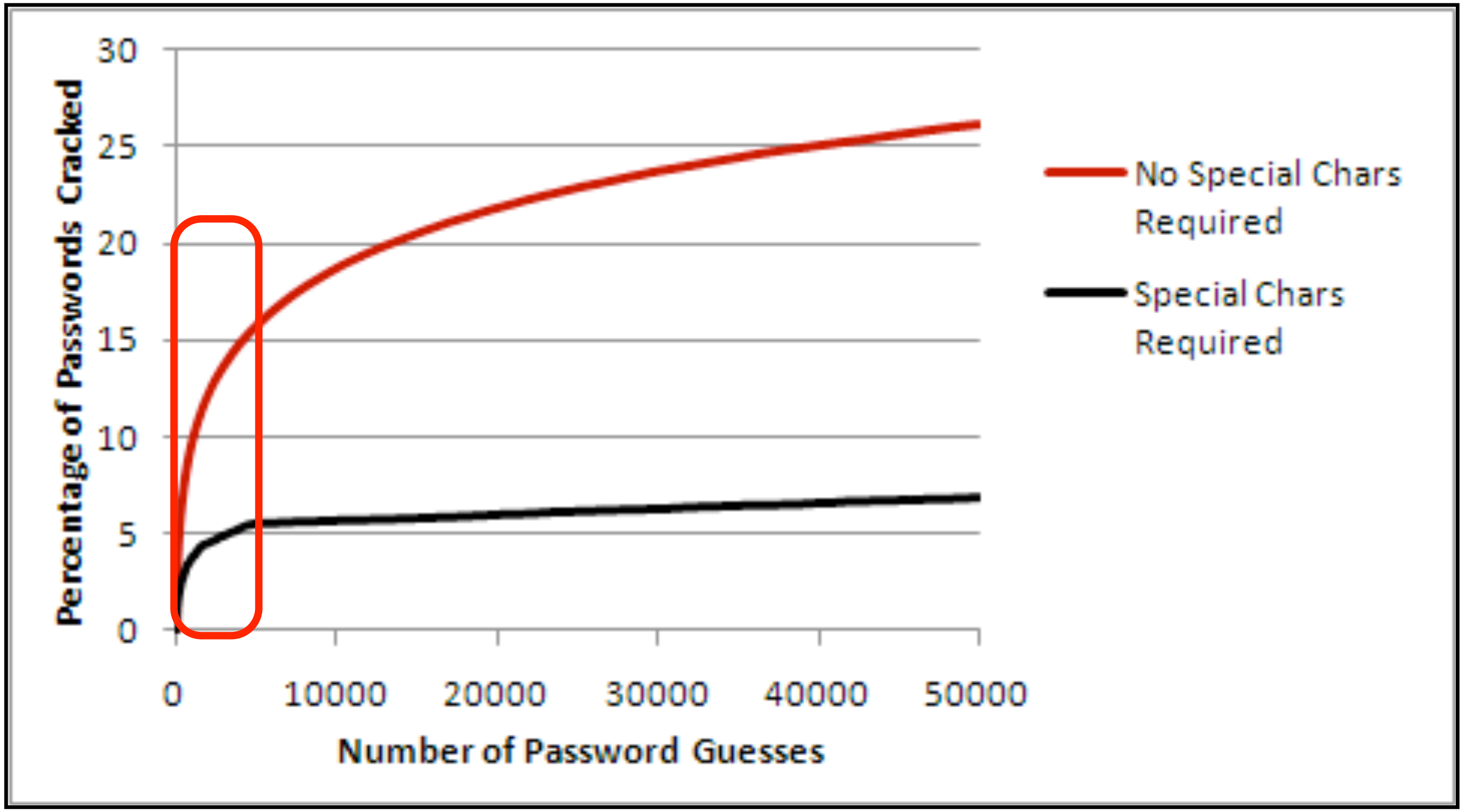
Requiring UpperCase - Shorter Cracking Session



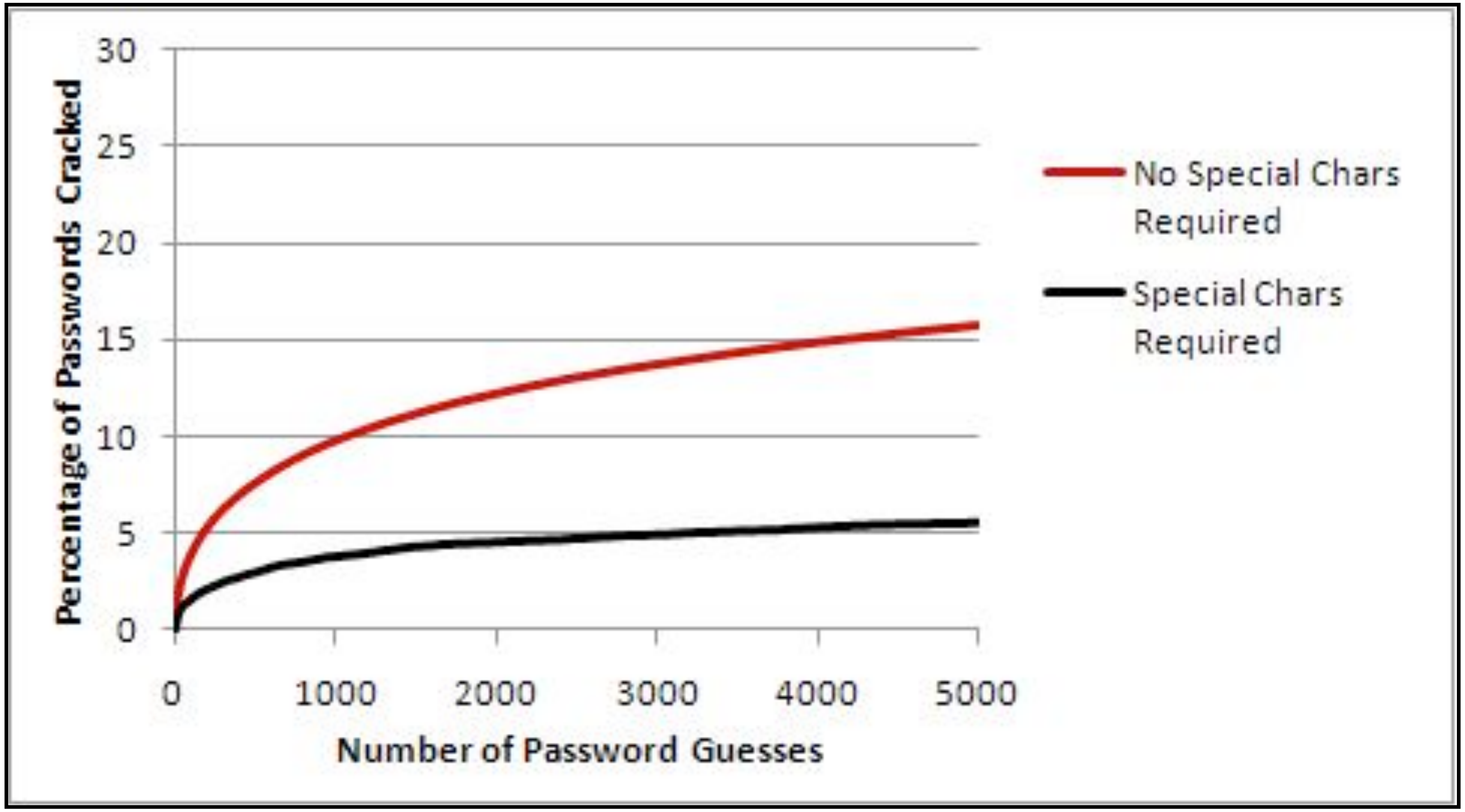
Top Ten Case Mangling Rules of 7 Char Strings

String: U=Upper, L=Lower	Probability
UUUUUUU	53.56%
ULLLLLL	35.69%
ULLLULL	1.05%
LLLLLLL - aka passwor!D	1.03%
ULLLLLU	0.90%
ULLULLL	0.85%
ULULULU	0.68%
LLLLLLU	0.62%
UULLLLL	0.61%
UUULLLL	0.59%

When Special Characters are Required



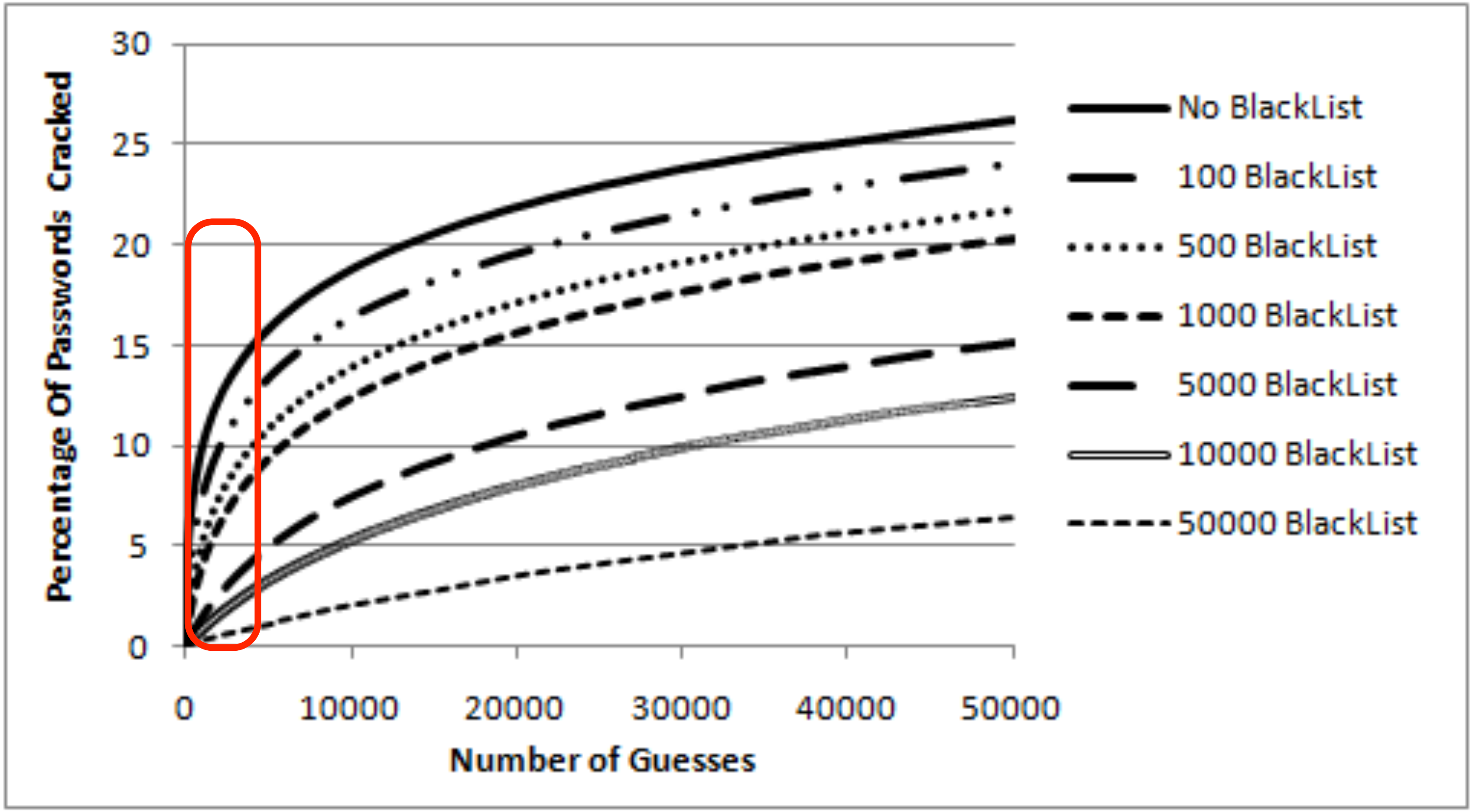
Special Chars Required - Shorter Cracking Session



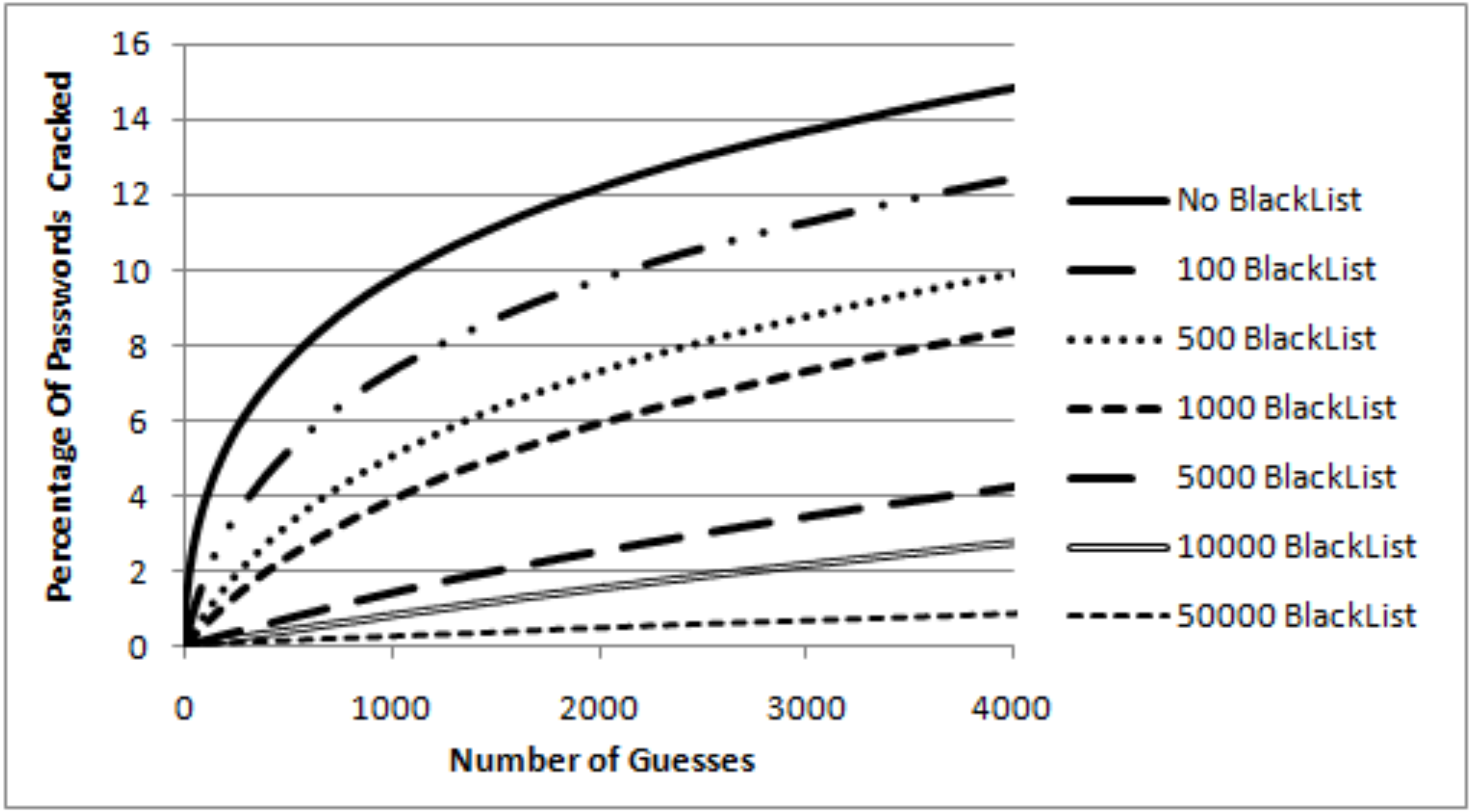
Top Ten Structures for Special Characters

String: A=Alpha, D=Digit, S=Special	Probability
AAAAAAS	28.50%
AAASAAA	7.87%
AAAASDD	6.32%
AAAAASD	6.18%
AASAAAA	3.43%
AAAASAA	2.76%
AAAAASA	2.64%
SAAAAAS	2.50%
ASAAAAA	2.38%
AAAAASS	2.17%

The Effect of BlackLists



A Closer View:

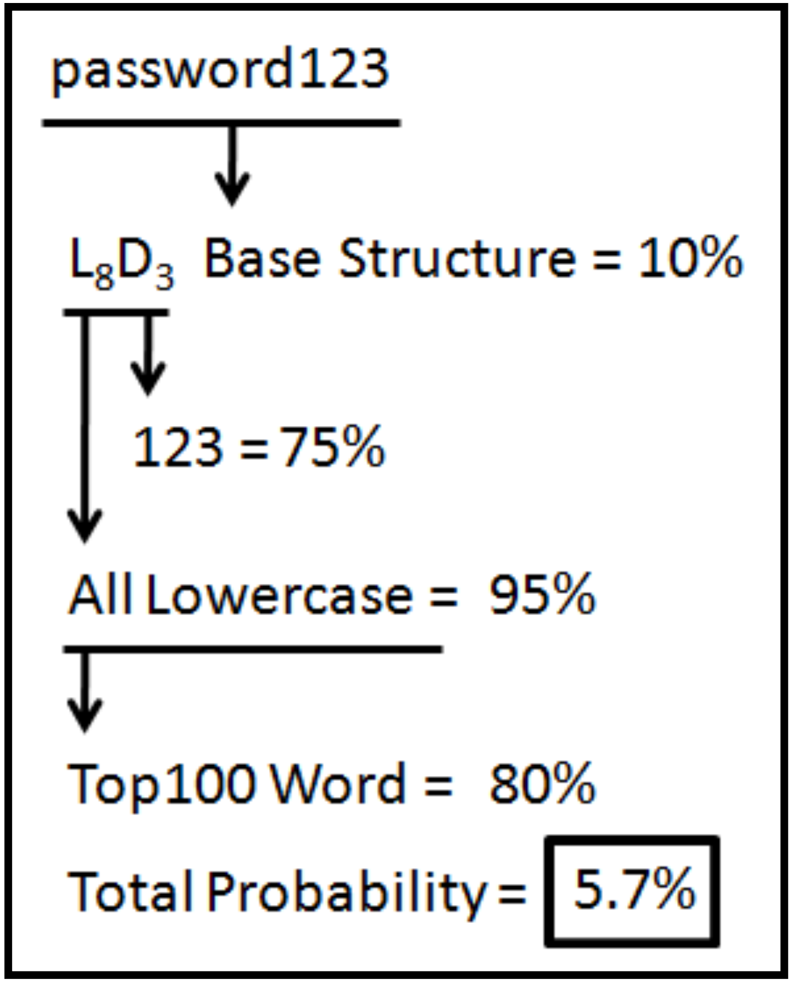


Building a Better Policy

- The hard part is determining what makes a 'strong' password
    - By definition, it is something the attacker does not try
- We can base our policies on known attack techniques
    - What happens when attackers change their techniques in response?

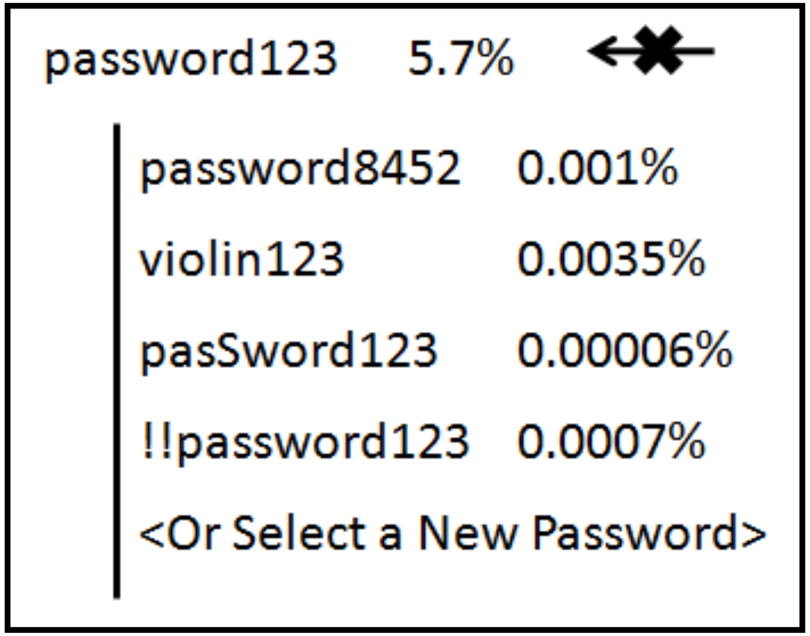
Modifying our Probabilistic PW Cracking Grammar:

- Currently used to attack password creation strategies
- Is trained on a known password list and creates a probabilistic context free grammar that generates password guesses
- We can easily use the same grammar to parse user passwords and assign them a probability as well



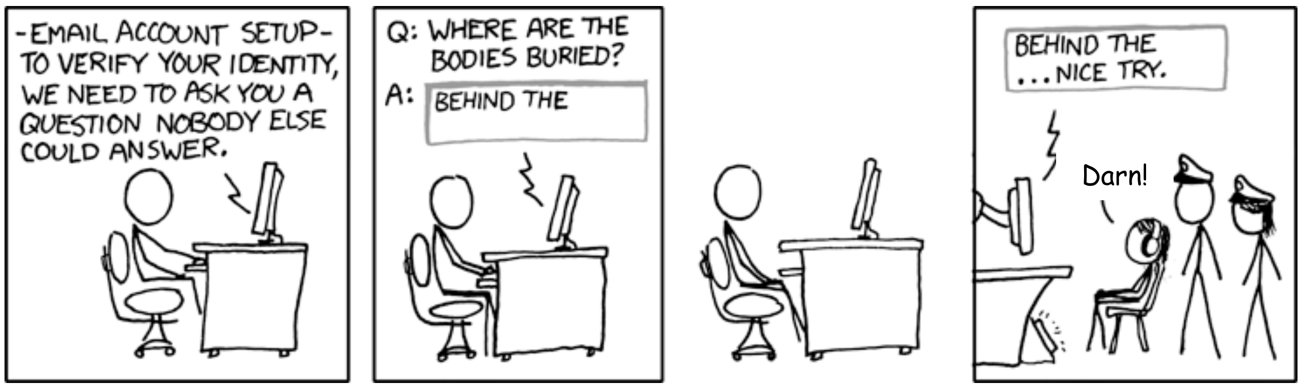
Reject High Probability Guesses

- Since it uses PCFGs it can randomly suggest lower probability replacements
- This way users can select passwords that they want to use with a minimum of interference
    - a long passphrase of all lower case letters
    - Short, but complex: TnW!2s
- This can be paired with a standard blacklist



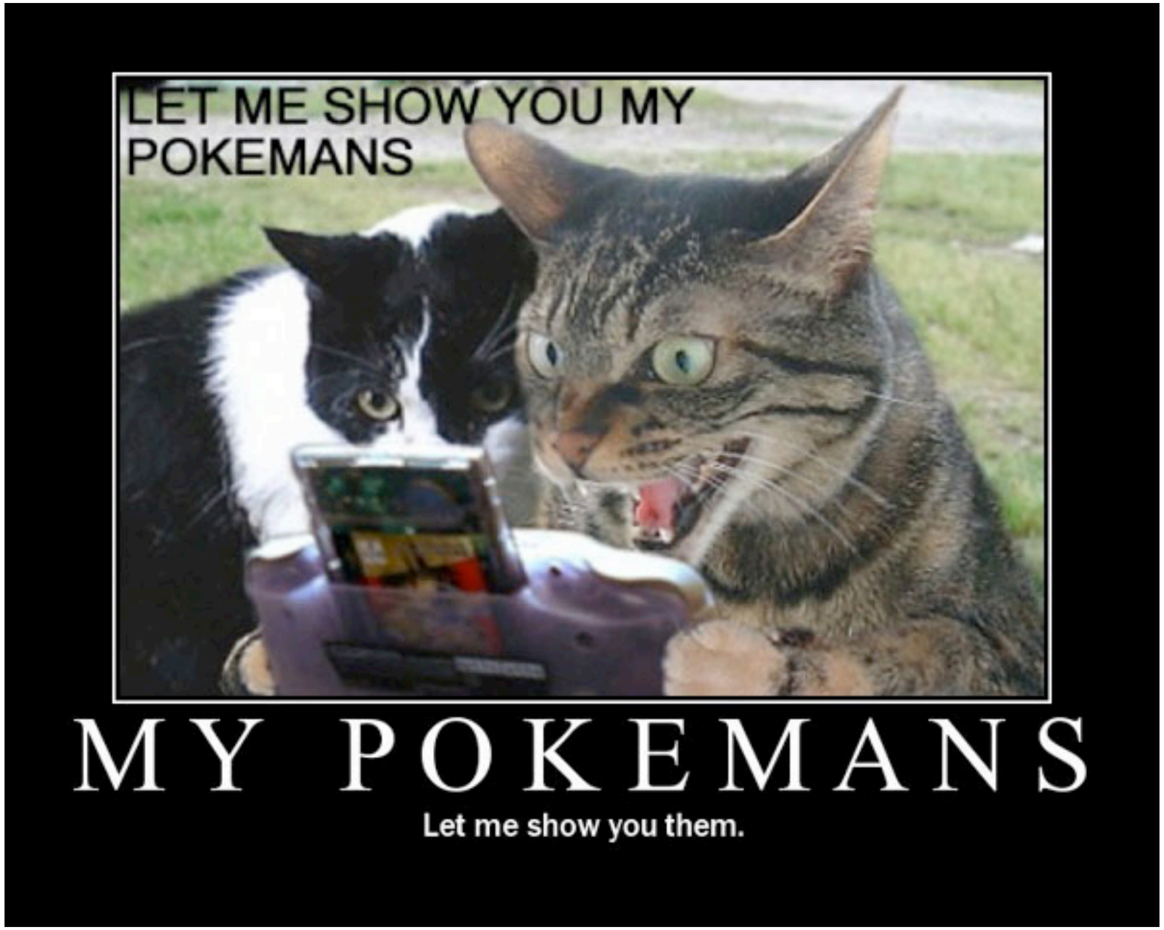
In Conclusion

- The metric of password entropy proposed in NIST 800-63 does not provide actionable information
- By themselves, explicit password creation rules, such as minimum password length and character requirements, still seem to leave systems vulnerable to online attacks
- Implicit password creation rules seem to provide much more security against online attacks

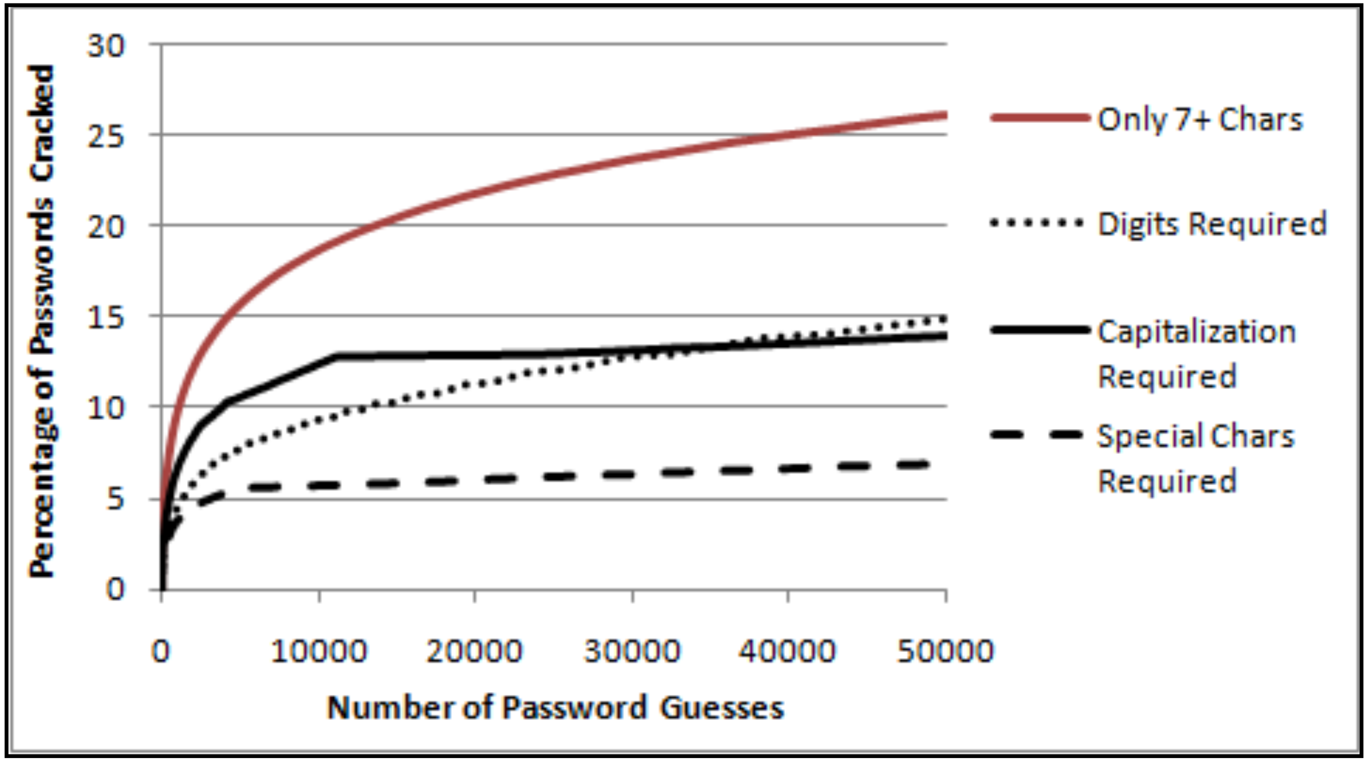


Additional Slides to Use Based on Questions

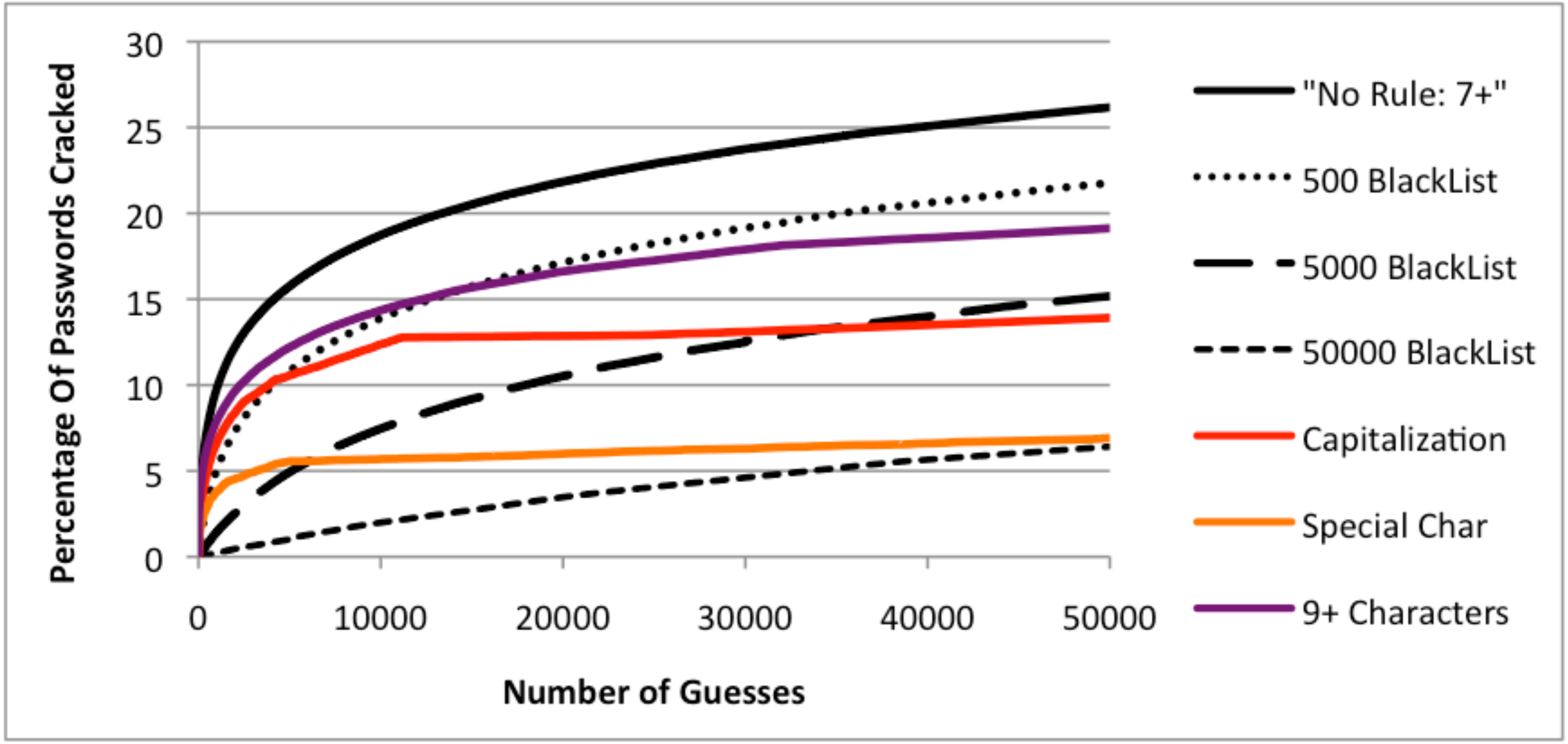
Alt Title: Some People Really Like Graphs...



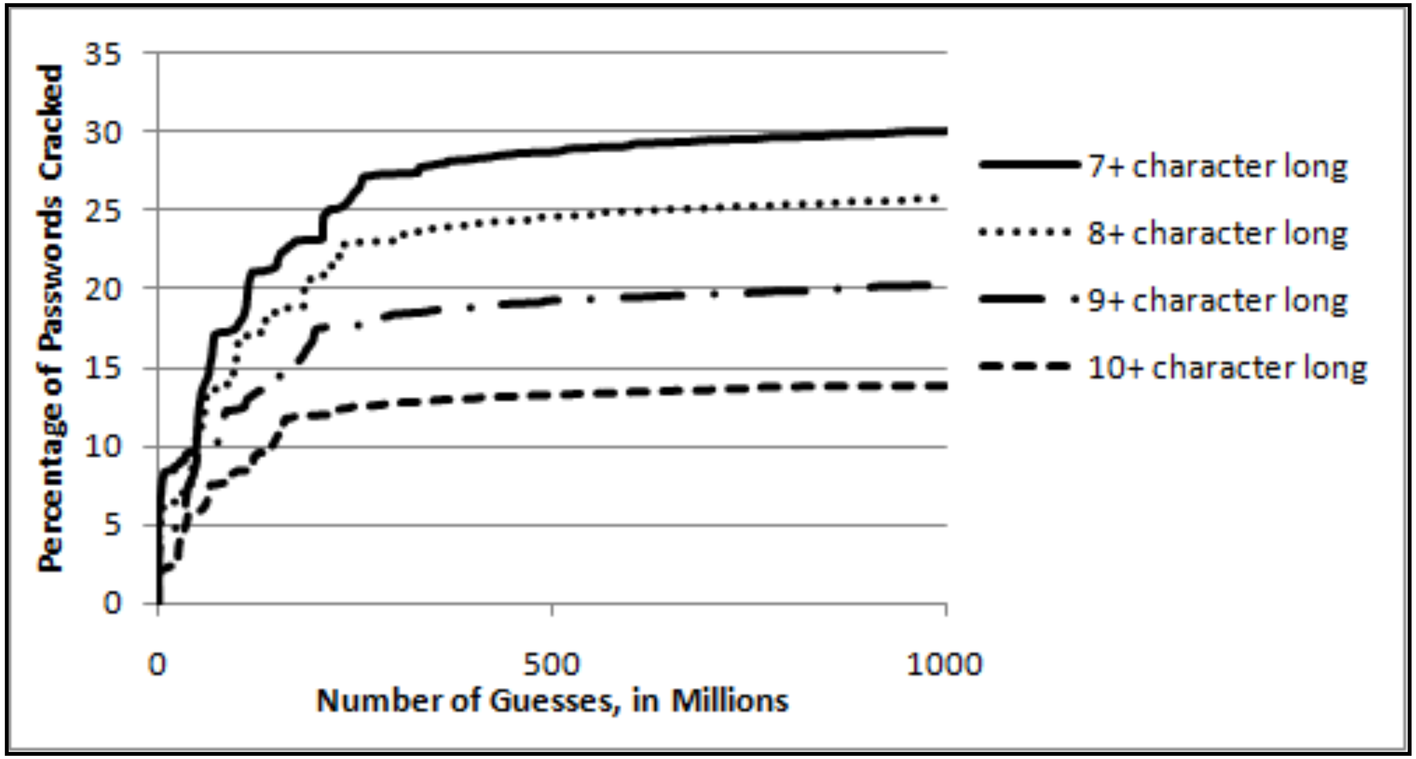
Comparison of Different Password Requirements



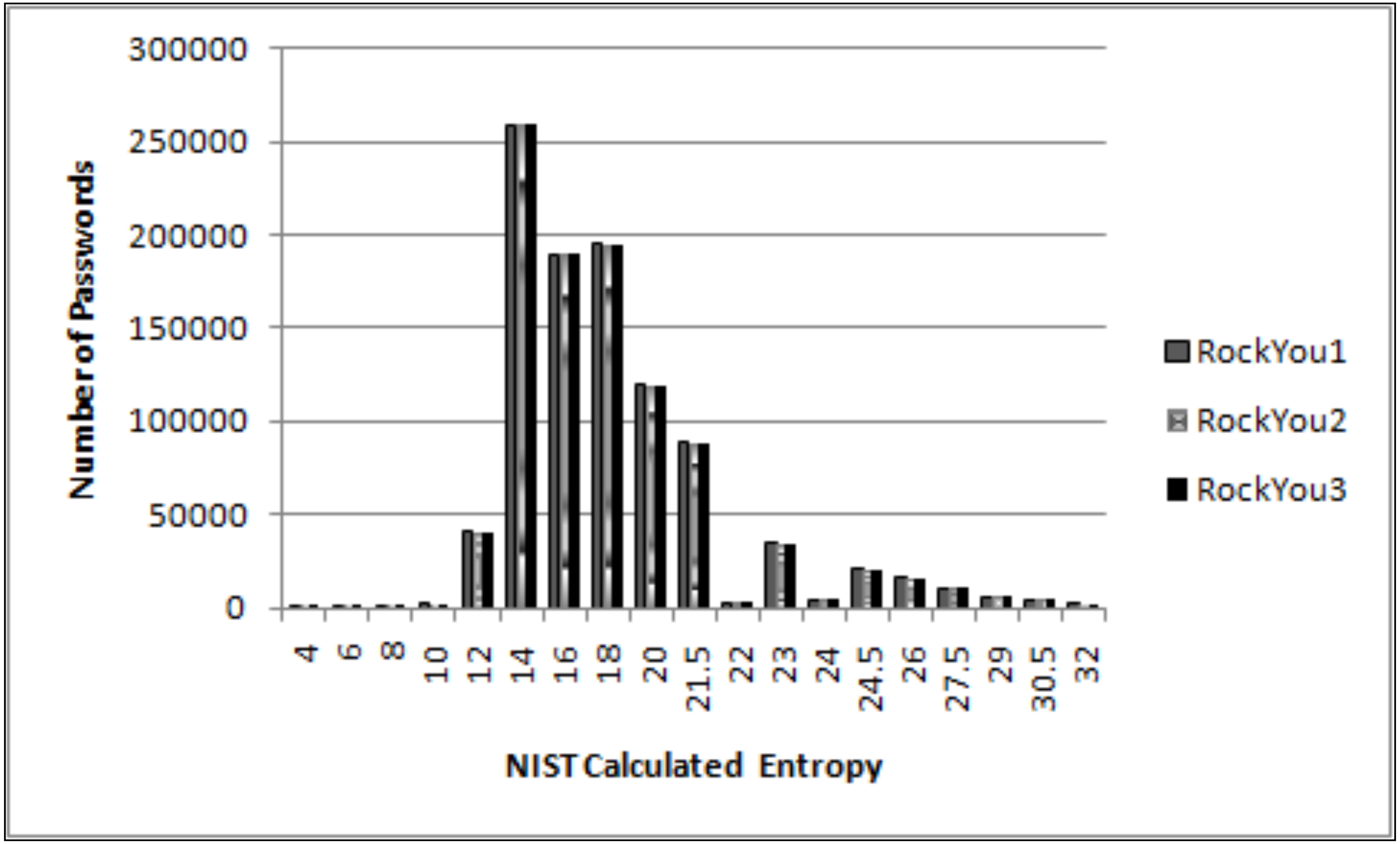
Common Mangling Rules and BlackLists



Standard Offline Password Cracking Attack



NIST Entropy Distribution:



Types of Password Creation Policies

No Guaranteed Minimum Level of Security

- **Explicit**
  - "Your password must be 7 characters long and contain a digit"
- **External**
  - Part of the password is assigned to you, aka a system generated password or two factor authentication
- **Implicit**
  - "Your password isn't strong enough, choose another"
  - Example: Blacklists