Password Authentication

Mike Just
Heriot-Watt University
COINS Summer School on Auth Ecosystems

31 July 2016

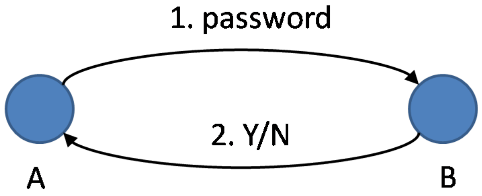
Outline

- Many, many (recent) publications on authentication
  - Several good publications
- I will focus on several publications in the past few years that I think are good, or at least interesting
  - Solve real problems
  - Interesting applications of computer science
  - Consideration of usability and security
- Some of the work is my own. Most is not.

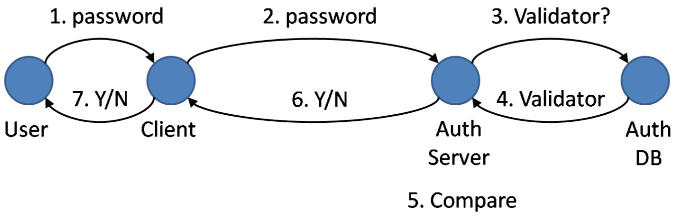
Password Authentication

- A password
  - String of alphanumeric characters
  - Usual length of 6-10 characters
- Purpose
  - Used to uniquely distinguish one entity from another
  - Password used as the “digital representation” of an entity
  - Usually known only to one entity, and a challenger
  - Typically used to control access for an entity to some resource
- Implementation variations
  - *Rules* regarding construction, length, disallowed substrings
  - *Policy* regarding update frequency, re-use
  - *Selection process* including user-chosen, random generation, hybrid

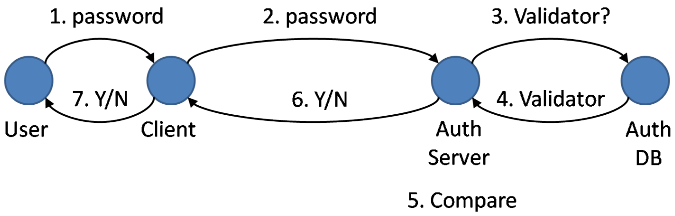
Password Authentication - A Simple Protocol



Password Authentication - Reality

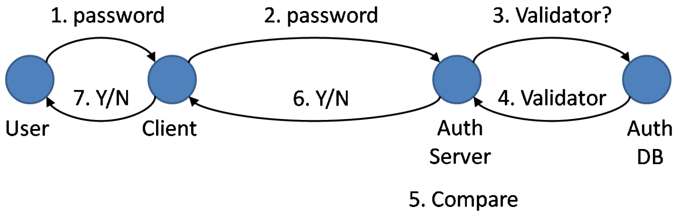


Password Authentication - Reality

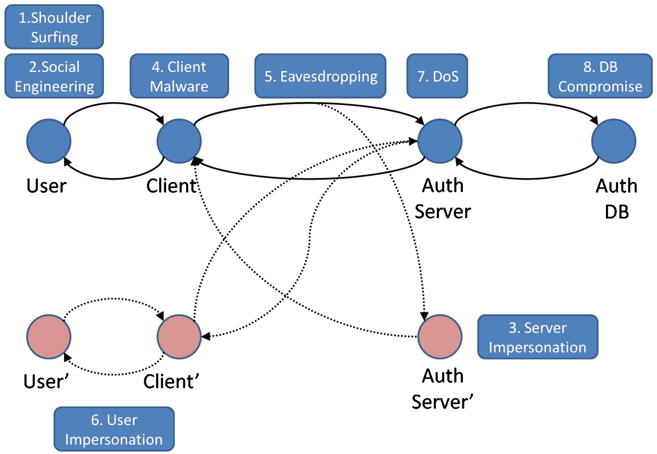


- Not shown
  - Multiple attempts to authenticate (state)
  - Persistent, authentic connection (“cookies”)
  - Multiple users/clients accessing through same server
  - Users/clients accessing multiple servers
  - “Common” protections, such as encryption
  - Multiple authentication factors
  - etc.

Password Authentication – Possible Attacks



Password Authentication – Possible Attacks

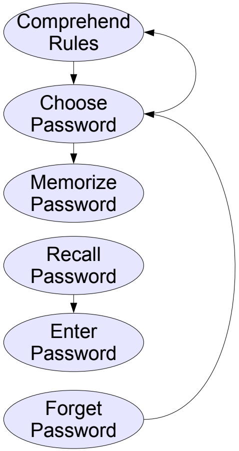


Password Usability I

Issuance

Use

Maintenance



Password Usability II

- Human cognition for selection, and memory for recall
- Different tasks over the lifetime of a password
  - *Creation*: Comprehension of password rules, selection of suitable password, memorization of chosen password
  - *Use*: Password recall, password entry
  - *Maintenance*: Forget password

Password Usability III

Passwords require “100% correct, unaided recall of a non-meaningful item” [Sasse, 2003]

The cause of usability issues are known

- Rules, rules, rules!
    - Length, e.g., of at least 8 characters
    - Diversity, e.g., one uppercase, one lowercase, one number, . . .
    - Update requirements, e.g., monthly
- Limited number of login attempts, e.g., 3
- Accurately memorize a non-meaningful item
- Dozens of accounts with passwords

Password Usability IV

- Human memory
  - Limited capacity of working memory
  - Items stored in memory decay over time
  - Frequent recall improves memorability of items (automatically)
  - Unaided recall is harder than cued recall
  - Non-meaningful items are harder to recall than meaningful ones
  - Similar items compete and are easily confused
  - Items linger in memory – humans cannot “forget on demand”

Password Burden

A heavy burden for users:

- Self-monitor behaviour: Don't share with anyone, don't re-use same password, don't write it down, make sure you're at the legitimate web site, don't respond to phishing, don't click on strange urls, . . .
- Magnified across multiple accounts, with different policies, rules, interfaces, . . .

Password Implementation Principles

A heavy burden for system owners:

- Security toolbelt
  - Encryption, hashing, policy and rules, network protection, anti-* software, user training/education/awareness
- Some implementation principles
  - Obfuscate password on client screen (shoulder surfing)
  - No urls in communication to users (social engineering)
  - Encrypt the network communication (eavesdropping)
  - Limit the number of login attempts (user impersonation)
  - Encrypt the password in DB (DB compromise)
  - Require update, e.g., every 3 months (user impersonation)
  - Disallow re-use, e.g., not recent 10 (user impersonation)
  - Prescribe construction, e.g., length (user impersonation)
  - Disallow substrings, e.g., username (user impersonation)
- Each needs to be balanced with *usability* as well

Password Implementation Decisions

- Proper password implementation can be complicated
  - Balance of (at least) security and usability
  - Follow *all* principles for *all* situations?
- How well are these principles adhered to today?
- How many principles are still valid today?
- Recent research provides evidence, primarily for web authentication

Password Security in Practice – “Common” Protections

- Cambridge study of password practice at 150 web sites
- 70% of sites don’t encrypt (crypto. hash) their passwords
    - Vulnerable to theft from server
- 41% don’t encrypt communication (addn’l 28% inconsistent)
    - Vulnerable to eavesdropping and hijacking
- 19% return error for wrong login username (80% at recovery)
    - Vulnerable to ID theft (for subsequent attacks)
- 84% do not throttle password guessing
    - Vulnerable to online guessing (user impersonation)

Password Security in Practice – Password Strength

- MS Research study of password strength at 75 web sites
- Discovered correlation between usability and password strength (size and rules)
    - No correlation to site size or value of resources
- Sites trying to attract/retain customers had lowest strength
- Sites where there was little choice (government, university) had the highest strength

Password security I

- How (in)secure are passwords?
- Typical approaches to measure the strength of a password distribution
- Counting the number of passwords, e.g.,
  - 8-character, alphabetic: $52^8 \sim 2^{46}$
  - 6-character, alphanumeric + special: $72^6 \sim 2^{37}$
- But this is an overestimate, and assumes that passwords are randomly chosen

Password security II

- Most texts (and NIST guidance) use *Shannon entropy* to measure an attacker's uncertainty in a password

$$\sum_{i=1}^{N} -p_i \log_2 p_i$$

- Requires knowledge of the password distribution
- Though this is not so problematic
  - Several password databases compromised in recent years
  - Bonneau (2012) has had some suggest at extrapolating a larger password distribution from several smaller samples

Password security III

- However, Shannon entropy overestimates password strength since it does not consider an attacker who gives up after β guesses
- Metrics by Bonneau, Just & Matthews (2010) and Bonneau (2012)

$$\begin{aligned} \lambda_\beta(\mathcal{X}) &= \sum_{i=1}^{\beta} p_i \\ \mu_\alpha(\mathcal{X}) &= \min\left\{ j \in [1, N] \middle| \sum_{i=1}^{j} p_i \geq \alpha \right\} \end{aligned}$$

Password security IV

- The *marginal success rate* (λ_β) is usually used for measuring *online attacks*
    - λ_3 might be 2% at each account by guessing top 3 passwords
    - With λ_{10}, Bonneau showed that Yahoo! account passwords had 10 bits of security
- The *marginal guesswork* (μ_α) is usually used for measuring *offline attacks*
    - $\mu_{0.5}$ might show that 1,000 guesses are required for a 50% success rate
    - With $\mu_{0.5}$, Bonneau showed that Yahoo! account passwords had 20 bits of security
    - Enforced password rules could improve security

Password security V

- Measuring the strength of an *individual password* comes down to heuristic techniques
    - Feedback on a particular password choice
    - Measurement techniques (e.g., *password meters*) can draw upon *password cracking* approaches (e.g., n-gram Markov models)
- Improving security often involves enforcing password rules
    - Rules (e.g., include a special character) are intended to increase the number of chosen passwords, and “flatten” the distribution
    - Password rules are complicated, and annoying to follow

Password authentication

- Is security improved by providing more information to users?
- Two design techniques to assist in choosing secure passwords
  - Password rules/guidance
  - Password strength feedback
- Do these practices help?

Password rules/guidance

- Password *guidance* suggests techniques that users may want to follow during password construction
- Password *rules* force requirements on the constructed password

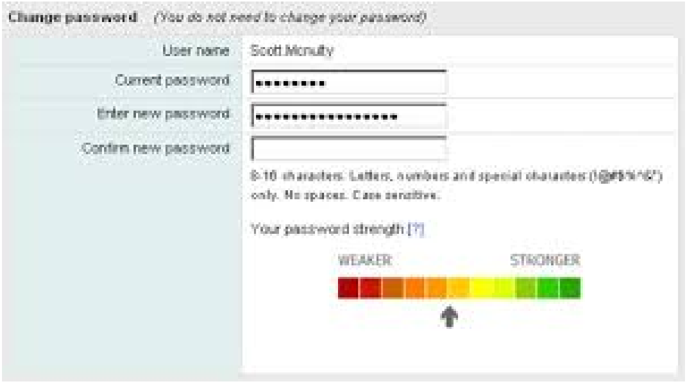
Password rules/guidance

- Password *guidance* suggests techniques that users may want to follow during password construction
- Password *rules* force requirements on the constructed password
- Guidance is typically not read, understood, followed
- Rules often lead users towards common paths of least resistance
  - "What's the simplest password that I can choose to meet the password rules?"
- Common, top passwords
  - `password`, `password1`, `p@ssw0rd1`
- To deal with such problems today, many systems provide feedback during password construction

Password construction feedback I

- Feedback on password construction can be provided either immediately, or in response to a password choice
- Design factors
  - How to display the rules, e.g., static, linked to password choice
  - Use of colour, e.g., red, yellow, and green (traffic light metaphor)
  - Strength indicators, e.g., weak, good, strong
- The basic idea is that without feedback, some people may not have known how to construct good passwords
- And feedback can provide positive confirmation of a choice, which may help to improve recall

Password construction feedback II



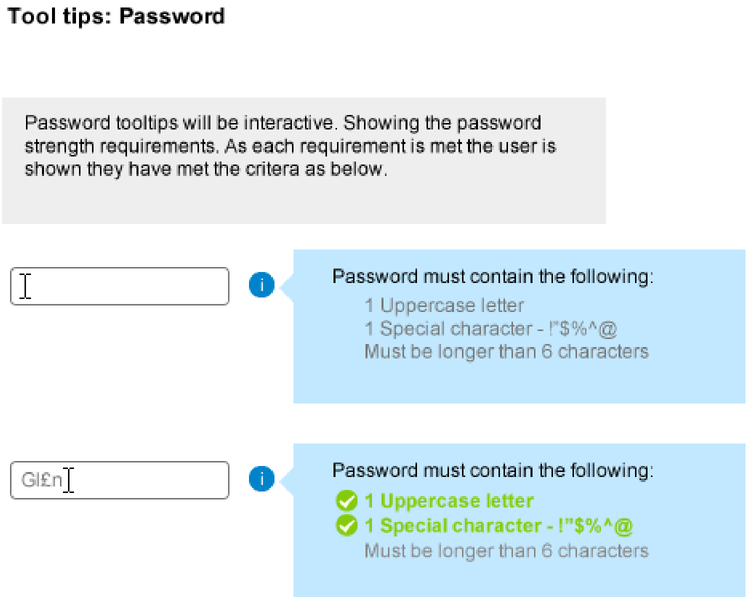
Change password *(You do not need to change your password)*

User name	Scott Mcnulty
Current password	••••••••
Enter new password	•••••••••••••••
Confirm new password	

8-10 characters. Letters, numbers and special characters (!@#$%^&*) only. No spaces. Case sensitive.

Your password strength [?]

Password construction feedback III



Tool tips: Password

Password tooltips will be interactive. Showing the password strength requirements. As each requirement is met the user is shown they have met the critera as below.

Password must contain the following:

1 Uppercase letter
1 Special character - !"$%^@
Must be longer than 6 characters

Gl£n

Password must contain the following:

✓ 1 Uppercase letter
✓ 1 Special character - !"$%^@
Must be longer than 6 characters

Password construction feedback IV

- Does feedback help?
- Yes, to some degree
    - It can increase efficiency, effectiveness and satisfaction (usability)
    - Though people can still choose weak passwords
- What about influencing password choices more strongly?
    - System chosen passwords are generally a bad idea (though there is interesting research in this area)
    - But what about *persuading* users to change a few characters in their password?

Password Attacks in Practice

- Increased examples of attacks using social engineering (phishing) and client malware (keystroke logging)
- Several high-profile examples of access that by-pass password (e.g., challenge questions)
- Several recent, substantial attacks involving DB theft (e.g., Gawker, RockYou)
  - Confirmation of basic oversights such as no password hashing
  - Great source of experimental data of password choices
  - Evidence of password re-use

Password Implementation

- Evolution of tools for password protection
- Evidence suggests a rethink of some previously held beliefs and principles, e.g., some attacks better mitigated today
  - *Shoulder surfing* mitigation with character obfuscation
  - *Server impersonation* with certificates and browser locks
  - *Eavesdropping* with network encryption
  - *DB Compromise* with DB protection, e.g., password hashing
  - Server monitoring and protection help mitigate several attacks
- Prioritize the threats: Phishing, malware, and password re-use

Improved Password Usability

- Improved usability begins with the identification of some security myths
- Some Password Myths

Improved Password Usability

- Improved usability begins with the identification of some security myths
- Some Password Myths
  1. Strong passwords are necessary
  2. Frequent password updates are necessary
  3. You must not write down your password
  4. Password masking (‘*’) is necessary
- Mythic quality varies over time

Myth 1 – Necessity of Strong Passwords

- Generally defined by a bitstrength of 60 bits or more
- Strong passwords (length and rules) are necessary

Myth 1 – Necessity of Strong Passwords

- Generally defined by a bitstrength of 60 bits or more
- Strong passwords (length and rules) are necessary
  - A Myth

Myth 1 – Necessity of Strong Passwords

- Generally defined by a bitstrength of 60 bits or more
- Strong passwords (length and rules) are necessary
    - A Myth
- Why?
    - Impact on usability, including productivity
    - There are alternative means of mitigation
    - No protection against phishing and keyboard logging
- Alternative means of mitigation
    - Increase strength of user ID's
    - Control number of attempts, moderate guessing (CAPTCHAs)
    - Use of network monitoring and protection
    - Use of 2nd factors, increased trust from frequent machines

Myth 2 – Necessity of Frequent Password Updates

- Requirement to periodically change password (e.g., monthly)
  - Prevention and recovery
- Frequent password updates are necessary

Myth 2 – Necessity of Frequent Password Updates

- Requirement to periodically change password (e.g., monthly)
    - Prevention and recovery
- Frequent password updates are necessary
    - A Myth

Myth 2 – Necessity of Frequent Password Updates

- Requirement to periodically change password (e.g., monthly)
  - Prevention and recovery
- Frequent password updates are necessary
  - A Myth
- Why?
  - Impact on usability, including productivity
  - There are alternative means of protection (user impersonation)
  - User-chosen transformations often very simple
- Alternative means of protection
  - Use of network monitoring and protection
  - User notification

Myth 3 – Don't Write Down Your Password

- Writing or recording of password as a memory aid
  - Evidence of more than half of users doing this
- You must not write down your password

Myth 3 – Don't Write Down Your Password

- Writing or recording of password as a memory aid
  - Evidence of more than half of users doing this
- You must not write down your password
  - A Myth

Myth 3 – Don't Write Down Your Password

- Writing or recording of password as a memory aid
  - Evidence of more than half of users doing this
- You must not write down your password
  - A Myth
- Why?
  - Encourages password re-use
  - Encourages weak passwords (within the rules)
  - Limited threat in many cases
  - Some evidence that users take sufficient precautions
- Inclination to record might be partially mitigated through first two myths

Myth 4 – Necessity of Password Masking

- Password masking (‘*’) obfuscates screen display from shoulder surfing
- Password masking is necessary

Myth 4 – Necessity of Password Masking

- Password masking (‘*’) obfuscates screen display from shoulder surfing
- Password masking is necessary
- Not really a myth for passwords
  - Though many services do give users control over the password visibility
  - Though mythical for some info-based alternatives, such as challenge questions
- For passwords
  - Only partially mitigates shoulder surfing
  - Users seem to have adapted (cost is *low*)
  - Clever solution for shoulder surfing mitigation

Evolution in Practice Today

- Some evidence of new (sometimes improved) practices
    - Password loathing difficult to assess in changing environment
- Reduced password strength as customer incentive
- Prominent examples of communication encryption
- Multiple factors
    - Mobile phones
    - Secondary passwords
    - Authentication of location
    - Increased trust for frequent machine
    - Multiple factors for certain situations
- Implementation decisions motivated by many dependencies
    - Security (incl. threat picture), usability, revenue, cost
    - Client profiles?

Password Alternatives

- There are numerous alternatives to passwords
    - Traditional passwords are not a long-term solution
    - And not a solution for every situation
- Biometrics continue to be a *niche* solution
    - Though some apparent success on mobile devices
- Some success with tokens (as a second factor)
- Yet information-based solutions remain ubiquitous
- Some information-based alternatives to traditional passwords
    1. Popularity-based passwords
    2. Password persuasion
    3. Honeywords
    4. Random passwords

Popularity-Based Authentication I

- Microsoft Research proposal
- Problem: Left to their own devices, users will choose “weak” (common) passwords
- Two general solutions
  1. Reduce the number of guesses an attacker can make
  2. Reduce the popularity of certain passwords
- The first can be done by limiting guesses, even from particular locations
- The second is currently achieved with password rules
  - General Solution: “Flatten” the password distribution

Popularity-Based Authentication II

- Password rules are a cat-and-mouse game
    - Common password: `password`

Popularity-Based Authentication II

- Password rules are a cat-and-mouse game
  - Common password: `password`
  - Require a number: `password1`

Popularity-Based Authentication II

- Password rules are a cat-and-mouse game
    - Common password: `password`
    - Require a number: `password1`
    - Require special character: `p@ssword1`

Popularity-Based Authentication II

- Password rules are a cat-and-mouse game
    - Common password: `password`
    - Require a number: `password1`
    - Require special character: `p@ssword1`
    - Require uppercase: `P@ssword1`

Popularity-Based Authentication II

- Password rules are a cat-and-mouse game
    - Common password: `password`
    - Require a number: `password1`
    - Require special character: `p@ssword1`
    - Require uppercase: `P@ssword1`
- Proposal: Popularity rules
    - Let users chose whatever password they want, so long as it's not too popular
- Challenge: How to keep track of list of popular passwords
    - Without making it a target for an attacker

Popularity-Based Authentication III

- Current password storage uses a cryptographic hash function $h()$ for the password p_i of each user u_i ("one-way")

$$u_i, s_i, h(s_i, p_i)$$

- A unique salt value s_i is normally added to reduce simultaneous guessing across all users

Popularity-Based Authentication III

- Current password storage uses a cryptographic hash function $h()$ for the password p_i of each user u_i ("one-way")

$$u_i, s_i, h(s_i, p_i)$$

- A unique salt value s_i is normally added to reduce simultaneous guessing across all users
- To track password frequency, we could store a frequency value f_i with each unsalted hashed password

$$h(p_i), f_i$$

- However, this information would give too much about the password distribution to an attacker

Popularity-Based Authentication III

- Current password storage uses a cryptographic hash function $h()$ for the password p_i of each user u_i ("one-way")

$$u_i, s_i, h(s_i, p_i)$$

- A unique salt value s_i is normally added to reduce simultaneous guessing across all users
- To track password frequency, we could store a frequency value f_i with each unsalted hashed password

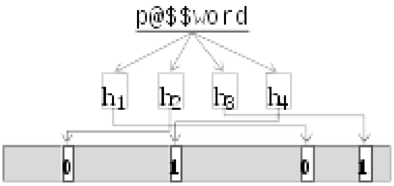
$$h(p_i), f_i$$

- However, this information would give too much about the password distribution to an attacker
- Rather, we could use a *Bloom filter*
  - With a t-bit binary vector, and k hash functions

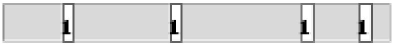
Popularity-Based Authentication IV

```
add(s)     for($i \in [1, k]$)
               $t[h_i(s)] \leftarrow 1$

query(s)   return  $^{k}_{i=1}$ $t[h_i(s)]$
```



(a)Before add(p@$$word).isMember(p@$$word)is *false*



(b)After add(p@$$word).isMember(p@$$word)is *true*

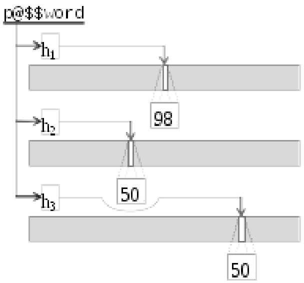
Popularity-Based Authentication V

- Bloom filters were used by Spafford in 1991 to track non-permitted dictionary words for passwords
- Though a Bloom filter only demonstrates existence, not frequency
- But a *count-min sketch* could be used
  - Each of k hash functions contribute to its own vector

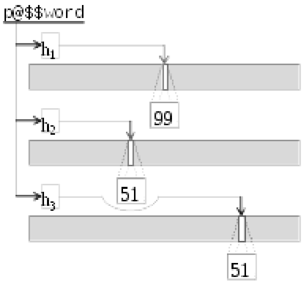
Popularity-Based Authentication VI

```
count(s)   Return min($t_1[h_1(s)], t_2[h_2(s)], \ldots, t_k[h_k(s)]$)

add(s)     For $i \in [1, k]$
               $t_i[h_i(s)] \leftarrow t_i[h_i(s)] + 1$
```



(a)Before add(p@$$word).count(p@$$word)=50



(b)After add(p@$$word).count(p@$$word)=51

Popularity-Based Authentication VII

- *False positives* for both Bloom filters and count-min sketch
    - Unpopular passwords will, in some cases, be deemed popular
- However, this is actually advantageous, limiting the information gained by an attacker

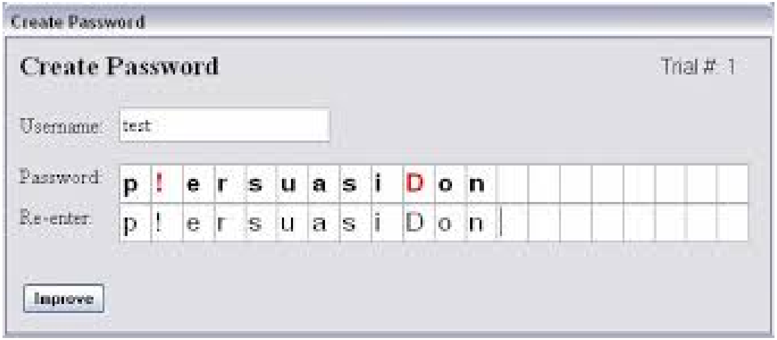
Popularity-Based Authentication VII

- *False positives* for both Bloom filters and count-min sketch
  - Unpopular passwords will, in some cases, be deemed popular
- However, this is actually advantageous, limiting the information gained by an attacker
- Biggest challenge of *popular password rules*
  - What *feedback* do you give to a user who has chosen a popular password?

Password persuasion I

- For persuading an improved password choice
  - User chooses password on their own
  - User is then presented with a modified version, e.g., with random characters inserted or added
  - Users can "shuffle" alternatives
- Goal is to allow user to choose variations to their password
  - Idea is that interaction with user will improve satisfaction and effectiveness (recall of modified password)

Password persuasion II


Create Password

Create Password

Trial #: 1

Username:	test
Password:	p ! e r s u a s i D o n
Re-enter:	p ! e r s u a s i D o n

Password persuasion III

- Three variants of arriving at a changed password were considered:
  - Pre-load: Users are presented with a small number of random characters prior to their password selection
  - Replace: After their initial password selection, a small number of their password characters are replaced with random characters
  - Insert: After their initial password selection, a small number of random characters are added to their password
- The “small number” of characters varied from 2 to 4

Password persuasion IV

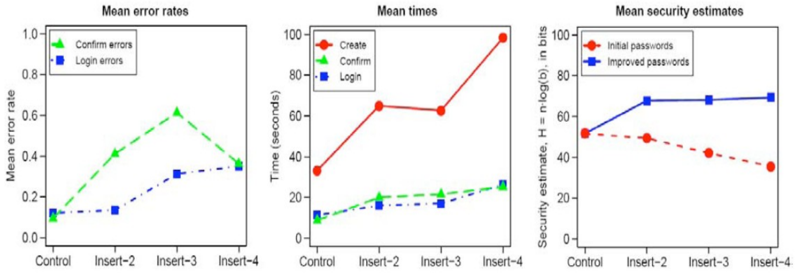


Figure 11: Summary of mean times, error rates, and a crude relative security estimate across conditions. The lines are present for ease of interpretation and do not represent continuous values.

Online vs. Offline Attacks

- Generally two types of *password guessing* attacks
  - Online: Guesses through the online authentication server
  - Offline: Guesses through the database, and password hashes
- Online attacks can be mitigated, e.g., limited guess attempts, multiple authentication factors
- Offline attacks are more difficult to mitigate
  - Multiple hashes (e.g., 10,000) have some effect, but slow authentication

Honey Passwords I

- Honey-* denotes a fake target used to lure an attacker, e.g., *honeynets*
- Such approaches have been proposed previously for authentication
    - Fake accounts in a password file, though may be detectable by attacker
    - Fake password files
- Juels and Rivest (2013) recently proposed “honeywords”

Honey Passwords II

- Rather than storing $u_i, h(p_i)$[1], we would store

$$(u_i, H_i)$$

where

$$\begin{aligned} H_i &= (v_{i,1}, v_{i,2}, \ldots, v_{i,k}) \\ v_{i,j} &= h(w_{i,j}) \end{aligned}$$

- and W_i is the set of "potential passwords", one of which is the real password

$$W_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,k})$$

[1]Salt ignored for simplicity

Honey Passwords III

- With a compromised password database, an attacker would now be faced with k password hashes for each user (e.g., k=20)
- Two advantages
  - Increases the work effort for an attacker
  - Allows detection by the server when a "sweet word" (that is not the correct password) is used
- Some implementation considerations, e.g., choosing believable sweet words

Honey Passwords III

- With a compromised password database, an attacker would now be faced with k password hashes for each user (e.g., k=20)
- Two advantages
  - Increases the work effort for an attacker
  - Allows detection by the server when a “sweet word” (that is not the correct password) is used
- Some implementation considerations, e.g., choosing believable sweet words
- However, the information regarding which password is the correct one has to be available somewhere!

Honey Passwords IV

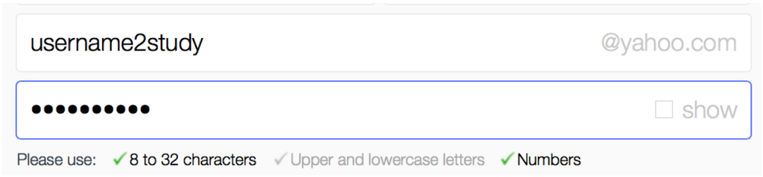
- To support the password validation, a secure, auxilliary *honeychecker* can be used
- For each user u_i, the honeychecker stores the index j such that $w_{i,j} = p_i$
    - In other words, the honeychecker knows the index that identifies the correct user password for each user
- Compromise of the honeychecker reduces security to the traditional password hashing protection method

Honey Passwords V

- Honeywords (and honey encryption) have also been proposed as a way to further protect *password managers/vaults*
- Password vaults are used to store a user's passwords, and are protected by a single master password
- Compromise of the master password would allow access to all other passwords
- Chatterjee et al. (2015) have extended the idea of honeywords with a *natural language encoder* such that unlocking the vault with another master password would produce a set of plausible passwords

Stronger Passwords

- Are there other ways to strengthen passwords?
- Better enforcement of password rules
- Requirements feedback seems to generally improve security



username2study @yahoo.com

●●●●●●●●●●● ☐ show

Please use: ✓ 8 to 32 characters ✓ Upper and lowercase letters ✓ Numbers

- Though password creation support results in weaker choices

Random Passwords I

- Bonneau and Schechter (2014) proposed the use of random passwords

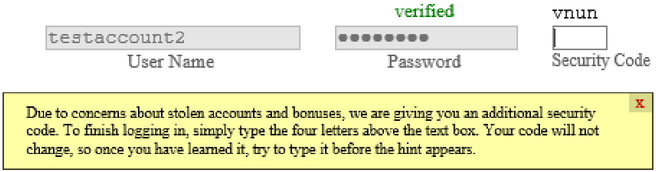
Random Passwords I

- Bonneau and Schechter (2014) proposed the use of random passwords
- Yes, that's right, random passwords

Random Passwords I

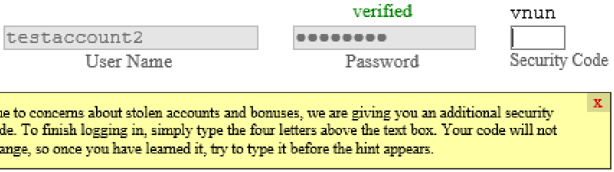
- Bonneau and Schechter (2014) proposed the use of random passwords
- Yes, that's right, random passwords
- Goal was to see if people could memorize a 56-bit code (six words)
- Approach
  - Memorization through *spaced repetition*
  - Experiment: Users asked to login with their own password over two-week period
  - At each login, they were provided with a new part of the code that they had to enter

Random Passwords II


verified
vnun
testaccount2
User Name
●●●●●●●●●
Password
Security Code
x

Due to concerns about stolen accounts and bonuses, we are giving you an additional security code. To finish logging in, simply type the four letters above the text box. Your code will not change, so once you have learned it, try to type it before the hint appears.

Random Passwords II

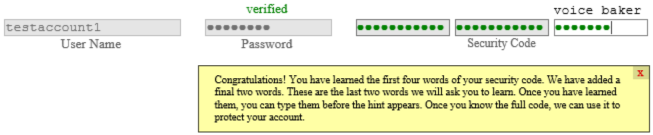


verified

vnun

testaccount2	●●●●●●●●●	
User Name	Password	Security Code

Due to concerns about stolen accounts and bonuses, we are giving you an additional security code. To finish logging in, simply type the four letters above the text box. Your code will not change, so once you have learned it, try to type it before the hint appears.



verified

voice baker

testaccount1	●●●●●●●●	●●●●●●●●●●●●	●●●●●●●●●●●●	●●●●●●●
User Name	Password		Security Code	

Congratulations! You have learned the first four words of your security code. We have added a final two words. These are the last two words we will ask you to learn. Once you have learned them, you can type them before the hint appears. Once you know the full code, we can use it to protect your account.

Random Passwords III

- Results (with around 225 participants)
  - 94% success learning codes after (median) 36 logins
  - 88% recall, three days after experiment
  - Added delay of $<$ 7s per login

Random Passwords III

- Results (with around 225 participants)
  - 94% success learning codes after (median) 36 logins
  - 88% recall, three days after experiment
  - Added delay of $<$ 7s per login

For those discouraged by the ample literature detailing the problems that can result when users and security mechanisms collide, we see hope for the human race.