Paper Review

• **Standard format**
  – Summary: a few sentences to summarize the paper
  – Strengths: a few bullets to summarize why you like this paper
  – Weaknesses: a few bullets to point out places that need to be improved
  – Detailed comments: elaborate on the strengths and weaknesses, provide detailed suggestions to revise the paper

• **Best Practice**
  – Write reviews that you would like to receive if you were the paper authors
  – Be constructive while being critical
  – Be specific, provide evidence for your arguments
  – Provide actionable things that can help the authors to improve the paper
How to Read Papers

• What is the key idea in this paper?
• What is the background/related works of this paper?
• What is the methodology?
  – experiments, data collection, system design
• How does this paper measure success?
  – eval results, findings, outcome
• What are the take-aways
  – new design principles, new discovery, new tools/data available
(My) Reading Habits

• Quick skim (10 mins)
  – Abstract and Intro, some about the methodology and results

• Detailed reading (2 ~ 3+ hours)
  – Take note
  – Mark things out on PDF/printed paper
  – Write down questions and comments

• Write down the review (1 hour)
  – Write the draft
  – Refine the review
Example Review

• Example review for the USENIX Security 2013 paper (week-1)
• https://gangw.cs.illinois.edu/class/cs598/sample-review.txt

• Our weekly review assignments
  – You don’t need to write long reviews, but you can practice to write helpful reviews
Paper Sign up (Second Round)

• Current class size
  – 28 students

• Each student will sign up for a second paper slot to lead the discussion

• Using the same Piazza post

• Finalize the slot by Friday night (11:00 pm) this week
Project Ideas

• Something you are relatively familiar with + ML
  – Encourage high-risk project (need to be well-thought-out)

• Skim through the paper list to find some ideas

• Do you have data for it? Do you have a way to quantify success?

• Be ethical, be legal, talk to me if you are not sure
Example Ideas

- Deepfake detection (video, audio, pictures)
- ML explanation for security applications
- Building a chat bot as a phishing honeypot
- Fingerprinting code at the Internet scale
- Generating new malware variants
- Malware: how can I tell I am being tested in a VM or Container?
- Classifying “fake news”
- Using side-channels to infer sensitive information
ML for attack (password)

- Beyond Credential Stuffing: Password Similarity Models using Neural Networks SP 2019
Password Guessing

- **Online guessing**
  - Usually has a rate limit
  - Must guess it correctly within a few attempts

- **Offline guessing**
  - To crack the hash
  - Leaked pwd databases where pwds are stored in a hashed format
  - Inefficient if the password is also “salted”
Background: Server-side Password Storing

• Worse way: storing password in plaintext
  – e.g.: username1, password_plaintext1

• Slight better, but not secure enough
  – E.g., username1, hashed_password1

• The right way: add salt to hashing
  – Salt: a fixed length random string
  – E.g., username1, hashed(password1+salt1), salt1
Existing Guessing Method

- John the Ripper
  - Dictionary + rules (somewhat manually crafted by experts)
- Hashcat
  - Password + 2 digits (user habits)
- Markov Models
  - Given “iloveyo”, what is the most likely next character?
- PCFGs
  - Learning some rules from a training sets (guess common passwords)
Figure 1: An example of using a neural network to predict the next character of a password fragment. The network is being used to predict a ‘d’ given the context ‘ba’. This network uses four characters of context. The probabilities of each next character are the output of the network. Post processing on the network can infer probabilities of uppercase characters.
Model Details

• RNN (Recurrent neural networks)
  – Handle sequential data, model the relationships within sequences

• An Important Design Choice
  – Do not leave everything to RNN
  – simple heuristics are used outside of RNN to speed up the guessing
    o Uppercases or rare Symbols

• Contexts
  – 10 characters (shorter: not enough, longer: too costly)
  – Determined empirically
Model Details (Cont.)

• Model complexity
  – A big network of 15,700,675 parameters
  – A smaller network with 682,851 parameters

• Transfer learning
  – Learn a generic model (not optimal to a specific type of password)
  – Fine-tune the last layers with specific type of passwords → specific guessing model that is specialized for this type of password
Password Policies

• Given a password policy, the process can be further optimized
• A website typically requires a password to be
  – At least X digits long
  – Must contain at least one Y character (Cap, special, number, English char)

• 1 character class and 8 characters minimum
• 4 character classes and 8 characters minimum
• 1 character class and 16 characters minimum
• 3 character class and 12 characters minimum
Evaluation

• Approach: measure # guessed passwords
  – Simulating cracking hashes in the offline setting

• Training data:
  – leaked password sets
  – 105 million passwords and 5.9 million natural-language words

• Testing data
  – MTurk study passwords: 1class8, 4class8, 1class16, 3class12
  – Real passwords: 000webhost password leak
Evaluation Results

MinGuess, represents an idealized guessing approach in which a password is considered guessed as soon as it is guessed by any of all the tested algorithms.
Performance on “3class12” Passwords

(a) 3class12 passwords
Practical Application (for good)

• Give feedback on “password strength”
  – when a user creates a password on the web browser
Discussion Question

- What if the password dataset has “salt”?  
  – Would it affect the guessing algorithm? In what way?

- What if the same salt is used on all the passwords?
ML for attack (password)

- Beyond Credential Stuffing: Password Similarity Models using Neural Networks SP 2019
This Paper Did Something Different

• Learn password similarity model using 1.4 billion email-password pairs

• Online guessing: compromises 16% of accounts in less than 1K guesses, if one of their other passwords known to the attacker

• Intuition: user passwords share similarities
  – Due to reuse or naïve modifications
Defense

• Personalized password strength meters (PPSMs)

• Ask users to set different-looking passwords
  – Different from other users’ passwords
  – Different from the other passwords of this same user

• Well … users can create such passwords, but can users memorize them?
The Model

• Based on sequence-to-sequence (seq2seq) algorithms

• Given a leaked password $w \sim$, learn other similar passwords $w_i$ from the same user

• The proposed model: password-to-path (pass2path)
  – “path” denoting the sequence of transformations

• Given a leaked password $w \sim$, the generative model produces a list of passwords $w_1, w_2, \ldots, w_q$ in decreasing order of likelihood
Model Specifics

\[ P (w \mid \tilde{w}) = P (T_{\tilde{w} \rightarrow w} \mid \tilde{w}) = \prod_{i=1}^{t} P (\tau_i \mid \tilde{w}, \tau_1, \ldots, \tau_{i-1}) , \]

where \( t \) is the minimum edit distance between two passwords \( w \) and \( \tilde{w} \), and \( T_{\tilde{w} \rightarrow w} = \tau_1, \ldots, \tau_t \).

For example, the path from ‘cats’ to ‘kates’ (edit distance of 2) is:
- (sub, ‘k’, 0)
- (ins, ‘e’, 3)

\[
P (w \mid \tilde{w}) = \prod_{i=1}^{t} P (\tau_i \mid v_0, \tau_1, \ldots, \tau_{i-1}) \\
= \prod_{i=1}^{t} P (\tau_i \mid v_{i-1}, \tau_{i-1}) ,
\]

26
Results

• Leaked datasets (using email/username to connect the “sane user”)
  – 48% of users’ accounts compromised in less than 1000 guesses
  – 16% if there is a basic countermeasure

• Case study: work with Cornell University’s IT Security Office
  – Credential stuffing countermeasures and other defenses
    o Multi-factor, CAPTCHA, IP blocking, Security questions
  – 8.4% of 15,665 Cornell accounts that appeared in public breaches
Guessing Results

% of passwords cracked

# of guesses

Pass2path

Untargeted

6x

3x
Cornell Study

![Graph showing leaked accounts and vulnerability odds for different groups: alumni, staff, faculty, and other. The graph indicates a significantly higher percentage of leaked accounts among alumni compared to other groups.](image-url)
Discussion Questions

• Can you think of anything to improve “online guessing”?

• How to prevent credential stuffing (online testing leaked passwords on other sites)?