CERIAS

The Center for Education and Research in Information Assurance and Security

Distribution-Aware Password Throttling Mechanisms

Alina Nesen, Jeremiah Blocki
Department of Computer Science, Purdue University

Problem: User Frustration

- Large-scale **online password guessing attacks** are wide-spread and remain a continuous **threat** for users and businesses
- The common method for preventing account hacking is user **lockout** after a fixed number of incorrect login attempts.
- **But when should the account be locked?**
  - A *high* number of allowed incorrect login attempts is *not efficient* to prevent account hack
  - *Low* number of allowed incorrect logins *annoys* the users by locking them out and in this way making the system less user-friendly.

Keep Users Safe & Happy?

Approach: Introduce *HitCount* parameter for every user increased at incorrect login attempt:

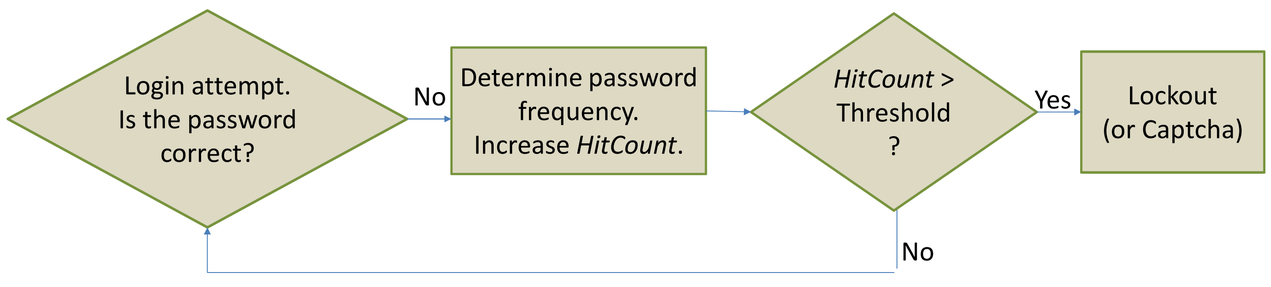
$HitCount_{User} = HitCount_{User} + Frequency[password]/N$:

Idea: Password frequency can show if it is the true user or an impostor who is trying to login.

Attackers pick popular passwords to increase their chances to hack an account

VS

Users forget passwords and/or make typos when trying to login



System Development

Challenge: Estimating password frequency

Approach 1. Using available dataset

- Use RockYou Dataset (32 mln pwds) to find frequency
- For password attempts not present in RockYou dataset (such as typos) set frequency = *1/2N*
- May not increase *HitCount* if pwd was attempted before

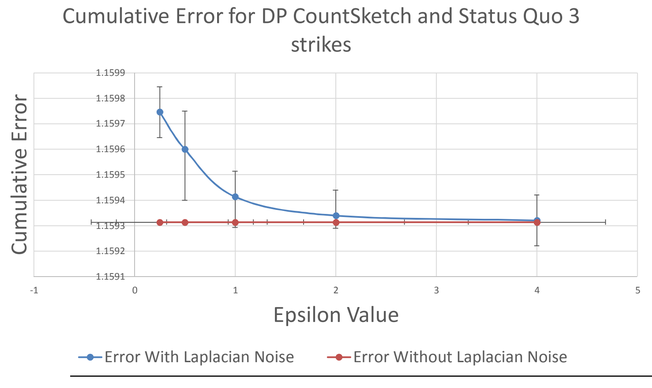
Rapid increase of *HitCount* means frequent passwords have been used for login attempts

Slow increase of *HitCount* shows that passwords used for login attempts were not frequent ones and could have been user's typos

Approach 2. Using CountSketch

- Can track evolution of popular passwords over time
- To decrease storage space, replace frequency table with **CountSketch** probabilistic data structure which allows to approximate the number of occurrences of a specific password in the dataset.
- To decrease security risks, enhance the Count Sketch data structure with differential privacy component by adding **Laplacian noise** to each cell: $Lap\left(\frac{d}{\varepsilon}\right)$, where *d* is the number of rows in the CountSketch.

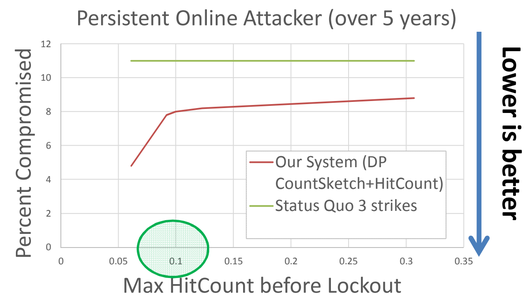
Preliminary Empirical Analysis

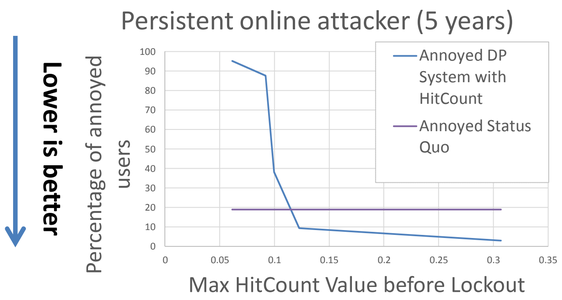


Comparison of the cumulative error for and with the different values of ε the differentially private CountSketch:

$$Error = \sum_{pwd \in RockYou} | f_{pwd} - \tilde{f}_{pwd} |$$

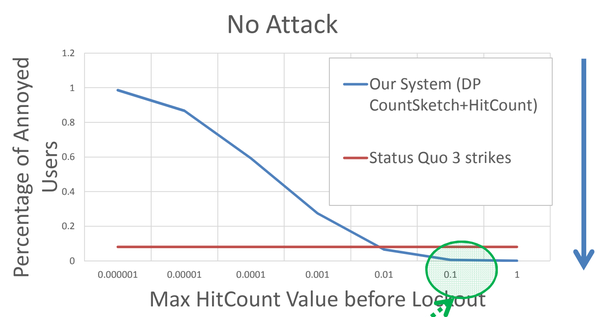
Comparison of the percentage of the compromised users in the systems *under attack* with soft lockout (lockout rate = 3) and differentially private system with CountSketch.





Comparison of the percentage of the annoyed users in the systems *under attack* with soft lockout (lockout rate = 3) and differentially private system with CountSketch.

Comparison of the percentage of annoyed users in the systems *without attack* with soft lockout (lockout rate =3) and differentially private system with CountSketch.



0% annoyed and fewer % compromised under attack!

References

1. Wang, Ding, et al. "Targeted online password guessing: An underestimated threat." *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016.
2. Charikar, Moses, Kevin Chen, and Martin Farach-Colton. "Finding frequent items in data streams." International Colloquium on Automata, Languages, and Programming. Springer, Berlin, Heidelberg, 2002.

PURDUE UNIVERSITY Discovery Park

CERIAS