Just In Time Hashing

Jeremiah Blocki (jblocki@purdue.edu) and Benjamin Harsha (bharsha@purdue.edu)

Key Problem
- Users are impatient and do not want to wait to log in
- The time users are willing to wait is not sufficient to provide protection against offline attacks

Just In Time with Memory Hardness
- Adversary gains an advantage if they can store previous attempts
- Using memory hard functions helps make storing all previous attempts infeasible

User Study: How much time do we have for key stretching?
- How long do users actually take to type their passwords?
- Is muscle memory involved? Is it consistent?
- How do they correct their typos? How often do they make typos?
- Precautions: Third party code review, only collecting timing and error correction data

The Just In Time Method
- Key idea: It takes users a second or two to type their password
- Start hashing when they start typing instead of when they stop
- Instead of storing the usual \( H(salt, pwd) \), store \( H(cn, H(cn-1, \ldots H(c1, salt) \ldots) \)

Security analysis
- Is there any advantage to an adversary facing a Just In Time hashed password in an offline attack?
- The adversary can use previous password attempts to speed up future attempts
- People pick passwords that are similar to each other!

Implementation
- Simple implementations exist, just use output of hash function and continue character by character
- Working it into the function itself helps, especially with memory hard functions
- Must deal with users making errors and correcting them
- Implementation based on Argon2, winner of Password Hashing Competition in 2015 [1,2]

Next Steps
- Finish modifications to Argon2
- Complete user study and analyze results
- Complete write up

References & Acknowledgements
2. https://password-hashing.net/

This project was funded by Intel through a CERIAS Research Assistantship

Fig 1: Several common passwords that share paths for JIT hashing