PolyPasswordHasher: Protecting Passwords in The Event of A Password File Disclosure

Prof. Justin Cappos, Santiago Torres-Arias, Shuyuan Luo
Tandon School of Engineering
New York University

Abstract

Over the years, we have witnessed various password-hash database breaches that have affected small and large companies, with a diversity of users and budgets. The industry standard, salted hashing (and even key stretching), has proven to be insufficient protection against attackers who now have access to clusters of GPU-powered password crackers. Although there are various proposals for better securing password storage, most do not offer the same adoption model (software-only, server-side) as salted hashing, which may impede adoption.

PolyPasswordHasher is a software-only, server-side password storage mechanism that requires minimal additional work for the server, but exponentially increases the attacker's effort. PolyPasswordHasher uses a threshold cryptosystem to interrelate stored password data so that passwords cannot be individually cracked. Our analysis shows that PolyPasswordHasher is memory and storage efficient, hard to crack, and easy to implement. In many realistic scenarios, cracking a PolyPasswordHasher-enabled database would be infeasible even for an adversary with millions of computers

RSA Conference Security Scholar

PolyPasswordHasher-For a Better Password Database

Prof. Justin Cappos (Santiago Torres-Arias & Shuyuan Luo)
New York University

Problem Statement and Goals

How do the server protect our passwords when storing them?

1.Password hashes use a one-way function.

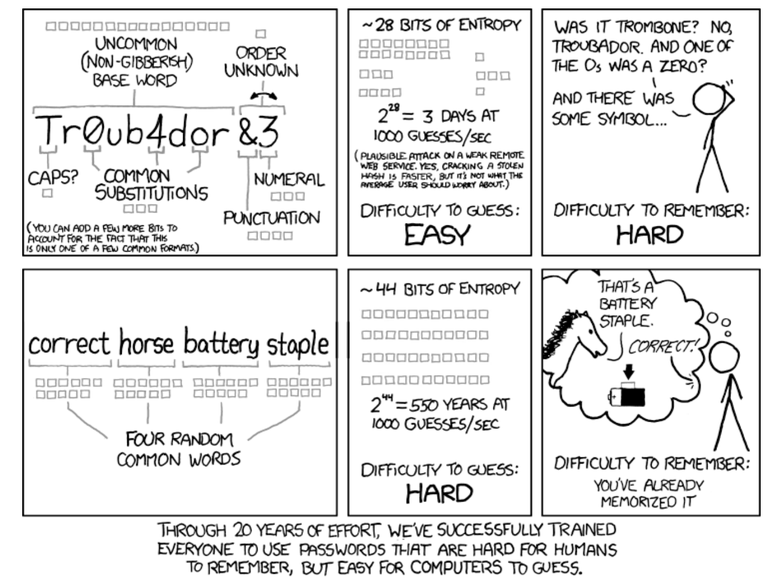
2.Encrypted database key management obstructs personnel rotation.

3.Use of hardware to store or verify passwords.

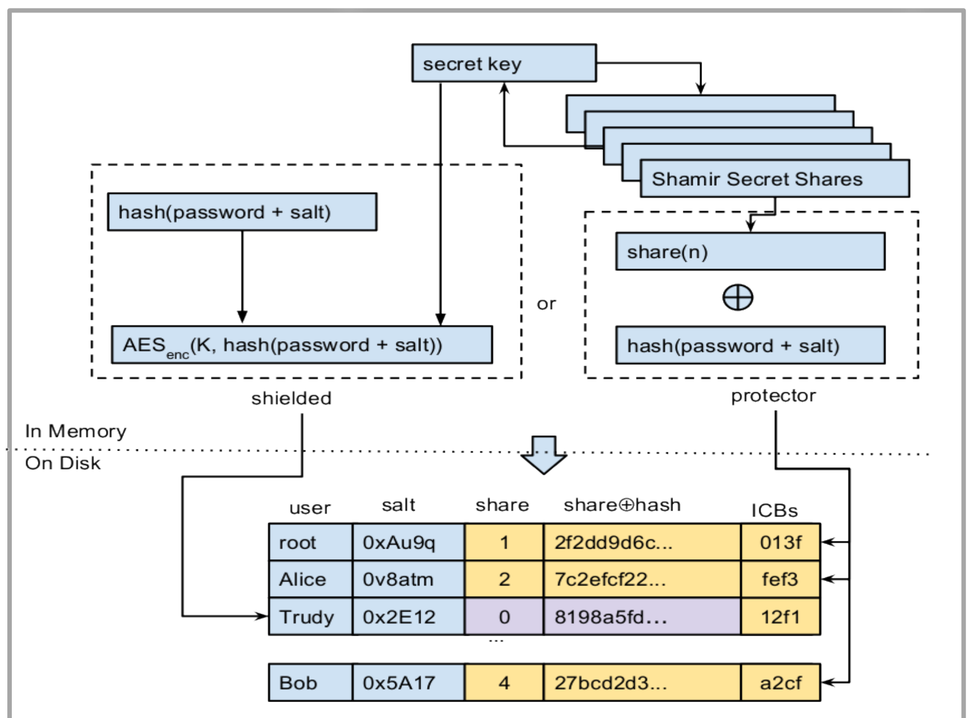
4.Client changes to the authentication mechanism.

Password database disclosures have caused companies billions of dollars in damages. The current industry security standard, salted hashing, has proven to be insufficient protection. Other alternatives either cost too much memory, money, requires hardware or user-end changes.

Weak passwords are the most vulnerable, hackers can crack those in million seconds!



Approach



How do PolyPasswordHasher protects password database?

Key Idea: Make passwords protect each other by using Shamir Secret Share Theorem!

- Each salted password hash is combined with a piece of the Secret key (like a dot in a line)
- The Secret key is revealed to the server only when a certain amount of protector accounts provide the correct passwords. (they are like the dots which can form the line)
- The server then hides the Secret key and use it to authenticate users
- The attacker needs to guess many trustworthy and hard passwords at the same time to recover the Secret key. Only with the Secret key, they can start regular salt and hash hacking

Results

With the same password database, the research and test shows that PPH is more secure.

With Regular salted hashes:

The password database can be cracked in less than an hour with an average laptop.

With PolyPasswordHasher (k=3):

The password database can be cracked around the age of the universe with every available (900M) computer. It's much harder because the hacker needs to get multiple hard passwords correct before starting salt and hash hacking.

Moreover, PolyPasswordHasher is easy to implement, software-only, server-change-only, and memory efficient.

Ongoing Work:

Research initiatives in 2016 summer have focused on providing easy to integrate libraries for different applications, including but not limited to a PPH Pluggable Authentication Module (PAM).

In many realistic scenarios, cracking a PolyPasswordHasher-enabled database would be infeasible even for an adversary with millions of computers.