

Authentication and Access Control

This lecture includes materials taken from “What Every Web Programmer Needs To Know About Security”, by Arkajit Dey and Neil Daswani. The content of this presentation is licensed under the Creative Commons 3.0 License.



Password Security

- Password systems ubiquitous, vulnerable
- Early password security studies (1979) - Morris, Thompson: 86% of passwords can be cracked
- Threats: Online & Offline Dictionary Attacks
- Solutions: Hashing & Salting

A Strawman Proposal

- Basic password system: file w/ username, password records (colon delimiter)

```
john:automobile  
mary:balloon  
joe:wepntkas
```

- Simple to implement, but risky
 - All users compromised if hacker gets the password file

Hashing

- Encrypt passwords, don't store "in the clear"
 - Could decrypt (e.g. DES) to check, key storage?
 - Even better: "one-way encryption", no way to decrypt
 - If file stolen, passwords not compromised
 - Use one-way hash function, h : preimage resistant
 - Ex: SHA-1 hashes stored in file, not plaintext passwd

```
john:9Mfsk4EQh+XD2lBcCAvputrIuVbWKqbxPgKla7u67oo=  
mary:AEd62KRDHUXW6tp+XazwhTLSUlADWXrinUPbxQEfnSI=  
joe:J3mhF7Mv4pnfjcnOHz1ZrUELjSBJFOolr6D6fx8tfwU=
```

Hashing Example



“What is your username & password?”

My name is john. My password is automobile. Does
h(automobile)
=
9Mfsk4EQ...
???

■ Hash: “One-way encryption”

- ☐ No need to (can't) decrypt
- ☐ Just compare hashes
- ☐ Plaintext password not in file, not “in the clear”

Dictionary Attacks

Attacker Obtains Password File:

joe	9Mfsk4EQ...
mary	AEd62KRD...
john	J3mhF7Mv...

- *Offline*: attacker steals file and tries combos
- *Online*: try combos against live system

mary has
password
balloon!

Attacker computes possible password hashes (using words from dictionary)

$h(\text{automobile})$	$= 9\text{Mfsk4EQ}...$
$h(\text{aardvark})$	$= \text{z5wcuJWE}...$
$h(\text{balloon})$	$= \text{AEd62KRD}...$
$h(\text{doughnut})$	$= \text{tvj/d6R4}$

Attacker



Salting

- *Salting* – include additional info in hash
- Add third field to file storing random # (*salt*)
- Example Entry: john with password automobile
john:ScF5GDhWeHr2q5m7mSDuGPVasV2NHZ4kuu5n5eyuMbo=:1515
- Hash of password concatenated with salt:
 $h(\text{automobile}|1515) = \text{ScF5GDhW} \dots$

Off-line Dictionary Attack Foiled!



h(automobile2975) = KNVXKOHDBDEBKOURX
h(automobile1487) = ZNBXLPOEWNVDEJOG
h(automobile2764) = ZMCXOSJNFKOFJHKDF
h(automobile4012) = DJKOINSLOKDKOLJUS
h(automobile3912) = CNVIUDONSOUIEPQN
...Etc...
h(aardvark2975) = DKOUOXKOUJWIOIQ
h(aardvark1487) = PODNJUIHDJSHYEJNU
...Etc...


/etc/passwd:

john	LPINSFRABXJYWONF	2975
mary	DOIIDBQBZIDRWNGK	1487
joe	LDHNSUNELDUALKDY	2764

**Too many
combinations!!!
Attack is
Foiled!**

Salting: Bad News

- What bad-guy can read password file?
 - Note that many attacks are committed by insiders.
- Ineffective against chosen-victim attack
 - Attacker wants to compromise particular account
 - Just hash dictionary words with victim's salt
- Attacker's job harder, not impossible
 - Easy for attacker to compute $2^k n$ hashes?
 - Then offline dictionary attack still a threat.



Additional Password Security Techniques

- Several other techniques to help securely manage passwords: Mix and match ones that make sense for particular app
 - Strong Passwords
 - “Honeypots”
 - Filtering
 - Aging
 - Captchas
 - Limiting Logins
 - Artificial Delays
 - Last Login
 - Image Authentication
 - One-Time Passwords

Strong Passwords

- Not concatenation of 1 or more dictionary words
- Long as possible: letters, numbers, special chars
- Can create from long phrases:
 - Ex: "Nothing is really work unless you would rather be doing something else" -> n!rWuUwrbds3
 - Use 1st letter of each word, transform some chars into visually or phonetically similar ones

“Honeypot” Passwords

- Simple username/password (guest/guest) combos as “honey” to attract attackers
- Bait attackers into trying simple combos
- Alert admin when “booby-trap” triggered
- Could be indication of attack
- ID the IP and track to see what they’re up to



Password Filtering

- Let user choose password
 - Within certain restrictions to guarantee stronger password
 - Ex: if in the dictionary or easy to guess
- May require mixed case, numbers, special chars
 - Can specify set of secure passwords through regular expressions
 - Also set a particular min length

Aging Passwords

- Encourage/require users to change passwords every so often
 - Every time user enters password, potential for attacker to eavesdrop
 - Changing frequently makes any compromised password of limited-time use to attacker
- Could “age” passwords by only accepting it a certain number of times
- But if require change too often, then users will workaround, more insecure

Limited Login Attempts

- Allow just 3-4 logins, then disable or lock account
 - Attacker only gets fixed number of guesses
 - Inconvenient to users if they're forgetful
 - Legitimate user would have to ask sys admin to unlock or reset their password
 - Potential for DoS attacks if usernames compromised and attacker guesses randomly for all, locking up large percentage of users of system

Artificial Delays

- Artificial delay when user tries login over network
- Wait 2^n seconds after n th failure from particular IP address
 - Only minor inconvenience to users (it should only take them a couple of tries, 10 seconds delay at most)
 - But makes attacker's guesses more costly, decreases number of guesses they can try in fixed time interval
- HTTP Proxies can be problematic
 - One user mistyping password may delay another user
 - Need more sophisticated way to delay



Last Login

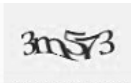
- Notify user of last login date, time, location each time they login
 - Educate them to pay attention
 - Tell user to report any inconsistencies
- Discrepancies = indications of attacks
- Catch attacks that may not have been noticed
 - Ex: Alice usually logs in monthly from CA
 - Last login was 2 weeks ago in Russia
 - Alice knows something's wrong, reports it

Image Authentication

- Combat phishing: images as second-factor
- Ask users to pick image during account creation
 - Display at login after username is entered
 - Phisher can't spoof the image
 - Educate user to not enter password if he doesn't see the image he picked
- Recently deployed by PassMark, used on `www.bofa.com` and other financial institutions

Captchas

- Challenge-response test used to ensure that the response is not generated by a computer.



One-Time Passwords

- Multiple uses of password gives attacker multiple opportunities to steal it
- OTP: login in with different password each time
- Devices generate passwords to be used each time user logs in
 - Device uses seed to generate stream of passwords
 - Server knows seed, current time, can verify password
- OTP devices integrated into PDAs, cell-phones

Summary

- Hashing passwords: don't store in clear
- Dictionary Attacks: try hashes of common words
- Salting: add a random #, then hash
 - Dictionary attack harder against arbitrary user
 - But doesn't help attack against particular victim
- Other Approaches:
 - Image Authentication
 - One-time Passwords
 - ...