Řízení bezpečnosti

PVA4 Programování a vývoj aplikací

Adam Fišer

Obsah

Úvod

- Autentizace
- Autorizace
- Session
- Přihlášení uživatele

Autentizace

- Autentizace – ověření identity subjektu neboli zjištění, zda-li daný subjekt je ten, za který se vydává.

  Subjektem může být osoba, systém, zpráva
- Je bezpečnostní opatření zajišťující ochranu před falešnou identikou.

Autentizace

Identita

Prokázaní identity lze rozdělit dle toho:

- Co subjekt má – identifikační karta, platební karta, klíč, telefon
- Co subjekt zná – PIN, heslo, fráze, klíče
- Čím subjekt je – biometrické údaje (otisk prstu, sítnice)

Faktory autentizace

- Jednofaktorová – ověření pouze dle jednoho způsobu. Typicky uživatelské jméno a heslo
- Dvoufaktorová – k ověření jsou použity dva prvky. Typicky co člověk má a zná např. mobilní telefon a SMS nebo token
- Třífaktorová – přidává oblast spojenou s osobou – biometrii.

> Čím vyšší bezpečnost služby vyžadují, tím je nutné použít silnější autentizaci.

Kde používáte dvou/tří-faktorovou autentizaci?

Autorizace

- Autorizace – oprávnění k provedení určité akce nebo k přístupu k určitému zdroji.
- Zajišťuje, že uživatel může provádět pouze ty akce, ke kterým má oprávnění.
- Autorizace je založena na autentizaci, kdy se ověří identita uživatele a na základě toho se mu přidělí oprávnění.
- Autorizace je nedílnou součástí řízení přístupových oprávnění tzv. identity management k datům, souborům, adresářům, operacím atd.

Přístupová práva

- Přístupová práva znamenají úroveň oprávnění zaměstnance nebo uživatele systému k něčemu přistoupit, něco využívat. Přístupová práva se udělují obvykle na základě oprávnění, svolení nebo svěřených pravomocí.
- Přístupová práva se mohou týkat různých oblastí, např. k datům, souborům, adresářům, operacím, službám, aplikacím, zařízením, prostředkům, informacím, údajům, zdrojům, systémům, sítím, službám, funkcím, úkolům, činnostem, procesům, úkonům, akcím, činnostem, postupům, metodám, technologiím, nástrojům

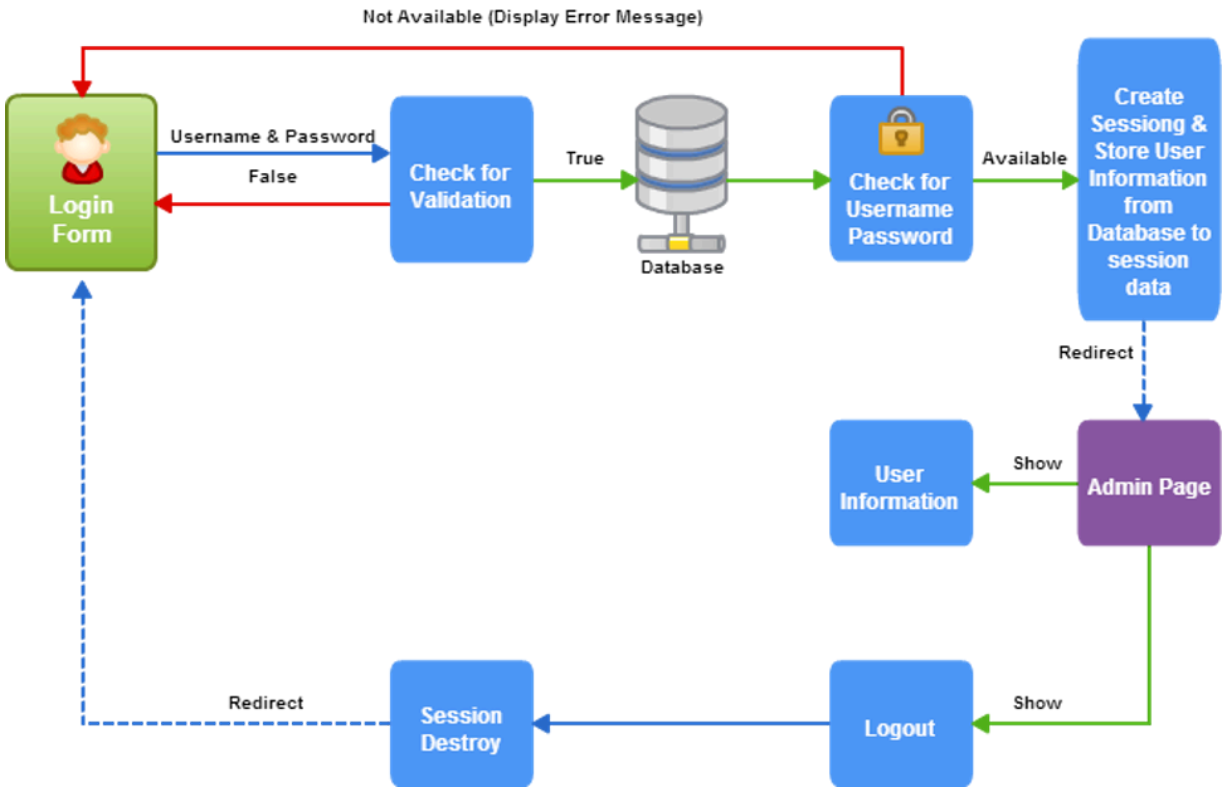
Authorization management

- Řízení oprávnění se zabývá přístupem osob k aktivům (různým objektům), nejčastěji k datům.
- Cílem je ochrana aktiv před neoprávěným užitím nebo vstupem.
- Díky řízení oprávnění tedy organizace chrání své zdroje.
- Řízení oprávnění se týká konkrétních osob, protože se vždy vztahuje ke konkrétnímu člověku. Může ale zahrnovat i oprávnění systémů (které zprostředkovaně umožňují autentizaci lidí - uživatelů). Příkladem je přihlašování uživatelů přes takzvané single-sign-on.

Navrhněte login proces pro webovou aplikaci

Adam Fišer

Proces přihlášení



Zdroj: 29.1.2022 https://www.pngkey.com/maxpic/u2e6w7r5o0u2w7t4/

Hesla

- Vždy předpokládejte, že se k heslům může dostat útočník.
- Hesla nikdy neukládáme jako plain-text (čistý text)
- Zabezpečení hesel:
  - Šifrováním – nutnost použití šifrovacího klíče, který v případě úniku umí dešifrovat hesla – riziková operace a v praxi se nepoužívá
  - Hash a jeho odvozeniny

Hashování hesel

- Hashování je proces, kdy se heslo převede na hashovou hodnotu.
- Hashování je jednosměrný proces, tzn. z hashové hodnoty nelze zpětně získat heslo.
- Hashovací funkce jsou jednosměrné matematické algoritmy používané k mapování dat jakékoli velikosti na bitový řetězec pevné velikosti.
- Kryptografické hašovací funkce se široce používají v praktikách zabezpečení informací, jako jsou digitální podpisy, kódy pro ověřování zpráv a jiné formy ověřování.

Základní hash

- Stejná zpráva vždy vede ke stejné hashovací hodnotě (tj. funkce je nedeterministická). - **POZOR: Zásadní bezpečnostní problém!**
- Hodnota hash se vypočítá rychle.
- Je nemožné mít dvě zprávy se stejnou hodnotou hash (známé jako „kolize“).
- Je nemožné úmyslně vytvořit zprávu, která poskytne danou hodnotu hash.
- Mírné změny ve zprávě by měly výslednou hodnotu hash značně změnit, aby se zdálo, že nesouvisí s původním hashem
- např. SHA-256, SHA-3, MD5, SHA-1

Nevýhody základního hash

- Tato metoda má však jednu nevýhodu, uživatelé, kteří si zadají stejné heslo, mají i stejné hashe.
- Uživatel, který si zadá stejné heslo na různých systémech, má pak na různých systémech stejný hash.
- Pokud je použito slabé heslo, lze takové heslo velice rychle prolomit za pomocí předvypočtených hashů (precomputed hashes) uložených v rainbow tables.

Solený hash

- Solený hash je metoda, která řeší problém s kolizemi a předvypočtenými hashi.
- K uživatelskému heslu se přidá unikátní řetězec tzv. sůl (salt). Každý uživatel má jiný řetězec.
- Použitím Salt Hash znemožníme útok za použití předvypočtených hashů
- Pokud si dva uživatelé zadají stejné heslo, budou mít v systému uloženy dva zcela odlišné hashe.

Jak se tvoří solený hash

- Heslo uživatele se spojí s náhodným řetězcem (sůl) a výsledný řetězec se zahashuje.
- Heslo zadané uživatelem a sůl vygenerovaná systémem se spojí do jednoho řetězce a až pro ten spočte hash.
- Sůl je možné vložit před heslo (prefix), za heslo (postfix) případně doprostřed či kombinace.
- Solí by měl být dostatečně náhodný řetězec znaků, o určité délce.
- Sůl není tajná informace a tudíž není potřeba ji chránit.
- Útočník by musel pro úspěšné lámání hesel zjistit, jak přesně algoritmus se solí zachází. To útočník může zjistit, ale také nemusí, záleží na tom, zda může provést analýzu kódu
- Hashovací funkce MD5 a SHA nejsou bezpečné!
- Vyvarujte se opakování hashování. Paradoxně čím častěji ji spouštíte, nezvyšujete bezpečnost.

Uložení dat

- Pro přihlášení uživatele budete mít v databázi uloženo:
  - Uživatelské jméno
  - Výsledek hashovací funkce
  - Počet cyklů (míra složitosti)
  - Samotná sůl

Login

- Přihlášení uživatele je proces, kdy uživatel zadá své uživatelské jméno a heslo.
- Systém si vyhledá, jaká sůl byla použita, k heslu ji přičte, spočte hash a výsledek porovná.
- Pokud se vypočtený hash bude shodovat s hashem uloženým v systému, uživatel zadal heslo správně, v opačném případě bylo heslo zadáno chybně

Hashování v praxi

```php
$password = 'tajne_heslo'; // Heslo uživatele, typicky z formuláře při registraci
```

```php
// Funkce pro osolení hesla
// Tajný algoritmus, jak zabezpečit heslo uživatele
function osoleneHeslo(string $password): string {
    $salt = 'urWfWwT9KAXuAUH*^w7sH&9@g';
    return $salt . $password . chunk_split($salt, 4 ,".");
}
```

...

```php
// Vygeneruje bezpečný hash s vyšší složitostí (výchozí je 10):
// Do funkce předáváme osolené heslo a nastavíme algoritmus na BCRYPT a složitost na 12
echo password_hash( osoleneHeslo($password), PASSWORD_BCRYPT, ['cost' => 12]);
// ukázka výstupu: $2y$12$tq.y6I1biT8I91a8hreuQeEmGXNtp2dsLuKqDdLu8sobhXHhqOpHG
```

Hashování v praxi

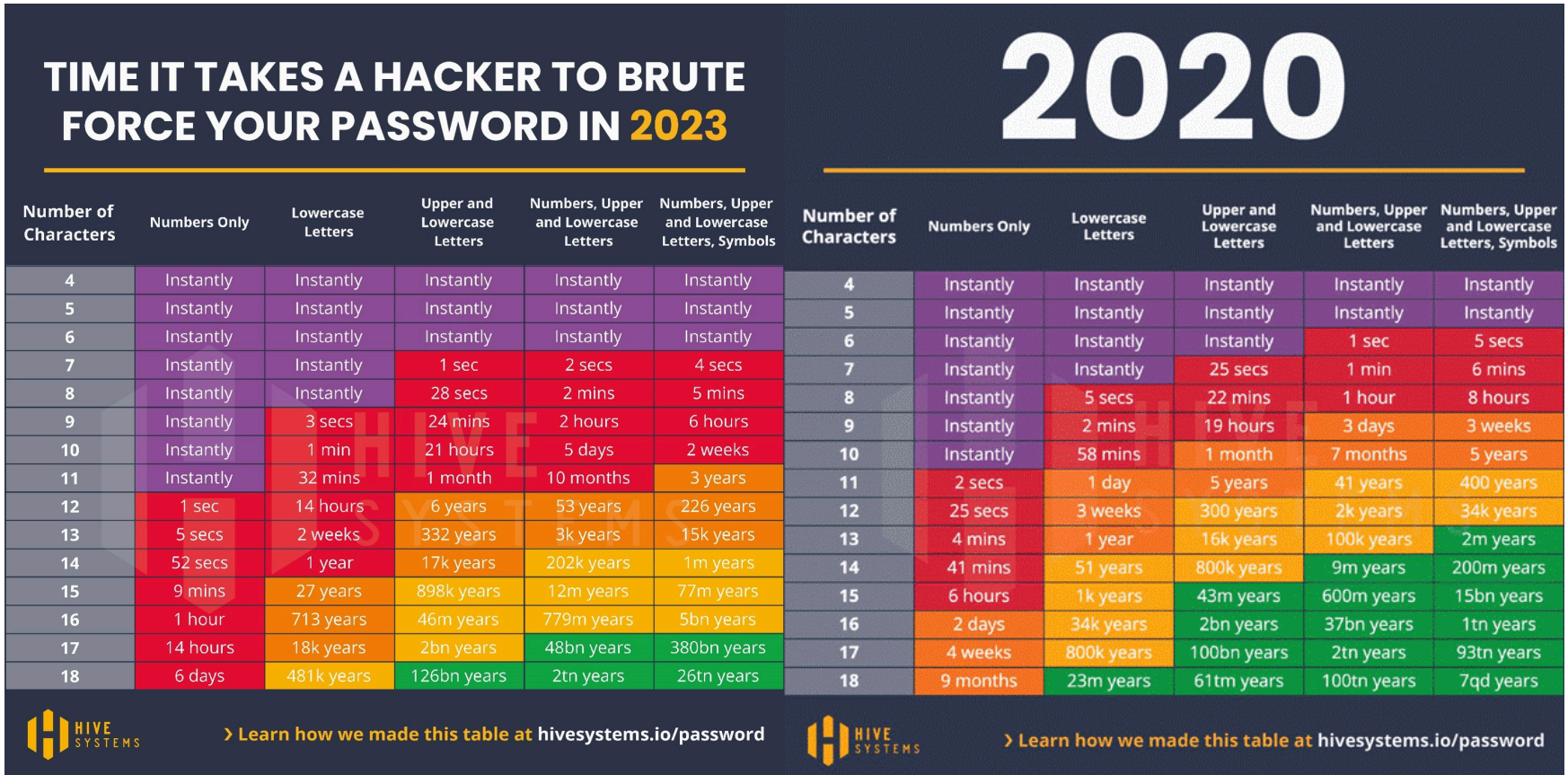
```php
// Hash hesla uživatele načteme standardně z databáze.
// Pro ukázku hash nyní vygenerujeme stejným algoritmem jako v předchozím slajdu
$ulozenyHash = password_hash($saltedPassFromForm, PASSWORD_BCRYPT, ['cost' => 12]);
```

```php
...
// Krok 2: ověření hesla
// Zkontrolujeme heslo zadané uživatelem ve formuláři s uloženým heslem

// Uživatelské heslo z formuláře osolíme stejným mechanismem jako při registraci
$saltedPassFromForm = osoleneHeslo('tajne_heslo');

// Funkcí password_verifi ověříme shodu osoleného hesla s získaným hashem původního hesla
if (password_verify($saltedPassFromForm, $ulozenyHash)) {
    echo 'Heslo je platné';
} else {
    echo 'Neplatné heslo.';
}
```

Heslo



TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD IN 2023

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	1 sec	2 secs	4 secs
8	Instantly	Instantly	28 secs	2 mins	5 mins
9	Instantly	3 secs	24 mins	2 hours	6 hours
10	Instantly	1 min	21 hours	5 days	2 weeks
11	Instantly	32 mins	1 month	10 months	3 years
12	1 sec	14 hours	6 years	53 years	226 years
13	5 secs	2 weeks	332 years	3k years	15k years
14	52 secs	1 year	17k years	202k years	1m years
15	9 mins	27 years	898k years	12m years	77m years
16	1 hour	713 years	46m years	779m years	5bn years
17	14 hours	18k years	2bn years	48bn years	380bn years
18	6 days	481k years	126bn years	2tn years	26tn years

HIVE SYSTEMS

› Learn how we made this table at hivesystems.io/password

2020

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	61tm years	100tn years	7qd years

HIVE SYSTEMS

› Learn how we made this table at hivesystems.io/password

Password hygiene

Consider passphrases

Require unique passwords

Employ password managers

Review cycle frequency

Use MFA every-where possible

The Consumer Authentication Strength Maturity Model (CASMM) V6

Správci hesel

Nejlepší heslo je žádné heslo, ale pokud už musíte mít heslo, tak alespoň spravujte hesla správně.

- 1password pro studenty zdarma
  - Podmínka je mít aktivní ověřený github student a ten už máme
  - https://blog.1password.com/github-student-developer-pack/
- Bitwarden
- Dashlane
- Keepasxc

Vyvarovat se:

- Ukládání hesel v prohlížeči
- Lastpass – správce hesel s mnoha úniky
- Používání stejného hesla na více místech

Děkuji za pozornost

Otázky?

Repository / Prezentace