Viac ťahákov nájdeš na

**bart.sk/tahaky**

## Databáza _

» **Skontroluj si fungovanie zálohovania.** Nastav čo najnižší interval, nikdy nevieš, kedy sa niečo pokazí doslova z minúty na minútu. A skontroluj aj to, či vieš zálohu bez problémov obnoviť. Bude ti na nič, ak ju nedokážeš nasadiť späť.

» **Je tvoj projekt v súlade s aktuálnym GDPR?** Pozri si pre istotu aktuálny zákon o ochrane osobných údajov (zvykne sa meniť) a uprav, čo treba, ešte teraz. Pomôže ti to vyhnúť sa mastnej pokute.

» **Ako sú na tom používateľské heslá? Zahashované?** Použi napríklad bezpečný algoritmus bcrypt.

» **Uchovávaš niekde citlivé údaje v celom znení (emaily, adresy...), napríklad kvôli newsletteru?** Ak ich už musíš mať, vytvor si ešte aj zoznam všetkých miest, kde sa nachádzajú (mailchimp databáza, office dokument, šanón v druhej zásuvke). Je to užitočné v prípade útokov aj úradných kontrol.

» **Zabráň SQL-injection.** Skontroluj si použitie správnych príkazov. Napríklad ak používaš NPM, nedávaj tam npm-mysql, ale nasaď npm-mysql2, ktorý podporuje “prepared statements”.

## APIs _

» **Monitoruješ nezvyčajnú aktivitu?** Nastav si notifikácie, ktoré ťa upozornia na správanie, ktoré by mohlo indikovať útok.

» **Skontroluj prístupy.** Uisti sa, že používatelia sú plne autentifikovaní a správne autorizovaní pre používanie API.

» **Aktualizuj.** Daj pozor na aktuálny software, staré protokoly zabezpečenia vymeň za nové hneď, ako je to možné, a ak to nemáš automatizované, do kalendára si daj reminder na včasnú obnovu platnosti digitálnych certifikátov.

» **Uisti sa, že ti útoky DOS na API neodstavia web.** Na dlhšie trvajúcich requestoch a autentifikácii na API použi rate limiter alebo load balancer.

» **Používaj overené a aktualizované balíčky.** Pre istotu ich môžeš pred nasadením preveriť v node napr. cez npm audit. Odhalí všetky zraniteľnosti.

» **Zaisti, aby neboli na verejných rozhraniach API vymenované žiadne zdroje.** V prípade identifikátorov nepoužívaj čísla, ale UUID (Universally Unique Identifier) v súlade s RFC 4122.

## Cloud a Server _

» **Máš zamknuté?** Použi SSL alebo novšiu TLS certifikáciu (zadarmo sa to dá napr. cez Let’s Encrypt) a implementuj HTTPS. Zároveň nezabudni presmerovať všetky HTTP na HTTPS.

» **Zavri porty!** Nechaj otvorený len minimálny počet portov a používaj tie neštandardné, nech sa pri prípadnom útoku hackeri aspoň poriadne potrápia.

» **Nedevelopuješ sám? Tak chráň svoj projekt!** Pre kolegov developerov používaj roly podľa ich pracovného zamerania a nie rootovské poverenia, zabezpeč prístupy cez ich IP adresy a donúť ich svoje heslá pravidelne striedať.

» **Aktualizuj rýchlo a bez výpadkov.** Priprav spôsob, ako aktualizovať plne automatizovaným spôsobom.

» **Najbezpečnejší server je ten vypnutý.** Mimo svojich pracovných hodín daj dev serverom oddýchnuť.

## Tipy na záver _

» **Vytvor a publikuj súbor, v ktorom nájdu používatelia informáciu o kanáloch určených na hlásenie bezpečnostných problémov.** Dobrý príklad nájdeš na:

```
google.com/.well-known/security.txt
```

» **Vytvor si cvičný plán bezpečnostných incidentov.** Nachystaj si dokument, kde v krokoch popíšeš, čo treba urobiť v prípade jednotlivých bezpečnostných hrozieb. Pomôže ti to neprepadnúť panike a nezabudnúť na nič podstatné.

» **Viac tipov a zdroje nájdeš na:**

```
bart.sk/bezpecnost
```

# Security checklist

bart

bart.sk/tahaky

## Pre-development

### Výber softvéru (programu, knižnice, aplikácie) _

» **Používaš na vývoj správny softvér?** Mal by byť jednoduchý, spoľahlivý, efektívny (rýchly a s malou spotrebou RAM), mal by mať pravidelné aktualizácie a nevyžadovať na svoj chod ďalšie programy. Ako ho spoznáš?

- **Je obsiahnutý v inštaláciách distribúcií ako CentOS, Fedora, Debian a pod.** Takéto balíčky používa veľa ľudí a ich bezpečnosť je preverená ďalšími vývojármi - odborníkmi.
- **Má webovú stránku** s možnosťou hlásenia chýb a archív stiahnutí starších verzií.
- **Je dostupná jeho EOL verzia s dlhodobou podporou.**
- **Má aktívny vývoj a viac vývojárov.** Skontrolovať si to môžeš napr. podľa posledných commitov.
- **Nemá priveľa otvorených chýb bez riešenia.**
- **Je open source alebo má otvorenú licenciu.** Takto je väčšia pravdepodobnosť, že ho firma odrazu nezruší. Pri closed source aplikácii sa môže stať, že sa nenájde nový vývojár, ktorý by ďalej riešil jej údržbu, a tým pádom appka zaniká.
- **Nemá závislosti.** Funguje samostatne, bez potreby doinštalovania ďalších programov na jej prevádzku. Ak tieto závislosti existujú, neinštaluj ich cez pip/npm, ale radšej požiadaj o inštaláciu z distribúcie. Pozor, každú ručne doinštalovanú knižnicu treba pravidelne aktualizovať a sledovať, či k nej nepribudli ďalšie závislosti.

## Post-development

### Poverenia, dáta, autentifikácia _

» **Máš všetky tajné údaje (napr. prístupy do databázy) uložené bezpečne?** Nikdy ich nenechávaj v GIT-e ani napevno v kóde. Radšej využi .env súbory alebo CI/CD variables.

» **Na ukladanie svojich hesiel používaj password manager**, ako 1Password alebo Keychain na iOS.

» **Skontroluj si pravidlá pre tvorbu hesla aj na strane používateľa.** Mal by si od neho chcieť aspoň 8 znakov, ideálne s číslom aj špeciálnym symbolom.

» **Zníž privilégiá!** Aplikácie a kontajnery spúšťaj vždy s tými minimálnymi, nikdy nie ako root. Pozor, Docker spúšťa appky ako root defaultne.

### <> Aplikácie _

» **Vyhni sa priamej manipulácii s DOM, stávaš sa tým terčom pre XSS útok:**

- Nepoužívaj `innerHTML` - ak chceš upravovať obsah HTML, použi radšej `textContent`, `innerText` alebo `setHTML()`.
- Keď je pre teba dôležité zobraziť content ako HTML kód, predtým, ako upravíš DOM, poriadne vstup verifikuj a to napríklad pomocou opensource knižnice DOMPurify a jej funkcie `sanitize()`.
- Ak máš nasadený React, vyhni sa bezhlavému používaniu property `dangerouslySetInnerHtml`.

» **Veríš vstupom používateľa (komentáre, príspevky)? Radšej stav na opatrnosť a formuláre ošetri:**

- Dáta kontrolované používateľom zakóduj - zabrániš tým exekúcii škodlivých kódov HTML parserom pri zobrazení v aplikácii. Ak používaš jeden z pokročilejších frameworkov, ako React, máš čiastočne vyhrané - tie už disponujú automatickým escapovaním a kódovaním.
- Nezabudni na ochranu proti botom a všetky formuláre ošetri reCaptchou.
- Pred odoslaním používateľských vstupov na server dáta prečisti od potenciálnych škodlivých častí kódu (je to napr. aj ochrana proti SQL Injection). Najjednoduchší spôsob, ako to docieliť, je opäť pomocou `DOMPurify.sanitize()`.

» **Skontroluj si aj chybové hlášky.** Na produkcii z nich vyhoď detaily, ktoré môžu hackeri zneužiť, a vypni tu aj debug. Namiesto toho nastav detailný zápis do logov pre správnu diagnostiku. Osobné informácie o používateľoch odtiaľ ale radšej vynechaj.

» **Maj správne cookies.** Ak obsahujú citlivé dáta, musia byť nastavené ako `httpOnly`.

» **Nastav si striktnú Content Security Policy (CSP).** Nikdy never všetkému, čo odošle server - nastav CSP hlavičku a striktne kontroluj zdroje, ktoré môže tvoja appka načítať.

» **Chráň v hlavičkách.** V odpovediach klientov používaj hlavičku X-XSS-Protection, prípadne to pokry striktne definovanou CSP so zákazom využívania inline JavaScriptu. Tiež nezabudni na použitie CSRF tokenov.

» **Nepoužívaj GET s citlivými údajmi alebo tokenmi v URL**, pretože sa logujú na serveri a proxy.

» **Definuj, ktoré externé súbory je možné na stránku doťahovať a odkiaľ.** Pokiaľ na to nemáš nástroj, využi štandardné pravidla zo `same-origin policy`.

» **Tip:** Svoj web môžeš preskenovať cez:

```
observatory.mozilla.org
```

Stránka ťa oznámkuje a ponúkne ti bezpečnostný report, ktorý ti pomôže projekt poriadne zabezpečiť.