Security

Checklist na zabezpečení webových aplikací

https://blog.horejsek.com/bezpecnost/

V poslední době jsem častokrát řešil bezpečnost webů a vzpomínal, cože vše to musím mít v cajku. Tak jsem se rozhodl si sepsat takový check-list. A možná bude užitečný i někomu jinému. :-)

Základem je HTTPS

Bez toho asi nemá smysl ani řešit nic jiného. Takže to je must-have první věc, kterou řešit, pokud chcete web zabezpečit. A ne jen nutně zabezpečit, cokoliv proběhne po drátech nešifrovaně, může kdekoliv po cestě někdo upravit. Třeba se na nás přiživovat vkládáním vlastní reklamy. Jop, i tací poskytovatelé internetu mezi námi existují.

HSTS

Když už je HTTPS nastavené, další krok je mít správně nastavený redirect z HTTP na HTTPS. Aby se nestalo, že lze nějak web prohlížet přes nezabezpečené vody. Když už jste si dali tu práci, ať to má smysl.

Redirect však není vše. Uživatelé často pečou na to, co do prohlížeče zadají, a tak útočník může vylákat oběť na HTTP, i když třeba máme redirecty pořešené. Proto nastavíme HSTS hlavičku, aby prohlížeč ani HTTP nezkoušel a v takovém případě sám přesměroval. (Ano, znamená to, že když podělám SSL, nebude vůbec fungovat web. Ale mnohdy lepší žádný, než nebezpečný.)

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

Session cookie

Web je nyní zabezpečený a snažíme se našeho uživatele držet od HTTP co nejdále. Ale nikdy si nemůžeme být jisti, co se útočníkovi povede udělat. Proto budeme držet to nejcenější v co největším bezpečí. Mluvím o session v cookie. Rozhodně by měla být nastavena jako HTTP, což znamená, že ji nelze přečíst JavaScriptem. Mít nastaveno taky Secure není taky na škodu. Znamená to, že cookina půjde pouze po šifrovaném spojení.

CSP

Neboli Content Security Policy je hlavička povolující jen určité zdroje. To znamená, že touto hlavičkou lze znemožnit útočníkovi vložit nebezpečný script. Resp. vložit může, ale nic to neudělá. Lze nastavit defaultně pro všechny, případně pro každý typ zvlášť (obrázek, fonty, JavaScripty, …).

Rozhodně je dobré se vyhnout unsafe-inline či dokonce unsafe-eval. Nejlépe dovolit jen vaši doménu, případně ještě ověřené CDNky. A samozřejmě nejen zakázat, ale také zakázané pokusy nahlásit – je dobré vědět, zda se někdo o něco pokouší.

```
Content-Security-Policy: default-src 'self'; report-uri /csp-report;
```

Případně lze využít verzi, která jen nebezpečný zdroj napráší. Vhodné do začátku, než se odladí již

běžící web, ale rychle bych se dostal do striktního módu.

```
Content-Security-Policy-Report-Only: default-src 'self'; report-uri /csp-report;
```

XSS

CSP je fajn, ale pokud se útočníkovi povede dostat do stránky script přes naši samotnou webovou aplikaci, jsme namydlení. Naštěstí prohlížeče obsahují zabudovanou ochranu proti XSS (Cross Site Scripting). Normálně bývá zapnutá, ale jistota je jistota… (A opět lze bonzovat, což je asi to nejužitečnější na této hlavičce.)

```
X-Xss-Protection: 1; mode=block; report=/xss-report;
```

Iframe

Iframy se sice už nepoužívají, ale neměli bychom na ně zapomínat. Skrývají totiž potenciální nebezpečí. Útočník může naši stránku dát do iframe, překrýt nějakou zajímavou hrou (třeba klikací reklama v podobě „chytni všechny míče a máš možnost vyhrát nový iPhone“) a donutit tak uživatele kliknout na místa na našem webu, jak potřebuje. Proto je nejlepší úplně zablokovat možnost naši stránku v nějakém iframe mít.

```
X-Frame-Options: DENY
```

Typy souboru

Ať už na webu máme nahrávání souborů, které pak i zobrazujeme, či nikoliv, nikdy si nemůžeme být jisti, co se útočníkovi povede. Proto je možnost mu zase jednoduše znepříjemnit práci a to tak, že typ souboru budeme vždy určovat my a prohlížeč se nebude snažit hádat. Tedy pokud útočník nahraje JavaScript či celou HTML stránku, ale my takové věci nepodporujeme, pak prostě jako HTML stránku nevydáme a tudíž se nic nestane.

```
X-Content-Type-Options: nosniff
```

CSRF

Už jste si všimli, že e-mailové aplikace se ptají, zda zobrazit obrázky? Je to z důvodu, že pokud jste například přihlášeni na Facebooku a útočník vám poslal e-mail s obrázkem, který není obrázek, ale jen odkaz na smazání účtu, a vy se na takový obrázek chcete podívat… máte po účtu. Teoreticky. Bude to platit na webech, kde takové akce lze provést obyčejným GET požadavkem, což Facebook

není.

Základem je veškeré akce nedělat GETem, ale POSTem či ostatními. To ale nezabrání místo obrázku využít například formulář, který uživatel vyplní nevědomky, neb si myslí, že je o něčem úplně jiném. Proto je dobré přidat ke každému požadavku ještě nějaký token. Na serveru vygenerovat token a ten při akci poslat zpět (a zvalidaovat, samozřejmě). Poté útočník musí nejprve zjistit i validní CSRF token, aby mohl takto uškodit. Což je podstatně složitější, neb web máme za SSL.

HPKP

Třešničkou na dortu je HPKP, což určuje, kterým certifikátům věřit. Defaultně prohlížeče věří všem certifikátům autorit, které mají v seznamu věrohodných. Jenže už se stalo, že byla certifikační autorita napadnuta…

```
Public-Key-Pins: pin-sha256='…'; pin-sha256='…'; max-age=5184000; report-
uri=/hpkp-report;
```

V ukázce je vidět dvakrát pin-sha256. To není překlep, ale první je aktuálně používaný a druhý backup. Například příprava na vyexpirování atp. A opět lze zároveň bonzovat pokusy a nebo pouze bonzovat.

```
Public-Key-Pins-Report-Only: pin-sha256='…'; pin-sha256='…'; max-age=5184000; report-
uri=/hpkp-report;
```

Hesla

Pokud se spravují hesla, musí se sledovat novinky a mít hesla co nejlépe zabezpečená. Tedy žádné MD5 hashe, ale co nejpomalejší hashovací algoritmus. Jakýkoliv algoritmus je oslaben proti jednomu typu útoku – brute force. Proto je dobré vybrat pomalý algoritmus, aby si nemohl útočník louskat hashe na jeho Raspberry po večerech. Často jsem používal PBKDF2 (default v Djangu), ale nic se nezkazí s bcrypt (náročné na CPU) či scrypt (náročné na paměť), které už jsou rozšířené. Mimochodem výkon lze stále lépe škálovat, než paměť.

Jelikož uniknutí databáze, kterou by mohl začít někdo louskat, nehrozí tak často, je potřeba se zamyslet taky nad přihlašovacím formulářem. Není dobré nechat útočníka v pohodě zkoušet louskat přímo na našem webu. Minimálně si monitorovat nebezpečné množství pokusů, raději s automatickým zpomalováním až blokováním. S blokováním však opatrně, neb se pak může zamezit přístup skutečným uživatelům. Například u naší „interní“ aplikaci víme, odkud se uživatelé přihlašují, takže můžeme podezřelá místa rovnou blokovat. U aplikací pro širokou veřejnost je lepší volbou zpomalování.

Logování

Pokud máte vše až potud, web máte dobře zabezpečen. :-) Ale dovolím si na závěr ještě jednu drobnost. A to, že i na druhé straně, tedy na vašem serveru, je potřeba k citlivým údajům přistupovat opatrně. Základem je mít vypnut debug mód v produkci (a nejlépe i na testu). I když má například Werkzeug debugger nově PIN k aktivaci, je to prostě díra.

Co je ale důležitější, je logování. Logujete si celé požadavky i se vstupními parametry? Všude, včetně

login obrazovky? Pak máte logy pravděpodobně plné hesel v čitelné podobě. My jsme si například ve Flasku přetížili Werkzeugový ImmutableOrderedMultiDict, který skryje nebezpečné hodnoty.

Další tip, pokud pracujete v kódu často s citlivými údaji, použít nějaký speciální objekt, který nedovolí nijak zobrazit vnitřní hodnotu. Můžete pak hodnotu přenášet všude možně a máte jistotu, že se po cestě nezaloguje. Něco ve stylu:

```
class SecretValue(object):

    def __init__(self, secret_value):

        self.__secret_value = secret_value



    def get_secret_value(self):

        return self.__secret_value



    def __str__(self):

        return '--secret-value--'



    def __repr__(self):

        return '<SecretValue>'



>>> v = SecretValue('aaa')

>>> str(v)

'--secret-value--'

>>> repr(v)

'<SecretValue>'

>>> str({'value': v})

"{'value': }"

>>> logging.info(p)

INFO:root:--secret-value--

>>> logging.info('value: {}'.format(v))

INFO:root:value: --secret-value--
```

Security

Poslední tip je na tracebacky. Python to například nedělá, ale PHP nahradí parametry funkcí za skutečné hodnoty. Docela užitečná věc pro debugování, ale nepraktická na citlivé údaje. Doporučuji takové chování vypnout, pokud lze. Pokud nelze, vybrat si rozumější nástroj. :-)

Unique solution ID: #1030
Author: n/a
Last update: 2020-09-09 23:43