

Hashing

In dit labo gaan we dieper in op hashing van bestanden en van paswoorden

Wat ga je leren in dit labo?

- Een hash laten berekenen van een gedownload bestand.
- Gebruik maken van de cryptojs library voor het hashen van paswoorden.
- Het nut van iterations en salt leren.

Vorbereiding

- Neem de slides over hashing nog eens uitvoerig door.
- Neem de gitbook pagina over de library crypto-js door:

<https://apwt.gitbook.io/g-pro-software-security/hashing/crypto.js>

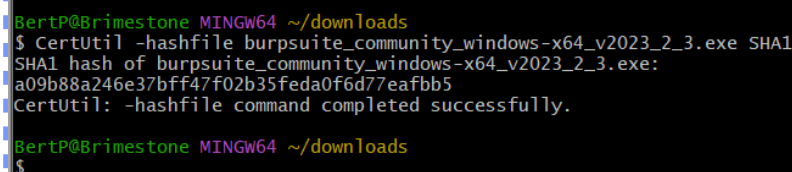
Stappenplan

1. Je hebt in vorig labo Burpsuite gedownload. Ga naar de download pagina van burpsuite (community edition) en probeer de SHA256 checksum van dit bestand te weten te komen. Geef deze hieronder in het tekstvak in:

`077acd9a00298aa5fbdc98ce9127fed1b9a242781b5a6a6741fb910d6c27f799`

2. Gebruik CertUtil (of openssl) commando om de SHA256 hash te berekenen van dit bestand.

Neem een screenshot van je terminal en sleep deze hieronder in:



```
BertP@Brimestone MINGW64 ~/downloads
$ CertUtil -hashfile burpsuite_community_windows-x64_v2023_2_3.exe SHA1
SHA1 hash of burpsuite_community_windows-x64_v2023_2_3.exe:
a09b88a246e37bff47f02b35feda0f6d77eafbb5
CertUtil: -hashfile command completed successfully.

BertP@Brimestone MINGW64 ~/downloads
$
```

3. Deze SHA256 hash moet hetzelfde als die je op de website hebt gevonden. Waarom?

Omdat het anders niet over dezelfde file gaat.
En dus niet betrouwbaar is.

4. Doe een git clone van de volgende repository:

https://github.com/similonap/software_security_2021.git

Ga naar de directory `labo_hashing` en voer het commando

```
npm install
```

uit.

- In dit deel van het labo zijn we aangenomen om de bank 'UNSAFE BANK INC.' te helpen met hun security problemen. Jij bent uiteraard na de cursus Software Security de aangewezen persoon om hun daarbij te helpen.
- Ze hebben ons een script aangeboden dat hun `users` databank afprint. Je kan het script laten lopen door

```
node password_db_print.js
```

uit te voeren in je terminal.

Wat merk je op in verband met de paswoorden?

```
Ze zijn in plaintext opgeslagen in de databank
```

- Om in te loggen hebben ze ook een script aangeboden. Je kan het laten uitvoeren door

```
node login.js
```

in de terminal uit te voeren. Log in met 1 van de users die je in de vorige stap hebt gezien.

- We moeten deze passwoorden nu `hashen` zodat deze niet meer leesbaar zijn voor hackers indien deze zouden gestolen worden.
- Ga naar het bestand `security.js` en pas de `hash` functie aan zodat deze het `PBKDF2` algoritme gebruikt om het passwoord te hashen. Zorg ervoor dat er maar 1 iteration wordt gedaan en dit de salt gebruikt die daar boven aangemaakt wordt. We willen hier de hexadecimale string representatie van.

Opgelet: gebruik hiervoor de `iterations` en `salt` constante die boven de functie al klaarstaat.

- Voer nu terug het

```
node password_db_print.js
```

script uit. Nu zal je zien dat de paswoorden niet meer zomaar uitleesbaar zijn.

Neem hier een screenshot van:

```
Bert@Brimestone MINGW64 ~/Documents/AP/SoftwareSecurity/software_security_2021/labo_hashing (main)
$ node.cmd password_db_print.js
[DEBUG] hash(admin123) took 4 ms
[DEBUG] hash(test123) took 1 ms
[DEBUG] hash(hunter2) took 0 ms
username | password
-----|-----
alice@bankinc.com | 02c5121796a42ec3c7707dd2a71f5d31
bob@bankinc.com | 303185aec3fffffa8ac69317d367c5831
cindy@bankinc.com | dd12fad1c1353d32a1eb4d3c50a8553a
```

11. Na je wijziging klagen de klanten van de bank dat ze niet meer kunnen inloggen met het `login.js` script.

Dit komt omdat het password dat je ingeeft ook eerst moet gehashed worden met de `hash` functie vooraleer je die aan de `login` functie moet meegeven. Pas het `login.js` bestand aan zodat je terug kan inloggen.

12. Je hebt al opgemerkt dat er in de output vaak:

```
[DEBUG] hash(admin123) took 1 ms
```

stond. Dit is omdat we ook de snelheid van het hashing algoritme willen testen. Met 1 iteratie ging dit heel snel. Veel te snel! Dit betekent dat een hacker dit zelf ook heel snel kan berekenen. Hij moet dit natuurlijk wel voor alle combinaties testen.

De bank wil dat jij het aantal iteraties van het hashing algoritme verhoogt zodat dit gemiddeld gezien ongeveer 200ms zal duren om een hash te berekenen.

Test de snelheid van je hashing functie door

```
node hashing_speed_test.js
```

uit te voeren. Indien het niet rond 200ms ligt, pas je het aantal iterations in `security.js` aan en herhaal je de stap.

13. We gaan nu weer even terug naar de juice shop. We hebben via een andere hacker een gehashed paswoord te pakken gekregen van de gebruiker `admin@juice-sh.op`:

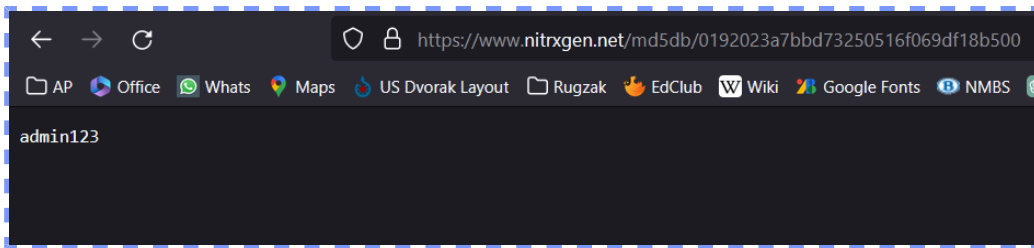
```
0192023a7bbd73250516f069df18b500
```

gebruik nu de website

```
https://www.nitrxgen.net/md5db/
```

om het originele paswoord te bepalen.

Neem een screenshot van je browser en sleep deze hieronder in:



14. Log in met deze gebruiker en verifieer of het paswoord klopt.

15. **Extra:** Bekijk de extra leerstof over hashcat op [gitbook](#) en probeer de md5 hash te achterhalen via hashcat.

Neem een screenshot van je terminal met het paswoord op en sleep deze hieronder in:



16. Print deze pagina af als PDF en slaag deze op als `naam_voornaam_labo_hashing.pdf`

Zip alle bestanden die je in dit labo hebt aangemaakt en stuur deze in via digitap. Deze bestanden zijn:

- security.js
- login.js
- naam_voornaam_labo_hashing.pdf