

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>

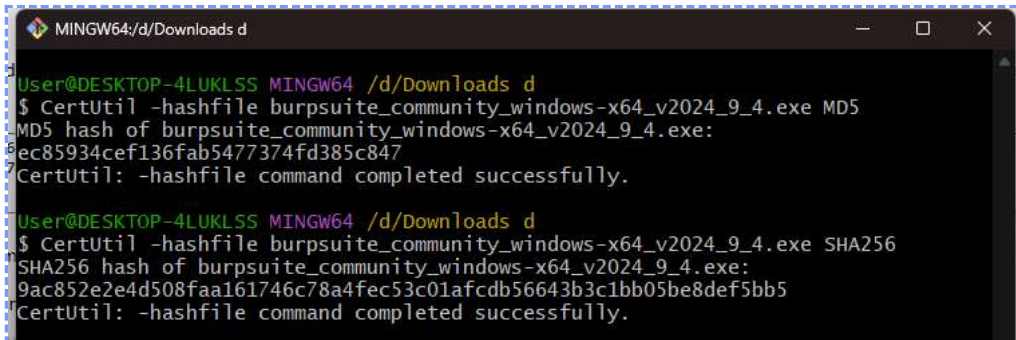
Stappenplan

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:

```
SHA256: 9ac852e2e4d508faa161746c78a4fec53c01afcdb56643b3c1bb05be8def5bb5
MD5: ec85934cef136fab5477374fd385c847
```

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:



```
MINGW64:/d/Downloads d
User@DESKTOP-4LUKL55 MINGW64 /d/Downloads d
$ CertUtil -hashfile burpsuite_community_windows-x64_v2024_9_4.exe MD5
MD5 hash of burpsuite_community_windows-x64_v2024_9_4.exe:
ec85934cef136fab5477374fd385c847
CertUtil: -hashfile command completed successfully.

User@DESKTOP-4LUKL55 MINGW64 /d/Downloads d
$ CertUtil -hashfile burpsuite_community_windows-x64_v2024_9_4.exe SHA256
SHA256 hash of burpsuite_community_windows-x64_v2024_9_4.exe:
9ac852e2e4d508faa161746c78a4fec53c01afcdb56643b3c1bb05be8def5bb5
CertUtil: -hashfile command completed successfully.
```

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

omdat dit anders een ander bestand of aangepast bestand is dan hetgene dat op de website staat. wat dus betekent dat de veiligheid niet meer gegarandeerd kan worden.

4. Download de meegeleverde zip-file, labo_hashing en extract de files.

Open de directory labo_hashing in Visual Studio Code en voer het commando

```
npm install
```

uit.

5. 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.
6. 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?

Deze zijn zeer kort en niet echt veilig, ze maken gebruik van geen enkele hoofdletter en enkele cijfers achteraan. Deze wachtwoorden kunnen zeer makkelijk achterhaald worden

7. 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.

8. We moeten deze passwords nu **hashen** zodat deze niet meer leesbaar zijn voor hackers indien deze zouden gestolen worden.

9. Ga naar het bestand `security.js` en pas de `hash` functie aan zodat deze het **PBKDF2** algoritme gebruikt om het password 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 hexadecimal string representatie van.

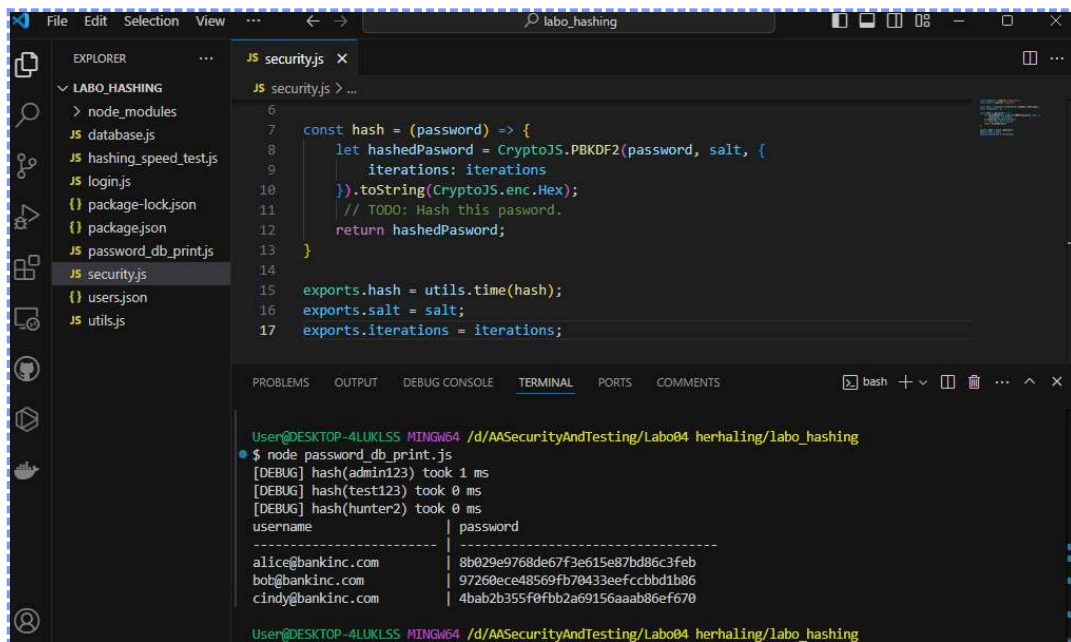
Opgelet: gebruik hiervoor de `iterations` en `salt` constante die boven de functie al klaarstaat.

10. Voer nu terug het

```
node password_db_print.js
```

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

Neem hier een screenshot van:



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 password te pakken gekregen van de gebruiker `admin@juice-sh.op`:

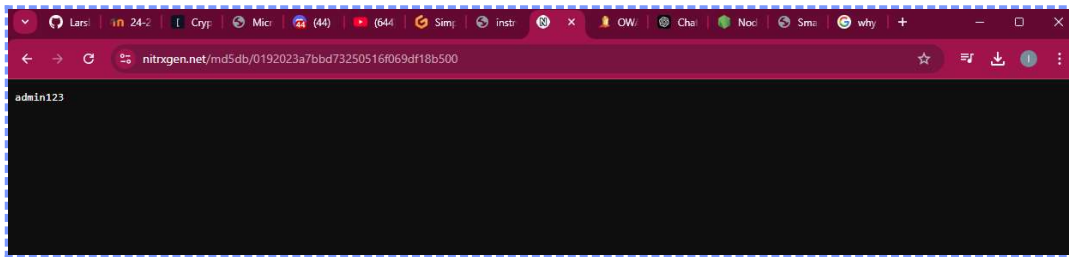
```
0192023a7bbd73250516f069df18b500
```

gebruik nu de website

<https://www.nitrxngen.net/md5db/>

om het originele password 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