# micro:bit Cybersecurity 5: Keer de "(Rainbow) Tables" om
# TI-84 Plus CE Python

## Achtergrondinformatie

- Een cryptografische hashfunctie is een eenrichtingsfunctie; het originele wachtwoord kan niet worden berekend uit de hash. Wanneer een hacker in een computer inbreekt, kunnen de wachtwoord-hashes gestolen worden, maar kunnen ze niet gebruikt worden om in te loggen. Een **brute-force** aanval, die veel plaintext-wachtwoorden test, kan langdurig zijn en vereist veel computerkracht en tijd. Een snellere methode is het doorzoeken van een bestaande tabel met wachtwoorden en hun hashes. ***Rainbow tables*** zijn vooraf berekende hashes van veelgebruikte wachtwoorden. Als een gestolen hash overeenkomt met een hash in de tabel, kan het bijbehorende plaintext-wachtwoord uit de tabel worden gehaald en door de hacker worden gebruikt om in te loggen op die computer.
- Een *rainbow table* is een uitgebreide Python-**woordenlijst** van plaintext-wachtwoorden, geïndexeerd met de bijbehorende berekende hashes.
- Meerdere computers berekenen *rainbow tables* vooraf en het duurt lange tijd om ze samen te stellen. Ze zijn beschikbaar voor hackers om te downloaden op het *darkweb*.
- Het doorzoeken van een *rainbow table* is snel, maar de tabellen kunnen enorm groot zijn.
- *Rainbow tables* worden vaak bijgewerkt met nieuwe wachtwoorden.
- Lange wachtwoorden met minder frequent gebruikte tekens zullen minder snel in een *rainbow table* voorkomen.
- Het regelmatig wijzigen van je wachtwoord helpt de kans te verkleinen dat het in een *rainbow table* voorkomt.
- Een ***salt*** is een sleutelzin die aan alle wachtwoorden op een bepaald platform wordt toegevoegd. Het is een extraatje wat wordt toegevoegd aan je wachtwoord net als wanneer je wat extra zout op je eten strooit. De *salt* die aan het wachtwoord wordt toegevoegd verschilt per platform, zoals een internetbankieren- of winkelaccount, en kan zelfs binnen één platform ook verschillen per wachtwoord. *Salts* worden toegevoegd aan het wachtwoord bij het instellen en tijdens de authenticatie.
- Als iemand hetzelfde wachtwoord op verschillende platforms gebruikt, zal de opgeslagen hash op elk platform verschillen omdat ze gehashed worden met verschillende *salts*.

## Wat is jouw opdracht?

1. Oefen op het doorzoeken van een *rainbow table*:
   De 10 meest voorkomende wachtwoorden:

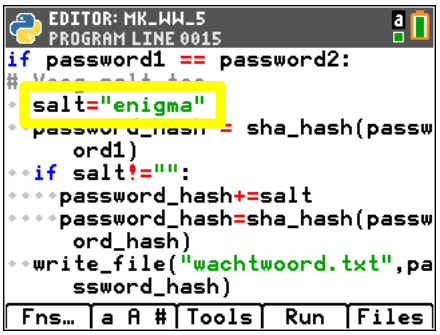
| | | | | |
|---|---|---|---|---|
| *password* | *Qwerty* | *111111* | *abc123* | *12345678* |
| *123456* | *Guest* | *123123* | *123456789* | *12345* |

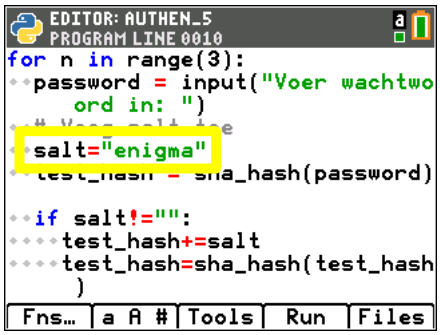
   a. De rainbow table 'rbt' bevat de hashes van deze tien wachtwoorden.\
   b. Open 'OEF_1_5' en voer het programma uit. De hash van een van de wachtwoorden uit de tabel hierboven wordt willekeurig gekozen en opgeslagen in de file 'oefen_wachtwoord' op je micro:bit.
   Open 'OEF_2_5'. Merk op dat nu 'rbt_5' (de rainbow table) wordt geïmporteerd. Voer het programma uit. De opgeslagen hash uit 1b wordt gelezen van de micro:bit en opgeslagen in de variabele "hash". Druk op de [var]-toets selecteer "hash" om het nogmaals af te drukken.
   c. Je kunt nu als volgt het gekozen wachtwoord vinden door in de shell rbt[hash] te typen. Let op de vierkante haakjes! Gebruik eventueel de [var]-toets om "hash" te typen.
   d. Herhaal stappen 1b tot 1d om nog een paar keer te oefenen met het gebruik van de rainbow table.

2. Stel een wachtwoord in op je micro:bit.
    a. **Kies een van de tien veelvoorkomende wachtwoorden** die zijn opgenomen in de *rainbow table* van deze activiteit.
    b. Open "MK_WW_5.py" en voer het programma uit om het wachtwoord naar keuze in te stellen op je micro:bit.
    c. Open “OEF_3_5.py” en voer het programma uit om de hash van het wachtwoord op het scherm af te drukken. Schrijf de eeste vijf tekens van de hash op een stuk papier. Dit heb je nodig in 4c.
    d. Open “AUTHEN_5.py” en voer het programma uit om je nieuwe wachtwoord te testen.

3. Remote login:
    - Zorg ervoor dat alle groepsleden hetzelfde groepsnummer gebruiken.
    - De **ontvanger**
        - **Fluister je wachtwoord naar de zender** zodat de zender kan inloggen op jouw micro:bit. Zorg ervoor dat je het **wachtwoord privé** houdt voor de hacker. Open 'ONTV_5.py', wijzig de groep naar het toegewezen nummer en voer het programma uit **voordat** de zender zijn programma uitvoert.
    - De **zender**
        - Open 'ZEND_5.py', wijzig de groep naar jouw toegewezen nummer en voer je programma uit **nadat** de ontvanger en hacker het programma hebben gestart. **Verstuur het wachtwoord van de ontvanger** om vanop afstand op hun micro:bit in te loggen.
    - De **hacker**
        - Ga naar ‘HACK_5.py’, wijzig de groep naar jouw toegewezen nummer en voer het programma uit **voordat** de zender het programma heeft uitgevoerd. Opmerking – dit programma **ontvangt de wachtwoord-hash** die door de afzender is verzonden om in te loggen op de micro:bit van de ontvanger.
        - Gebruik de gestolen hash en de rainbow table, zoals je deed in stap 1c, om het wachtwoord van de ontvanger op te zoeken uit de onderschepte hash.
    - Zodra de hacker het wachtwoord van de ontvanger heeft gekraakt, moet de ontvanger het ‘ONTV_5.py’ programma opnieuw uitvoeren om te testen of de hacker toegang kan krijgen tot je micro:bit.
    - De hacker moet het ‘ZEND_5.py’ uitvoren gaan en het gekraakte plaintext-wachtwoord versturen. Opmerking – als de hacker succesvol is, zou hij in staat moeten zijn om in te loggen op de micro:bit van de ontvanger zonder ooit het wachtwoord te hebben gekregen!

4. Oefen op het toevoegen van *salt* aan wachtwoorden:
   a. Open 'MK_WW_5.py' opnieuw. Zoek de regel op waar de variabele **salt = ""** staat. Verander de lege string in een woord bijvoorbeeld "enigma". Doe hetzelfde in het programma 'AUTHEN_5.py'. In beide programma's moet de salt hetzelfde zijn!. Merk op dat de salt wordt toegevoegd door de programmeur en niet door de gebruiker.

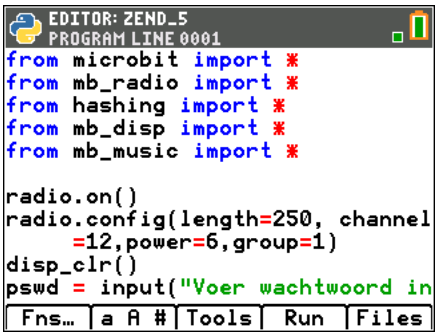


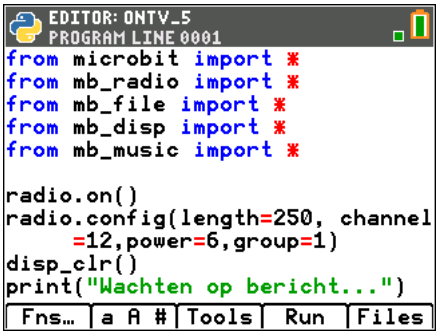

   b. Open opnieuw 'MK_WW_5.py' en maak weer een wachtwoord aan gebruik weer dezelde als in onderdeel 2. Deze keer echter wordt door het programma de salt toegevoegd.
   c. Open 'OEF_3_5.py' en voer het programma uit om de hash weer te geven die op je micro:bit is opgeslagen. Hoewel het wachtwoord hetzelfde is als het wachtwoord in 2b, merk je dat het een andere hash heeft dan de hash in 2c.
   d. Gebruik de *rainbow table* genaamd 'rbt' om het wachtwoord op te zoeken dat overeenkomt met de hash. Om de tabel te gebruiken, typ je **rbt[hash]** in de Python-shell (zoals in 1d). Opmerking – de zoekopdracht van 'rbt' zou moeten mislukken met een foutmelding "**KeyError**:" omdat er geen hash-index in de woordenlijst staat, zelfs niet als je een van de tien veelvoorkomende wachtwoorden hebt gebruikt. Onthoud dat de *salt* die aan de zender en ontvanger is toegevoegd de hash heeft veranderd en deze dus niet meer in de *rainbow table* staat.

## De code

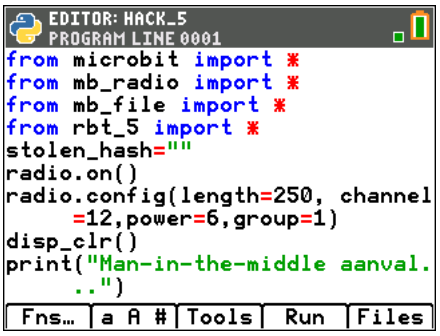
Zender




Ontvanger




Hacker



## Extra uitdagingen

- Laat elk teamlid elke rol een keer spelen en zo proberen het wachtwoord van de ander te hacken.
- Laat elk teamlid stap 4 herhalen met een andere *salt*, maar hetzelfde veelvoorkomende wachtwoord. Laat hen vervolgens van micro:bit wisselen binnen de groep en testen of hun platform (rekenmachine) de micro:bit van de ander opent. Opmerking – iedereen in het team heeft hetzelfde wachtwoord gebruikt, maar de authenticatie werkt niet op de calculator van iemand anders, omdat elke calculator een andere *salt* gebruikt.

## Samengevat

- Een *rainbow table* is een hulpmiddel voor hackers dat wordt gebruikt om ongeautoriseerde toegang te krijgen tot een account of apparaat.
- Een *rainbow table* is een Python-woordenlijst van hashes met het bijbehorende plaintext-wachtwoord.
- Het toevoegen van een *salt* aan een wachtwoord is een methode die een programmeur kan gebruiken om een extra beschermingslaag tegen hackers toe te voegen.
- Het regelmatig wijzigen van je wachtwoord en het gebruiken van minder frequent gebruikte tekens helpt je wachtwoord te beschermen tegen het verschijnen in de *rainbow table* van een hacker.

## Tips voor als het misgaat

- Controleer of iedereen in het team hun toegewezen groepsnummer gebruikt.
- Zorg ervoor dat de ontvanger en hacker hun programma's uitvoeren en wachten voordat de zender het bericht verzendt.
- Zorg ervoor dat de wachtwoorden succesvol zijn ingesteld op elke micro:bit.
- Onthoud, de zender probeert in te loggen op de micro:bit van de ontvanger en moet het wachtwoord voor die micro:bit sturen.

## Bestanden

- Zet de onderstaande programma's op je rekenmachine m.b.v. de TI Connect CE software. De link om deze software te downloaden staat hier. Telkens als je met een nieuw onderdeel begint kun je het beste eerst de gebruikte programma's wissen en daarna de programma's voor het nieuwe ondedeel weer op je rekenmachine zetten.

| Naam | Beschrijving |
|---|---|
| OEF_1_5.py | Kies een willekeurig wachtwoord uit de tabel met de 10 meest voorkomende wachtwoorden. |
| OEF_2_5.py | Leest het oefen_wachtwoord en oefent met de rainbow table. |
| OEF_3_5.py | Leest het wachtwoord dat is aangemaakt met MK_WW_5. |
| MK_WW_5.py | Vraagt om een wachtwoord en slaat dit op op de micro:bit. |
| AUTHEN_5.py | Vergelijkt de hash van een ingevoerd wachtwoord met de hash die op de micro:bit is opgeslagen. |
| ZEND_5.py | Stuurt een hash van het wachtwoord naar de ontvanger voor authenicatie . |
| ONTV_5.py | Ontvangt de hash van een wachtwoord en vort een authenicatie uit. |
| HACK_5.py | "Man-in-the-middle-aanval" tussen zender en ontvanger |
| RBT_5.py | Python dictionary met 10 meest voorkomende wachtwoorden met als index de hashes. |
| HASHING.8xv | SHA-256 hashing module. |