

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*:
 - a. Ga naar de pagina met "**hack_wachtwoord_simulatie.py**" en voer het programma uit.
 - b. Een willekeurige hash wordt gestolen uit het bestand 'oefen_wachtwoord.txt' dat op de micro:bit is opgeslagen en wordt opgeslagen in de Python-shell in de variabele genaamd 'hacked_hash'.
 - c. 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[hacked_hash]` in de Python-shell bij de **REPL >>> prompt** op de volgende pagina. Tip – gebruik de [var] toets van de Nspire om de variabele "hacked_hash" uit een menu te selecteren.
 - d. Herhaal stappen 1b en 1c 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.

<i>Password</i>	<i>qwerty</i>	<i>111111</i>	<i>abc123</i>	<i>12345678</i>
<i>123456</i>	<i>guest</i>	<i>123123</i>	<i>123456789</i>	<i>12345</i>
 - b. Ga naar de pagina met "**stel_wachtwoord_in.py**" en voer het programma uit om het wachtwoord naar keuze in te stellen op je micro:bit.

TI-Nspire CX II

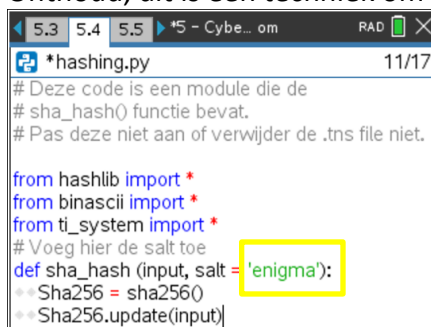
- c. Ga naar “`lees_wachtwoord_hash.py`” en voer het programma uit om de hash die op je micro:bit is opgeslagen weer te geven.
- d. Herhaal stap 1c om het wachtwoord dat je zojuist hebt ingesteld in 2a te hacken. Opmerking - de RBT-zoekopdracht zou het wachtwoord moeten teruggeven dat je hebt ingesteld in 2b. Begrijp je hoe RBT wordt gebruikt om het wachtwoord van een hash te vinden?
- e. Ga naar “`authenticatie.py`” en voer het programma uit om je nieuwe wachtwoord te testen.

3. Remote login:

- De **ontvanger**
 - **Fluister je wachtwoord naar de zender** zodat de zender kan inloggen op je micro:bit. Zorg ervoor dat je het **wachtwoord privé** houdt voor de hacker. Ga naar '`student_ontvanger.py`', wijzig de groep naar het toegewezen nummer en voer het programma uit **voordat** de zender het programma uitvoert.
- De **zender**
 - Ga naar '`student_zender.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 '`student_hacker.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 '`student_ontvanger.py`' programma opnieuw uitvoeren om te testen of de hacker toegang kan krijgen tot je micro:bit.
- De hacker moet naar de '`student_zender.py`' 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. Ga naar de pagina met de module “**hashing.py**” en voeg een extra parameter, de *salt*, toe aan de “`Sha256`” functie. Voeg als *salt* een eenvoudig woord toe, zoals “*enigma*” Opmerking: dit moet alleen worden gedaan in de programma’s `student_zender.py` en `student_ontvanger.py`. Onthoud, dit is een techniek om hackers te blokkeren.



```

5.3 5.4 5.5 *5 - Cybe... om RAD
*hashing.py 11/17
# Deze code is een module die de
# sha_hash() functie bevat.
# Pas deze niet aan of verwijder de .tns file niet.

from hashlib import *
from binascii import *
from ti_system import *
# Voeg hier de salt toe
def sha_hash(input, salt = 'enigma'):
    Sha256 = sha256()
    Sha256.update(input)
    
```

- Na het bewerken van de functie, moet je de code opnieuw compileren door de blauwe [ctrl] toets in te drukken en vervolgens de [B] toets. Zodra dit is voltooid, zou het bericht "hash.py succesvol opgeslagen." moeten verschijnen. Als het bestand niet succesvol is opgeslagen, controleer dan de syntaxis op fouten en zorg ervoor dat de argumenten door een komma worden gescheiden en de *salt* tussen aanhalingstekens staat.
- Ga terug naar de pagina met het programma “**stel_wachtwoord_in.py**” en voer het uit om het wachtwoord in te stellen met hetzelfde wachtwoord als je net gebruikte, uit de reeks van de tien meest voorkomende wachtwoorden. Deze keer zal de *salt* echter aan het wachtwoord worden toegevoegd.
- Ga naar “**lees_wachtwoord_hash.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.
- 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[hacked_hash]** in de Python-shell bij de >>> prompt op de volgende pagina. 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

```
student_zender.py 11/11
from microbit_radio import *
from hashing import *
# Gebruik het wachtwoord van de ontvanger
kanaal = 1
groep = 1
clear_history()
wachtwoord = input("Voer wachtwoord in: ")
wachtwoord_hash = sha_hash(wachtwoord)
tx(wachtwoord_hash,kanaal,groep)
```

Ontvanger

```
student Ontvanger.py 11/15
from microbit_radio import *
from hashing import *
# Deel je wachtwoord met de zender
kanaal = 1
groep = 1
clear_history()
test_hash = rx(kanaal,groep)
authentic_hash = read_file("wachtwoord.txt")
if test_hash == authentic_hash:
    display.show(Image.YES)
    music.play(music.BA_DING)
```

Hacker

```
student_hacker.py 13/13
# De hacker zal gebruik maken van de rainbow
# table om het wachtwoord wat verzonden was
# aan de ontvanger te vinden
kanaal = 1
groep = 1
clear_history()
print("Man-in-the-middle attacking!")
hacked_hash = rx(kanaal,groep)
print("hash string = {}".format(hacked_hash))
# Typ op de volgende pagina rbt[hacked_hash]
# in de Python shell prompt >>>.
```

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.