Achtergrondinformatie

- Zoals je hebt geleerd in de activiteiten vier en vijf, is een **cryptografische hashfunctie** een eenrichtingsberekening; het oorspronkelijke wachtwoord kan niet omgekeerd worden berekend vanuit de hash. Wanneer een hacker in een computer inbreekt en de wachtwoord-hashes steelt, kunnen ze deze niet gebruiken om in te loggen, maar ze kunnen wel gebruikt worden in een ***rainbow table*** of een brute-force aanval.
- Een brute-force aanval creëert alle **permutaties** van mogelijke **platte tekst** wachtwoorden, berekent de hash en vergelijkt deze met de gestolen hash. Een brute-force aanval is een langdurig proces dat **veel rekentijd** vereist.
- Een reden waarom deze aanval zoveel rekentijd vereist, is het aantal permutaties voor een **wachtwoordregel**.
- De permutaties van een wachtwoordregel vertegenwoordigen alle manieren waarop het wachtwoord kan worden gevormd op basis van het aantal en de soorten vereiste tekens in de regel.
- Bijvoorbeeld, een wachtwoordregel met precies vier tekens, die alleen hoofdletters gebruikt, heeft 1 van de 26 verschillende tekens op elke positie. Het aantal manieren waarop een wachtwoord op basis van deze regel kan worden gevormd, is 26 x 26 x 26 x 26 = 456.976, of 26^4 verschillende wachtwoordpermutaties!
- Het bovenstaande voorbeeld kan veralgemeend worden als:

$$\textit{aantal mogelijke tekens}^{\textit{lengte van het wachtwoord}}$$

- Regels die lange wachtwoorden vereisen die zijn gevormd uit veel mogelijke tekens, zijn moeilijker te breken via een brute-force aanval, simpelweg omdat er veel meer mogelijke wachtwoorden zijn om te testen. Bijvoorbeeld, een regel die acht tekens vereist, bestaande uit een kleine letter, een hoofdletter, een cijfer en een speciaal teken (er zijn 32 speciale tekens) heeft $(26+26+10+32)^8$ = 6.095.689.385.410.816 mogelijke wachtwoorden!
- Lange wachtwoorden die uit veel verschillende soorten tekens bestaan en die regelmatig worden veranderd, verkleinen de kans dat je slachtoffer wordt van een brute-force aanval.

Wat is jouw opdracht?

1. Oefenen met wachtwoordsterkte (permutaties):
    a. Open het programma 'OEF_1_6', en voer het programma uit. Druk op de [var] toets en selecteer de functie p(). Deze functie vraagt twee argumenten, het aantal mogelijke tekens en de lengte van het wachtwoord, en berekent het aantal permutaties. Controleer het voorbeeld uit de introductie, p(26+26+10+32, 8). Probeer ook eens andere combinaties.
    b. Open 'OEF_2_6', en voer het programma uit. Dit programma maakt een grafiek met op de horizontale as de lengte van het wachtwoord en het aantal pemutaties op de verticale as.
    c. Open 'OEF_2_6', en bekijk de Python-code. Zie je hoe drie geneste lussen elk mogelijk wachtwoord genereren waarbij drie hoofdletters mogen worden gebruik? Voer het programma uit en observeer hoe het werkt. Zie je hoe de drie geneste lussen de permutaties creëren? Hoeveel mogelijke wachtwoorden kunnen er worden gemaakt van drie hoofdletters? Opmerking: Dit programma duurt ongeveer tien minuten om te voltooien! Als je ongeduldig wordt, houd dan de [on]-toets ingedrukt om het programma te onderbreken.

©Texas Instruments 2024

d. Open weer 'OEF_1_6' en bereken hiermee het aantal permutaties van het voorbeeld uit 1c. Hoeveel verschillende wachtwoorden konden er worden gemaakt? Hoe lang deed de rekenmachine erover. Kun je je voorsellen hoe lang het zal duren om 6.095.689.385.410.816 permutaties te berekenen?

2. Stel een wachtwoord in op je micro:bit.
   a. Verbind je micro:bit met je rekenmachine.
   b. Open 'MK_WW_6.py' en voer het programma uit. Maak een wachtwoord met als voorwaarde: drie tekens bestaande uit alleen kleine letters. Opmerking: Kies voor een wachtwoord met een eerste letter uit het begin van het alfabet om de rekentijd later in deze activiteit te verkorten. Bijvoorbeeld, "aab" of "aba" zal VEEL sneller gekraakt worden dan "zaa" of "zba".

3. De brute-force aanval:
   a. Wissel jouw micro:bit uit met die van een groepslid. *Vertel hem/haar jouw wachtwoord niet!*
   b. Open 'BRUTE_6.py' en bekijk de Python code. Merk op dat er geen printopdrachten binnen de drie geneste lussen staan. Uitvoer van welke aard dan ook, zoals het afdrukken naar het scherm, is relatief langzaam. Het weglaten van print opdrachten maakt het programma veel sneller. Het **optimaliseren** van vaak herhaalde code is essentieel bij het schrijven van programma's.
   c. Open 'OEF_4_6.py' en voer het programma uit. De hash van het wachtwoord wordt gelezen uit file "wachtwoord.txt" en op het scherm getoond. Dit is de hash die de hacker gebruikt om de micro:bit te kraken. Een hash kan niet omgekeerd gebruikt worden. Het programma brobeert systematisch alle hashes van 3-letterige wachtwoorden uit. Dit wordt "brute force" genoemd.

4. Authenticatie:
   a. Open 'AUTHEN_6.py' en voer het programma uit. Voer drie keer een verkeerd wachtwoord in. Na het bekijken van de Python-code, kun je uitleggen hoe het programma herhaalde authenticatiepogingen voorkomt? Hoe draagt het beperken van het aantal pogingen bij aan de veiligheid?
   b. Voer het programma opnieuw uit met het gekraakte platte tekst-wachtwoord om het resultaat van de brute-force aanval te testen. Is het hacken gelukt?

5. Remote login:
   - Zorg ervoor dat alle groepsleden hetzelfde groepsnummer gebruiken.
   - De **ontvanger**
     - Wijzig het wachtwoord zoals beschreven in activiteit 2. Oefen goede **wachtwoordconventie** en hergebruik hetzelfde wachtwoord niet. Kies een wachtwoord dat aan het begin van het alfabet ligt. Fluister je wachtwoord naar de verzender zodat de verzender in kan loggen op je micro:bit. Zorg ervoor dat je het wachtwoord geheim houdt voor hackers. Open 'ONTV_6.py', wijzig de groep naar het toegewezen nummer en voer het programma uit voordat de zender het programma heeft uitgevoerd.
   - De **zender**
     - Ga naar 'ZEND_6.py', wijzig de groep naar het toegewezen nummer en voer je programma uit nadat de ontvanger en hacker hun programma's hebben gestart. Gebruik het wachtwoord van de ontvanger om vanop afstand op hun micro:bit in te loggen.

©Texas Instruments 2024

- De **hacker**
  - Open HACK_6.py', wijzig de groep naar het toegewezen nummer en voer het programma uit voordat de zender het programma heeft uitgevoerd. Opmerking: dit programma ontvangt de wachtwoord-hash die door de zender wordt verzonden om in te loggen op de micro:bit van de ontvanger.
  - Gebruik de gestolen hash en de brute functie om het wachtwoord van de ontvanger te kraken vanuit de onderschepte hash. Type in de shell: "brute(hacked_hash)". De variabele "hacked_hash" kan geselecteerd worden met de [var]-toets.
- Nadat de hacker het wachtwoord van de ontvanger heeft gekraakt, moet de ontvanger het 'ONTV_6.py'-programma opnieuw uitvoeren om te testen of de hacker in staat is om toegang te krijgen tot je micro:bit. De hacker moet het 'ZEND_6.py'-programma uitvoeren en het gekraakte platte tekst-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!

De code

Zender

```
EDITOR: ZEND_6
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from hashing import *
from mb_disp import *
from mb_music import *

radio.on()
radio.config(length=250, channel
    =12,power=6,group=1)
disp_clr()
pswd = input("Voer wachtwoord in
Fns… | a A # | Tools | Run | Files
```

Ontvanger

```
EDITOR: ONTV_6
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from mb_file import *
from mb_disp import *
from mb_music import *

radio.on()
radio.config(length=250, channel
    =12,power=6,group=1)
disp_clr()
print("Wachten op wachtwoord..."
Fns… | a A # | Tools | Run | Files
```

Hacker

```
EDITOR: HACK_6
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from hashing import *
from brute_6 import *

_chr_set_26 = "abcdefghijklmnopq
    rstuvwxyz"

stolen_hash=""
radio.on()
radio.config(length=250, channel
Fns… | a A # | Tools | Run | Files
```

Extra uitdagingen

- Elk teamlid zou iedere rol een keer geprobeerd moeten hebben, om zo elkaars wachtwoord te hacken.
- Stel het micro:bit wachtwoord in op 'aaa' en kijk met het programma 'OEF_4_6.py' hoe lang het duurt om het te kraken. Doe dit ook voor de wachtwoorden 'aaf', 'aak' en 'aap'. Kun je voorspellen hoe lang het duurt om 'aaz' te kraken?

Samengevat

- Een brute-force aanval is een rekenintensieve hack die de hash berekent van alle permutaties van wachtwoorden voor een gegeven wachtwoordregel en deze vergelijkt met een gestolen wachtwoordhash. Als de hashes hetzelfde zijn, wordt het platte tekst-wachtwoord voor de gestolen hash ontdekt.
- Het regelmatig veranderen van je wachtwoord, het gebruiken van niet frequent gebruikte tekens, het niet hergebruiken van hetzelfde wachtwoord voor verschillende accounts, en het niet delen of opschrijven van wachtwoorden helpt je accounts te beschermen tegen hackers.

©Texas Instruments 2024

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 verzender het bericht verzendt.
- Onthoud, de zender probeert in te loggen op de micro:bit van de ontvanger en moet het wachtwoord voor de micro:bit van de ontvanger 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_6.py	Bereken het aantal permutaties van een gegeven wachtwoord.
OEF_2_6.py	Teken een grafiek van het aantal permutaties als functie van de lengte van een wachtwoord.
OEF_3_6.py	Geeft alle permutaties van een drie letter wachtwoord met alleen kleine letters weer.
OEF_4_6.py	Importeert de module brute_6 om een wachtwoord op de micro:bit te kraken met de brut force methode.
MK_WW_6.py	Vraagt om een 3 letterig wachtwoord met alleen kleine letters en slaat dit op op de micro:bit
AUTHEN_6.py	Vergelijkt de hash van een ingevoerd wachtwoord met de hash die op de micro:bit is opgeslagen.
ZEND_6.py	Stuurt een hash van het wachtwoord naar de ontvanger voor authenicatie .
ONTV_6.py	Ontvangt de hash van een wachtwoord en vort een authenicatie uit.
HACK_6.py	"Man-in-the-middle-aanval" tussen zender en ontvanger en gebruikt de brute-functie om de ontvangen hash te kraken.
BRUTE_6.py	Programma om met brute force een wachtwoord te kraken.
HASHING.8xv	SHA-256 hashing module.

©Texas Instruments 2024