





Správa IT a kybernetická bezpečnost – skripty

 Správa identit a hesel	2
1. Detekce prošlých hesel v AD (PowerShell)	2
2. Vyhledání účtů bez nutnosti měnit heslo (Password never expires)	2
3. Seznam neaktivních účtů v doméně	3
4. Vyhledání účtů s administrátorskými právy mimo standardní skupiny	3
 Správa počítačů a serverů	4
5. Detekce neaktivních počítačů v AD (PowerShell)	4
6. Kontrola nainstalovaných aktualizací (PowerShell)	4
7. Kontrola antivirové ochrany (PowerShell + WMI)	5
8. Detekce neautorizovaných lokálních účtů	5
9. Port scan vybraného subnetu (Bash + nmap) Popis:	6
10. ARP / MAC detekce nových zařízení (Bash) Popis:	6
11. Log parser firewallu / proxy (Python)	7
12. Kontrola expirace certifikátů (Python + OpenSSL)	7
 Logy a monitoring	8
13. Automatický report z Event Logu (PowerShell)	8
14. Analýza logů z Linuxu (/var/log/auth.log) (Python/Bash)	9
15. Shromažďování a odesílání logů do SIEMu (Python)	9
 Zabezpečení systémů	10
16. Kontrola integrity souborů (Bash s sha256sum)	10
17. Audit GPO nastavení (PowerShell)	10
18. Kontrola běžících služeb proti whitelistu (Python / PowerShell)	11
19. Inventarizační skript (PowerShell / Python)	11

Správa identit a hesel

1. Detekce prošlých hesel v AD (PowerShell)

Popis:

- Načte všechny uživatele v AD.
- Kontroluje, že účet je povolený, heslo není nastaveno na *Password never expires*.
- Spočítá datum expirace hesla pomocí `msDS-UserPasswordExpiryTimeComputed`.
- Pokud je expirace v minulosti → vypíše.
- Zobrazí: **Name, SamAccountName, PasswordLastSet, PasswordExpiryDate**.
- Možnost exportu do CSV.

```
Import-Module ActiveDirectory
```

```
$ExpiredUsers = Get-ADUser -Filter * -Properties PasswordExpired,
PasswordLastSet, msDS-UserPasswordExpiryTimeComputed, PasswordNeverExpires
|
    Where-Object {
        $_.Enabled -eq $true -and
        $_.PasswordNeverExpires -eq $false -and
        ([datetime]::FromFileTime($_.msDS-
UserPasswordExpiryTimeComputed)) -lt (Get-Date)
    } |
    Select-Object Name, SamAccountName, PasswordLastSet,

@{Name="PasswordExpiryDate";Expression={[datetime]::FromFileTime($_.msDS-
UserPasswordExpiryTimeComputed)}}

$ExpiredUsers | Export-Csv -Path "C:\Reports\ExpiredPasswords.csv" -
NoTypeInformation -Encoding UTF8
```

2. Vyhledání účtů bez nutnosti měnit heslo (Password never expires)

Popis:

- Najde účty, kde je zaškrtnuto *Password never expires*.
- Seznam se exportuje do CSV.

```
Import-Module ActiveDirectory
```

```
$NeverExpire = Get-ADUser -Filter * -Properties PasswordNeverExpires |
    Where-Object { $_.PasswordNeverExpires -eq $true } |
    Select-Object Name, SamAccountName, Enabled

$NeverExpire | Export-Csv -Path "C:\Reports\PasswordNeverExpires.csv" -
NoTypeInformation -Encoding UTF8
```

3. Seznam neaktivních účtů v doméně

Popis:

- Zjistí účty, které se nepřihlásily 90 dní.
- Používá LastLogonDate.

```
Import-Module ActiveDirectory

$Days = 90
$DateLimit = (Get-Date).AddDays(-$Days)

$InactiveUsers = Get-ADUser -Filter * -Properties LastLogonDate |
    Where-Object { $_.LastLogonDate -lt $DateLimit -and $_.Enabled -eq
    $true } |
    Select-Object Name, SamAccountName, LastLogonDate

$InactiveUsers | Export-Csv -Path
"C:\Reports\InactiveUsers_{Days}days.csv" -NoTypeInfoation -Encoding
UTF8
```

4. Vyhledání účtů s administrátorskými právy mimo standardní skupiny

Popis:

- Zkontroluje členství uživatelů v privilegovaných skupinách.
- Najde účty mimo běžné admin skupiny, které mají oprávnění navíc.

```
Import-Module ActiveDirectory

# Standardní skupiny
$AdminGroups = "Domain Admins","Enterprise Admins","Administrators"

$PrivUsers = foreach ($Group in $AdminGroups) {
    Get-ADGroupMember -Identity $Group -Recursive |
        Select-Object Name, SamAccountName, ObjectClass,
    @{Name="Group";Expression={$Group}}
}

$PrivUsers | Export-Csv -Path "C:\Reports\AdminGroupMembers.csv" -
NoTypeInfoation -Encoding UTF8
```

Správa počítačů a serverů

5. Detekce neaktivních počítačů v AD (PowerShell)

Popis:

- Najde počítače, které se nehlásily určitou dobu (např. 30/60/90 dní).
- Vypíše: **ComputerName, LastLogonDate, OperatingSystem**.

```
Import-Module ActiveDirectory

$DaysInactive = 90
$DateLimit = (Get-Date).AddDays(-$DaysInactive)

$InactiveComputers = Get-ADComputer -Filter * -Properties LastLogonDate,
OperatingSystem |
    Where-Object { $_.LastLogonDate -lt $DateLimit } |
    Select-Object Name, LastLogonDate, OperatingSystem

$InactiveComputers | Export-Csv -Path
"C:\Reports\InactiveComputers_${DaysInactive}days.csv" -NoTypeInformation -
Encoding UTF8
```

6. Kontrola nainstalovaných aktualizací (PowerShell)

Popis:

- Kontroluje, zda jsou na počítačích poslední Windows Update.
- Využívá WMI třídu Win32_QuickFixEngineering.
- Seznamuje: **ComputerName, HotFixID, InstalledOn**.

```
$Computers = Get-ADComputer -Filter * | Select-Object -ExpandProperty Name
$Report = @()

foreach ($Computer in $Computers) {
    try {
        $Updates = Get-WmiObject -Class Win32_QuickFixEngineering -
ComputerName $Computer
        foreach ($u in $Updates) {
            $Report += [PSCustomObject]@{
                ComputerName = $Computer
                HotFixID      = $u.HotFixID
                InstalledOn   = $u.InstalledOn
            }
        }
    } catch {
        Write-Warning "Nelze připojit k $Computer"
    }
}

$Report | Export-Csv -Path "C:\Reports\InstalledUpdates.csv" -
NoTypeInformation -Encoding UTF8
```

7. Kontrola antivirové ochrany (PowerShell + WMI)

Popis:

- Kontroluje, zda je na počítačích zapnutá ochrana AV/EDR.
- Používá WMI třídu `AntiVirusProduct` z `root\SecurityCenter2`.
- Seznamuje: **ComputerName**, **DisplayName**, **ProductState**.

```
$Computers = Get-ADComputer -Filter * | Select-Object -ExpandProperty Name
$AVReport = @()

foreach ($Computer in $Computers) {
    try {
        $AVs = Get-WmiObject -Namespace "root\SecurityCenter2" -Class
        AntiVirusProduct -ComputerName $Computer
        foreach ($av in $AVs) {
            $AVReport += [PSCustomObject]@{
                ComputerName = $Computer
                AVName        = $av.displayName
                ProductState  = $av.productState
            }
        }
    } catch {
        Write-Warning "Nelze získat AV info z $Computer"
    }
}

$AVReport | Export-Csv -Path "C:\Reports\AVStatus.csv" -NoTypeInformation -
Encoding UTF8
```

8. Detekce neautorizovaných lokálních účtů

Popis:

- Bere jen členy skupiny **Administrators**.
- Bezpečně extrahuje jména uživatelů z formátu `PartComponent`.
- Vypíše čistý seznam lokálních administrátorů, který můžeš použít k porovnání s autorizovanými účty.

Powershell:

```
# Definice povolených administrátorských účtů
$AllowedAdmins = @("Administrator", "ITSupport", "Helpdesk")

# Získání seznamu počítačů z AD
$Computers = Get-ADComputer -Filter * | Select-Object -ExpandProperty Name

# Výsledek
$LocalAdminReport = @()

foreach ($Computer in $Computers) {
    try {
        # Získání členů lokální skupiny Administrators
        $Admins = Get-WmiObject -Class Win32_GroupUser -ComputerName
        $Computer |
```

```

        Where-Object { $_.GroupComponent -like '"Administrators"' } |
        ForEach-Object {
            ($_ .PartComponent -replace '.*Name="([^\"]+)"', '$1')
        }

# Filtrování nepovolených účtů
$Unauthorized = $Admins | Where-Object { $_ -notin $AllowedAdmins }

foreach ($u in $Unauthorized) {
    $LocalAdminReport += [PSCustomObject]@{
        ComputerName      = $Computer
        UnauthorizedAdmin = $u
    }
}
}
catch {
    Write-Warning "Nelze načíst lokální adminy z $Computer"
}
}

# Export výsledku
$LocalAdminReport | Export-Csv -Path
"C:\Reports\UnauthorizedLocalAdmins.csv" -NoTypeInformation -Encoding UTF8

```

9. Port scan vybraného subnetu (Bash + nmap)

Popis:

- Rychlá kontrola otevřených portů v subnetu.
- Užitečné pro odhalení služeb a neočekávaných otevřených portů.

```

#!/bin/bash
SUBNET="192.168.1.0/24"
OUTPUT="port_scan_report.txt"

nmap -p 1-65535 -T4 -oN $OUTPUT $SUBNET
echo "Port scan dokončen. Výsledky uloženy do $OUTPUT"

```

10. ARP / MAC detekce nových zařízení (Bash)

Popis:

- Sleduje nové zařízení připojené do sítě přes ARP tabulku.
- Vypíše IP, MAC, a čas detekce.

```

#!/bin/bash
KNOWN_DEVICES="known_devices.txt"
CURRENT_DEVICES="current_devices.txt"

arp -a | awk '{print $2, $4}' | sed 's/[()]/g' > $CURRENT_DEVICES

for device in $(cat $CURRENT_DEVICES); do
    if ! grep -q "$device" $KNOWN_DEVICES; then
        echo "Nové zařízení detekováno: $device"
        echo $device >> $KNOWN_DEVICES
    fi
done

```

11. Log parser firewallu / proxy (Python)

Popis:

- Analýza logů firewallu nebo proxy.
- Vypíše **top 10 cílových IP nebo domén** podle počtu přístupů.

```
from collections import Counter

log_file = "/var/log/firewall.log"

with open(log_file) as f:
    lines = f.readlines()

targets = [line.split()[2] for line in lines] # předpoklad: 3. sloupec =
cílová IP/doména
counter = Counter(targets)
top10 = counter.most_common(10)

print("Top 10 cílových IP/domén:")
for target, count in top10:
    print(f"{target}: {count}")
```

12. Kontrola expirace certifikátů (Python + OpenSSL)

Popis:

- Kontrola SSL/TLS certifikátů domén a služeb.
- Vypíše **doménu, datum expirace, počet dní do expirace**.

```
import ssl
import socket
from datetime import datetime

domains = ["example.com", "google.com"]

for domain in domains:
    context = ssl.create_default_context()
    with socket.create_connection((domain, 443)) as sock:
        with context.wrap_socket(sock, server_hostname=domain) as ssock:
            cert = ssock.getpeercert()
            exp_date = datetime.strptime(cert['notAfter'], "%b %d %H:%M:%S
%Y %Z")

            days_left = (exp_date - datetime.utcnow()).days
            print(f"{domain} - expires on {exp_date}, in {days_left} days")
```

Logy a monitoring

13. Automatický report z Event Logu (PowerShell)

Popis:

- Shromažďuje a zobrazuje vybrané události ze **Security Event Logu** za zadané časové období (výchozí je posledních 7 dní) a sleduje **události 4624 a 4625** (úspěšné a neúspěšné přihlášení).
- Filtruje pouze RDP relace pomocí **RemoteInteractive**.
- Extrahuje **uživatelské jméno, IP a čas přihlášení**.
- Vypíše přehlednou tabulku pro report.

Powershell:

```
# Nastavení časového období (např. posledních 7 dní)
$startDate = (Get-Date).AddDays(-7)

# Vyhledání selhaných přihlášení
$failedLogons = Get-EventLog -LogName Security -After $startDate |
    Where-Object { $_.EventID -eq 4625 } |
    Select-Object TimeGenerated, Message, UserName

# Uzamčené účty
$lockedAccounts = Get-EventLog -LogName Security -After $startDate |
    Where-Object { $_.EventID -eq 4740 } |
    Select-Object TimeGenerated, Message, TargetUserName

# RDP přístupy (úspěšné)
$rdpLogons = Get-EventLog -LogName Security -After $startDate |
    Where-Object { $_.EventID -eq 4624 -and $_.Message -like "*RemoteInteractive*" } |
    ForEach-Object {
        $msg = $_.Message
        $null = $msg -match 'Account Name:\s+(\S+)'
        $user = $matches[1]
        $null = $msg -match 'Source Network Address:\s+(\S+)'
        $ip = $matches[1]

        [PSCustomObject]@{
            TimeGenerated = $_.TimeGenerated
            UserName      = $user
            IPAddress     = $ip
        }
    }

# Výpis
$failedLogons
$lockedAccounts
$rdpLogons
```

14. Analýza logů z Linuxu (/var/log/auth.log) (Python/Bash)

Popis:

- Sleduje **neúspěšné SSH přihlášení**.
- Vypíše **top IP adresy podle počtu pokusů**.

```
#!/bin/bash
log_file="/var/log/auth.log"

grep "Failed password" $log_file | awk '{print $(NF-3)}' | sort | uniq -c |
sort -nr | head -10
```

Python alternativa:

```
from collections import Counter

log_file = "/var/log/auth.log"

with open(log_file) as f:
    ips = [line.split()[-3] for line in f if "Failed password" in line]

counter = Counter(ips)
for ip, count in counter.most_common(10):
    print(f"{ip}: {count}")
```

15. Shromažďování a odesílání logů do SIEMu (Python)

Popis:

- Jednoduchý forwarder logů do SIEMu.
- Připojení přes TCP nebo UDP, odesílá logy v reálném čase.

```
import socket

SIEM_HOST = "siem.example.com"
SIEM_PORT = 514

log_file = "/var/log/syslog"

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP

with open(log_file) as f:
    for line in f:
        sock.sendto(line.encode(), (SIEM_HOST, SIEM_PORT))

print("Logy byly odeslány do SIEMu.")
```

🛡 Zabezpečení systémů

16. Kontrola integrity souborů (Bash s sha256sum)

Popis:

- Vytváří hash souborů (SHA256) v určených adresářích (např. /etc, /usr/bin, C:\Windows\System32 na Windows) a porovnává je s referenčním souborem baseline.sha256.
- Výsledek: Pokud se aktuální hash liší od uloženého, znamená to, že soubor byl změněn, a skript to vypíše.
- Použití: Slouží pro detekci neautorizovaných úprav souborů, např. po útoku malwarem nebo při omylu správce.

```
#!/bin/bash

# Cesta k adresáři, který chceme kontrolovat
TARGET_DIR="/cesta/k/souborům"

# Soubor s referenčními SHA256 kontrolními součty
BASELINE_FILE="baseline.sha256"

# Generování kontrolních součtů aktuálních souborů
find "$TARGET_DIR" -type f -exec sha256sum {} \; > current.sha256

# Porovnání aktuálních součtů s referenčním souborem
echo "Kontrola integrity souborů:"
sha256sum -c "$BASELINE_FILE" 2>/dev/null | while read line; do
    if echo "$line" | grep -q "OK"; then
        echo "$line"
    else
        echo "UPOZORNĚNÍ: Změna detekována! $line"
    fi
done

# Volitelně: aktualizace baseline po ověření
# cp current.sha256 baseline.sha256
```

17. Audit GPO nastavení (PowerShell)

Popis:

- Vypsání důležitých bezpečnostních parametrů:
 - Lockout policy
 - Délka hesla
 - Heslo historie
- Sleduje, zda GPO odpovídají doporučené bezpečnostní politice.

```
# Získání politiky pro místní počítač
$polices = Get-ADDefaultDomainPasswordPolicy

$polices | Select-Object LockoutThreshold, LockoutDuration,
MaxPasswordAge, MinPasswordLength, PasswordHistoryCount
```

18. Kontrola běžících služeb proti whitelistu (Python / PowerShell)

Popis:

- Sleduje běžící služby / procesy.
- Upozorní, pokud se objeví **neznámý nebo nepovolený proces** mimo whitelist.

PowerShell:

```
# Definovaný whitelist služeb
$whitelist = @("WinDefend", "Spooler", "wuauserv")

# Získání všech běžících služeb
$services = Get-Service | Where-Object { $_.Status -eq "Running" }

# Filtrace služeb mimo whitelist
$services | Where-Object { $whitelist -notcontains $_.Name } | Select-
Object Name, DisplayName, Status
```

Python (Linux) alternativa:

```
import psutil

whitelist = ["sshd", "systemd", "cron"]

for proc in psutil.process_iter(['pid', 'name']):
    if proc.info['name'] not in whitelist:
        print(proc.info)
```

Užitečné utility

19. Inventarizační skript (PowerShell / Python)

Popis:

- Shromažďuje informace o hardwaru, OS verzích a patch levelu.
- Vhodné pro rychlý přehled infrastruktury.

PowerShell:

```
Get-ComputerInfo | Select-Object CsName, OsName, OsVersion, OsArchitecture,  
WindowsProductName, WindowsVersion
```

```
Get-WmiObject Win32_ComputerSystem | Select-Object Manufacturer, Model,  
TotalPhysicalMemory
```

Python (cross-platform):

```
import platform  
import psutil  
  
print("OS:", platform.system(), platform.release())  
print("Architektura:", platform.machine())  
print("CPU count:", psutil.cpu_count())  
print("RAM (GB):", round(psutil.virtual_memory().total / (1024**3), 2))
```
