



Desenvolvimento de Scripts

Scripts em Linux

Aula 01 – Introdução ao desenvolvimento de scripts em Linux

- **O que é Script?**
 - Série de instruções
 - Otimizar processos
- **Por que usar scripts?**
- **Automatização de processos.**

- **Importância da automação de tarefas:**
 - Eficiência;
 - Consistência;
 - Resposta rápida a incidentes;
 - Análise de dados;
 - Monitoramento contínuo;
 - Escalonamento

- Noções da Linguagem Bash:

Comentários: Linhas começando com `#` são comentários e são ignorados pelo interpretador.

Variáveis: Variáveis são usadas para armazenar valores. Exemplo: `nome="Usuário"`

Comandos: Comandos do sistema podem ser executados usando crases (```) ou `$()`. Exemplo: `data= $(date)`

Estruturas Condicionais: `if`, `elif` e `fi` são usados para tomada de decisões baseadas em condições.

Estruturas de Loop: `for` e `while` permitem repetir ações várias vezes.

Redirecionamento de Saída: `>` (redireciona saída para um arquivo) e `>>` (anexa saída a um arquivo).

Testes e Comparação de Strings: O comando `test` (ou `[]`) é usado para comparar strings, números e arquivos.

Funções: Permitem definir blocos de código reutilizáveis.

Argumentos da Linha de Comando: Variáveis especiais como `$1`, `$2`

Interpolação de Variáveis: Variáveis podem ser usadas dentro de strings com aspas duplas para interpolação.

Aula 02 – Exemplos práticos:

- scripts básicos para automação de tarefas de segurança cibernética em Linux;

- Um script que analisa os logs de autenticação para identificar tentativas de login falhadas pode ajudar a detectar possíveis tentativas de invasão. O comando `grep` pode ser usado para procurar padrões de logs.

Exemplo de verificação de Logs de Autenticação:

```
#!/bin/bash
```

```
echo "Procurando por tentativas de login falhadas nos logs:"
```

```
grep "Failed password" /var/log/auth.log
```

- Automatizar a verificação e instalação de atualizações do sistema é importante para manter o sistema seguro. O uso de comandos como `apt-get` (para sistemas baseados em Debian) ou `yum` (para sistemas baseados em Red Hat) pode ser incorporado em um script.

Exemplo de Verificação de Atualizações do Sistema:

```
#!/bin/bash
```

```
echo "Verificando atualizações disponíveis..."
```

```
apt-get update
```

```
apt-get upgrade -s | grep "upgraded"
```


- Usando ferramentas como o `curl`, você pode criar um script que verifica automaticamente a presença de vulnerabilidades conhecidas em sites.

Exemplo de Verificação de Vulnerabilidades em Sites:

```
#!/bin/bash

site="https://www.google.com"

vulnerabilidade="CVE-2014-6271"

echo

resultado=$(curl -s -I "$site" | grep "$vulnerabilidade")

if [ -n "$resultado" ]; then

    echo "Vulnerabilidade Shellshock detectada no site $site."

else

    echo "Site seguro, nenhuma vulnerabilidade Shellshock detectada."

fi
```

Aula 03 – Introdução ao desenvolvimento de scripts em Linux

- Nessa abordagem, você coloca o comando entre parênteses e precede-o com o símbolo de dólar seguido de parênteses. O resultado do comando é armazenado em uma variável.

Por exemplo:

```
```bash
```

```
resultado=$(ls /caminho/do/diretorio)
```

```
echo "Conteúdo do diretório: $resultado"
```

```
```
```

Substituição de Comando com Backticks (`):

- Essa é uma abordagem mais antiga, onde você coloca o comando entre crases (backticks) e o resultado é novamente armazenado em uma variável.

Por exemplo:

```
```bash
```

```
resultado=`ls /caminho/do/diretorio`
```

```
echo "Conteúdo do diretório: $resultado"
```

```
```
```

Geralmente, a primeira abordagem com `\$()` é preferida por ser mais legível e mais fácil de ser aninhada.

Exemplo de Execução de Comando Externo em Script:

Suponha que você queira verificar o espaço livre em disco usando o comando ``df -h``. Você pode incorporar esse comando em um script da seguinte maneira:

```
```bash
#!/bin/bash
espaco_livre=$(df -h)
echo "Espaço livre em disco:"
echo "$espaco_livre"
```
```

Nesse exemplo, o comando ``df -h`` é executado e seu resultado é armazenado na variável ``espaco_livre``. Em seguida, o resultado é exibido usando o comando ``echo``.

1. Criando um Diretório e Copiando um Arquivo:

```
```bash
#!/bin/bash
echo "Criando um diretório e copiando um arquivo..."
mkdir novo_diretorio
cp arquivo.txt novo_diretorio/
echo "Diretório criado e arquivo copiado."```
```

## 2. Listando Conteúdo de um Diretório e Renomeando um Arquivo:

```
``bash
#!/bin/bash
echo "Listando conteúdo do diretório e renomeando um
arquivo..."
ls /caminho/do/diretorio
mv antigo_nome.txt novo_nome.txt
echo "Arquivo renomeado."
``
```

## 3. Removendo um Arquivo e Executando um Comando Externo:

```
```bash
#!/bin/bash
echo "Removendo um arquivo e executando um comando
externo..."
rm arquivo_removido.txt
resultado=$(ls /caminho/do/diretorio)
echo "Conteúdo do diretório: $resultado"
```


Aula 04 – Introdução ao desenvolvimento de scripts em Linux

Script de Detecção de Portas Abertas:



1. Um script que verifica portas abertas em um sistema e alerta sobre portas não autorizadas ou suspeitas.

```
#!/bin/bash
```

```
host="127.0.0.1"
```

```
portas=("80" "22" "443" "3389")
```

```
for porta in "${portas[@]}; do
```

```
    nc -zv "$host" "$porta" > /dev/null 2>&1
```

```
    if [ $? -eq 0 ]; then
```

```
        echo "Porta $porta está aberta em $host"
```

```
    else
```

```
        echo "Porta $porta está fechada em $host"
```

```
    fi
```

```
done
```

2. Um script que analisa logs de autenticação em busca de padrões suspeitos, como múltiplas tentativas de login falhadas.

```
#!/bin/bash
log_file="/var/log/auth.log"
padrao="Failed password"
if grep -q "$padrao" "$log_file"; then
    echo "Padrão suspeito encontrado nos logs de autenticação."
else
    echo "Nenhum padrão suspeito encontrado nos logs de autenticação."
fi
```

Exemplos de Scripts com Foco em Segurança Cibernética:

3. Script de Detecção de Ataques de Força Bruta:

- Um script que monitora logs em busca de atividades de força bruta em tentativas de acesso, como tentativas repetidas de login.

```
#!/bin/bash
log_file="/var/log/auth.log"
limite_tentativas=5
tentativas=$(grep "Failed password" "$log_file" | wc -l)
if [ "$tentativas" -ge "$limite_tentativas" ]; then
    echo "Possível ataque de força bruta detectado."
else
    echo "Nenhuma atividade de força bruta detectada."
fi
```

4. Script de Verificação de Vulnerabilidades em Sistemas:

Um script que verifica sistemas em busca de vulnerabilidades conhecidas, usando bancos de dados de CVEs (Vulnerabilidades e Exposições Comuns).

```
#!/bin/bash
sistema="Ubuntu"
versao="20.04"
vulnerabilidade=$(grep "$sistema $versao" cve_database.txt)
if [ -n "$vulnerabilidade" ]; then
    echo "Vulnerabilidade conhecida encontrada no sistema
    $sistema $versao."
else
    echo "Nenhuma vulnerabilidade conhecida encontrada."
fi
```

5. Script de Monitoramento de Arquivos Críticos:

Um script que monitora alterações em arquivos críticos do sistema e alerta sobre quaisquer modificações não autorizadas.

```
#!/bin/bash
```

```
diretorio="/diretorio_critico"
```

```
while true; do
```

```
    changes=$(inotifywait -e modify,create,delete  
"$diretorio")
```

```
    echo "Alterações detectadas em: $changes"
```

```
done
```

6. Script de Análise Forense para Coleta de Evidências:

Um script que automatiza a coleta de evidências de um sistema comprometido para análise forense posterior.

```
#!/bin/bash
```

```
output_dir="/evidencias"
```

```
log_file="/var/log/syslog"
```

```
mkdir -p "$output_dir"
```

```
cp "$log_file" "$output_dir/syslog_copia.log"
```

```
echo "Evidências coletadas em $output_dir"
```





Desenvolvimento de Scripts

WINDOWS

Introdução ao desenvolvimento de scripts

O que são Scripts?

Scripts são arquivos de texto com comandos executáveis por um interpretador.



Fonte: Desenvolvimento de Script.
Disponível em: https://as2.ftcdn.net/v2/img/06/21/25/65/1000_F_621256585_6wkBrdVt6w2peMgKzUHSUo7i5MHf6bVz.jpg.
Acesso em 31 Julho. 2023

- Automação de tarefas
- Padronização de processos
- Gerenciamento de Sistemas e Redes
- Manipulação de Dados
- Personalização e Extensibilidade
- Redução de Erros
- Eficiência e Produtividade



Fonte: Desenvolvimento de Script.

Disponível em: https://as2.ftcdn.net/v2/img/06/21/25/65/1000_F_621256585_6wkBrdVt6w2peMgKzUHSUo7i5MHf6bVz.jpg.
Acesso em 31 Julho. 2023

Linguagens populares de script para Windows

PowerShell: Baseada em objetos, manipula recursos do sistema operacional e interage com outras plataformas.

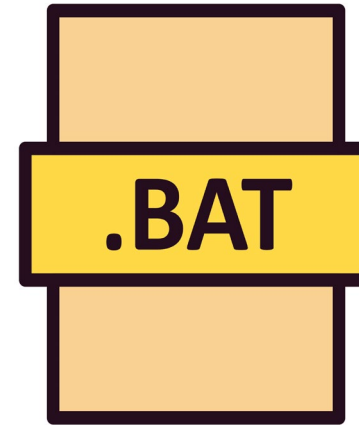


Fonte: PowerShell arquivo.

Disponível em: <https://stock.adobe.com/br/images/programming-language-powershell-inscription-on-the-background-of-computer-code/280856935>. Acesso em 31 Julho. 2023

Linguagens populares de script para Windows

Batch: Usa comandos do prompt de comando para tarefas simples ou complexas.



Fonte: Bat arquivo.
Disponível em: https://as1.ftcdn.net/v2/jpg/04/97/91/80/1000_F_497918010_pQr2cQRx0EoIDLI5T0GcPrjd6FJWBU0.jpg.
Acesso em 31 Julho. 2023

Linguagens populares de script para Windows

VBScript: Baseada em Visual Basic, usada para scripts no Windows Script Host e aplicativos como Internet Explorer, Excel, Word.



Fonte: VBScript arquivo.
Disponível em: <https://stock.adobe.com/br/images/programming-language-vbscript-inscription-on-the-background-of-computer-code/278184401>. Acesso em 31 Julho. 2023

Linguagens populares de script para Windows

JavaScript: Baseada em ECMAScript, usada para scripts no Windows Script Host e aplicativos web ou desktop.



Fonte: JavaScript arquivo.

Disponível em: <https://stock.adobe.com/br/images/moscow-russia-1-june-2020-javascript-js-logo-sign-with-program-code-on-background-illustrative-editorial/390711002>. Acesso em 31 Julho, 2023

Vantagens do uso de scripts

Agilidade: Automatiza tarefas complexas com poucas linhas de código e sem compilação.

Flexibilidade: Pode ser adaptado às necessidades do desenvolvedor com parâmetros, variáveis, loops, etc.



Vantagens do uso de scripts

Portabilidade: Executável em diferentes versões ou plataformas do Windows.

Integração: Comunica-se com outros programas por meio de interfaces, usando bibliotecas externas.



Riscos: Malware, vulnerabilidades e permissões elevadas.

Boas práticas: Verificar origem e integridade, analisar conteúdo e comportamento, limitar permissões e recursos, atualizar e proteger sistemas.



Exemplos de scripts

PowerShell

```
# Script para listar os processos em execução no sistema
Get-Process | Format-Table -AutoSize

# Script para criar um arquivo zip com os arquivos de uma pasta
$source = "C:\Users\user\Documents"
$destination = "C:\Users\user\Documents.zip"
Compress-Archive -Path $source -DestinationPath $destination

# Script para enviar um email usando o Outlook
$Outlook = New-Object -ComObject Outlook.Application
$Mail = $Outlook.CreateItem(0)
$Mail.To = "destinatario@email.com"
$Mail.Subject = "Assunto do email"
$Mail.Body = "Corpo do email"
$Mail.Send()
```

Exemplos de scripts



Batch

```
:: Script para exibir a data e a hora atuais  
echo %date% %time%  
  
:: Script para copiar um arquivo para outra pasta  
copy C:\Users\user\file.txt D:\Backup\file.txt  
  
:: Script para executar um ping em um site  
ping www.bing.com
```

Exemplos de scripts



VBScript

```
' Script para exibir uma mensagem na tela
MsgBox "Olá mundo!"

' Script para criar uma pasta no desktop
Set objFSO = CreateObject("Scripting.FileSystemObject")
strDesktop = objFSO.GetSpecialFolder(0)
objFSO.CreateFolder strDesktop & "\Nova Pasta"

' Script para abrir o Internet Explorer e navegar para um site
Set objIE = CreateObject("InternetExplorer.Application")
objIE.Visible = True
objIE.Navigate "https://www.bing.com"
```

Exemplos de scripts



JavaScript

```
// Script para exibir uma mensagem na tela
WScript.Echo("Olá mundo!");

// Script para criar um arquivo de texto no desktop
var fso = new ActiveXObject("Scripting.FileSystemObject");
var desktop = fso.GetSpecialFolder(0);
var file = fso.CreateTextFile(desktop + "\\Arquivo.txt", true);
file.WriteLine("Este é um arquivo de texto criado por um script.");
file.Close();

// Script para abrir o Notepad e escrever algo nele
var shell = new ActiveXObject("WScript.Shell");
shell.Run("notepad.exe");
WScript.Sleep(1000);
shell.SendKeys("Este é um texto escrito por um script.");
```

Conclusão



Nesta aula, aprendemos os conceitos básicos de scripts em ambientes Windows, as principais ferramentas e linguagens de script para Windows, os riscos e as boas práticas de segurança cibernética ao desenvolver e executar scripts. Esperamos que esta aula tenha sido útil e interessante para você.



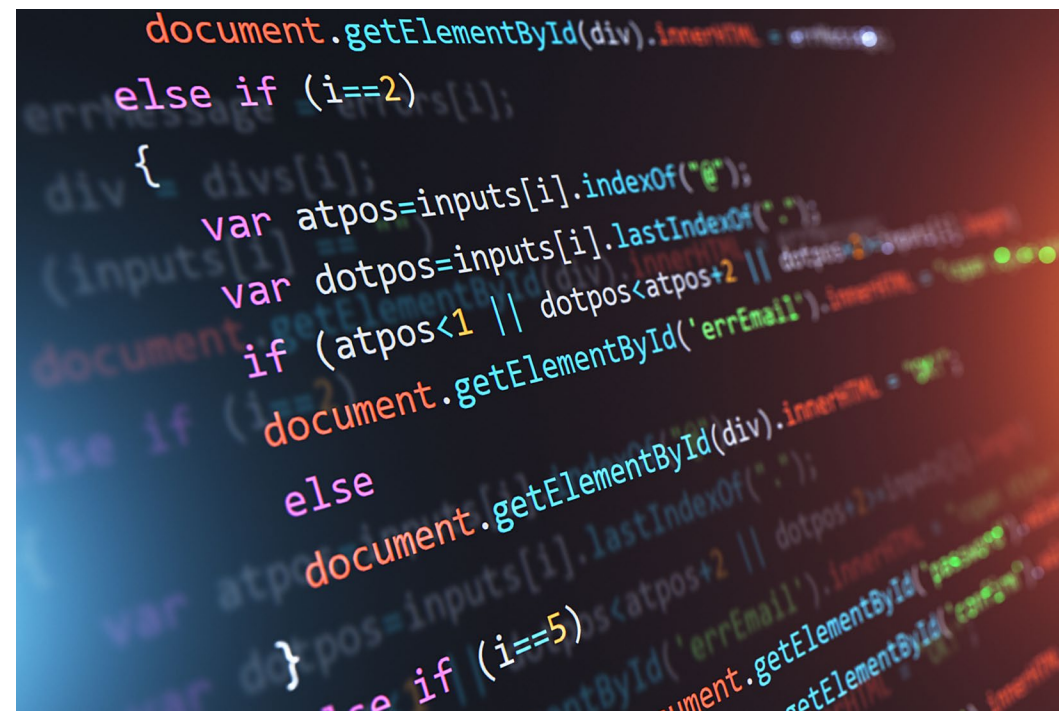
Desenvolvimento de Scripts

WINDOWS

Desenvolvimento de scripts avançados

Técnicas avançadas de script

Uso de parâmetros e argumentos: Permite tornar os scripts mais flexíveis e reutilizáveis ao receberem valores através de parâmetros e argumentos, adaptando-se a diferentes situações ou entradas.

A blurred screenshot of JavaScript code, likely from a web browser's developer console. The code includes DOM manipulation methods like `document.getElementById` and `document.getElementsByTagName`, along with conditional logic using `if` and `else if` statements. The text is out of focus, but some keywords like `document`, `getElementById`, `innerHTML`, `if`, `else if`, `var`, and `indexOf` are visible.

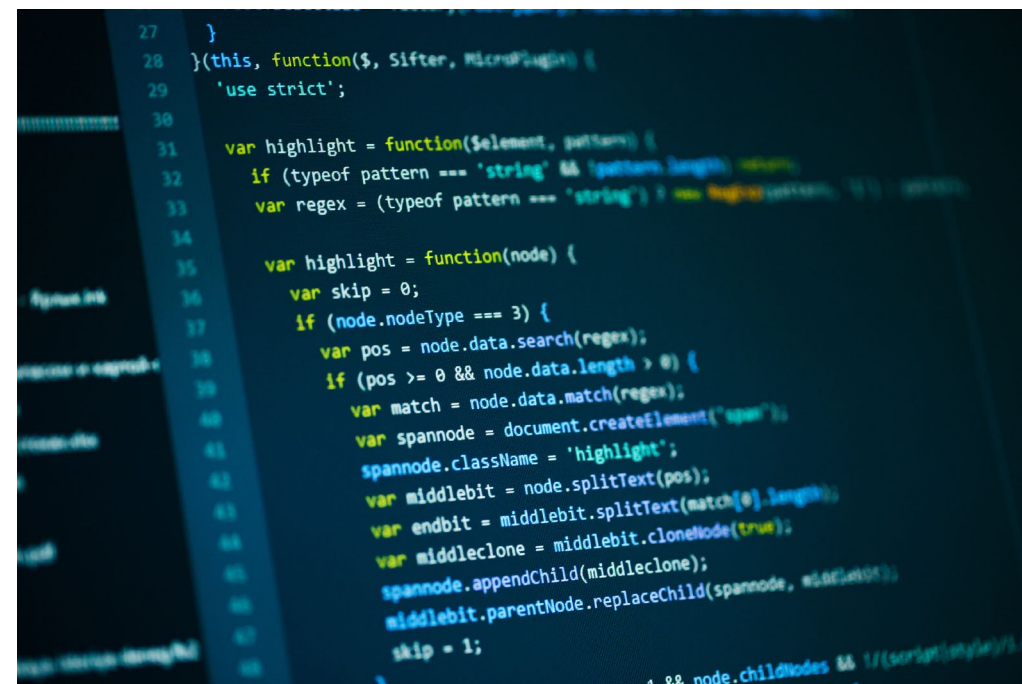
Fonte: Desenvolvimento de Script.

Disponível em: https://as2.ftcdn.net/v2/jpg/02/25/08/49/1000_F_225084966_hhswkk9GgkAKcr2p1n69aail1jETbZO9.jpg.

Acesso em 31 Julho. 2023

Técnicas avançadas de script

Uso de funções e módulos: Organiza os scripts em partes menores e mais compreensíveis através de blocos de código (funções) que executam tarefas específicas, e possibilita a importação de módulos contendo funções relacionadas em outros scripts, evitando repetição de código.

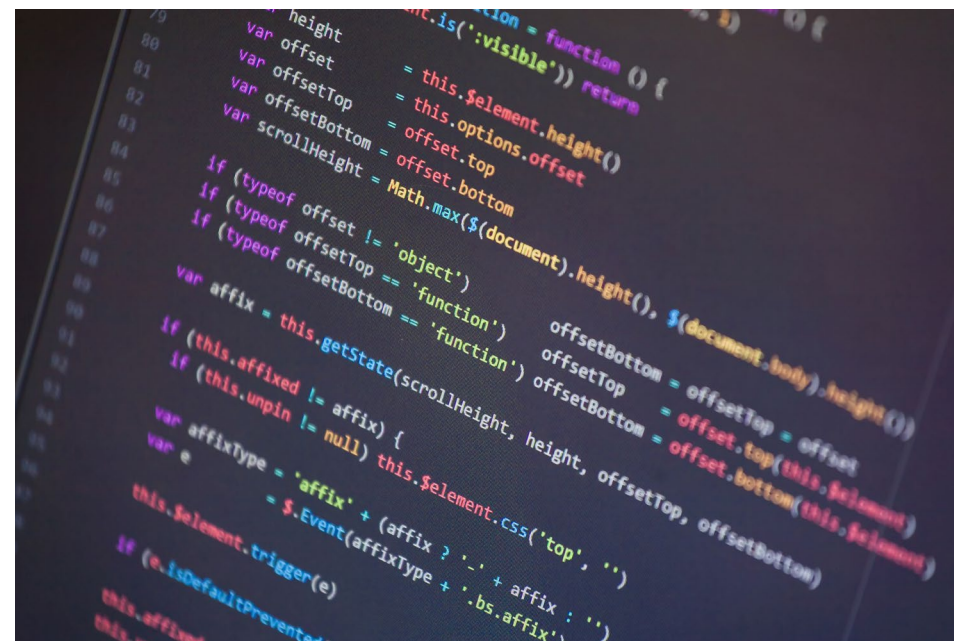
A screenshot of a code editor showing JavaScript code. The code is wrapped in an IIFE (Immediately Invoked Function Expression) and uses 'use strict'. It defines a 'highlight' function that takes a 'node' as an argument. Inside the function, it searches for a regex pattern in the node's data, creates a 'span' element with the class 'highlight', splits the text around the match, clones the middle part, and replaces the original text with the highlighted version. The code is numbered from 27 to 48.

```
27 }  
28 }(this, function($, Sifter, MicroPlugin) {  
29   'use strict';  
30  
31   var highlight = function($element, pattern) {  
32     if (typeof pattern === 'string' && (pattern.length > 0)) {  
33       var regex = (typeof pattern === 'string') ? new RegExp(pattern, 'g') : pattern;  
34  
35       var highlight = function(node) {  
36         var skip = 0;  
37         if (node.nodeType === 3) {  
38           var pos = node.data.search(regex);  
39           if (pos >= 0 && node.data.length > 0) {  
40             var match = node.data.match(regex);  
41             var spannode = document.createElement("span");  
42             spannode.className = 'highlight';  
43             var middlebit = node.splitText(pos);  
44             var endbit = middlebit.splitText(match[0].length);  
45             var middleclone = middlebit.cloneNode(true);  
46             spannode.appendChild(middleclone);  
47             middlebit.parentNode.replaceChild(spannode, middlebit);  
48             skip = 1;  
49           }  
50         }  
51         if (node.childNodes && 1 / (script|style)/i.test(node.nodeName)) {  
52           for (var i = 0; i < node.childNodes.length; i++) {  
53             highlight(node.childNodes[i]);  
54           }  
55         }  
56       };  
57       highlight($element);  
58     }  
59   }  
60 }  
61 }  
62 }  
63 }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }
```

Fonte: Desenvolvimento de Script.
Disponível em: https://as2.ftcdn.net/v2/jpg/01/67/19/37/1000_F_167193773_n13NaWJMBdTTvz1EcBmqvjoeAW0WGzlu.jpg.
Acesso em 31 Julho. 2023

Técnicas avançadas de script

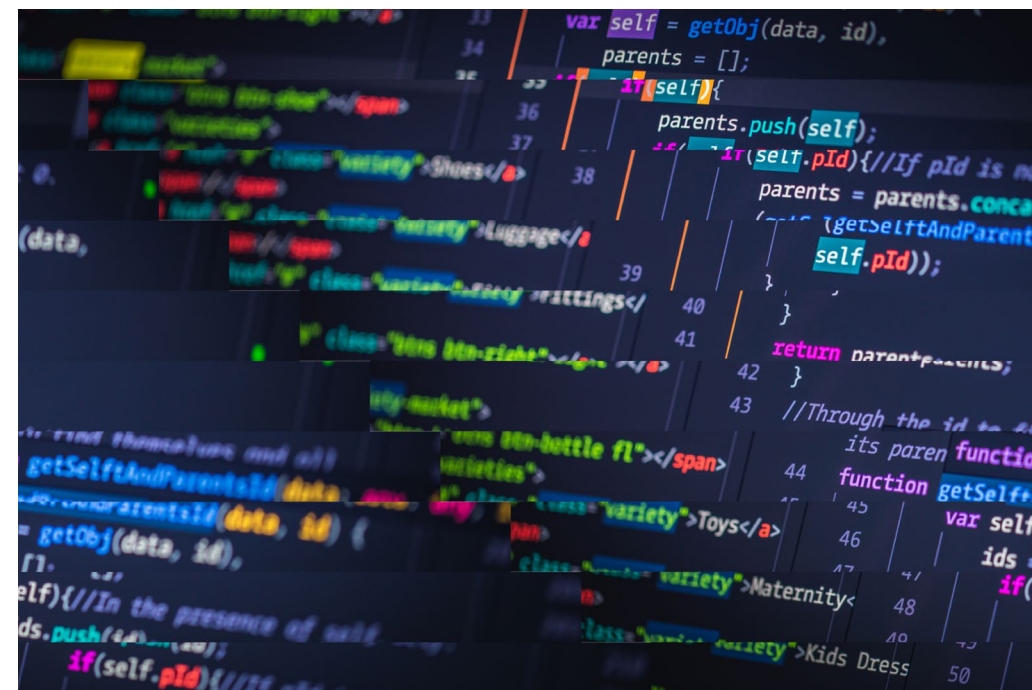
Uso de estruturas condicionais e iterativas: Permite controlar o fluxo de execução dos scripts utilizando estruturas condicionais (execução baseada em condições) e iterativas (repetições), facilitando o tratamento de diferentes cenários ou dados.



Fonte: Desenvolvimento de Script.
Disponível em: https://as1.ftcdn.net/v2/img/01/07/96/20/1000_F_107962078_G7Y5A5EGKraZaLkObrm3eMvbq89OJEnP.jpg
Acesso em 31 Julho. 2023

Técnicas avançadas de script

Uso de expressões regulares: Manipula textos de forma eficiente e precisa ao definir padrões de busca ou substituição em um texto. Possibilita extrair, validar, formatar ou modificar dados conforme necessário.



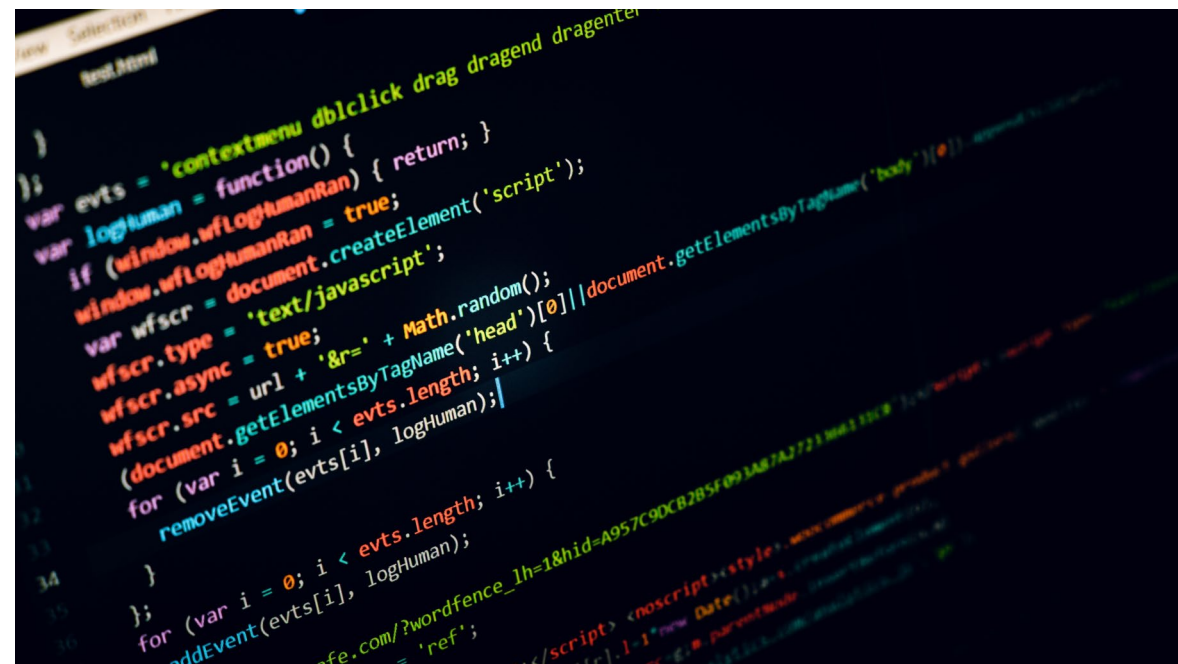
Fonte: Desenvolvimento de Script.

Disponível em: https://as2.ftcdn.net/v2/img/02/78/33/57/1000_F_278335772_JkLllyneRhJr2CJgzkGP9o9rPRosQj1.jpg

Acesso em 31 Julho. 2023

Técnicas avançadas de script

Uso de bibliotecas externas: Permite ampliar as funcionalidades dos scripts através de conjuntos de funções ou classes disponibilizados por outros desenvolvedores. Acesso a recursos que não estão disponíveis nas linguagens ou ferramentas nativas.



Fonte: Desenvolvimento de Script.

Disponível em: https://as1.ftcdn.net/v2/jpg/03/63/62/58/1000_F_363625802_x7s5KlbqPhCjrUW7HZgOQRnDq2RS7KF.jpg.

Acesso em 31 Julho. 2023

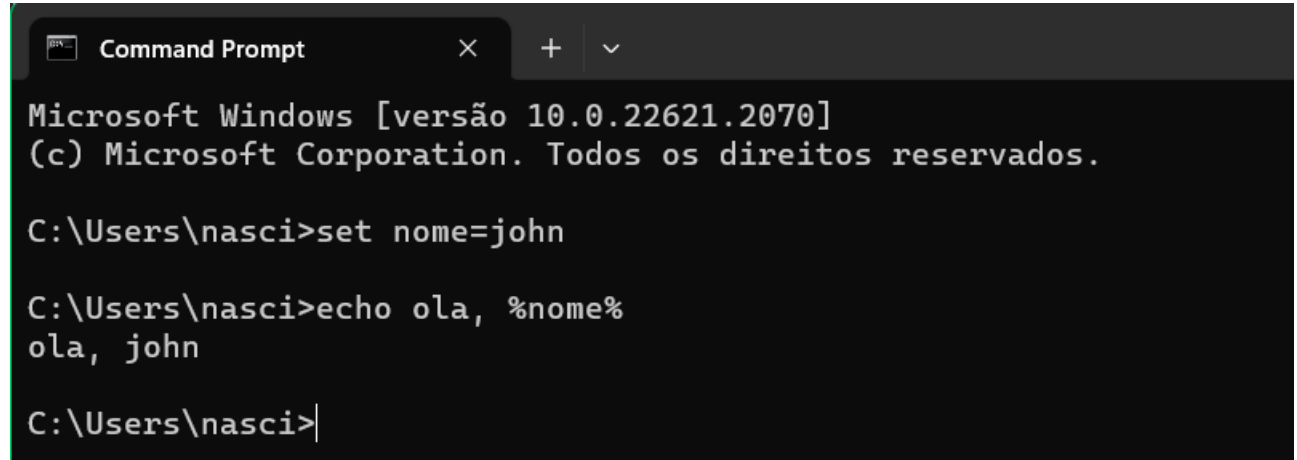
Escrever mensagem em tela

Batch

Abre o CMD – command prompt do Windows, e digite:

set nome=João

echo Olá, %nome%!



```
Command Prompt
Microsoft Windows [versão 10.0.22621.2070]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\nasci>set nome=john

C:\Users\nasci>echo ola, %nome%
ola, john

C:\Users\nasci>
```

Fonte: Autor, 08 de agosto de 2023

Cálculo de variáveis

Batch

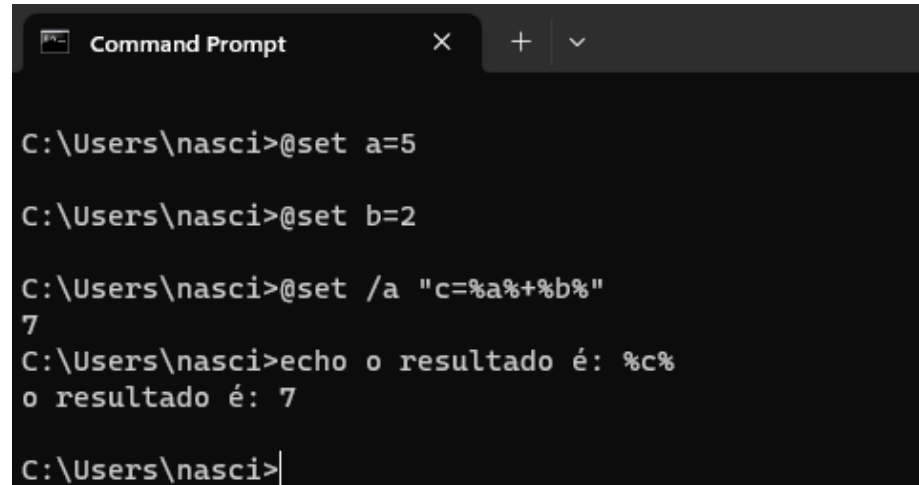
Abre o CMD – command prompt do Windows, e digite:

```
@set a=5
```

```
@set b=2
```

```
@set /a "c=%a%+%b%"
```

```
Echo O resultado é: %c%
```



```
Command Prompt
C:\Users\nasci>@set a=5
C:\Users\nasci>@set b=2
C:\Users\nasci>@set /a "c=%a%+%b%"
7
C:\Users\nasci>echo o resultado é: %c%
o resultado é: 7
C:\Users\nasci>
```

Fonte: Autor, 08 de agosto de 2023



Desenvolvimento de Scripts

WINDOWS

Desenvolvimento de scripts avançados

Operadores relacionais



Batch

Abre o CMD – command prompt do Windows, e digite:

```
@set n1=5
```

```
@set n2=2
```

```
if %n1% gtr %n2% (echo O número 1 é maior que o  
número 2.) else if %n1% lss %n2% (echo O número 1 é  
menor que o número 2.) else (echo Os números são  
iguais.)
```

```
Command Prompt
C:\Users\nasci>@set n1=5
C:\Users\nasci>@set n2=6
C:\Users\nasci>if %n1% gtr %n2% (echo O número 1 é maior que o número 2.) else
if %n1% lss %n2% (echo O número 1 é menor que o número 2.) else (echo Os número
s são iguais.)
O número 1 é menor que o número 2.
```

Fonte: Autor, 08 de agosto de 2023

Manipulação de arquivos



REM Criar uma pasta chamada "MinhaPasta"

```
mkdir MinhaPasta
```

REM Criar arquivo txt

```
type nul > meuarquivo.txt
```

REM Copiar um arquivo chamado "meuarquivo.txt" para a pasta

```
copy meuarquivo.txt MinhaPasta
```

Manipulação de arquivos



REM Mover a pasta "MinhaPasta" para outro local

```
move MinhaPasta NovaLocalizacao\MinhaPasta
```

REM Renomear o arquivo dentro da pasta

```
ren NovaLocalizacao\MinhaPasta\meuarquivo.txt novoarquivo.txt
```

REM Deletar a pasta e seu conteúdo

```
rmdir /s /q NovaLocalizacao\MinhaPasta
```



Desenvolvimento de Scripts

WINDOWS

Desenvolvimento de scripts avançados


```
Write-Host "Hello, World!"
```

```
if ($a -gt $b) {  
    Write-Host "A é maior que B"  
} else {  
    Write-Host "B é maior que A"  
}
```

```
for ($i=1; $i -le 5; $i++) {  
    Write-Host "Iteração $i"  
}
```

```
function Saudacao {  
    param (  
        [string]$nome  
    )  
    Write-Host "Olá, $nome!"  
}
```

```
# Chamando a função  
Saudacao -nome "Amigo"
```

```
function Saudacao {  
    param (  
        [string]$nome,  
        [string]$cumprimento = "Olá"  
    )  
    Write-Host "$cumprimento, $nome!"  
}  
  
# Chamando a função  
Saudacao -nome "Amigo"  
Saudacao -nome "Amigo" -cumprimento "Bom dia"
```



Desenvolvimento de Scripts

WINDOWS

Desenvolvimento de scripts avançados

Script de Remoção de Malwares no Registro



```
$chaveMaliciosa = "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run\EncryptorMalicioso"
```

```
if (Test-Path $chaveMaliciosa) {  
    Write-Host "Removendo chave maliciosa do Registro..."  
    Remove-Item -Path $chaveMaliciosa -Force  
    Write-Host "Chave maliciosa removida com sucesso."  
} else {  
    Write-Host "Nenhuma chave maliciosa encontrada no Registro."  
}
```

Criptografia e descriptografia de mensagem



```
# Encryption Key (16, 24, or 32 bytes)
```

```
$encryptionKey = [Text.Encoding]::UTF8.GetBytes("MySecretEncryptionKey")
```

```
# Text to encrypt
```

```
$textToEncrypt = "SENAI 123."
```

```
# Convert the text to bytes
```

```
$bytesToEncrypt = [Text.Encoding]::UTF8.GetBytes($textToEncrypt)
```

```
# Create AES encryption object
```

```
$aes = [System.Security.Cryptography.Aes]::Create()
```

```
$aes.Mode = [System.Security.Cryptography.CipherMode]::CBC
```

```
$aes.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7
```

```
$aes = [System.Security.Cryptography.Aes]::Create()
```

```
$aes.Mode = [System.Security.Cryptography.CipherMode]::CBC
```

```
$aes.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7
```

Criptografia e descriptografia de mensagem



```
# Generate a random IV (Initialization Vector)
$aes.GenerateIV()

# Create an encryption stream
$encryptor = $aes.CreateEncryptor()

# Encrypt the bytes
$encryptedBytes = $encryptor.TransformFinalBlock($bytesToEncrypt, 0, $bytesToEncrypt.Length)

# Convert the encrypted bytes to Base64 for storage
$encryptedText = [Convert]::ToBase64String($aes.IV + $encryptedBytes)

# Display the encrypted text
Write-Host "Encrypted Text: $encryptedText"
```




Desenvolvimento de Scripts

WINDOWS

Desenvolvimento de scripts avançados

Criptografia e descriptografia de mensagem



```
# Convert the Base64 encrypted text back to bytes
$encryptedBytesWithIV = [Convert]::FromBase64String($encryptedText)
$iv = $encryptedBytesWithIV[0..15]
$encryptedBytesOnly = $encryptedBytesWithIV[16..($encryptedBytesWithIV.Length - 1)]

# Set the IV and decrypt
$aes.IV = $iv
$decryptor = $aes.CreateDecryptor()
$decryptedBytes = $decryptor.TransformFinalBlock($encryptedBytesOnly, 0, $encryptedBytesOnly.Length)

# Convert the decrypted bytes back to text
$decryptedText = [Text.Encoding]::UTF8.GetString($decryptedBytes)

# Display the decrypted text
Write-Host "Decrypted Text: $decryptedText"
```

Verificação de Políticas de Segurança



Get-ExecutionPolicy
Get-ProcessMitigation -System
Get-MpPreference | Select-Object -Property *Detection*

Este script exibe informações sobre a política de execução, políticas de mitigação de processo e preferências de detecção do Windows Defender.

Varredura de Portas e Conexões Ativas



```
Get-NetTCPConnection | Select-Object -Property LocalAddress, LocalPort, RemoteAddress, RemotePort, State  
Test-NetConnection -ComputerName localhost -Port 80
```

Este script lista as conexões TCP ativas e verifica a conectividade com uma porta específica.



Desenvolvimento de Scripts

WINDOWS

Desenvolvimento de scripts avançados

Verificação de Atividades de Logon



```
Get-WinEvent -LogName Security | Where-Object { $_.Id -eq 4624 -or $_.Id -eq 4634 } | Select-Object -Property TimeCreated, Id, Message
```

Este script analisa eventos de logon no log de segurança do Windows.

Análise de Processos Suspeitos



```
Get-Process | Where-Object { $_.Path -eq $null -and $_.Handles -gt 500 -and $_.CPU -gt 50 }
```

Este script lista processos que não têm um caminho de arquivo associado, têm um número significativo de identificadores de objeto ou estão usando uma quantidade considerável de CPU.

Conclusão



Lembre-se de que esses scripts são apenas pontos de partida e podem precisar ser ajustados para atender às necessidades específicas do seu ambiente. Execute scripts de segurança com cuidado e teste-os primeiro em um ambiente controlado antes de implantá-los em um ambiente de produção.