$scriptPath = "C:\Scripts\NetworkSecurityScript.ps1"

$logPath = "C:\Scripts\SecurityLog.txt"

New-Item -ItemType Directory -Path "C:\Scripts" -Force | Out-Null

$MyInvocation.MyCommand.ScriptBlock | Out-File -FilePath $scriptPath -Force

"Script initialized at $(Get-Date)" | Out-File -FilePath $logPath -Append

# Register scheduled task for startup

$action = New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument "-NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File `"$scriptPath`""

$startupTrigger = New-ScheduledTaskTrigger -AtStartup

$principal = New-ScheduledTaskPrincipal -UserId "SYSTEM" -LogonType ServiceAccount -RunLevel Highest

$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -ExecutionTimeLimit (New-TimeSpan -Hours 0)

try {

Register-ScheduledTask -TaskName "NetworkSecurityMonitor" -Action $action -Trigger $startupTrigger -Principal $principal -Settings $settings -Force -ErrorAction Stop | Out-Null

"Scheduled task registered at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to register scheduled task: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned -Force -ErrorAction Stop

"Set ExecutionPolicy to RemoteSigned" | Out-File -FilePath $logPath -Append

$vulnScan = Get-CimInstance -ClassName Win32\_ComputerSystem -ErrorAction Stop | Select-Object -Property \*

$osVersion = (Get-CimInstance Win32\_OperatingSystem -ErrorAction Stop).Version

$hotfixes = Get-HotFix -ErrorAction Stop | Select-Object HotFixID, InstalledOn

"System scan completed: OS Version $osVersion, Hotfixes count: $($hotfixes.Count)" | Out-File -FilePath $logPath -Append

$services = @("Telnet", "XblAuthManager", "XblGameSave", "XboxGipSvc")

foreach ($service in $services) {

if (Get-Service -Name $service -ErrorAction SilentlyContinue) {

try {

Set-Service -Name $service -StartupType Disabled -ErrorAction Stop

Stop-Service -Name $service -Force -ErrorAction SilentlyContinue

"Disabled and stopped service: $service" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable service ${service}: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

}

netsh advfirewall set allprofiles state on | Out-Null

netsh advfirewall set allprofiles firewallpolicy blockinbound,allowoutbound | Out-Null

try {

New-NetFirewallRule -DisplayName "Block Common Ports" -Direction Inbound -Action Block -Protocol TCP -LocalPort 135,139,445 -ErrorAction Stop | Out-Null

New-NetFirewallRule -DisplayName "Block Common Ports UDP" -Direction Inbound -Action Block -Protocol UDP -LocalPort 135,139,445 -ErrorAction Stop | Out-Null

"Firewall rules configured" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to configure firewall rules: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

AuditPol /set /category:"Account Logon" /success:enable /failure:enable | Out-Null

AuditPol /set /category:"Logon/Logoff" /success:enable /failure:enable | Out-Null

AuditPol /set /category:"Object Access" /success:enable /failure:enable | Out-Null

"Audit policies enabled" | Out-File -FilePath $logPath -Append

try {

Set-SmbServerConfiguration -EnableSMB1Protocol $false -Force -ErrorAction Stop

Disable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol -NoRestart -ErrorAction Stop

"SMBv1 disabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable SMBv1: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

Set-ExecutionPolicy -ExecutionPolicy AllSigned -Scope LocalMachine -Force -ErrorAction Stop

"Set ExecutionPolicy to AllSigned for LocalMachine" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to set ExecutionPolicy: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$hostsPath = "$env:windir\System32\drivers\etc\hosts"

$monitor = New-Object System.IO.FileSystemWatcher

$monitor.Path = Split-Path $hostsPath -Parent

$monitor.Filter = Split-Path $hostsPath -Leaf

$monitor.EnableRaisingEvents = $true

Register-ObjectEvent $monitor Changed -SourceIdentifier HostsFileMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "Hosts file modified."

"Hosts file change detected at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"Hosts file monitoring enabled" | Out-File -FilePath $logPath -Append

$users = Get-LocalUser -ErrorAction Stop | Where-Object { $\_.Enabled -eq $true }

foreach ($user in $users) {

if ($user.LastLogon -lt (Get-Date).AddDays(-90)) {

try {

Disable-LocalUser -Name $user.Name -ErrorAction Stop

"Disabled inactive user: $($user.Name)" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable user $($user.Name): $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

}

try {

$newAdminName = "DisabledAdmin$(Get-Random -Minimum 1000 -Maximum 9999)"

if (Get-LocalUser -Name "Administrator" -ErrorAction SilentlyContinue) {

Set-LocalUser -Name "Administrator" -Description "Disabled Default Admin" -AccountNeverExpires -ErrorAction Stop

Rename-LocalUser -Name "Administrator" -NewName $newAdminName -ErrorAction Stop

net user $newAdminName /active:no | Out-Null

"Administrator account disabled and renamed to $newAdminName" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to rename/disable Administrator account: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$tasks = Get-ScheduledTask -ErrorAction Stop | Where-Object { $\_.State -eq "Ready" }

foreach ($task in $tasks) {

if ($task.Principal.UserId -notlike "NT AUTHORITY\\*" -and $task.Principal.UserId -notlike "SYSTEM") {

try {

Disable-ScheduledTask -TaskName $task.TaskName -TaskPath $task.TaskPath -ErrorAction Stop

"Disabled suspicious task: $($task.TaskName)" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable task $($task.TaskName): $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

}

try {

Set-MpPreference -EnableExploitProtection $true -ErrorAction Stop

Set-MpPreference -EnableNetworkProtection Enabled -ErrorAction Stop

"Exploit and Network Protection enabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to enable Exploit/Network Protection: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$regPath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"

$regMonitor = [Microsoft.Win32.RegistryKey]::OpenBaseKey([Microsoft.Win32.RegistryHive]::LocalMachine, [Microsoft.Win32.RegistryView]::Default)

$regMonitor = $regMonitor.OpenSubKey("SOFTWARE\Microsoft\Windows\CurrentVersion\Run", $true)

$regMonitor.EnableRaisingEvents = $true

Register-ObjectEvent $regMonitor Changed -SourceIdentifier RegistryMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4657 -Message "Registry Run key modified."

"Registry Run key change detected at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"Registry monitoring enabled" | Out-File -FilePath $logPath -Append

$dnsQueries = Get-DnsClientCache -ErrorAction Stop

$dnsLog = New-Object System.IO.FileSystemWatcher

$dnsLog.Path = "$env:windir\System32\LogFiles"

$dnsLog.Filter = "\*.log"

$dnsLog.EnableRaisingEvents = $true

Register-ObjectEvent $dnsLog Changed -SourceIdentifier DNSLogMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4662 -Message "DNS log file modified."

"DNS log change detected at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"DNS monitoring enabled" | Out-File -FilePath $logPath -Append

$criticalFiles = @("$env:windir\System32\cmd.exe", "$env:windir\System32\powershell.exe")

foreach ($file in $criticalFiles) {

try {

$hash = Get-FileHash -Path $file -Algorithm SHA256 -ErrorAction Stop

$hash | Export-Clixml -Path "$($env:temp)\$($file.Replace('\','\_')).xml" -ErrorAction Stop

"File hash generated for: $file" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to generate hash for ${file}: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

try {

Disable-PSRemoting -Force -ErrorAction Stop

"PowerShell Remoting disabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable PowerShell Remoting: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

$sysmonConfig = @"

<Sysmon schemaversion="4.81">

<EventFiltering>

<ProcessCreate onmatch="include"/>

<FileCreateTime onmatch="include"/>

<NetworkConnect onmatch="include"/>

</EventFiltering>

</Sysmon>

"@

$sysmonConfig | Out-File -FilePath "$($env:temp)\sysmonconfig.xml" -ErrorAction Stop

Invoke-WebRequest -Uri "https://download.sysinternals.com/files/Sysmon.zip" -OutFile "$($env:temp)\Sysmon.zip" -ErrorAction Stop

Expand-Archive -Path "$($env:temp)\Sysmon.zip" -DestinationPath "$($env:temp)\Sysmon" -ErrorAction Stop

Start-Process -FilePath "$($env:temp)\Sysmon\Sysmon64.exe" -ArgumentList "-accepteula -i `"$($env:temp)\sysmonconfig.xml`"" -Wait -ErrorAction Stop

"Sysmon installed and configured" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to install Sysmon: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$logReview = Get-WinEvent -LogName "Security" -MaxEvents 1000 -ErrorAction Stop | Where-Object { $\_.LevelDisplayName -eq "Error" -or $\_.LevelDisplayName -eq "Warning" }

"Security log review completed, events found: $($logReview.Count)" | Out-File -FilePath $logPath -Append

try {

$dgReadiness = Get-CimInstance -ClassName Win32\_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard -ErrorAction Stop

if ($dgReadiness.AvailableSecurityProperties -contains 8) {

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\LSA" -Name "LsaCfgFlags" -Value 2 -ErrorAction Stop

"Credential Guard enabled" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to enable Credential Guard: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$adminGroup = Get-LocalGroupMember -Group "Administrators" -ErrorAction Stop

foreach ($member in $adminGroup) {

if ($member.Name -notlike "\*\Administrator" -and $member.Name -notlike "\*\DisabledAdmin\*") {

try {

Remove-LocalGroupMember -Group "Administrators" -Member $member.Name -ErrorAction Stop

"Removed excessive admin: $($member.Name)" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to remove admin $($member.Name): $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

}

try {

if ((Get-WmiObject Win32\_OperatingSystem).ProductType -ne 1) { # Check if not Windows Home

$applockerPolicy = @"

<AppLockerPolicy Version="1">

<RuleCollection Type="Exe" EnforcementMode="Enabled">

<FilePathRule Id="921cc481-6e17-4653-8f75-050b80acca20" Name="All executables in Program Files" Description="" UserOrGroupSid="S-1-1-0" Action="Allow">

<Conditions>

<FilePathCondition Path="%PROGRAMFILES%\\*" />

</Conditions>

</FilePathRule>

</RuleCollection>

</AppLockerPolicy>

"@

$applockerPolicy | Out-File -FilePath "$($env:temp)\applocker.xml" -ErrorAction Stop

Set-AppLockerPolicy -XmlPolicy "$($env:temp)\applocker.xml" -ErrorAction Stop

"AppLocker policy applied" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to apply AppLocker policy: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

$filter = Set-WmiInstance -Class \_\_EventFilter -Namespace "root\subscription" -Arguments @{Name="WMIActivity";EventNamespace="root\cimv2";QueryLanguage="WQL";Query="SELECT \* FROM \_\_InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32\_Process'"} -ErrorAction Stop

$consumer = Set-WmiInstance -Class CommandLineEventConsumer -Namespace "root\subscription" -Arguments @{Name="WMIActivityConsumer";CommandLineTemplate="powershell.exe -Command \`"Write-EventLog -LogName Security -Source 'Microsoft-Windows-Security-Auditing' -EventId 4688 -Message 'WMI Process Created'\`""} -ErrorAction Stop

$binding = Set-WmiInstance -Class \_\_FilterToConsumerBinding -Namespace "root\subscription" -Arguments @{Filter=$filter;Consumer=$consumer} -ErrorAction Stop

"WMI monitoring enabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to enable WMI monitoring: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$execMonitor = New-Object System.IO.FileSystemWatcher

$execMonitor.Path = "$env:windir\System32"

$execMonitor.Filter = "\*.exe"

$execMonitor.EnableRaisingEvents = $true

Register-ObjectEvent $execMonitor Changed -SourceIdentifier ExecMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "System executable modified."

"Executable change detected at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"Executable monitoring enabled" | Out-File -FilePath $logPath -Append

try {

$ipBlacklist = @("1.1.1.1", "8.8.8.8")

foreach ($ip in $ipBlacklist) {

New-NetFirewallRule -DisplayName "Block Suspicious IP $ip" -Direction Outbound -Action Block -RemoteAddress $ip -ErrorAction Stop | Out-Null

"Blocked IP: $ip" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to block IPs: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

Set-MpPreference -EnableControlledFolderAccess Enabled -ErrorAction Stop

"Controlled Folder Access enabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to enable Controlled Folder Access: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$hiddenTasks = Get-ScheduledTask -ErrorAction Stop | Where-Object { $\_.Settings.Hidden -eq $true }

foreach ($task in $hiddenTasks) {

try {

Disable-ScheduledTask -TaskName $task.TaskName -TaskPath $task.TaskPath -ErrorAction Stop

"Disabled hidden task: $($task.TaskName)" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable hidden task $($task.TaskName): $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

$installedPrograms = Get-WmiObject -Class Win32\_Product -ErrorAction Stop | Select-Object Name, Version

"Installed programs scanned: $($installedPrograms.Count)" | Out-File -FilePath $logPath -Append

Register-WmiEvent -Query "SELECT \* FROM \_\_InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32\_UserAccount' AND TargetInstance.SID LIKE 'S-1-5-21%-%-500'" -SourceIdentifier NewAdminMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4720 -Message "New admin account created."

"New admin account detected at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"Admin account monitoring enabled" | Out-File -FilePath $logPath -Append

$secPolMonitor = New-Object System.IO.FileSystemWatcher

$secPolMonitor.Path = "$env:windir\System32\GroupPolicy"

$secPolMonitor.EnableRaisingEvents = $true

Register-ObjectEvent $secPolMonitor Changed -SourceIdentifier SecPolMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4719 -Message "Security policy modified."

"Security policy change detected at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"Security policy monitoring enabled" | Out-File -FilePath $logPath -Append

$startupFolders = @("$env:ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp", "$env:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup")

foreach ($folder in $startupFolders) {

Get-ChildItem -Path $folder -ErrorAction SilentlyContinue | ForEach-Object {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "Startup folder item detected: $($\_.Name)"

"Startup folder item detected - $($\_.Name) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

try {

Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer" -Name "RestrictActiveXInstall" -Value 1 -ErrorAction Stop

"ActiveX installation restricted" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to restrict ActiveX installation: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$processes = Get-Process -ErrorAction Stop | Where-Object { $\_.CPU -gt 1000 -or $\_.WorkingSet64 -gt 100MB }

foreach ($proc in $processes) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4688 -Message "High resource process detected: $($proc.Name)"

"High resource process - $($proc.Name) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

$compressionTools = @("WinRAR.exe", "7z.exe")

foreach ($tool in $compressionTools) {

$process = Get-Process -Name $tool -ErrorAction SilentlyContinue

if ($process) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4688 -Message "Compression tool detected: $tool"

"Compression tool detected - $tool at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

$certMonitor = Get-ChildItem -Path Cert:\LocalMachine\Root -ErrorAction Stop

foreach ($cert in $certMonitor) {

if ($cert.NotAfter -lt (Get-Date)) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 5061 -Message "Expired root certificate detected: $($cert.Subject)"

"Expired certificate - $($cert.Subject) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

$updateSession = New-Object -ComObject Microsoft.Update.Session

$updateSearcher = $updateSession.CreateUpdateSearcher()

$updates = $updateSearcher.Search("IsInstalled=0 and Type='Software' and IsHidden=0")

foreach ($update in $updates.Updates) {

Write-EventLog -LogName System -Source "Microsoft-Windows-WindowsUpdateClient" -EventId 19 -Message "Missing update: $($update.Title)"

"Missing update - $($update.Title) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

wevtutil set-log Security /rt:true /q:true | Out-Null

"Security event log retention enabled" | Out-File -FilePath $logPath -Append

try {

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server" -Name "fDenyTSConnections" -Value 1 -ErrorAction Stop

"Remote desktop connections disabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable Remote Desktop: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$criticalFolders = @("$env:windir\System32", "$env:ProgramFiles")

foreach ($folder in $criticalFolders) {

$acl = Get-Acl -Path $folder -ErrorAction Stop

$acl.Access | Where-Object { $\_.IdentityReference -notlike "NT AUTHORITY\\*" } | ForEach-Object {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "Non-system account with access to $folder"

"Non-system access to $folder detected at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

AuditPol /set /subcategory:"File System" /success:enable /failure:enable | Out-Null

"File system auditing enabled" | Out-File -FilePath $logPath -Append

$executables = Get-ChildItem -Path "$env:ProgramFiles" -Recurse -Include \*.exe -ErrorAction Stop

foreach ($exe in $executables) {

try {

$sig = Get-AuthenticodeSignature -FilePath $exe.FullName -ErrorAction Stop

if ($sig.Status -ne "Valid") {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4657 -Message "Invalid signature on executable: $($exe.FullName)"

"Invalid executable signature - $($exe.FullName) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to check signature for $($exe.FullName): $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

}

try {

Disable-WindowsOptionalFeature -Online -FeatureName MicrosoftWindowsPowerShellV2Root -NoRestart -ErrorAction Stop

"PowerShell v2 disabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable PowerShell v2: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$services = Get-WmiObject -Class Win32\_Service -ErrorAction Stop

foreach ($service in $services) {

$sd = $service.GetSecurityDescriptor().Descriptor

if ($sd.DACL -notlike "\*SYSTEM\*") {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4670 -Message "Service with excessive permissions: $($service.Name)"

"Excessive service permissions - $($service.Name) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

$lnkFiles = Get-ChildItem -Path $env:USERPROFILE -Recurse -Include \*.lnk -ErrorAction Stop

foreach ($lnk in $lnkFiles) {

$shell = New-Object -ComObject WScript.Shell

$shortcut = $shell.CreateShortcut($lnk.FullName)

if ($shortcut.TargetPath -notlike "$env:ProgramFiles\*") {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "Suspicious LNK file detected: $($lnk.FullName)"

"Suspicious LNK file - $($lnk.FullName) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

try {

Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Office\16.0\Word\Security" -Name "VBAWarnings" -Value 4 -ErrorAction Stop

Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Office\16.0\Excel\Security" -Name "VBAWarnings" -Value 4 -ErrorAction Stop

"Office macros restricted" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to restrict Office macros: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$netstat = netstat -ano | Select-String "ESTABLISHED"

foreach ($conn in $netstat) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 5156 -Message "Established connection: $conn"

"Network connection - $conn at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

try {

Set-MpPreference -AttackSurfaceReductionRules\_Ids "56a863a9-875e-4185-98a7-b882c64b5ce5" -AttackSurfaceReductionRules\_Actions Enabled -ErrorAction Stop

"ASR rules enabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to enable ASR rules: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

$secPol = @"

[Unicode]

Unicode=yes

[Version]

signature="$CHICAGO$"

Revision=1

[Security]

MinimumPasswordLength=12

PasswordComplexity=1

LockoutBadCount=5

"@

$secPol | Out-File -FilePath "$($env:temp)\secpol.inf" -Encoding ascii -ErrorAction Stop

if (Test-Path "$($env:temp)\secpol.sdb") { Remove-Item "$($env:temp)\secpol.sdb" -Force -ErrorAction Stop }

secedit /configure /db "$($env:temp)\secpol.sdb" /cfg "$($env:temp)\secpol.inf" /overwrite /quiet | Out-Null

"Security policy applied" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to apply security policy: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\USBSTOR" -Name "Start" -Value 4 -ErrorAction Stop

"USB storage disabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to disable USB storage: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$honeypotPath = "$($env:temp)\honeypot.txt"

New-Item -Path $honeypotPath -ItemType File -ErrorAction Stop | Out-Null

$honeypotMonitor = New-Object System.IO.FileSystemWatcher

$honeypotMonitor.Path = Split-Path $honeypotPath -Parent

$honeypotMonitor.Filter = Split-Path $honeypotPath -Leaf

$honeypotMonitor.EnableRaisingEvents = $true

Register-ObjectEvent $honeypotMonitor Changed -SourceIdentifier HoneypotMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "Honeypot file accessed."

"Honeypot file accessed at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"Honeypot monitoring enabled" | Out-File -FilePath $logPath -Append

$serviceAccounts = Get-WmiObject -Class Win32\_Service -ErrorAction Stop | Where-Object { $\_.StartName -notlike "LocalSystem" }

foreach ($account in $serviceAccounts) {

if ($account.StartName -like "\*Administrator\*") {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4670 -Message "Service account with admin privileges: $($account.Name)"

"Admin service account - $($account.Name) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

$tempFolders = @("$env:temp", "$env:APPDATA")

foreach ($folder in $tempFolders) {

Get-ChildItem -Path $folder -Recurse -File -ErrorAction Stop | Where-Object { $\_.Extension -in @(".exe", ".ps1", ".bat") } | ForEach-Object {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "Suspicious file in $folder - $($\_.Name)"

"Suspicious file in $folder - $($\_.Name) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

$report = "Security Report - $(Get-Date -Format 'yyyy-MM-dd')"

$report += "`nVulnerabilities: $($vulnScan | Out-String)"

$report += "`nHotfixes: $($hotfixes | Out-String)"

$report += "`nSuspicious Logs: $($logReview | Out-String)"

$report | Out-File -FilePath "$($env:temp)\security\_report\_$(Get-Date -Format 'yyyyMMdd').txt" -ErrorAction Stop

"Security report generated at $(Get-Date)" | Out-File -FilePath $logPath -Append

$emailTraffic = Get-NetTCPConnection -ErrorAction Stop | Where-Object { $\_.RemotePort -eq 25 }

foreach ($conn in $emailTraffic) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 5156 -Message "SMTP connection detected: $($conn.RemoteAddress)"

"SMTP connection - $($conn.RemoteAddress) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

$fimPaths = @("$env:windir\System32", "$env:ProgramFiles")

foreach ($path in $fimPaths) {

$fimMonitor = New-Object System.IO.FileSystemWatcher

$fimMonitor.Path = $path

$fimMonitor.EnableRaisingEvents = $true

Register-ObjectEvent $fimMonitor Changed -SourceIdentifier FIMMonitor -Action {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 4663 -Message "File modified in critical path: $path"

"File change in $path at $(Get-Date)" | Out-File -FilePath "C:\Scripts\SecurityLog.txt" -Append

} | Out-Null

"File integrity monitoring enabled for $path" | Out-File -FilePath $logPath -Append

}

try {

$bitLocker = Get-BitLockerVolume -MountPoint $env:SystemDrive -ErrorAction Stop

if ($bitLocker.ProtectionStatus -eq "Off" -and (Get-CimInstance -ClassName Win32\_Tpm -Namespace root\cimv2\Security\MicrosoftTpm)) {

Enable-BitLocker -MountPoint $env:SystemDrive -EncryptionMethod Aes256 -UsedSpaceOnly -TpmProtector -ErrorAction Stop

"BitLocker enabled" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to enable BitLocker: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

try {

$networkTools = @("nmap.exe", "wireshark.exe")

foreach ($tool in $networkTools) {

New-NetFirewallRule -DisplayName "Block $tool" -Direction Outbound -Action Block -Program $tool -ErrorAction Stop | Out-Null

"Blocked network tool: $tool" | Out-File -FilePath $logPath -Append

}

}

catch {

"Failed to block network tools: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$exfilMonitor = Get-NetTCPConnection -ErrorAction Stop | Where-Object { $\_.RemoteAddress -like "pastebin.com" -or $\_.RemoteAddress -like "\*.telegram.org" }

foreach ($conn in $exfilMonitor) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId 5156 -Message "Potential data exfiltration to: $($conn.RemoteAddress)"

"Potential data exfiltration to - $($conn.RemoteAddress) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

try {

Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer" -Name "NoControlPanel" -Value 1 -ErrorAction Stop

"Control Panel access restricted" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to restrict Control Panel: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

$env:\_\_PSLockdownPolicy = 4

"PowerShell constrained mode enabled" | Out-File -FilePath $logPath -Append

$psLogs = Get-WinEvent -LogName "Microsoft-Windows-PowerShell/Operational" -MaxEvents 1000 -ErrorAction Stop | Where-Object { $\_.Id -in @(4103, 4104) }

foreach ($log in $psLogs) {

Write-EventLog -LogName Security -Source "Microsoft-Windows-Security-Auditing" -EventId $log.Id -Message "PowerShell activity: $($log.Message)"

"PowerShell activity logged - $($log.Id) at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

try {

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager" -Name "SafeDLLSearchMode" -Value 1 -ErrorAction Stop

"Safe DLL search mode enabled" | Out-File -FilePath $logPath -Append

}

catch {

"Failed to enable Safe DLL search mode: $($\_.Exception.Message)" | Out-File -FilePath $logPath -Append

}

while ($true) {

Start-Sleep -Seconds 60

"Script still running at $(Get-Date)" | Out-File -FilePath $logPath -Append

}

}

catch {

$errorMessage = "Error occurred: $($\_.Exception.Message) at $(Get-Date)"

$errorMessage | Out-File -FilePath $logPath -Append

Write-EventLog -LogName Application -Source "Application" -EventId 1000 -Message $errorMessage -ErrorAction SilentlyContinue

}