BASIC CONFIGURATION BENCHMARK CHECKS

Ensure Web content is on a Non-System Partition

```
$test = Get-Content (Join-Path -Path $Env:SystemRoot -ChildPath
'System32\inetsrv\config\applicationHost.config')

If (
    (Test-Path -Path (Join-Path -Path $Env:SystemDrive -ChildPath 'inetpub')) -And
    $test -Match [RegEx]::Escape((Join-Path -Path $Env:SystemDrive -ChildPath
'inetpub'))
) {
    $true
} Else {
    $false
}
```

Ensure ‘Directory Browsing’ is Set to Disable

```
Get-Website | Foreach-Object {
  [PSCustomObject]@{
    "Name" = $_.Name
    "DirectoryBrowsing"  = (Get-WebConfigurationProperty -Filter
"/system.webServer/directoryBrowse" -PSPath "IIS:\Sites\$($_.Name)" -Name
"enabled").Value
}
}
```

Ensure ‘Application pool identity’ is configured for all application pools

```
$AppPools = Get-ChildItem 'IIS:\AppPools'

$AppPools | Foreach-Object {
  $processModels = Get-ItemProperty "IIS:\AppPools\$($_.Name)" | Select-Object
-ExpandProperty 'processModel'
```

```
  If ( $processModels.identityType -NE 'ApplicationPoolIdentity' ) {
    $false
  } Else {
    $true
  }
}
```

Ensure ‘unique application pools’ is set for sites

```
[Bool](Get-WebApplication | Group-Object -Property 'applicationPool' | Where-Object 'count' -GT 1)

#6 - Ensure ‘application pool identity’ is configured for anonymous user identity
Get-ChildItem 'IIS:\Sites' | Foreach-Object {
  $anonAuth = (Get-WebConfigurationProperty -Filter '/system.webServer/security/authentication/anonymousAuthentication' -PSPath "IIS:\Sites\$($_.Name)" -Name "userName").Value

  [PSCustomObject]@{
    "Name" = $_.Name
    "AnonAuth" = $(If($anonAuth -EQ '') {$false}Else{$true})
  }
}
```

Ensure WebDav feature is disabled

```
[Bool](Get-WindowsFeature -Name 'Web-DAV-Publishing' | Where-Object Installed -EQ $true)
```

Ensure ‘global authorization rule’ is set to restrict access

```
if ((Get-WindowsFeature Web-Url-Auth).Installed -EQ $true) {
  Get-WebSite | ForEach-Object {
     $site = $_.Name
     $config = Get-WebConfiguration -Filter "system.webServer/security/authorization" -PSPath "IIS:\Sites\$($_.Name)"

     $config.GetCollection() | ForEach-Object {
     $accessType = ($_.Attributes | Where-Object Name -EQ 'accessType').Value
```

```
     $users = ($_.Attributes | Where-Object Name -EQ
'users').Value
     $roles = ($_.Attributes | Where-Object Name -EQ
'roles').Value

     If (($accessType -EQ "Allow" -Or $accessType -EQ 0) -And
($users -eq "*" -or $roles -eq "?")) {
          [PSCustomObject]@{
               "Name" = $site
               "AccessType" = $accessType
               "Users" = $users
               "Roles" = $roles
               "Pass" = $false
          }
     } Else {
          [PSCustomObject]@{
               "Name" = $site
               "AccessType" = $accessType
               "Users" = $users
               "Roles" = $roles
               "Pass" = $true
          }
     }
     }
  }
}
```

Ensure access to sensitive site features is restricted to authenticated principals only

```
Get-Website | Foreach-Object {
  $mode = (Get-WebConfiguration -Filter
'system.web/authentication' -PSPath
"IIS:\sites\$($_.Name)").mode

  If (($mode -NE 'forms') -And ($mode -NE 'Windows')) {
     [PSCustomObject]@{
     "Name" = $_.Name
     "Principals" = $false
```

```
    }
  } Else {
    [PSCustomObject]@{
    "Name" = $_.Name
    "Principals" = $true
    }
  }
}
```

Ensure ‘forms authentication’ requires SSL

```
Get-Website | Foreach-Object {
  $config = (Get-WebConfiguration -Filter
'system.web/authentication' -PSPath "IIS:\sites\$($_.Name)")

  If ($config.mode -EQ 'forms') {
    [PSCustomObject]@{
        "Name" = $_.Name
        "SSL"  = $config.Forms.RequireSSL
    }
  }
}
```

Ensure ‘forms authentication’ is set to use cookies

```
Get-Website | Foreach-Object {
  $config = (Get-WebConfiguration -Filter
'/system.web/authentication' -PSPath "IIS:\sites\$($_.Name)")

  If ($config.mode -EQ 'forms') {
    [PSCustomObject]@{
        "Name"    = $_.Name
        "CookieMode" = $(If($config.Forms.Cookieless -NE
'UseCookie') { $false } Else { $true })
    }
  }
}
```

Ensure ‘cookie protection mode’ is configured for forms authentication

```
Get-Website | Foreach-Object {
  $config = (Get-WebConfiguration -Filter
'/system.web/authentication' -PSPath "IIS:\sites\$($_.Name)")

  If ($config.mode -EQ 'forms') {
     [PSCustomObject]@{
          "Name" = $_.Name
          "CookieProtection" = $(If($config.Forms.protection -NE
'All') { $false } Else { $true })
     }
  }
}
```

Ensure transport layer security for ‘basic authentication’ is configured

```
Get-Website | Foreach-Object {
  $ssl   = (Get-WebConfiguration -Filter
"/system.webServer/security/access" -PSPath
"IIS:\sites\$($_.Name)").SSLFlags
  $basic = (Get-WebConfigurationProperty -filter
"/system.WebServer/security/authentication/basicAuthentication"
-name Enabled -PSPath "IIS:\sites\$($_.Name)").Value

  If ($basic) {
     [PSCustomObject]@{
          "Name" = $_.Name
          "SSL"  = $(If($ssl -EQ 'Ssl') { $true } Else { $false
})
     }
  }
}
```

Ensure ‘passwordFormat’ is not set to clear

```
# Individual site config
Get-Website | Foreach-Object {
```

```
  $config = (Get-WebConfiguration -Filter
'/system.web/authentication' -PSPath "IIS:\sites\$($_.Name)")
  $format = (Get-WebConfiguration -Filter
'/system.web/authentication/forms/credentials' -PSPath
"IIS:\sites\$($_.Name)").passwordFormat

  If ($config.mode -EQ 'forms') {
     [PSCustomObject]@{
          "Name" = $_.Name
          "Format" = $(If($format -EQ 'clear'){ $false } Else {
$true })
     }
  }
}

# Machine Config
$machineConfig =
[System.Configuration.ConfigurationManager]::OpenMachineConfigur
ation()
$passwordFormat =
$machineConfig.GetSection("system.web/authentication").forms.cre
dentials.passwordFormat
If ($passwordFormat -EQ 'clear') {
  $false
} Else {
  $true
}
```

Ensure ‘credentials’ are not stored in configuration files

```
Get-Website | Foreach-Object {
  $config = (Get-WebConfiguration -Filter
'/system.web/authentication' -PSPath "IIS:\sites\$($_.Name)")
  $stored = (Get-WebConfiguration -filter
'/system.web/authentication/forms/credentials' -PSPath
"IIS:\sites\$($_.Name)").IsLocallyStored

  If ($config.mode -EQ 'forms') {
     [PSCustomObject]@{
```

```
            "Name" = $_.Name
            "LocalStored" = $stored
        }
    }
}
```

ASP.NET BENCHMARK CHECKS

Ensure ‘deployment method retail’ is set

```
$machineConfig =
[System.Configuration.ConfigurationManager]::OpenMachineConfigur
ation()
$deployment = $machineConfig.GetSection("system.web/deployment")
$deployment.Retail
```

Ensure ‘debug’ is turned off

```
Get-Website | Foreach-Object {
  [PSCustomObject]@{
    "Site"  = $_.Name
     "Debug" = (Get-WebConfiguration -Filter
'/system.web/compilation' -PSPath "IIS:\sites\$($_.Name)").Debug
  }
}
```

Ensure custom error messages are not off

```
Get-Website | Foreach-Object {
  $mode = (Get-WebConfiguration -Filter
'/system.web/customErrors' -PSPath "IIS:\sites\$($_.Name)").Mode

  [PSCustomObject]@{
     "Site"  = $_.Name
     "Mode"  = $mode
     "CustomErrors" = $(If($mode -EQ 'off'){$false}Else{$true})
  }
}
```

Ensure IIS HTTP detailed errors are hidden from displaying remotely

```
Get-Website | Foreach-Object {
  $errorMode = (Get-WebConfiguration -Filter
'/system.webServer/httpErrors' -PSPath
"IIS:\sites\$($_.Name)").errorMode

  [PSCustomObject]@{
     "Site"  = $_.Name
     "ErrorMode" = $errorMode
     "Errors" = $(If(($errorMode -NE 'Custom') -And ($errorMode
-NE 'DetailedLocalOnly') ){ $False } Else { $True })
  }
}
```

Ensure ASP.NET stack tracing is not enabled

```
# Individual Site Config
Get-Website | Foreach-Object {
  [PSCustomObject]@{
     "Site"  = $_.Name
     "Trace" = (Get-WebConfiguration -Filter '/system.web/trace'
-PSPath "IIS:\sites\$($_.Name)").enabled
  }
}

# Machine Config
$machineConfig =
[System.Configuration.ConfigurationManager]::OpenMachineConfigur
ation()
$deployment = $machineConfig.GetSection("system.web/trace")
$deployment.enabled
```

Ensure ‘httpcookie’ mode is configured for session state

```
Get-Website | Foreach-Object {
  $sessionState = (Get-WebConfiguration -Filter
'/system.web/sessionState' -PSPath
"IIS:\sites\$($_.Name)").cookieless

  [PSCustomObject]@{
```

```
      "Site"  = $_.Name
      "SessionState" = $sessionState
      "CookieLess" = $(If(($sessionState -NE "UseCookies") -And
($sessionState -NE "False")) { $false } Else { $true })
   }
}
```

Ensure ‘cookies’ are set with HttpOnly attribute

```
Get-Website | Foreach-Object {
   [PSCustomObject]@{
      "Site"  = $_.Name
      "httpCookies" = (Get-WebConfiguration -Filter
'/system.web/httpCookies' -PSPath
"IIS:\sites\$($_.Name)").httpOnlyCookies
   }
}
```

Ensure ‘MachineKey validation method – .Net 3.5’ is configured

```
Get-Website | Foreach-Object {
   $site = $_
   $applicationPool = $_.applicationPool
   If ($applicationPool) {
      $pools = Get-WebApplication -Site $_.Name

      $pools | ForEach-Object {
           $appPool  = ($_.Attributes | Where-Object Name -EQ
'applicationPool').Value
           $properties = Get-ItemProperty -Path
"IIS:\AppPools\$appPool" | Select-Object *
           $version  = $properties.managedRuntimeVersion

           If ($version -Like "v2.*") {
                $validation = (Get-WebConfiguration -Filter
'/system.web/machineKey' -PSPath
"IIS:\sites\$($site.Name)").Validation

                [PSCustomObject]@{
                "Site"    = $site.Name
```

```
                "AppPool" = $appPool
                  "Version"    =
$properties.managedRuntimeVersion
                "Validation" = $validation
                }
          }
     }
  }
}
```

Ensure ‘MachineKey validation method – .Net 4.5’ is configured

```
Get-Website | Foreach-Object {
  $site = $_
  $applicationPool = $_.applicationPool
  If ($applicationPool) {
     $pools = Get-WebApplication -Site $_.Name

     $pools | ForEach-Object {
          $appPool  = ($_.Attributes | Where-Object Name -EQ
'applicationPool').Value
          $properties = Get-ItemProperty -Path
"IIS:\AppPools\$appPool" | Select-Object *
          $version  = $properties.managedRuntimeVersion

          If ($version -Like "v4.*") {
               $validation = (Get-WebConfiguration -Filter
'/system.web/machineKey' -PSPath
"IIS:\sites\$($site.Name)").Validation

                [PSCustomObject]@{
                "Site"    = $site.Name
                "AppPool" = $appPool
                "Version" = $version
                "Validation" = $validation
                }
          }
     }
  }
```

```
}
```

Ensure global .NET trust level is configured

```
Get-Website | Foreach-Object {
  $site = $_
  $applicationPool = $_.applicationPool
  If ($applicationPool) {
     $pools = Get-WebApplication -Site $_.Name

     $pools | ForEach-Object {
          $appPool  = ($_.Attributes | Where-Object Name -EQ
'applicationPool').Value
          $properties = Get-ItemProperty -Path
"IIS:\AppPools\$appPool" | Select-Object *
          $version  = $properties.managedRuntimeVersion

         $level = (Get-WebConfiguration -Filter
'/system.web/trust' -PSPath "IIS:\sites\$($site.Name)").level

          [PSCustomObject]@{
               "Site"    = $site.Name
               "AppPool" = $appPool
               "Version" = $version
               "Level"   = $Level
          }
     }
  }
}
```

Ensure X-Powered-By Header is removed

```
Get-Website | Foreach-Object {
  $site = $_
  $config = Get-WebConfiguration -Filter
'/system.webServer/httpProtocol/customHeaders' -PSPath
"IIS:\sites\$($site.Name)"

  $customHeaders = $config.GetCollection()
```

```
  If ($customHeaders) {
     $customHeaders | ForEach-Object {
          [PSCustomObject]@{
               "Site" = $site.Name
               "X-Powered-By" = ($_.Attributes | Where-Object
Name -EQ name).Value -match 'x-powered-by'
          }
     }
  }
}
```

Ensure Server Header is removed

```
Get-Website | Foreach-Object {
  $site = $_
  $config = Get-WebConfiguration -Filter
'/system.webServer/httpProtocol/customHeaders' -PSPath
"IIS:\sites\$($site.Name)"

  $customHeaders = $config.GetCollection()

  If ($customHeaders) {
     $customHeaders | ForEach-Object {
          [PSCustomObject]@{
               "Site" = $site.Name
               "Server" = ($_.Attributes | Where-Object Name -EQ
name).Value -match 'server'
          }
     }
  }
}
```

REQUEST FILTERING AND OTHER RESTRICTION MODULES BENCHMARK CHECKS

Ensure ‘maxAllowedContentLength’ is configured

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
     [PSCustomObject]@{
          "Site" = $site.Name
          "maxAllowedContentLength" = (((Get-WebConfiguration
-Filter 'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)").requestLimits).Attributes |
Where-Object Name -EQ 'maxAllowedContentLength').Value
     }
  }
}
```

Ensure ‘maxURL request filter’ is configured

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
     [PSCustomObject]@{
          "Site" = $site.Name
          "maxURL" = (((Get-WebConfiguration -Filter
'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)").requestLimits).Attributes |
Where-Object Name -EQ 'maxURL').Value
     }
  }
}
```

Ensure ‘MaxQueryString request filter’ is configured

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
     [PSCustomObject]@{
          "Site" = $site.Name
          "MaxQueryString" = (((Get-WebConfiguration -Filter
'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)").requestLimits).Attributes |
Where-Object Name -EQ 'maxQueryString').Value
     }
  }
}
```

Ensure non-ASCII characters in URLs are not allowed

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
     [PSCustomObject]@{
          "Site" = $site.Name
          "allowHighBitCharacters" = (Get-WebConfiguration
-Filter 'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)").allowHighBitCharacters
     }
  }
}
```

Ensure Double-Encoded requests will be rejected

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
     [PSCustomObject]@{
          "Site" = $site.Name
          "allowDoubleEscaping" = (Get-WebConfiguration -Filter
'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)").allowDoubleEscaping
```

```
      }
  }
}
```

Ensure ‘HTTP Trace Method’ is disabled

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_

     $config = (Get-WebConfiguration -Filter
'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)")

     $config.verbs.Attributes | Where-Object {
     $_.Name -EQ 'trace'
     }
  }
}
```

Ensure Unlisted File Extensions are not allowed

```
If ((Get-WindowsFeature Web-Filtering).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
     [PSCustomObject]@{
          "Site" = $site.Name
          "allowUnlisted" = (((Get-WebConfiguration -Filter
'system.webServer/security/requestFiltering' -PSPath
"IIS:\sites\$($site.Name)").fileExtensions).Attributes |
Where-Object Name -EQ 'allowUnlisted').Value
     }
  }
}
```

Ensure Handler is not granted Write and Script/Execute

```
Get-Website | Foreach-Object {
     $site = $_
```

```
     [PSCustomObject]@{
     "Site" = $site.Name
     "accessPolicy" = (Get-WebConfiguration -Filter
'system.webServer/handlers' -PSPath
"IIS:\sites\$($site.Name)").accessPolicy
     }
}
```

Ensure ‘notListedIsapisAllowed’ is set to false

```
Get-Website | Foreach-Object {
  $site = $_

  [PSCustomObject]@{
     "Site" = $site.Name
     "notListedIsapisAllowed" = (Get-WebConfiguration -Filter
'system.webServer/security/isapiCgiRestriction' -PSPath
"IIS:\sites\$($site.Name)").notListedIsapisAllowed
  }
}
```

Ensure ‘notListedCgisAllowed’ is set to false

```
Get-Website | Foreach-Object {
  $site = $_

  [PSCustomObject]@{
     "Site" = $site.Name
     "notListedIsapisAllowed" = (Get-WebConfiguration -Filter
'system.webServer/security/isapiCgiRestriction' -PSPath
"IIS:\sites\$($site.Name)").notListedCgisAllowed
  }
}
```

Ensure ‘Dynamic IP Address Restrictions’ is enabled

```
If ((Get-WindowsFeature Web-Ip-Security).Installed -EQ $true) {
  Get-Website | Foreach-Object {
     $site = $_
```

```
     $config = Get-WebConfiguration -Filter
'/system.webServer/security/dynamicIpSecurity' -PSPath
"IIS:\sites\$($site.Name)"

     [PSCustomObject]@{
     "Site" = $site.Name
     "denyByConcurrentRequests" =
$config.denyByConcurrentRequests.enabled
     "denyByRequestRate" = $config.denyByRequestRate.enabled
     }
  }
}
```

IIS LOGGING BENCHMARK CHECKS

Ensure Default IIS weblog location is moved

```
Get-Website | Foreach-Object {
  $site = $_

  [PSCustomObject]@{
     "Site" = $site.Name
     "Location" = $Site.logFile.Directory
  }
}
```

Ensure ‘ETW Logging’ is enabled

```
Get-Website | Foreach-Object {
  $site = $_

  [PSCustomObject]@{
     "Site" = $site.Name
     "logTargetW3C" = $Site.logFile.logTargetW3C
  }
}
```

FTP REQUEST BENCHMARK CHECKS

Ensure FTP requests are encrypted

```
Get-Website | Foreach-Object {
  $site = $_
  $FTPBindings = $site.bindings.collection | Where-Object
-Property Protocol -eq FTP
  If ($FTPBindings) {
     $config = (Get-WebConfiguration -Filter
'system.applicationHost/sites' -PSPath
"IIS:\sites\$($site.Name)").siteDefaults.ftpServer.security.ssl

     ($config.Attributes | Where-Object Name -EQ
'controlChannelPolicy').Value
     ($config.Attributes | Where-Object Name -EQ
'dataChannelPolicy').Value
  }
}
```

Ensure FTP Logon attempt restrictions is enabled

```
Get-Website | Foreach-Object {
  $site = $_
  $FTPBindings = $site.bindings.collection | Where-Object
-Property 'Protocol' -eq 'FTP'
  If ($FTPBindings) {
     $config = (Get-WebConfiguration -Filter
'system.ftpServer/security/authentication' -PSPath
"IIS:\sites\$($site.Name)").denyByFailure
     [PSCustomObject]@{
     "Site"         = $site.Name
     "Enabled"      = $config.enabled
     "MaxFailures" = $config.maxFailure
```

```
      "EntryExp"      = ($config.entryExpiration).ToString()
      "Logging"       = $config.loggingOnlyMode
      }
  }
}
```

TRANSPORT ENCRYPTION BENCHMARK CHECKS

Ensure HSTS Header is set

```
Get-Website | Foreach-Object {
  $site   = $_
  $config = (Get-WebConfiguration -Filter
'/system.webServer/httpProtocol' -PSPath
"IIS:\sites\$($site.Name)").customHeaders
  $value  = ($config.Attributes | Where-Object Name -EQ
'Strict-Transport-Security').Value

  $value | Where-Object { $_ -Match "max-age" }
}
```

Ensure SSLv2 is Disabled

```
$path =
"HKLM:\System\CurrentControlSet\Control\SecurityProviders\SCHANN
EL\Protocols\SSL 2.0"

If ((Test-Path -Path $path) -and (Test-Path -Path
"$path\Server")) {
     $Key = Get-Item "$path\Server"

     if ($null -ne $Key.GetValue("Enabled", $null)) {
```

```
          $value = Get-ItemProperty "$path\Server" |
Select-Object -ExpandProperty "Enabled"
          # Ensure it is set to 0
          if ($value -ne 0) {
          $false
          } Else {
          $true
          }
     }
}
```

Ensure SSLv3 is Disabled

```
$path =
"HKLM:\System\CurrentControlSet\Control\SecurityProviders\SCHANN
EL\Protocols\SSL 3.0"

If ((Test-Path -Path $path) -and (Test-Path -Path
"$path\Server")) {
     $Key = Get-Item "$path\Server"

     if ($null -ne $Key.GetValue("Enabled", $null)) {
          $value = Get-ItemProperty "$path\Server" |
Select-Object -ExpandProperty "Enabled"
          # Ensure it is set to 0
          if ($value -ne 0) {
          $false
          } Else {
          $true
          }
     }
}
```

Ensure TLS 1.0 is Disabled

```
$path =
"HKLM:\System\CurrentControlSet\Control\SecurityProviders\SCHANN
EL\Protocols\TLS 1.0\Server"
```

```
If ((Test-Path -Path $path)) {
     $Key = Get-Item "$path"

     if ($null -ne $Key.GetValue("Enabled", $null)) {
          $value = Get-ItemProperty "$path\Server" |
Select-Object -ExpandProperty "Enabled"
          # Ensure it is set to 0
          if ($value -ne 0) {
          $false
          } Else {
          $true
          }
     }
}
```

Ensure TLS 1.1 is Disabled

```
$path =
"HKLM:\System\CurrentControlSet\Control\SecurityProviders
\SCHANNEL\Protocols\TLS 1.1\Server"
If ((Test-Path -Path $path)) {
     $Key = Get-Item "$path"

     if ($null -ne $Key.GetValue("Enabled", $null)) {
          $value = Get-ItemProperty "$path\Server" |
Select-Object -ExpandProperty "Enabled"
          # Ensure it is set to 0
          if ($value -ne 0) {
          $false
          } Else {
          $true
          }
     }
}
```

https://calcomsoftware.com/automating-iis-hardening-with-powershell/

https://www.drewhjelm.com/2015/07/20/easy-iis-ssl-hardening.html

https://techcommunity.microsoft.com/t5/itops-talk-blog/windows-server-101-hardening-iis-via-security-control/ba-p/329979

https://docs.microsoft.com/en-us/powershell/module/webadminstration/backup-webconfiguration?view=winserver2012-ps

https://docs.microsoft.com/en-us/powershell/module/webadminstration/restore-webconfiguration?view=winserver2012-ps