链滴

黑客派自动签到脚本

作者：YxxXlv0COaxl

原文链接：https://ld246.com/article/1558789271479

来源网站：链滴

一个已知问题:脚本会长时间卡住(大概率).多次运行脚本仍无法继续.

问题原因:由于Invoke-WebRequest命令使用的是IE内核进行DOM解析,获取网页之后,会进行DOM析,才能进一步使用ParsedHtml方法中的getElementsByClassName等网页方法,脚本就卡在了这个方.

具体原因未知,可能是该站完全放弃了IE,未进行兼容导致的.

*.ps1文件

```
<#

.Synopsis
   自动签到的PowerShell脚本
.DESCRIPTION
   自动签到
.EXAMPLE
   自动签到程序.ps1 [-Task]
   自动签到程序.ps1 [-DelUser] <动态值>
.EXAMPLE
   自动签到程序.ps1 [-WebName] <动态值> [-UserName] <String> [-PassWord] <String>

.INPUTS
   到此 cmdlet 的输入(如果有)
.OUTPUTS
   来自此 cmdlet 的输出(如果有)
.NOTES
   一般注释
.COMPONENT
   此 cmdlet 所属的组件
.ROLE
   此 cmdlet 所属的角色
.FUNCTIONALITY
   最准确描述此 cmdlet 的功能
#>
# 该脚本声明必须在开头,其他函数及命令之前
# 用于手动添加账号密码
[CmdletBinding(DefaultParameterSetName)] # 设置默认参数组为空
Param(

    [Parameter(ParameterSetName="adds",Position=1,Mandatory=$true)]
    [ValidateNotNullOrEmpty()]
    [string]$UserName,

    [Parameter(ParameterSetName="adds",Position=2,Mandatory=$true)]
    [ValidateNotNullOrEmpty()]
    [string]$PassWord,

    [Parameter(ParameterSetName="Task")]
    [switch]$Task,

    [Parameter(DontShow,ParameterSetName="None")]
    [switch]$None
)
```

原文链接：黑客派自动签到脚本

```
#动态参数
DynamicParam
{

    $filepath="$($PSScriptRoot)\cookie.user"
    if (-not (Test-Path -Path $filepath))
    {
        New-Item -ItemType "file" -Path $filepath | Out-Null
    }
    #主
    $PD = New-Object -Type System.Management.Automation.RuntimeDefinedParameterDictionary
    $modName=(Get-ChildItem -Path $($PSScriptRoot) -Name -Include 签到-*.psm1) -replace 签到-','' -replace '.psm1',''
    #$modName=$mods
    #当有模块时,动态添加模块列表
    if ($modName.Count){

        #容器
        $a = New-Object -TypeName System.Collections.ObjectModel.Collection[System.Attribute]
        #属性
        $b = New-Object System.Management.Automation.ParameterAttribute
        $b.Mandatory = $true
        $b.ParameterSetName='adds'
        $b.Position=0
        $b.HelpMessage='可选参数:'+$modName
        $a.Add($b)
        #参数值
        $c=New-Object System.Management.Automation.ValidateSetAttribute($modName)
        $a.Add($c)

        $d = New-Object -TypeName System.Management.Automation.RuntimeDefinedParameter('WebName',[string], $a)
        $PD.Add('WebName', $d)
    }
    #账号删除功能
    [PSObject].Assembly.GetType('System.Management.Automation.TypeAccelerators')::Add('S', [System.Collections.ArrayList])
    $pp=(Get-Content $filepath) | Convertfrom-Json
    $ar = New-Object -TypeName SA
    foreach ($s in $pp)
    {

        $ar.Add($s.WebName+":"+$s.UserName)|Out-Null
    }
    if ($ar.Count) {

        #容器
        $a = New-Object -TypeName System.Collections.ObjectModel.Collection[System.Attribute]
        #属性
        $b = New-Object System.Management.Automation.ParameterAttribute
        $b.Mandatory = $true
```

```
        $b.ParameterSetName='Deleted'
        $a.Add($b)

        #参数值
        $c=New-Object System.Management.Automation.ValidateSetAttribute($ar)
        $a.Add($c)

        $d = New-Object -TypeName System.Management.Automation.RuntimeDefinedParame
er('DelUser',[string], $a)
        $PD.Add('DelUser',$d)
    }
    return $PD
}

begin
{
    $T=1
    #cookie存储文件位置
    $filepath="$($PSScriptRoot)\cookie.user"
    if (-not (Test-Path -Path $filepath))
    {
        New-Item -ItemType "file" -Path $filepath | Out-Null
    }
    #签到结果记录路径文件位置
    $LogPath="{0}\Log\" -f $PSScriptRoot
    if (-not (Test-Path -Path $LogPath))
    {
        New-Item -ItemType "directory" -Path $LogPath | Out-Null
    }
    $global:Log="{0}{1:yyyyMMdd}.log" -f $LogPath,(get-date)
    #可用签到模板
    $modName=(Get-ChildItem -Path $($PSScriptRoot) -Name -Include 签到-*.psm1) -replace
签到-','' -replace '.psm1',''
    #所有用户信息
    [array]$pp=(Get-Content $filepath) | Convertfrom-Json
    #添加计划任务
    if ($task)
    {
        # 任务操作   程序 ,工作目录,参数
        $AA = New-ScheduledTaskAction -Execute "PowerShell"  -WorkingDirectory "$($PSScrip
Root)" -Argument "$($Script:MyInvocation.InvocationName)"
        #$BB = New-ScheduledTaskAction -Execute "PowerShell"  -WorkingDirectory "$($PSScri
tRoot)" -Argument "-WindowStyle Hidden -Command `"& { Get-WinEvent -FilterHashtable @
LogName='论坛签到';StartTime=(Get-Date -Hour 0 -Minute 0 -Second 1)} | sort  TimeCreated
Descending| SELECT-Object TimeCreated,id,leveldisplayname,ProviderName,message | Out-Gr
dView -Wait -Title 今日签到记录}`""
        $tem=Get-ScheduledTask -TaskName '论坛自动签到任务' -ErrorAction SilentlyContinue -
arningAction SilentlyContinue
        if ($tem)
        {
            if (($tem.Actions.Arguments -eq $AA.Arguments ) -and ($tem.Actions.WorkingDirecto
y -eq $AA.WorkingDirectory))
            {
                Write-Warning '该计划任务已存在'
```

```
            }
        }
        else
        {
            # 触发器    早上6点到晚上23点
            $T = New-ScheduledTaskTrigger -Daily -At 6:0:0  -RandomDelay 17:0:0
            # 创建者  当前用户名
            $P = New-ScheduledTaskPrincipal "$env:userdomain\$env:username"
            # 设置集    各种相关设置
            $S = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -WakeToRun -Priority 4
DontStopIfGoingOnBatteries -DontStopOnIdleEnd -RunOnlyIfNetworkAvailable -MultipleInst
nces IgnoreNew -StartWhenAvailable -Compatibility win8  -ExecutionTimeLimit 0   -Disallow
ardTerminate
            # 新任务    最后是任务描述  DisallowStartIfOnBatteries
            $D = New-ScheduledTask -Action $AA -Principal $P -Trigger $T -Settings $S  -Descript
on "论坛自动签到脚本"
            # 注册任务
            Register-ScheduledTask -TaskName "论坛自动签到任务" -InputObject $D |Out-Null
            Write-Host '计划任务已经添加'
        }
        return
    }
    #删除账号
    if ($PSBoundParameters.DelUser)
    {
        $DU=$PSBoundParameters.DelUser -split ":"
        Remove-Item $filepath
        foreach ($s in $pp)
        {
            if (-not(($s.WebName -eq $DU[0]) -and ($s.UserName -eq $DU[1])))
            {
                ConvertTo-Json -Compress -InputObject $s |  Out-File -append $filepath
            }
        }
        return
    }
    "签到脚本开始运行"
    #设置UserAgent
    enum 浏览器
    {
        Chrome
        FireFox
        InternetExplorer
        Opera
        Safari
    }
    function Get-UA ([浏览器]$ua)
    {
        return [Microsoft.PowerShell.Commands.PSUserAgent]::[浏览器]$ua
    }
    $global:UserAgents=Get-UA('Chrome')

    #写入到文件
    function global:Write-file ($Account){
```

```
    ConvertTo-Json -Compress -InputObject $Account |  Out-File -append $filepath
}

#写入到日志文件
function global:Write-Log ($Account){
    Out-File -append $Log -InputObject $Account.ToString()
}

function global:HttpRest ($S)
{
    Start-Sleep -Seconds $T
    $b=try{
        Invoke-RestMethod  @s
    }catch{
        $_.Exception.Response
    }
    return $b
}

function global:HttpWeb ($S)
{
    Start-Sleep -Seconds $T
    $b=try{
        Invoke-WebRequest  @s
    }catch{
        $_.Exception.Response
    }
    return $b
}
#进行url编码
function global:UrlEncode ([string]$pa)
{
    return [uri]::EscapeUriString($pa)
}

#添加cookie
function global:AddCookie($Tok,$WebHost)
{
    # 创建cookie 数据包
    $Loadion = New-Object Microsoft.PowerShell.Commands.WebRequestSession
    $y=($Tok| Get-Member -MemberType NoteProperty ).Name
    # 循环添加所有 cookie
    foreach ($tk in $y)
    {
        $cookie = New-Object System.Net.Cookie($tk,$Tok.$tk,"",$WebHost)
        $Loadion.Cookies.Add($cookie)
    }
    return $Loadion
}
#转换cookie为散列表
function global:CookieToHashTable ([Microsoft.PowerShell.Commands.WebRequestSession
$cookie,$uri)
{
    $d=@{}
```

```
        $i=$cookie.Cookies.GetCookieHeader($uri)
        $i -split '; '|%{
            $temp=$_ -split '='
            $d.Add($temp[0],$temp[1])
        }
        return $d
    }
    #计算MD5
    function global:MD5
    {
         param
        (
            [parameter(mandatory=$true)]
            [ValidateNotNullOrEmpty()]
            [String] $String
        )
        #希哈计算
        $StringBuilder = New-Object System.Text.StringBuilder
        [System.Security.Cryptography.HashAlgorithm]::Create("MD5").ComputeHash([System.Te
t.Encoding]::UTF8.GetBytes($String))|%{
            [Void]$StringBuilder.Append($_.ToString("x2"))
        }
        return $StringBuilder.ToString()
    }

    $color=@{
        ForegroundColor='Black'
        BackgroundColor='White'
        NoNewline=$true
    }

    Write-Host -NoNewline "当前可用签到模板: "
    foreach ($item in $modName)
    {
        Write-Host "$item" @color
        Write-Host " " -NoNewline
        import-module "$($PSScriptRoot)\签到-$($item).psm1" -Prefix $item -Force
    }
    Write-Host ''
    #$pp=(Get-Content $filepath) | Convertfrom-Json
    Write-Host -NoNewline "共有 "
    Write-Host -Object $pp.Count  @color
    Write-Host " 个账号`n"

    #通过命令行添加新账号
    if ($PSBoundParameters.Count -eq 3)
    {
        foreach ($item in $pp)
        { #通过遍历,如果输入的用户名已存在,且与网站都吻合,则提示该用户已存在
            if (($item.UserName -eq $PSBoundParameters.UserName) -and  ($item.WebName -eq
$PSBoundParameters.WebName))
            {
                Write-Warning  "该用户已存在"
                return
```

```
            }
        }
        $NewA=&($PSBoundParameters.WebName+'Account')  $PSBoundParameters -news
        $NewA.LogIn()
        if ($NewA.Token)
        {
            "登陆成功"
        }elseif($NewA.Msg){
            $NewA.Msg
#|Format-Table -AutoSize
        }
        return
    }

    #-----------自动签到----------------
    "开始自动签到"
    foreach ($item in $pp)
    {
        #如果该账号缺少对应模块,则直接跳过
        if ($modName -notcontains $item.WebName) {continue}
        $Account=&($item.WebName+'Account') $item
        $Account.CheckIn()
        $Account.Result | Format-Table -AutoSize

    }
    Start-Sleep -Seconds 3;pause
}
<#
process
{

}
#>
end
{


"运行结束"
}
```

签到-黑客派.psm1

```
#网络地址

$WebName='黑客派'

[uri]$login_uri='https://hacpai.com/api/v2/login'
[uri]$login_OTH_uri='https://hacpai.com/api/v2/login/2fa'
[uri]$user_info_uri='https://hacpai.com/api/v2/user/{0}'
#[uri]$user_point_uri='https://hacpai.com/api/v2/notifications/point?p=1'
[uri]$user_checkin_uri='https://hacpai.com/activity/checkin'
$WebHost=$login_uri.Host
```

原文链接：黑客派自动签到脚本

```
#网络连接的基类
class GetWeb:Hashtable
{
    [uri]$Uri
    [validateset("Get", "Post", "Head")]
    [string]$Method='Get'
    hidden [int] $TimeoutSec
    hidden [string] $UserAgent

    GetWeb ([uri]$uri) {
        [uri]$this.Uri=$uri
        $this.Method='Get'
        $this.TimeoutSec= 7
        $this.UserAgent=$global:UserAgents
    }
}

class Result
{
    [string]$WebName=$Script:WebName
    [string]$UserName=$UserName
    [int]$StatusCode=-1
    [string]$Status
    [int]$Sc=-1
    [string]$Msg
    [int]$Points=-1
    [int]$AllPoint=-1
    [int]$Now=-1
    [int]$Long=-1
    Result($UserName){}
    [string]ToString(){
        $g="时间: {0:HH:mm'ss"}`t网站: {1}`t用户: {2}`t积分: {3}({4})`t次数: {5}/{6}" -f (get-date),$this
WebName,$this.UserName,$this.AllPoint,$this.points,$this.now,$this.Long
        return $g
    }
}

class WebSession:GetWeb
{
    $WebSession

    WebSession ([uri]$uri,$WebSession):base($uri){
        $this.WebSession=$WebSession
    }

}

class PostAndWebSession:WebSession
{
    $WebSession
    PostAndWebSession ([uri]$uri,$WebSession):base($uri,$WebSession){
        $this.Method='Post'
    }
}
```

```
#用于登录的body
class login_Body
{
    [string]$userName
    [string]$userPassword
    $captcha
    login_Body ([string]$userName,[string]$MD5) {
        $this.userName=$userName
        $this.userPassword=$MD5
        $this.captcha=$NULL
    }
}

class Head:Hashtable
{
    [string]$Referer
    Head(){
        $this.Referer=$Script:user_checkin_uri
    }
}

#post加body
class PostAndBody:GetWeb
{
    $Body
    PostAndBody ([uri]$uri,$body):base($uri){
        $this.Method='Post'
        $this.Body=$body
    }
}

#发送两步验证的请求参数
class PostAndBodyAndWebSession:PostAndBody
{
    $WebSession
    PostAndBodyAndWebSession ([uri]$uri,$body,$WebSession):base($uri,$body){
        $this.WebSession=$WebSession
    }
}

#建立以用户为主的类,其中包含登录,签到等网络操作的方法
 class 账户:Hashtable
 {
    [string]$WebName=$Script:WebName
    [ValidateNotNullOrEmpty()]
    [string]$UserName
    [ValidateNotNullOrEmpty()]
    [string]$MD5
    [PSCustomObject]$Token
    #输入方式添加
    账户 ([string]$username,[string]$userPassword){
        $this.WebName=$Script:WebName
        $this.UserName=$username
```

```
    $this.MD5=MD5($userPassword)
    #$this.Result=[Result]::new($UserName)
}
#从文件读取
账户 ([PSCustomObject]$Account){
    $this.WebName=$Script:WebName
    $this.UserName=$Account.UserName
    $this.MD5=$Account.MD5
    $this.Token=$Account.Token
    $this.Result=[Result]::new($Account.UserName)
}
#登录
LogIn()
{
    $body=[login_Body]::new($this.UserName,$this.MD5)
    $login=[PostAndBody]::new($Script:login_uri,$body)

    $b=HttpRest $login

    if ($b.sc -ne $NULL)
    {
        switch ($b.sc)
        {
            0
            {
                $this.Token=@{symphony=$b.token}
                Write-file -Account $this
                return
            }
            10
            {#两步验证
                $r = new-object psobject -Property @{symphony=$b.token}
                $this.OTH($r)
                return
            }
            Default
            {
                $this.Sc=$b.sc
                $this.Msg=$b.msg
            }
        }
    }elseif($b.StatusCode.value__ -ne $NULL){
        $this.StatusCode=$b.StatusCode.value__
        $this.Status=$b.StatusCode
        switch ($b.StatusCode.value__)
        {
            401
            {
                $this.Status='需要登录'
                return
            }
            403
            {
                $this.Status='权限不足'
```

```
                return
            }
        }
    }
}

#两步验证
hidden OTH ($r)
{
    $Loadion=AddCookie -Tok $r -WebHost $Script:WebHost
    $OTH = Read-Host '两步验证码'
    $body=@{twofactorAuthCode=$OTH}
    $TOTH=[PostAndBodyAndWebSession]::new($Script:login_OTH_uri,$body,$Loadion)
    $c=HttpRest $TOTH

    if ($c.sc -ne $NULL)
    {
        switch ($c.sc)
        {
            0
            {
                $this.Token=@{symphony=$c.token}
                Write-file -Account $this
                return
            }
            Default
            {
                $this.Sc=$c.sc
                $this.Msg=$c.msg
            }
        }
    }elseif($c.StatusCode.value__ -ne $NULL){
        $this.StatusCode=$c.StatusCode.value__
        $this.Status=$c.StatusCode
        switch ($c.StatusCode.value__)
        {
            401
            {
                $this.Msg='需要登录'
                return
            }
            403
            {
                $this.Msg='权限不足'
                return
            }
        }
    }
}

#主要的签到操作流程
CheckIn()
{
    $this.Loadion=AddCookie -Tok $this.Token -WebHost $Script:WebHost
```

```
    $this.get_check_uri()
    $this.get_info()
    Write-Log $this.Result
}

#获取积分信息
hidden get_info()
{
    $e=[WebSession]::new($Script:user_info_uri -f $this.UserName,$this.Loadion)
    $q=HttpRest $e
    if ($q.sc -ne $NULL)
    {
        $this.Result.Sc=$q.sc
        $this.Result.Msg=$q.msg
        if ($q.sc -eq 0)
        {
            $this.Result.Long=$q.data.user.userLongestCheckinStreak
            $this.Result.Now=$q.data.user.userCurrentCheckinStreak
            $this.Result.AllPoint=$q.data.user.userPoint
            return
        }
    }elseif($q.StatusCode.value__ -ne $NULL){
        $this.Result.StatusCode=$q.StatusCode.value__
        $this.Result.Status=$q.StatusCode
        switch ($q.StatusCode.value__)
        {
            401
            {
                $this.Result.Msg='需要登录'
                return
            }
            403
            {
                $this.Result.Msg='权限不足'
                return
            }
        }
    }
}

#获取签到地址
hidden get_check_uri()
{
    $e=[WebSession]::new($Script:user_checkin_uri,$this.Loadion)
    $e2=$e.Clone()
    $s=HttpWeb $e

    $this.Result.StatusCode=$s.StatusCode
    $this.Result.Status=$s.StatusDescription
    :ss switch ($s.StatusCode)
    {
        200
        {
            $re=$s.ParsedHtml.body.getElementsByClassName('btn green')[0].href
```

```
            if ($re)
            {
                $e.Uri=$re
                $e.Headers=[head]::new()
                Break :ss
            }
        }
        401
        {
            $this.Result.Msg='需要登录'
            return
        }
        403
        {
            $this.Result.Msg='权限不足'
            return
        }
    }

    $s=HttpWeb $e
    $this.Result.StatusCode=$s.StatusCode
    $this.Result.Status=$s.StatusDescription
    switch ($s.StatusCode)
    {
        200
        {
            $re=$s.ParsedHtml.body.getElementsByTagName('code')[0].innerText
            if ($re)
            {
                $this.Result.Points=$re
                return
            }
        }
        401
        {
            $this.Result.Msg='需要登录'
            return
        }
        403
        {
            $this.Result.Msg='权限不足'
            return
        }
    }
  }
}

function Account ()
{
  param
  (
    $Account,
    [switch] $news
  )
```

```
    if ($news){
        $f=[账户]::new($Account.UserName,$Account.PassWord)
    }else{
        $f=[账户]::new($Account)
    }
    return $f
}

Export-ModuleMember -Function Account
```