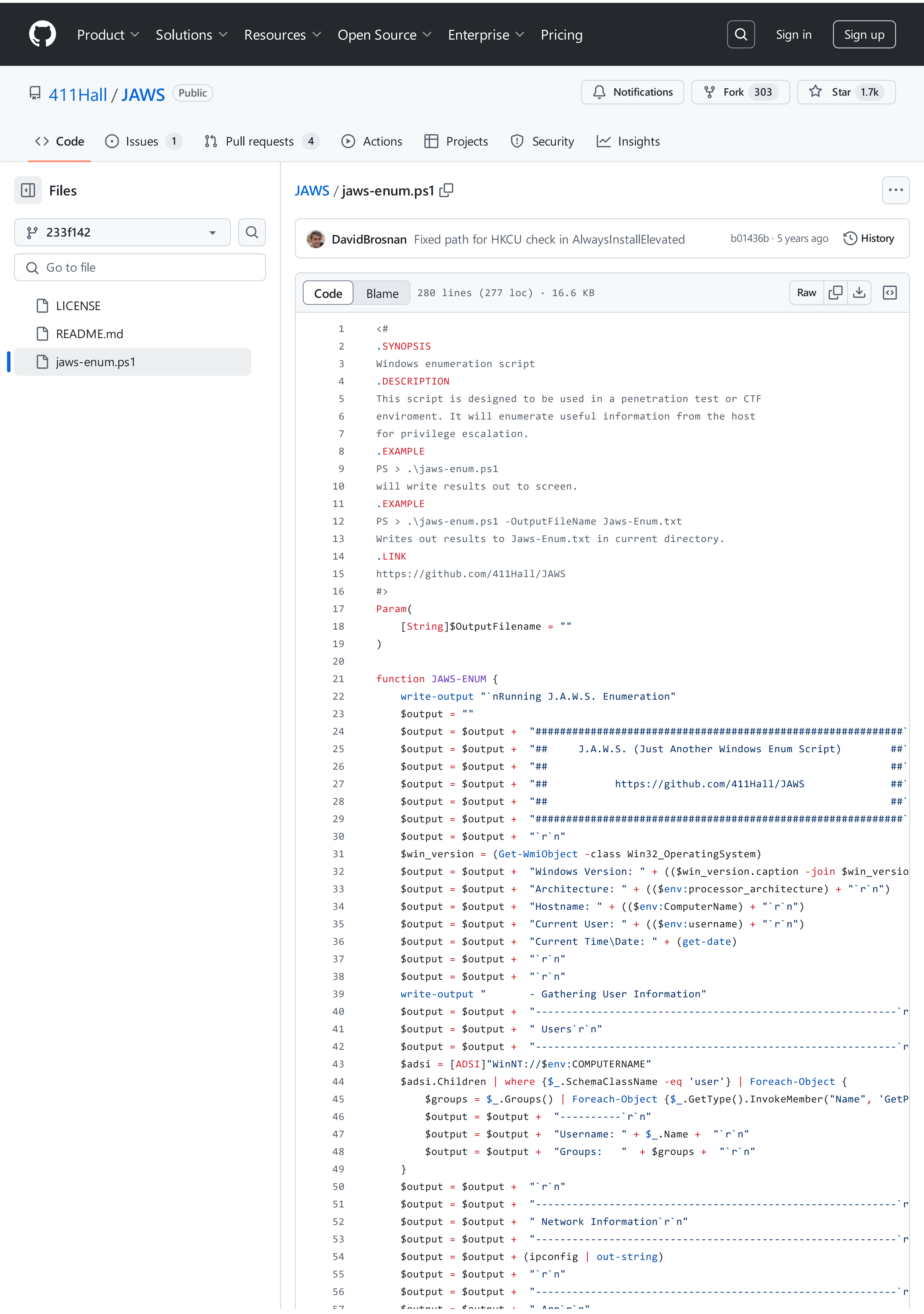

```
<#
.SYNOPSIS
Windows enumeration script
.DESCRIPTION
This script is designed to be used in a penetration test or CTF
enviroment. It will enumerate useful information from the host
for privilege escalation.
.EXAMPLE
PS > .\jaws-enum.ps1
will write results out to screen.
.EXAMPLE
PS > .\jaws-enum.ps1 -OutputFileName Jaws-Enum.txt
Writes out results to Jaws-Enum.txt in current directory.
.LINK
https://github.com/411Hall/JAWS
#>
Param(
    [String]$OutputFilename = ""
)

function JAWS-ENUM {
    write-output "`nRunning J.A.W.S. Enumeration"
    $output = ""
    $output = $output +  "############################################################`
    $output = $output +  "##     J.A.W.S. (Just Another Windows Enum Script)        ##`
    $output = $output +  "##                                                        ##`
    $output = $output +  "##           https://github.com/411Hall/JAWS              ##`
    $output = $output +  "##                                                        ##`
    $output = $output +  "############################################################`
    $output = $output +  "`r`n"
    $win_version = (Get-WmiObject -class Win32_OperatingSystem)
    $output = $output +  "Windows Version: " + (($win_version.caption -join $win_versio
    $output = $output +  "Architecture: " + (($env:processor_architecture) + "`r`n")
    $output = $output +  "Hostname: " + (($env:ComputerName) + "`r`n")
    $output = $output +  "Current User: " + (($env:username) + "`r`n")
    $output = $output +  "Current Time\Date: " + (get-date)
    $output = $output +  "`r`n"
    $output = $output +  "`r`n"
    write-output "       - Gathering User Information"
    $output = $output +  "-----------------------------------------------------------`r
    $output = $output +  " Users`r`n"
    $output = $output +  "-----------------------------------------------------------`r
    $adsi = [ADSI]"WinNT://$env:COMPUTERNAME"
    $adsi.Children | where {$_.SchemaClassName -eq 'user'} | Foreach-Object {
        $groups = $_.Groups() | Foreach-Object {$_.GetType().InvokeMember("Name", 'GetP
        $output = $output +  "----------`r`n"
        $output = $output +  "Username: " + $_.Name +  "`r`n"
        $output = $output +  "Groups:    "  + $groups +  "`r`n"
    }
    $output = $output +  "`r`n"
    $output = $output +  "-----------------------------------------------------------`r
    $output = $output +  " Network Information`r`n"
    $output = $output +  "-----------------------------------------------------------`r
    $output = $output + (ipconfig | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-----------------------------------------------------------`r
    $output = $output +  " App`r`n"
```

```
        $output = $output +  " Arp`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output + (arp -a | out-string)
        $output = $output +  "`r`n"
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " NetStat`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output + (netstat -ano | out-string)
        $output = $output +  "`r`n"
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Firewall Status`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  "`r`n"
        $Firewall = New-Object -com HNetCfg.FwMgr
        $FireProfile = $Firewall.LocalPolicy.CurrentProfile
        if ($FireProfile.FirewallEnabled -eq $False) {
            $output = $output +  ("Firewall is Disabled" + "`r`n")
            } else {
            $output = $output +  ("Firwall is Enabled" + "`r`n")
            }
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " FireWall Rules`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        Function Get-FireWallRule
        {Param ($Name, $Direction, $Enabled, $Protocol, $profile, $action, $grouping)
        $Rules=(New-object -comObject HNetCfg.FwPolicy2).rules
        If ($name)      {$rules= $rules | where-object {$_.name     -like $name}}
        If ($direction) {$rules= $rules | where-object {$_.direction  -eq $direction}}
        If ($Enabled)   {$rules= $rules | where-object {$_.Enabled    -eq $Enabled}}
        If ($protocol)  {$rules= $rules | where-object {$_.protocol   -eq $protocol}}
        If ($profile)   {$rules= $rules | where-object {$_.Profiles -bAND $profile}}
        If ($Action)    {$rules= $rules | where-object {$_.Action     -eq $Action}}
        If ($Grouping)  {$rules= $rules | where-object {$_.Grouping -like $Grouping}}
        $rules}
        $output = $output +  (Get-firewallRule -enabled $true | sort direction,applicationN
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Hosts File Content`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  "`r`n"
        $output = $output + ((get-content $env:windir\System32\drivers\etc\hosts | out-stri
        $output = $output +  "`r`n"
        write-output "       - Gathering Processes, Services and Scheduled Tasks"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Processes`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  ((Get-WmiObject win32_process | Select-Object Name,ProcessID,@
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Scheduled Tasks`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  "Current System Time: " + (get-date)
        $output = $output + (schtasks /query /FO CSV /v | convertfrom-csv | where { $_.Task
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Services`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output + (get-service | Select Name,DisplayName,Status | sort status | F
        $output = $output +  "`r`n"
        write-output "       - Gathering Installed Software"
        $output = $output +  "`r`n"
```

```
        $output = $output +  " System Files with Passwords`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $files = ("unattended.xml", "sysprep.xml", "autounattended.xml","unattended.inf", "
        $output = $output +  (get-childitem C:\ -recurse -include $files -EA SilentlyContin
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " AlwaysInstalledElevated Registry Key`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $HKLM = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer"
        $HKCU =  "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer"
        if (($HKLM | test-path) -eq "True")
        {
            if (((Get-ItemProperty -Path $HKLM -Name AlwaysInstallElevated).AlwaysInstallEl
            {
                $output = $output +   "AlwaysInstallElevated enabled on this host!"
            }
        }
        if (($HKCU | test-path) -eq "True")
        {
            if (((Get-ItemProperty -Path $HKCU -Name AlwaysInstallElevated).AlwaysInstallEl
            {
                $output = $output +   "AlwaysInstallElevated enabled on this host!"
            }
        }
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Stored Credentials`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output + (cmdkey /list | out-string)
        $output = $output +  "`r`n"
        $output = $output +  "-----------------------------------------------------------`r
        $output = $output +  " Checking for AutoAdminLogon `r`n"
        $output = $output + "-----------------------------------------------------------`r`
        $Winlogon = "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
        if (get-itemproperty -path $Winlogon -Name AutoAdminLogon -ErrorAction SilentlyCont
            {
            if ((get-itemproperty -path $Winlogon -Name AutoAdminLogon).AutoAdminLogon -eq
                {
                $Username = (get-itemproperty -path $Winlogon -Name DefaultUserName).Defaul
                $output = $output + "The default username is $Username `r`n"
                $Password = (get-itemproperty -path $Winlogon -Name DefaultPassword).Defaul
                $output = $output + "The default password is $Password `r`n"
                $DefaultDomainName = (get-itemproperty -path $Winlogon -Name DefaultDomainN
                $output = $output + "The default domainname is $DefaultDomainName `r`n"
                }
            }
        $output = $output +  "`r`n"
        if ($OutputFilename.length -gt 0)
           {
            $output | Out-File -FilePath $OutputFileName -encoding utf8
            }
        else
            {
            clear-host
            write-output $output
            }
    }

    if ($OutputFilename.length -gt 0)
        {
            Try
                {
                    [io.file]::OpenWrite($OutputFilename).close()
                    JAWS-ENUM
                }
            Catch
                {
                    Write-Warning "`nUnable to write to output file $OutputFilename, Check
                }
        }
    else
        {
        JAWS-ENUM
        }
```