How to mirror the list of server blocks from another Mastodon instance

If you're bringing up a new Mastodon server, there will probably come a time when you want, or have to block another server. I've heard there are some unsavoury ones out there. AD-blocking software had had the concept of a blocklist of many years, but I haven't really found a good central repository of blocked servers, and most importantly, the *reasons* they are blocked.

This set of PowerShell functions will let you pull the server block list from another Mastodon instance. If we ever decide to put together a master block list, these can easily be adapted to use that list as a source.

Get-MastodonServerBlock.ps1

```
# You need to run the .ps file in PowerShell so that the functions get
# loaded but after that you can just use them like regular commands.
#
# e.g.
# Get-MastodonServerBlock `
#     -domain source.social `
#     -accessToken "source_domain_token" |
# Foreach-Object {
#     Add-MastodonServerBlock `
#         -domain target.social `
#         -accessToken "target_domain_token" `
#         -block $_
# }
#
# If you don't have admin access to another server, and that server is
# publicly allowing their blocks to bee seen, you can leave off the
# -accessToken parameter and get that version of the list. Some of the
```

```
# domains may be obfuscated in this case, and the obfuscated domains
# will be ignored by Add-MastodonServerBlock.

#############################################################################

# Creates an object used by Add-MastodonServerBlock from its constituent
# parameters. Really just a helper method.
function New-MastodonServerBlock {
    param (
        [Parameter(Mandatory = $true)][string] $domain,
        [Parameter(Mandatory = $true)][string] $severity,
        [Parameter(Mandatory = $false)][string] $justification = "",
        [Parameter(Mandatory = $false)][string] $comment = "",
        [Parameter(Mandatory = $false)][switch] $rejectMedia = $false,
        [Parameter(Mandatory = $false)][switch] $rejectReports = $false,
        [Parameter(Mandatory = $false)][switch] $obfuscate = $false
    )

    $block = @{
        domain          = $domain
        severity        = $severity
        public_comment  = $justification
        private_comment = $comment
        reject_media    = $rejectMedia
        reject_reports  = $rejectReports
        obfuscate       = $obfuscate
    }

    New-Object -TypeName PSObject -Property $block
}

# Retrieves the list of server blocks from a server.
# If the accessToken is specified, it will attempt to use the admin
# API otherwise if will use the instance API where domains names may
# be obfuscated.
function Get-MastodonServerBlock {
    param (
        [Parameter(Mandatory = $true)][string] $domain,
```

```
        [Parameter(Mandatory = $false)][string] $accessToken = $null
    )
    if ($accesstoken -ne $null) {
        (Invoke-WebRequest `
            -Uri "https://$domain/api/v1/admin/domain_blocks" `
            -Method GET `
            -Headers @{ Authorization = "Bearer $accesstoken" }).Content | ConvertFrom-Json
    }
    else {
        (Invoke-WebRequest `
            -Uri "https://$domain/api/v1/instance/domain_blocks" `
            -Method GET).Content | ConvertFrom-Json
    }
}

# Adds a server block to a server.
# Requires an access token with appropriate access.
function Add-MastodonServerBlock {
    param (
        [Parameter(Mandatory = $true)][string] $domain,
        [Parameter(Mandatory = $true)][string] $accesstoken,
        [Parameter(Mandatory = $true, ValueFromPipeline = $true)] $block
    )

    if ($block.domain.IndexOf("*") -ge 0) {
        Write-Warning "Ignoring obfuscated domain $($_.domain)"
        return
    }

    Invoke-WebRequest `
        -Uri "https://$domain/api/v1/admin/domain_blocks" `
        -Method POST `
        -Headers @{ Authorization = "Bearer $accesstoken" } `
        -Body $($_ | ConvertTo-Json) `
        -ContentType "application/json"
}
```

Revision #3
Created 2022-11-28 16:57:50 AST by Steve Dinn

Updated 2023-05-01 19:33:08 ADT by Steve Dinn