```
    ____    _____________________________________________________________________________________________________
   //¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
   \\__//¯¯¯ Get-CiscoMerakiIpAddress                                                                        ___//¯¯\\
    ¯¯¯\\__________________________________________________________________________________________________//¯¯\\__//
        ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯

\________________________________________________________________________________________________________________________/
  Start /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯
```

Develop several classes that reproduces the input/output of this Market 32 Cisco Meraki authentication hyperlink.

https://n137.network-auth.com/splash/?mac=88%3A15%3A44%3AA3%3AB7%3A10&real_ip=192.168.0.81&client_ip=10.201.240.180&client_mac=9C:B7:0D:20:08:FE&vap=0&a=a17554a0d2b15a664c0e73900184544f19e70227&b=17468474&auth_version=5&key=834c46c40a4248fae0dec59501aef3f0327e6738&acl_ver=P4903858V2&continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect

If you would like to refer to the actual SCRIPT that is exhibited below...?

```
    ______________________________________________________________________________________________________________
    | Cisco Meraki IP | https://github.com/mcc85s/FightingEntropy/blob/main/Scripts/Get-CiscoMerakiIPAddress.ps1 |
    ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

                                                                                                                  _______/
\________________________________________________________________________________________________________________/ Start
  Overview /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯
```

That's such a LONG URL that there's really next to no way for a HUMAN to be able to REMEMBER all of that info.
Let's break it down. Cast it to a variable, $String

```
# // ___________________________________________________________________
# // | Get the Cisco Meraki IP Address from the wireless access point, |
# // | or else we won't receive any birthday party invitations         |
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

$String = ( "https://n137.network-auth.com/splash/?mac=88%3A15%3A44%3AA3%3AB7"+
           "%3A10&real_ip=192.168.0.81&client_ip=10.201.240.180&client_mac=9"+
           "C:B7:0D:20:08:FE&vap=0&a=a17554a0d2b15a664c0e73900184544f19e7022"+
           "7&b=17468474&auth_version=5&key=834c46c40a4248fae0dec59501aef3f0"+
           "327e6738&acl_ver=P4903858V2&continue_url=http%3A%2F%2Fwww.msftco"+
           "nnecttest.com%2Fredirect" -join '')
```

Alright, let's use the color formatting from Visual Studio Code, to proceed with the lesson plan.

```
# // ___________________________________________________________________________________________________
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | $String is a really long string. Let's SPLIT it, so that it makes MORE SENSE                   |
# // |________________________________________________________________________________________________|
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | Property             | Type    | Value                                                         |
# // |______________________|_________|_______________________________________________________________|
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | Base                 | String  | https://n137.network-auth.com/splash/?                        |
# // | Network Mac Address  | String  | mac=88%3A15%3A44%3AA3%3AB7%3A10&                              |
# // | Network IP Address   | String+ | real_ip=192.168.0.81&                                         |
# // | Client IP  Address   | String+ | client_ip=10.201.240.180&                                     |
# // | Client Mac Address   | String  | client_mac=9C:B7:0D:20:08:FE&                                 |
# // | Virtual Access Point | Integer | vap=0&                                                        |
# // | Private A Variable   | String+ | a=a17554a0d2b15a664c0e73900184544f19e70227&                   |
# // | Private B Variable   | Integer | b=17468474&                                                   |
# // | Auth. Version        | Integer | auth_version=5&                                               |
# // | Auth. Key            | String+ | key=834c46c40a4248fae0dec59501aef3f0327e6738&                 |
# // | Access Control List  | String+ | acl_ver=P4903858V2&                                           |
# // | Continue URL         | String+ | continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect |
# // |______________________|_________|_______________________________________________________________|
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
                                                                                                               __________/
 \____________________________________________________________________________________________________________/ Overview
```

GolubCorpNetworkAuth /----------\
/----------

```
# // ------------------------------------------------------------------
# // | Create a verbatim copy of the class represented by the URL string |
# // ------------------------------------------------------------------

Class GolubCorpNetworkAuth
{
    Hidden [String] $base
    [String] $mac
    [String] $real_ip
    [String] $client_ip
    [String] $client_mac
    [UInt32] $vap
    [String] $a
    [UInt32] $b
    [UInt32] $auth_version
    [String] $key
    [String] $acl_ver
    [String] $continue_url
    GolubCorpNetworkAuth([String]$String)
    {
        # https://n137.network-auth.com/splash/?
        # mac=88%3A15%3A44%3AA3%3AB7%3A10&
        # real_ip=192.168.0.81&
        # client_ip=10.201.240.180&
        # client_mac=9C:B7:0D:20:08:FE&
        # vap=0&
        # a=a17554a0d2b15a664c0e73900184544f19e70227&
        # b=17468474&
        # auth_version=5&
        # key=834c46c40a4248fae0dec59501aef3f0327e6738&
        # acl_ver=P4903858V2&
        # continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect

        # // ------------------------------------------------------------
        # // | Use the Regex base class to catch+trim+split the URL query |
        # // ------------------------------------------------------------

        $E             = [Regex]::Matches($String,"\?.+").Value.TrimStart("?").Split("&")

        # // ------------------------------------------------------------------
        # // | Test the input, if it does not have (11) components, throw an error |
        # // ------------------------------------------------------------------

        If ($E.Count -ne 11)
        {
            Throw "Invalid entry"
        }

        $This.mac          = $This.Tx($E[0])
        $This.real_ip      = $This.Tx($E[1])
        $This.client_ip    = $This.Tx($E[2])
        $This.client_mac   = $This.Tx($E[3])
        $This.vap          = $This.Tx($E[4])
        $This.a            = $This.Tx($E[5])
        $This.b            = $This.Tx($E[6])
        $This.auth_version = $This.Tx($E[7])
        $This.key          = $This.Tx($E[8])
        $This.acl_ver      = $This.Tx($E[9])
        $This.continue_url = $This.Tx($E[10])

        $This.base         = [Regex]::Matches($String,".+\?").Value.TrimEnd("?")
    }
    [String] Tx([String]$Entry)
    {
        # Slice assignment
        $0, $1 = $Entry -Split "="

        # Property message
```

```
        Write-Host "Setting [~] Property: [$0], Value: [$1]" -ForegroundColor 10

        # Property assignment
        Return $1
    }
    [String] Out()
    {
        Return @( $This.PSObject.Properties | % { $_.Name, $_.Value -join "=" } ) -join "&"
    }
    [String] ToString()
    {
        Return "{0}?{1}" -f $This.Base, $This.Out()
    }
}
```

```
# // ____________________________________________________________________________________________________
# // | Now create an instantiation of the above class with the variable $String as it's only parameter |
# // ----------------------------------------------------------------------------------------------------

$Test = [GolubCorpNetworkAuth]$String

# OR... you can use
# $Test = [GolubCorpNetworkAuth]::New($String)
```

```
PS Prompt:\> $Test = [GolubCorpNetworkAuth]$String
Setting [~] Property: [mac], Value: [88%3A15%3A44%3AA3%3AB7%3A10]
Setting [~] Property: [real_ip], Value: [192.168.0.81]
Setting [~] Property: [client_ip], Value: [10.201.240.180]
Setting [~] Property: [client_mac], Value: [9C:B7:0D:20:08:FE]
Setting [~] Property: [vap], Value: [0]
Setting [~] Property: [a], Value: [a17554a0d2b15a664c0e739001845 44f19e70227]
Setting [~] Property: [b], Value: [17468474]
Setting [~] Property: [auth_version], Value: [5]
Setting [~] Property: [key], Value: [834c46c40a4248fae0dec59501aef3f0327e6738]
Setting [~] Property: [acl_ver], Value: [P4903858V2]
Setting [~] Property: [continue_url], Value: [http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect]

PS Prompt:\> $Test


mac          : 88%3A15%3A44%3AA3%3AB7%3A10
real_ip      : 192.168.0.81
client_ip    : 10.201.240.180
client_mac   : 9C:B7:0D:20:08:FE
vap          : 0
a            : a17554a0d2b15a664c0e739001845 44f19e70227
b            : 17468474
auth_version : 5
key          : 834c46c40a4248fae0dec59501aef3f0327e6738
acl_ver      : P4903858V2
continue_url : http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect
```

```
# // ____________________________________________________________________________________________________
# // |--------------------------------------------------------------------------------------------------|
# // | Cool. Now some of that information has to be processed into an object.                           |
# // | It IS an object right now, but- it actually has some issues such as ...                          |
# // | _____________________                                                                            |
# // | | %3A = : | %2F = / |                                                                            |
# // | ---------------------                                                                            |
# // | Uh-oh. Those are gonna cause problems if we use them VERBATIM. Because...                        |
# // | ...those are actually CHARACTER CODES so that the browser can process the input string.          |
# // | If we were to REPLACE EVERY "%" symbol with a [char]0x, we can get back the actual character.    |
# // |__________________________________________________________________________________________________|
# // ----------------------------------------------------------------------------------------------------
```

```
# // ____________________________________________________________________________________________________
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | PS Prompt:\> [char]0x2f                                                                          |
# // | /                                                                                                |
# // | PS Prompt:\> [char]0x3a                                                                          |
# // | :                                                                                                |
# // |__________________________________________________________________________________________________|
# // |¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
# // | So, now what I'm going to do, is create MULTIPLE CLASSES so that I can cleanly organize the      |
# // | components of the input string, in order to get a more complex object BACK from it.              |
# // |__________________________________________________________________________________________________|
# // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

```
                                                                                                              ______________________/
\____________________________________________________________________________________________________________/ GolubCorpNetworkAuth
  IPInterface /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This class will be for handling EACH of the NETWORK and the CLIENT addresses, (IP + Mac Address)

```
Class IpInterface
{
    [String] $Type
    [Object]   $IP
    [String]  $Mac
    IpInterface([UInt32]$Type,[String]$IPAddress,[String]$MacAddress)
    {
        # // ________________________________________________________________
        # // | Tests whether we're specifying a NETWORK or CLIENT interface |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($Type -notin 0..1)
        {
            Throw "Invalid address type"
        }

        # // ______________________________________________________________________
        # // | Assign the type based on the input integer (0: Network, 1: Client) |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Type = @("Network","Client")[$Type]

        # // _________________________________________________________
        # // | Tests whether the IP address matches IPv4 conventions |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($IPAddress -notmatch "(\d+\.\d+\.\d+\.\d+)")
        {
            Throw "Invalid IP Address"
        }

        # // ____________________________________________________________________________
        # // | Assigns the property "Ip", while also converting the string to an object |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        $This.Ip   = [IPAddress]$IPAddress

        # // _____________________________________________________
        # // | Tests whether the Mac address matches conventions |
        # // ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯

        If ($MacAddress -notmatch (@("[A-F0-9]{2}")*6 -join ":"))
        {
            Throw "Invalid Mac Address"
        }
```

```
        # // ______________________________
        # // | Assigns the property "Mac" |
        # // ------------------------------

        $This.Mac  = $MacAddress
    }
    [String] ToString()
    {
        Return $This.Type, $This.Ip, $This.Mac -join "&"
    }
}
```

```
                                                                                                            _____________/
\_________________________________________________________________________________________________________/ IPInterface
  ApAuthenticationToken /-----------------------------------------------------------------------------------------------\
/-----------------------
```

This class will handle the information that is being sent/received by the Cisco Meraki Wireless Lan Controller on the backend, which MAY be in the store...? Or, it may be at Pchop HQ...

```
__________________________________________________________
| Golub Corporation | 461 Nott St. Schenectady, NY 12308 |
----------------------------------------------------------
```

Then again, maybe it's still sitting over at:

```
_________________________________________________________
| Nfrastructure | 5 Enterprise Lane, Halfmoon, NY 12065 |
---------------------------------------------------------
```

Look, I have no idea where it is specifically, or whether they work with Nfrastructure still...
I just know that they both extensively use CISCO equipment.
It's what EXPERTS use.

```
Class ApAuthenticationToken
{
    [UInt32] $Index
    [String] $A
    [UInt32] $B
    [UInt32] $Version
    [String] $Key
    [String] $Acl
    ApAuthenticationToken([UInt32]$Index,[String]$A,[UInt32]$B,[UInt32]$Version,[String]$Key,[String]$Acl)
    {
        # // ______________________________________________________________________________________
        # // | Access Point Index is essentially "which access point is this being issued to...?" |
        # // | on the CISCO MERAKI WIRELESS LAN CONTROLLER...                                     |
        # // --------------------------------------------------------------------------------------

        $This.Index = $Index

        # // ____________________________________________________________________________
        # // | A is a 40-digit HEXADECIMAL address, which is 8 digits longer than a GUID |
        # // ----------------------------------------------------------------------------

        If ($A -notmatch "[a-f0-9]{40}")
        {
            Throw "Not a valid A"
        }

        # // ____________________________________
        # // | Now we can ASSIGN the property A |
        # // ------------------------------------

        $This.A = $A

        # // ___________________________________________________________________________________
        # // | B in the example, appears to be an 8-digit Integer/[UInt32], though it could be  |
        # // | LARGER or SMALLER. The parameter input will automatically test whether it is the |
        # // | correct type. Now we can ASSIGN the property B                                   |
        # // -----------------------------------------------------------------------------------
```

```
            $This.B = $B

            # // ____________________________________________________________________________________
            # // | Auth. Version in the example is a 1-digit integer, but perhaps it could be larger |
            # // | The parameter input will automatically test whether it is the correct type.       |
            # // | Now we can ASSIGN the property Version                                            |
            # // ------------------------------------------------------------------------------------

            $This.Version = $Version

            # // ____________________________________________________________________________
            # // | Key is a 40-digit HEXADECIMAL address, which is 8 digits longer than a GUID |
            # // ----------------------------------------------------------------------------

            If ($Key -notmatch "[a-f0-9]{40}")
            {
                Throw "Not a valid key"
            }

            # // ___________________
            # // | Assigns the key |
            # // -------------------

            $This.Key = $Key

            # // ____________________________________________________________________________________
            # // | ACL stands for access control list, it's apparently a string, since P+V are NOT |
            # // | hexadecimal characters                                                          |
            # // ------------------------------------------------------------------------------------

            $This.Acl = $Acl
        }
    }
```

```
                                                                                                                    ________________________/
\___________________________________________________________________________________________________________________/ ApAuthenticationToken
  GolubCorpNetworkAuth2 /-------------------------------------------------------------------------------------------------------------------\
/------------------------
```

So, this will be a more complex class that has main “branch” properties to make it APPEAR to be much simpler. It'll essentially have all of the same information as the original GolubCorpNetworkAuth class, but it will be FORMATTED differently.

```
    Class GolubCorpNetworkAuth2
    {
        [Object] $Network
        [Object] $Client
        [Object] $Token
        [String] $Continue
        [String] Cx([String]$String)
        {
            $A       = [Char[]]$String
            $X       = 0
            Return @( Do
            {
                Switch -Regex ($A[$X])
                {
                    "\%"
                    {
                        ("[Char]0x{0}{1}" -f $A[$X+1],$A[$X+2]) | Invoke-Expression; $X ++; $X ++
                    }
                    Default
                    {
                        $A[$X]
                    }
                }
                $X ++
            }
            Until ($X -eq $String.Length)) -join ''
        }
```

```
        GolubCorpNetworkAuth2([String]$String)
        {
            # // ____________________________________________________________
            # // | Here's the input, but split...                           |
            # // | https://n137.network-auth.com/splash/?                   |
            # // | mac=88%3A15%3A44%3AA3%3AB7%3A10&                         |
            # // | real_ip=192.168.0.81&                                    |
            # // | client_ip=10.201.240.180&                                |
            # // | client_mac=9C:B7:0D:20:08:FE&                            |
            # // | vap=0&                                                   |
            # // | a=a17554a0d2b15a664c0e739001845 44f19e70227&             |
            # // | b=17468474&                                              |
            # // | auth_version=5&                                          |
            # // | key=834c46c40a4248fae0dec59501aef3f0327e6738&            |
            # // | acl_ver=P4903858V2&                                      |
            # // | continue_url=http%3A%2F%2Fwww.msftconnecttest.com%2Fredirect |
            # // ------------------------------------------------------------

            # // ________________________________________________________________
            # // | Use the method Cx to convert any % input to the full character |
            # // ----------------------------------------------------------------

            $E = ForEach ($Item in [Regex]::Matches($String,"\?.+").Value.TrimStart("?").Split("&"))
            {
                $This.Cx($Item.Split("=")[1])
            }

            # // ___________________________________
            # // | Assign the network information |
            # // -----------------------------------

            $This.Network      = [IPInterface]::New(0,$E[1],$E[0])

            # // _________________________________
            # // | Assign the client information |
            # // ---------------------------------

            $This.Client       = [IPInterface]::New(1,$E[2],$E[3])

            # // __________________________________
            # // | Compile the token information |
            # // ----------------------------------

            $This.Token        = [ApAuthenticationToken]::New($E[4],$E[5],$E[6],$E[7],$E[8],$E[9])

            # // ________________________
            # // | Set the continue URL |
            # // ------------------------

            $This.Continue     = $E[10]
        }
    }
```

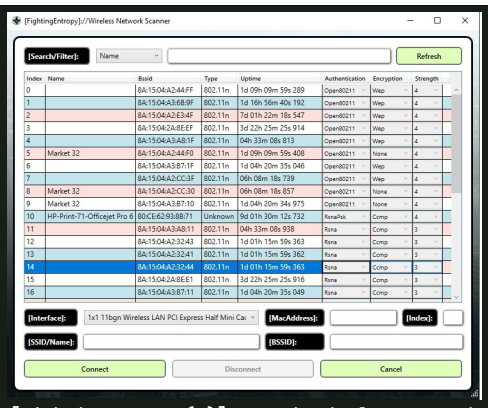
GolubCorpNetworkAuth2 Conclusion

Alright, now's the time to scope out the output of the above information. The information is NOT exactly the same, however, the NETWORK and CLIENT strings are EASIER to UNDERSTAND. That doesn't necessarily mean that will be the INTENDED result of the output string. There's plenty more to do with these classes, but that'll be another lesson.

```
PS Prompt:\> $Test2 | Format-List

Network  : Network&192.168.0.81&88:15:44:A3:B7:10
Client   : Client&10.201.240.180&9C:B7:0D:20:08:FE
Token    : ApAuthenticationToken
Continue : http://www.msftconnecttest.com/redirect
```

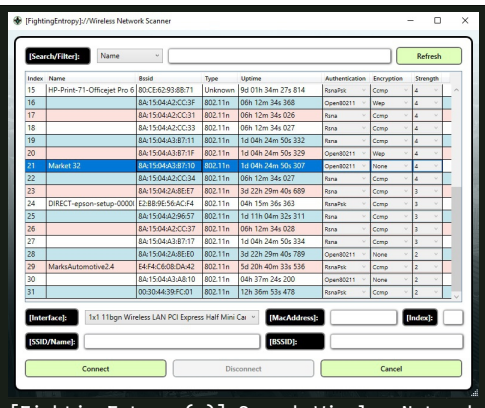
```
# // ---------------------------------------------------------
# // | Let's look at the SUBPROPERTIES of these properties  |
# // ---------------------------------------------------------

PS Prompt:\> $Test2.Network

Type    IP           Mac
----    --           ---
Network 192.168.0.81 88:15:44:A3:B7:10

PS Prompt:\> $Test2.Client

----   --            ---
Client 10.201.240.180 9C:B7:0D:20:08:FE

PS Prompt:\> $Test2.Token

Index   : 0
A       : a17554a0d2b15a664c0e73900184544f19e70227
B       : 17468474
Version : 5
Key     : 834c46c40a4248fae0dec59501aef3f0327e6738
Acl     : P4903858V2

PS Prompt:\> $Test2.Continue
http://www.msftconnecttest.com/redirect
```

There is additional work that needs to be done with this particular function to have it reproduce the ORIGINAL information, however, this code is effectively what is being done on the BACKEND of the server. The Cisco Meraki WLAN Controller has to maintain an encrypted/authenticated connection, EVEN IF IT IS OVER OPEN WIFI...
...and from what I can tell...



Index	Name	Bssid	Type	Uptime	Authentication	Encryption	Strength
0		8A:15:04:A2:44:FF	802.11n	1d 09h 09m 59s 289	Open80211	Wep	4
1		8A:15:04:A3:6B:9F	802.11n	1d 16h 56m 40s 192	Open80211	Wep	4
2		8A:15:04:A2:E3:4F	802.11n	7d 01h 22m 18s 547	Open80211	Wep	4
3		8A:15:04:2A:8E:EF	802.11n	3d 22h 25m 25s 914	Open80211	Wep	4
4		8A:15:04:A3:A8:1F	802.11n	04h 33m 08s 813	Open80211	Wep	4
5	Market 32	8A:15:04:A2:44:F0	802.11n	1d 09h 09m 59s 408	Open80211	None	4
6		8A:15:04:A3:B7:1F	802.11n	1d 04h 20m 35s 046	Open80211	Wep	4
7		8A:15:04:A2:CC:3F	802.11n	06h 08m 18s 739	Open80211	Wep	4
8	Market 32	8A:15:04:A2:CC:30	802.11n	06h 08m 18s 857	Open80211	None	4
9	Market 32	8A:15:04:A3:B7:10	802.11n	1d 04h 20m 34s 975	Open80211	None	4
10	HP-Print-71-Officejet Pro 6	80:CE:62:93:8B:71	Unknown	9d 01h 30m 12s 732	RsnaPsk	Ccmp	4
11		8A:15:04:A3:A8:11	802.11n	04h 33m 08s 938	Rsna	Ccmp	3
12		8A:15:04:A2:32:43	802.11n	1d 01h 15m 59s 363	Rsna	Ccmp	3
13		8A:15:04:A2:32:41	802.11n	1d 01h 15m 59s 362	Rsna	Ccmp	3
14		8A:15:04:A2:32:44	802.11n	1d 01h 15m 59s 363	Rsna	Ccmp	3
15		8A:15:04:2A:8E:E1	802.11n	3d 22h 25m 25s 916	Rsna	Ccmp	3
16		8A:15:04:A3:B7:11	802.11n	1d 04h 20m 35s 049	Rsna	Ccmp	3

[FightingEntropy(π)] Search-WirelessNetwork
Golub Corporation Cisco Meraki Results (1/2)
(Click to expand)



Index	Name	Bssid	Type	Uptime	Authentication	Encryption	Strength
15	HP-Print-71-Officejet Pro 6	80:CE:62:93:8B:71	Unknown	9d 01h 34m 27s 814	RsnaPsk	Ccmp	4
16		8A:15:04:A2:CC:3F	802.11n	06h 12m 34s 368	Open80211	Wep	4
17		8A:15:04:A2:CC:31	802.11n	06h 12m 34s 026	Rsna	Ccmp	4
18		8A:15:04:A2:CC:33	802.11n	06h 12m 34s 027	Rsna	Ccmp	4
19		8A:15:04:A3:B7:11	802.11n	1d 04h 24m 50s 332	Rsna	Ccmp	4
20		8A:15:04:A3:B7:1F	802.11n	1d 04h 24m 50s 329	Open80211	Wep	4
21	Market 32	8A:15:04:A3:B7:10	802.11n	1d 04h 24m 50s 307	Open80211	None	4
22		8A:15:04:A2:CC:34	802.11n	06h 12m 34s 027	Rsna	Ccmp	4
23		8A:15:04:2A:8E:E7	802.11n	3d 22h 29m 40s 689	Rsna	Ccmp	3
24	DIRECT-epson-setup-0000(	E2:BB:9E:56:AC:F4	802.11n	04h 15m 36s 363	RsnaPsk	Ccmp	3
25		8A:15:04:A2:96:57	802.11n	1d 11h 04m 32s 311	Rsna	Ccmp	3
26		8A:15:04:A2:CC:37	802.11n	06h 12m 34s 028	Rsna	Ccmp	3
27		8A:15:04:A3:B7:17	802.11n	1d 04h 24m 50s 334	Rsna	Ccmp	3
28		8A:15:04:2A:8E:E0	802.11n	3d 22h 29m 40s 789	Open80211	None	2
29	MarksAutomotive2.4	E4:F4:C6:08:DA:42	802.11n	5d 20h 40m 33s 536	RsnaPsk	Ccmp	2
30		8A:15:04:A3:A8:10	802.11n	04h 37m 24s 200	Open80211	None	2
31		00:30:44:39:FC:01	802.11n	12h 36m 53s 478	RsnaPsk	Ccmp	2

[FightingEntropy(π)] Search-WirelessNetwork
Golub Corporation Cisco Meraki Results (2/2)
(Click to expand)

`“88:15:44:A3:B7:10”` is in the second results box. The function Search-WirelessNetwork is available in the PowerShell module, [FightingEntropy(π)]. The instructions to install the module are located at...

```
 ______________________________________________________________________
| [FightingEntropy(π)] | https://github.com/mcc85s/FightingEntropy |
 ----------------------------------------------------------------------
                                                                                                  ___________/
\________________________________________________________________________________________________/ Conclusion
```

```
 ___________________________________________________________
|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|
|                                     Michael C. Cook Sr. |
|                                       Security Engineer |
|                                    Secure Digits Plus LLC |
|_________________________________________________________|
 ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```