T1016 - System Network Configuration Discovery

Description from ATT&CK

> Adversaries may look for details about the network configuration and settings, such as IP and/or MAC addresses, of systems they access or through information discovery of remote systems. Several operating system administration utilities exist that can be used to gather this information. Examples include [Arp](https://attack.mitre.org/software/S0099), [ipconfig](https://attack.mitre.org/software/S0100)/[ifconfig](https://attack.mitre.org/software/S0101), [nbtstat](https://attack.mitre.org/software/S0102), and [route](https://attack.mitre.org/software/S0103).
> Adversaries may also leverage a Network Device CLI on network devices to gather information about configurations and settings, such as IP addresses of configured interfaces and static/dynamic routes.(Citation: US-CERT-TA18-106A)(Citation: Mandiant APT41 Global Intrusion)
>
> Adversaries may use the information from System Network Configuration Discovery during automated discovery to shape follow-on behaviors, including determining certain access within the target network and what actions to do next.

Atomic Tests

- Atomic Test #1 - System Network Configuration Discovery on Windows
- Atomic Test #2 - List Windows Firewall Rules

- Atomic Test #5 - List Open Egress Ports
- Atomic Test #6 - Adfind - Enumerate Active Directory Subnet Objects
- Atomic Test #7 - Qakbot Recon
- Atomic Test #8 - List macOS Firewall Rules

Atomic Test #1 - System Network Configuration Discovery on Windows

T1003.004
T1003.005
T1003.006
T1003.007
T1003.008
T1003
T1006
T1007
T1010
T1012
T1014
T1016
src
T1016.md
T1016.yaml
T1018
T1020
T1021.001
T1021.002
T1021.003
T1021.006
T1027.001
T1027.002
T1027.004
T1027
T1030
T1033
T1036.003
T1036.004
T1036.005
T1036.006
T1036
T1037.001
T1037.002
T1037.004

Identify network configuration information

Upon successful execution, cmd.exe will spawn multiple commands to list network configuration settings. Output will be via stdout.

Supported Platforms: Windows

auto_generated_guid: 970ab6a1-0157-4f3f-9a73-ec4166754b23

Attack Commands: Run with `command_prompt`!

```
ipconfig /all
netsh interface show interface
arp -a
nbtstat -n
net config
```

Atomic Test #2 - List Windows Firewall Rules

Enumerates Windows Firewall Rules using netsh.

Upon successful execution, cmd.exe will spawn netsh.exe to list firewall rules. Output will be via stdout.

Supported Platforms: Windows

auto_generated_guid: 038263cb-00f4-4b0a-98ae-0696c67e1752

Attack Commands: Run with `command_prompt`!

```
netsh advfirewall firewall show rule name=all
```

Atomic Test #3 - System Network Configuration Discovery

Identify network configuration information.

Upon successful execution, sh will spawn multiple commands and output will be via stdout.

Supported Platforms: macOS, Linux

auto_generated_guid: c141bbdb-7fca-4254-9fd6-f47e79447e17

Attack Commands: Run with `sh`!

```
if [ -x "$(command -v arp)" ]; then arp -a; else echo "arp is missing fr
if [ -x "$(command -v ifconfig)" ]; then ifconfig; else echo "ifconfig i
if [ -x "$(command -v ip)" ]; then ip addr; else echo "ip is missing fro
if [ -x "$(command -v netstat)" ]; then netstat -ant | awk '{print $NF}'
```

Dependencies: Run with `sh`!

Description: Check if arp command exists on the machine

Check Prereq Commands:

```
if [ -x "$(command -v arp)" ]; then exit 0; else exit 1; fi;
```

Get Prereq Commands:

```
(which yum && yum -y install net-tools)||(which apt-get && DEBIAN_FRONTE
```

Atomic Test #4 - System Network Configuration Discovery (TrickBot Style)

Identify network configuration information as seen by Trickbot and described here https://www.sneakymonkey.net/2019/10/29/trickbot-analysis-part-ii/

Upon successful execution, cmd.exe will spawn `ipconfig /all`, `net config workstation`, `net view /all /domain`, `nltest /domain_trusts`. Output will be via stdout.

Supported Platforms: Windows

auto_generated_guid: dafaf052-5508-402d-bf77-51e0700c02e2

Attack Commands: Run with `command_prompt`!

```
ipconfig /all
net config workstation
net view /all /domain
nltest /domain_trusts
```

Atomic Test #5 - List Open Egress Ports

This is to test for what ports are open outbound. The technique used was taken from the following blog: https://www.blackhillsinfosec.com/poking-holes-in-the-firewall-egress-testing-with-allports-exposed/

Upon successful execution, powershell will read top-128.txt (ports) and contact each port to confirm if open or not. Output will be to Desktop\open-ports.txt.

Supported Platforms: Windows

auto_generated_guid: 4b467538-f102-491d-ace7-ed487b853bf5

Inputs:

Name	Description	Type	Default Value
output_file	Path of file to write port scan results	Path	$env:USERPROFILE\Desktop\open-ports.txt
portfile_url	URL to top-128.txt	Url	https://github.com/redcanaryco/atomic-red-team/raw/master/atomics/T1016/src/top-128.txt
port_file	The path to a text file containing ports to be scanned, one port per line. The default list uses the top 128 ports as	Path	PathToAtomicsFolder\T1016\src\top-128.txt

	defined by Nmap.		

Attack Commands: Run with `powershell`!

```
$ports = Get-content #{port_file}
$file = "#{output_file}"
$totalopen = 0
$totalports = 0
New-Item $file -Force
foreach ($port in $ports) {
    $test = new-object system.Net.Sockets.TcpClient
    $wait = $test.beginConnect("allports.exposed", $port, $null, $null)
    $wait.asyncwaithandle.waitone(250, $false) | Out-Null
    $totalports++ | Out-Null
    if ($test.Connected) {
        $result = "$port open"
        Write-Host -ForegroundColor Green $result
        $result | Out-File -Encoding ASCII -append $file
        $totalopen++ | Out-Null
    }
    else {
        $result = "$port closed"
        Write-Host -ForegroundColor Red $result
        $totalclosed++ | Out-Null
        $result | Out-File -Encoding ASCII -append $file
    }
}
$results = "There were a total of $totalopen open ports out of $totalpor
$results | Out-File -Encoding ASCII -append $file
Write-Host $results
```

Cleanup Commands:

```
Remove-Item -ErrorAction ignore "#{output_file}"
```

Dependencies: Run with `powershell`!

Description: Test requires #{port_file} to exist

Check Prereq Commands:

```
if (Test-Path "#{port_file}") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory (split-path #{port_file}) -ErrorAction ignore |
Invoke-WebRequest "#{portfile_url}" -OutFile "#{port_file}"
```

Atomic Test #6 - Adfind - Enumerate Active Directory Subnet Objects

Adfind tool can be used for reconnaissance in an Active directory environment. This example has been documented by ransomware actors enumerating Active Directory Subnet Objects reference- http://www.joeware.net/freetools/tools/adfind/, https://www.fireeye.com/blog/threat-research/2019/04/pick-six-intercepting-a-fin6-intrusion.html

Supported Platforms: Windows

auto_generated_guid: 9bb45dd7-c466-4f93-83a1-be30e56033ee

Inputs:

Name	Description	Type	Default Value
adfind_path	Path to the AdFind executable	Path	PathToAtomicsFolder\T1087.002\src\AdFind.exe

Attack Commands: Run with `command_prompt`!

```
#{adfind_path} -f (objectcategory=subnet)
```

Dependencies: Run with `powershell`!

Description: AdFind.exe must exist on disk at specified location (#{adfind_path})

Check Prereq Commands:

```
if (Test-Path #{adfind_path}) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Invoke-WebRequest -Uri "https://github.com/redcanaryco/atomic-red-team/r
```

Atomic Test #7 - Qakbot Recon

A list of commands known to be performed by Qakbot for recon purposes

Supported Platforms: Windows

auto_generated_guid: 121de5c6-5818-4868-b8a7-8fd07c455c1b

Inputs:

Name	Description	Type	Default Value
recon_commands	File that houses list of commands to be executed	Path	PathToAtomicsFolder\T1016\src\qakbot.bat

Attack Commands: Run with `command_prompt`!

```
#{recon_commands}
```

Atomic Test #8 - List macOS Firewall Rules

"This will test if the macOS firewall is enabled and/or show what rules are configured. Must be run with elevated privileges. Upon successful execution, these commands will output various information about the firewall configuration, including status and specific port/protocol blocks or allows.

Using `defaults`, additional arguments can be added to see filtered details, such as `globalstate` for global configuration ("Is it on or off?"), `firewall` for common application allow rules, and `explicitauths` for specific rules configured by the user.

Using `socketfilterfw`, flags such as --getglobalstate or --listapps can be used for similar filtering. At least one flag is required to send parseable output to standard out.

Supported Platforms: macOS

auto_generated_guid: ff1d8c25-2aa4-4f18-a425-fede4a41ee88

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo defaults read /Library/Preferences/com.apple.alf
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --getglobalstate
```