Introducing Slinky Cat - Living off the AD Land

labs.lares.com/introducing-slinkycat

Andy Gill

July 7, 2023

penetrationtesting

Slinky Cat has been developed to automate some of the methods introduced in living off the land and to supplement ScrapingKit. To help security and IT teams reduce their AD exposures and uncover quick wins and fixes designed for pen-testers and defenders alike.

Slinky Cat has been developed to automate some of the methods introduced in our previous blog post https://labs.lares.com/living-off-the-land/. It also goes hand in hand with our ScrapingKit tooling released last month.

Why SlinkyCat?

Funny name, right? We thought so, too; Neil has a cat called Slinky hence the tool's name, plus it's got a spooks esq naming ;).

SlinkyCat has been designed to help security, and IT teams reduce their AD exposures and uncover quick wins and fixes designed for pentesters and defenders alike. It also forms part of our offensive sysadmin toolkit that we have released as dual-purpose tooling; there is an accompanying blog post for that suite too, which can be found here(tool and blog post will be published after our talk on Saturday 8th July) ;

https://labs.lares.com/offensive-sysadmin/

Two Faces of the Same Coin

SlinkyCat has been designed to supplement both offense and defensive teams with the ability to offer an audit approach and additional quick wins for the penetration testers. The tooling will give the option of an HTML output that is digestible and searchable and gives fix options where applicable or highlights potential other attack paths that might be worth investigating.

Using The Tooling

Slinky Cat attempts to give users an easy-to-navigate menu offering predefined Active Directory Service Interfaces (ADSI) and .NET `System.DirectoryServices.AccountManagement` namespace queries which can be used to enumerate a Windows domain.

`[ADSISearcher]` is a type of accelerator for the class **System.DirectoryServices.DirectorySearcher.** It is used to search for one or more objects based on a filter. Whereas `[ADSI]` represents the class `System.DirectoryServices.DirectoryEntry`. It binds directly to objects such as an AD group, user, or computer. It can then be used to update the attributes of a particular object.

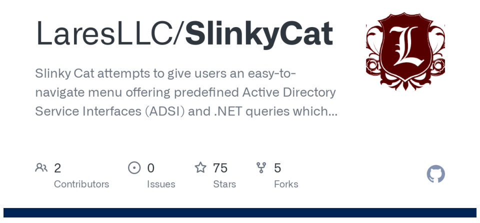
GitHub - LaresLLC/SlinkyCat: Slinky Cat attempts to give users an easy-to-navigate menu offering predefined Active Directory Service Interfaces (ADSI) and .NET queries which can be used to enumerate a Windows domain.

Slinky Cat attempts to give users an easy-to-navigate menu offering predefined Active Directory Service Interfaces (ADSI) and .NET queries which can be used to enumerate a Windows domain. - GitHub…

GitHubLaresLLC

The following lab was created to demonstrate Slinky Cat:

```
Domain - hacklab.local
Domain Controller - Microsoft Windows Server
2022 10.0.20348 N/A Build 20348
Domain Host 1 - Microsoft Windows 11 Enterprise
10.0.22621 N/A Build 22621
Domain Host 2 - Microsoft Windows 11 Enterprise
10.0.22621 N/A Build 22621
```


LaresLLC/**SlinkyCat**

Slinky Cat attempts to give users an easy-to-navigate menu offering predefined Active Directory Service Interfaces (ADSI) and .NET queries which...

2 Contributors · 0 Issues · 75 Stars · 5 Forks

How to Execute

It’s good practice to monitor which PowerShell modules have already been loaded into a session; you can do this by executing the following.

```
PS C:\Users\user1> Get-Module
ModuleType Version Name ExportedCommands
---------- ------- ---- ----------------
Manifest 3.1.0.0 Microsoft.PowerShell.Management {Add-Computer, Add-Content,
Checkpoint-Computer, Clear-Content...}

Script     2.0.0      PSReadLine {Get-PSReadLineKeyHandler, Get-PSReadLineOption,
Remove-PSReadLineKeyHandler, Set-PSReadLineKeyHandler...}
```

By default, PowerShell is very restrictive of which scripts can be imported; you must run the following line to bypass this for one session, allowing you to run your scripts.

```
PS C:\Users\user1> powershell.exe -nop -exec bypass
```

The `-nop`: (NoProfile) parameter instructs PowerShell not to load the user's profile script; this enables PowerShell to start up faster since it doesn't have to process any customizations or configurations defined in the profile.

The `-exec bypass`: This parameter is used to bypass the PowerShell execution policy.

For a more permanent solution, the execution policy can be set with `Set-ExecutionPolicy Unrestricted`; note that this is not recommended on non-attacking machines as it reduces the machine's security by setting an unrestricted security policy.

Then to import Slinky Cat type the following:

```
PS C:\Users\user1\Desktop\Scripts\Land> Import-Module .\SlinkyCat.ps1
Invoke-SlinkyCat
```

This should then load the user menu; we built a menu system to show the user helpful information and double up as a sysadmin tool rather than using something like AD Users and Computers, ADExplorer, or BloodHound.

```
=== Menu ===
1. ADSI Enumeration
2. Dot NET System.DirectoryServices.AccountManagement Namespace Enumeration
Q. Quit
```

Options

Selecting 1 reveals the predefined ADSI domain enumeration queries; these can be used to help reveal information about a Windows domain.

```
1. ADSI Enumeration

=== ADSI Enumeration ===
ADSI Options Menu Please select an option:
1. Enumerate All Domain Hosts
2. Enumerate All Domain Controllers
3. Enumerate All Domain Users
4. List All Users in the Domain admins group
5. List All accounts with an SPN
6. List All Domain groups
7. List All password set to never expire
8. List All Users which do not require a password
9. List All Users with password must change at next logon
10. List All Computers that are not Domain Controllers and are Windows 7
11. List All Computers that are not Domain Controllers and are Windows 10
12. List All Computers that are not Domain Controllers and are Windows 11
13. List All Servers
14. List All Server 2008
15. List All Server 2012
16. List All Server 2016
17. List All Server 2019
18. List All Server 2022
19. List Domain Groups which are a Member of the local admin group
20. List All Trusts Established with a Domain
21. List All Exchange Servers
22. List All Accounts that have never logged in
23. List All Domain User Accounts which have a completed AD description field
24. List All accounts that reference 'pass' in their AD description field
25. List All Users who have not changed their password in over 1 year
26. List All Users' last password change date and time
27. List All Systems with WinRM Open (Not OPSEC SAFE!)
28. List All Systems with RDP Open (Not OPSEC SAFE!)
29. Find All Machines where the current user has local admin access (Not OPSEC
Safe, will list All Computers then attempt to mount C$)
A. Run All Functions and Export to an Output Folder Full of txt Files
Q. Quit
```

It is not unheard of for system administrators to write verbose comments in a domain user's description field; any personalised comment can reveal helpful information.

```
Select an option: 23

Option: ADSI List all accounts that reference pass in their AD description field

Username: Administrator
Description: Built-in account for administering the computer/domain

Username: Guest
Description: Built-in account for guest access to the computer/domain

Username: krbtgt
Description: Key Distribution Center Service Account

Username: t.matkinson
Description: New users’ password is set to Welcome123

Press Enter to continue
```

There is also an option to scrape any reference of the word password across all user’s description fields.

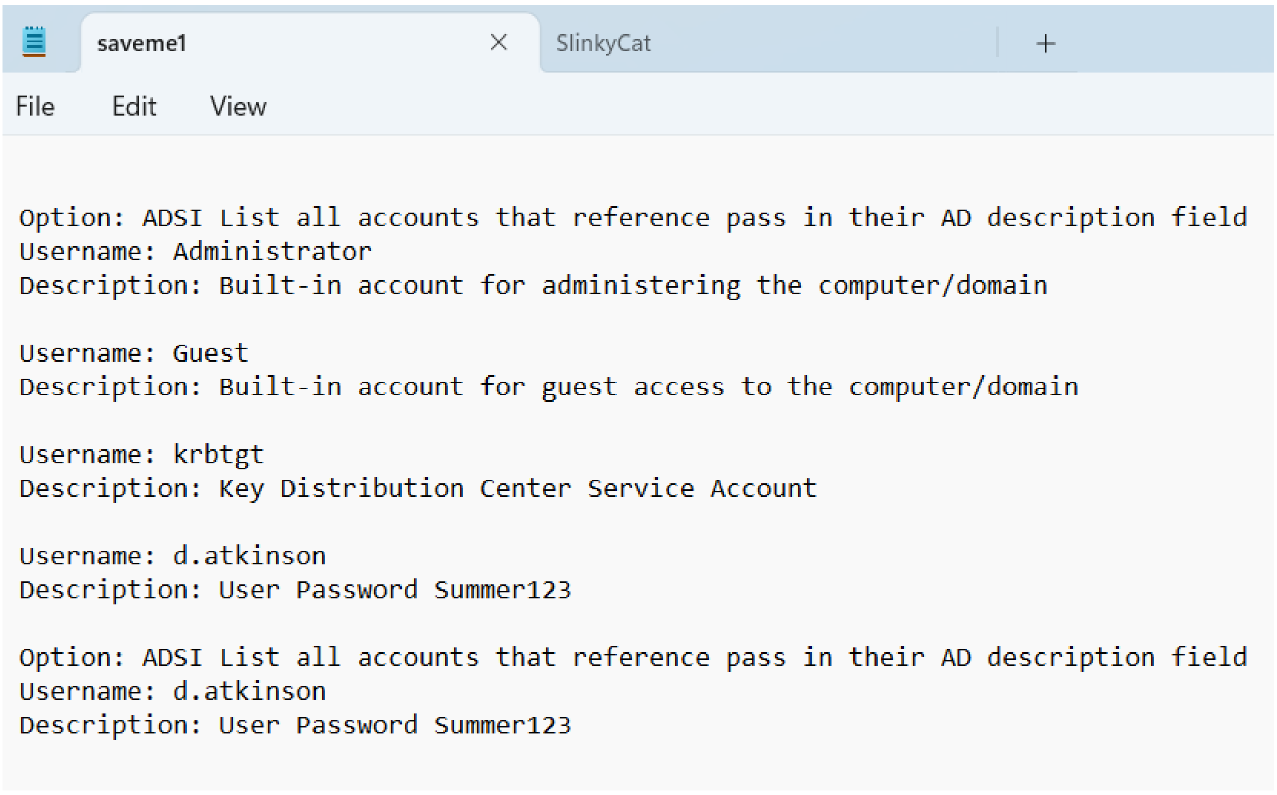
```
Select an option: 24
Option: ADSI List all accounts that reference pass in their AD description field

Username: m.sparkles
Description: User Password Summer123

Press Enter to continue
```

While all responses are echoed to the console, you can save all commands and their results to disk by selecting option `1` then `A` which will run all functions and make an output directory with various txt files; the script will warn you before it does this.

```
=== Menu ===
1. ADSI Enumeration
2. Dot NET System.DirectoryServices.AccountManagement Namespace Enumeration
3. List Available Individual Functions
Q. Quit
Select an option:
```



```
Option: ADSI List all accounts that reference pass in their AD description field
Username: Administrator
Description: Built-in account for administering the computer/domain

Username: Guest
Description: Built-in account for guest access to the computer/domain

Username: krbtgt
Description: Key Distribution Center Service Account

Username: d.atkinson
Description: User Password Summer123

Option: ADSI List all accounts that reference pass in their AD description field
Username: d.atkinson
Description: User Password Summer123
```

Screenshot of the recorded results

After quitting, you can execute SlinkyCat again by simply typing the following.

```
PS C:\Users\user1\Desktop\Scripts\Land> Invoke-SlinkyCat
```

Rebooting the machine will delete Slinky Cat from memory, but if you wish to unload it manually, you can do so by typing the following.

```
PS C:\Users\user1\Desktop\Scripts\Land> Remove-Module SlinkyCat
```

So how does SlinkyCat work?

The `CombinedMenu` the function loads the primary menu, which contains different options.

The `ADSIMenu` function displays the ADSI submenu options.

Active Directory Service Interfaces (ADSI) is a set of programming interfaces that enables interaction with the directory services provided by Microsoft Active Directory.

The `DotNetMenu` function displays the submenu for the .NET System.DirectoryServices.AccountManagement namespace enumeration options.

The `System.DirectoryServices.AccountManagement` namespace is a part of the .NET Framework and enables interaction with directory services, such as Active Directory.

The `$domain` variable stores the domain name queried by environmental variable but can also be specified by running `Invoke-SlinkyCat -Domain lares.com` while on a domain-joined machine.

The script uses loops to return to the menus after execution; it continues the main loop until the user chooses "Q" to quit. Each option gives different enumeration options with OPSEC warnings in some instances where functions will take a while or trigger many alerts.

Wrapping Up

The techniques demonstrated are nothing revolutionary but they use built-in functionality without the need for installing additional modules. The tooling will enable pentesters and sysadmins to spot potential misconfiguration within an Active Directory estate and pick up on quick wins.