Detecting Common User/Domain Recon

Domain Reconnaissance

`Active Directory (AD) domain reconnaissance` represents a pivotal stage in the cyberattack lifecycle. During this phase, adversaries endeavor to gather information about the target environment, seeking to comprehend its architecture, network topology, security measures, and potential vulnerabilities.

While conducting AD domain reconnaissance, attackers focus on identifying crucial components such as Domain Controllers, user accounts, groups, trust relationships, organizational units (OUs), group policies, and other vital objects. By gaining insights into the AD environment, attackers can potentially pinpoint high-value targets, escalate their privileges, and move laterally within the network.

User/Domain Reconnaissance Using Native Windows Executables

An example of AD domain reconnaissance is when an adversary executes the `net group` command to obtain a list of `Domain Administrators`.

```
C:\Users\Administrator>net group "Domain Admins" /domain
The request will be processed at a domain controller for domain lab.internal.local
.

Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            BRUCE_GEORGE             CHANCE_ARMSTRONG
HOPE_ADKINS              TYLER_MORRIS
The command completed successfully.
```

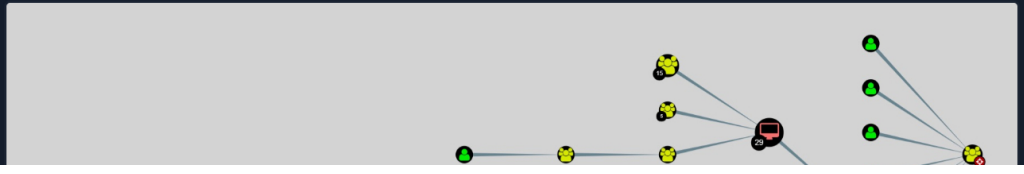
Common native tools/commands utilized for domain reconnaissance include:

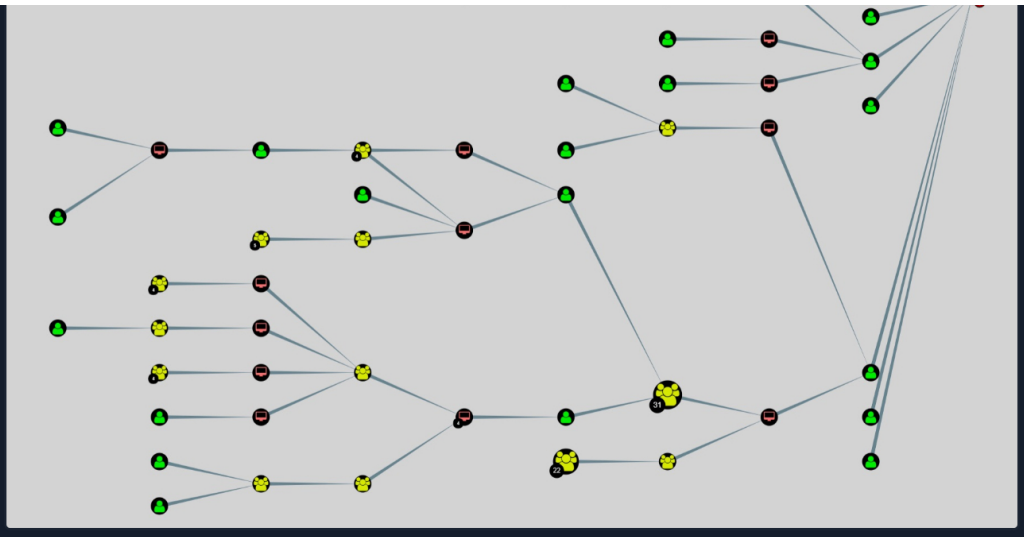
- `whoami /all`
- `wmic computersystem get domain`
- `net user /domain`
- `net group "Domain Admins" /domain`
- `arp -a`
- `nltest /domain_trusts`

For detection, administrators can employ PowerShell to monitor for unusual scripts or cmdlets and process command-line monitoring.

User/Domain Reconnaissance Using BloodHound/SharpHound

BloodHound is an open-source domain reconnaissance tool created to analyze and visualize the Active Directory (AD) environment. It is frequently employed by attackers to discern attack paths and potential security risks within an organization's AD infrastructure. BloodHound leverages graph theory and relationship mapping to elucidate trust relationships, permissions, and group memberships within the AD domain.





Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Sharphound is a C# data collector for BloodHound. An example of usage includes an adversary running Sharphound with all collection methods (`-c all`).

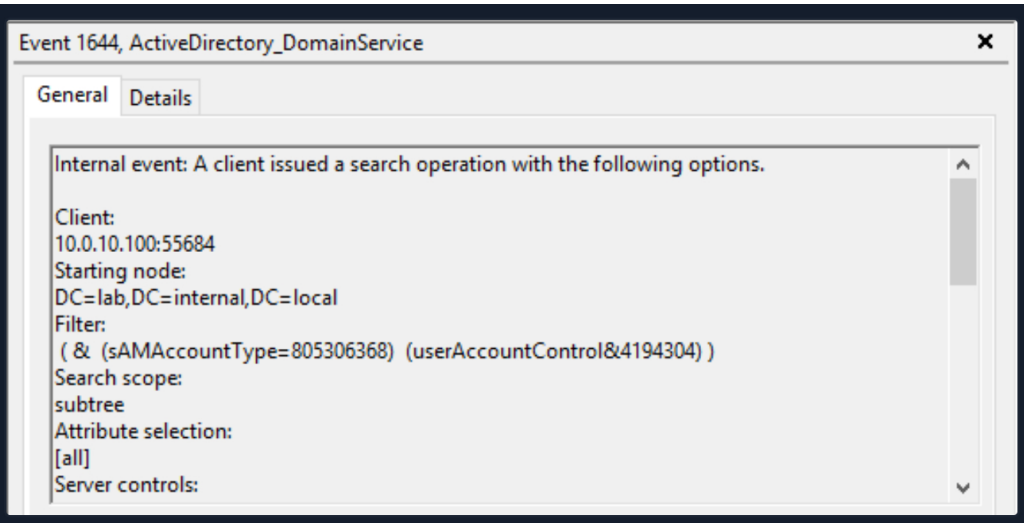
```
PS C:\Users\JENNY_HICKMAN\tools> .\Sharphound3.exe -c all
----------------------------------------------
Initializing SharpHound at 4:29 PM on 3/9/2021
----------------------------------------------

Resolved Collection Methods: Group, Sessions, LoggedOn, Trusts, ACL, ObjectProps, L
s, Container

[+] Creating Schema map for domain LAB.INTERNAL.LOCAL using path CN=Schema,CN=Confi
ternal,DC=local
[+] Cache File not Found: 0 Objects in cache

[+] Pre-populating Domain Controller SIDS
Status: 0 objects finished (+0) -- Using 20 MB RAM
Status: 3385 objects finished (+3385 564.1667)/s -- Using 48 MB RAM
Enumeration finished in 00:00:06.0237191
Compressing data to .\20210309162903_BloodHound.zip
You can upload this file directly to the UI

SharpHound Enumeration Completed at 4:29 PM on 3/9/2021! Happy Graphing!

PS C:\Users\JENNY_HICKMAN\tools>
```

BloodHound Detection Opportunities

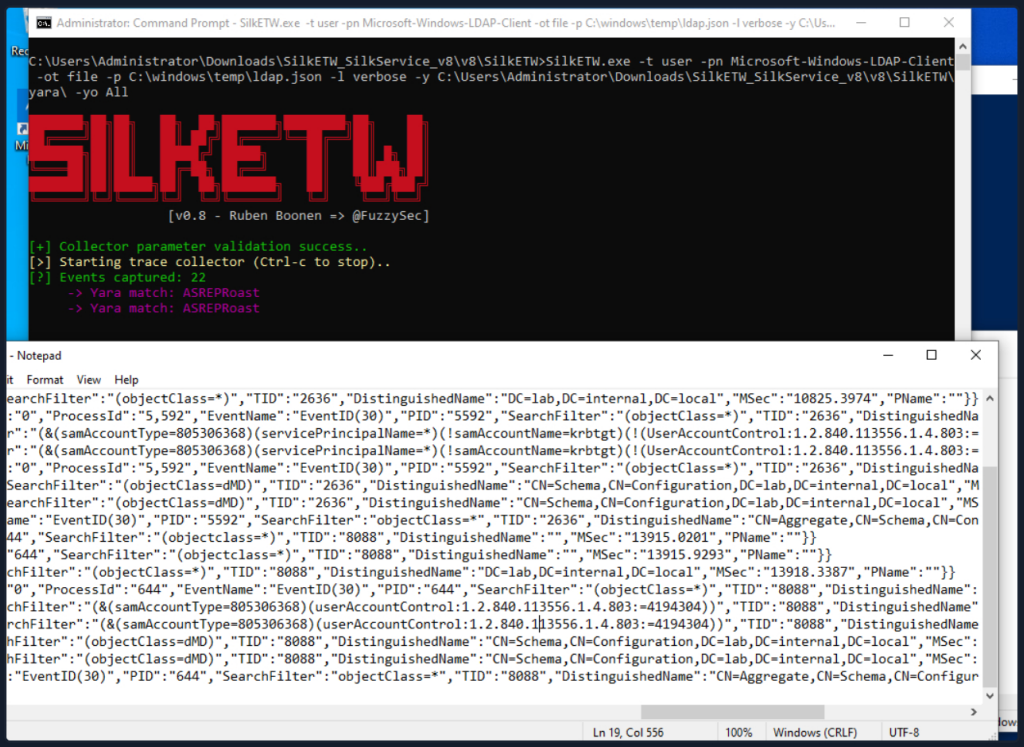
Under the hood, the BloodHound collector executes numerous LDAP queries directed at the Domain Controller, aiming to amass information about the domain.

```
if ((methods & ResolvedCollectionMethod.ACL) != 0)
{
    filterparts.Add("(|(samAccountType=805306368)(samAccountType=805306369)(samAccountType=268435456)(samAccountType=268435457)(samAccountType=536870912)(samAcc
    props.AddRange(new[]
    {
        "samaccountname", "distinguishedname", "dnshostname", "samaccounttype", "ntsecuritydescriptor", "displayname", "objectclass", "objectsid", "name"
    });
}

if ((methods & ResolvedCollectionMethod.ObjectProps) != 0)
{
    filterparts.Add("(|(samaccounttype=268435456)(samaccounttype=268435457)(samaccounttype=536870912)(samaccounttype=536870913)(samaccounttype=805306368)(samacc
    props.AddRange(new[]
    {
        "samaccountname", "distinguishedname", "samaccounttype", "pwdlastset", "lastlogon", "lastlogontimestamp", "objectsid",
        "sidhistory", "useraccountcontrol", "dnshostname", "operatingsystem",
        "operatingsystemservicepack", "serviceprincipalname", "displayname", "mail", "title",
        "homedirectory","description","admincount","userpassword","gpcfilesyspath","objectclass",
        "msds-behavior-version","objectguid", "name", "gpoptions", "msds-allowedToDelegateTo", "msDS-AllowedToActOnBehalfOfOtherIdentity"
    });
}

if ((methods & ResolvedCollectionMethod.GPOLocalGroup) != 0)
{
    filterparts.Add("(&(objectCategory=groupPolicyContainer)(name=*)(gpcfilesyspath=*))");
    props.AddRange(new[]
    {
        "displayname", "name", "gpcfilesyspath", "objectclass"
    });
}
```

However, monitoring LDAP queries can be a challenge. By default, the Windows Event Log does not record them. The best option Windows can suggest is employing `Event 1644` - the LDAP performance monitoring log. Even with it enabled, BloodHound may not generate many of the expected events.



A more reliable approach is to utilize the Windows ETW provider `Microsoft-Windows-LDAP-Client`. As showcased previously in the `SOC Analyst` path, SilkETW & SilkService are versatile C# wrappers for ETW, designed to simplify the intricacies of ETW, providing an accessible interface for research and introspection. `SilkService` supports output to the Windows Event Log, which streamlines log digestion. Another useful feature is the ability to employ `Yara` rules for hunting suspicious LDAP queries.



In addition, Microsoft's ATP team has compiled a list of LDAP filters frequently used by reconnaissance tools.

Recon tool	Filter
enum_ad_user_comments (Metasploit)	(&(&(objectCategory=person)(objectClass=user))(|(description=*pass*)(comment=*pass*)))
enum_ad_computers (Metasploit)	(&(objectCategory=computer)(operatingSystem=*server*))
enum_ad_groups (Metasploit)	(&(objectClass=group))
	(&(objectClass=group)(managedBy=*)),

enum_ad_managedby_groups (Metasploit)	(&(objectClass=group)(managedBy=*)(groupType:1.2.840.113556.1.4.803:=2147483648))
Get-NetComputer (PowerView)	(&(sAMAccountType=805306369)(dnshostname=*))
Get-NetUser - Users (Powerview)	(&(samAccountType=805306368)(samAccountName=*)
Get-NetUser - SPNs (Powerview)	(&(samAccountType=805306368)(servicePrincipalName=*)
Get-DFSshareV2 (Powerview)	(&(objectClass=msDFS-Linkv2))
Get-NetOU (PowerView)	(&(objectCategory =organizationalUnit)(name=*))
Get-DomainSearcher (Empire)	(samAccountType=805306368)

Armed with this list of LDAP filters, BloodHound activity can be detected more efficiently.

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, access the Splunk interface at http://[Target IP]:8000 and launch the Search & Reporting Splunk application. The vast majority of searches covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

Detecting User/Domain Recon With Splunk

You'll observe that a specific timeframe is given when identifying each attack. This is done to concentrate on the relevant events, avoiding the overwhelming volume of unrelated events.

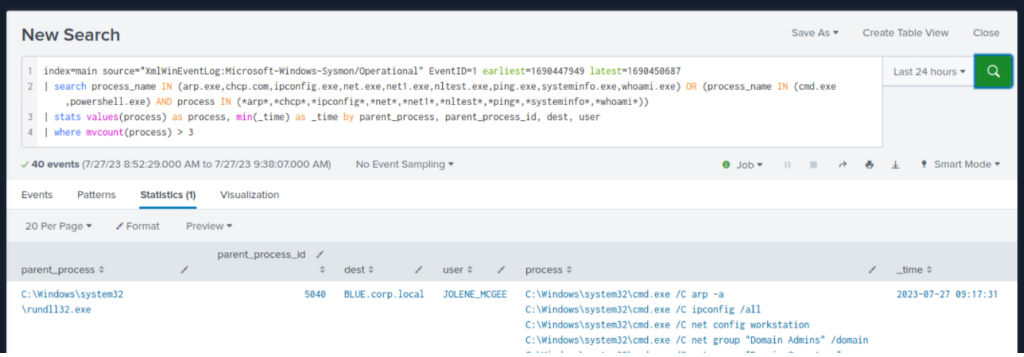
Now let's explore how we can identify the recon techniques previously discussed, using Splunk.

Detecting Recon By Targeting Native Windows Executables

Timeframe: `earliest=1690447949 latest=1690450687`

Detecting Common User/Domain Recon

```
index=main source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventID=1 earliest=16904479
| search process_name IN (arp.exe,chcp.com,ipconfig.exe,net.exe,net1.exe,nltest.exe,ping.exe,system
| stats values(process) as process, min(_time) as _time by parent_process, parent_process_id, dest,
| where mvcount(process) > 3
```



```
C:\Windows\system32\cmd.exe /C net group "Domain Computers"
/domain
C:\Windows\system32\cmd.exe /C net group "domain admins" /dom
C:\Windows\system32\cmd.exe /C net group "enterprise admins" /dom
C:\Windows\system32\cmd.exe /C net localgroup "administrators"
/dom
C:\Windows\system32\cmd.exe /C net view /all
C:\Windows\system32\cmd.exe /C net view /all /domain
C:\Windows\system32\cmd.exe /C net1 config workstation
C:\Windows\system32\cmd.exe /C nltest /domain_trusts
C:\Windows\system32\cmd.exe /C nltest /domain_trusts /all_trusts
C:\Windows\system32\cmd.exe /C systeminfo
C:\Windows\system32\cmd.exe /C whoami
C:\Windows\system32\cmd.exe /C whoami /upn
```

Search Breakdown:

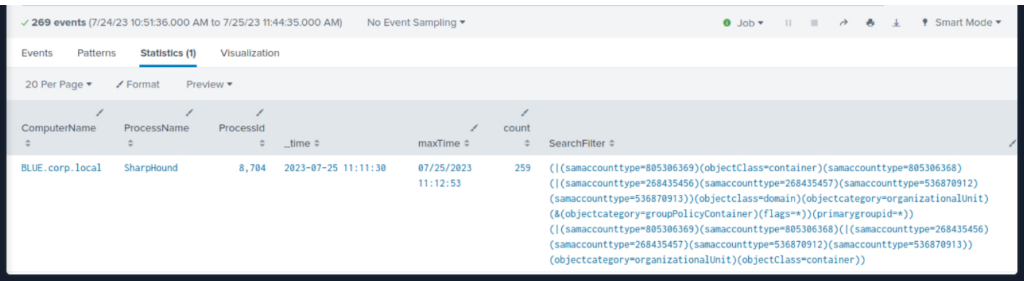
- `Filtering by Index and Source`: The search begins by selecting events from the main index where the source is `XmlWinEventLog:Microsoft-Windows-Sysmon/Operational`, which is the XML-formatted Windows Event Log for Sysmon (System Monitor) events. Sysmon is a service and device driver that logs system activity to the event log.
- `EventID Filter`: The search is further filtered to only select events with an `Event ID` of `1`. In Sysmon, Event ID 1 corresponds to `Process Creation` events, which log data about newly created processes.
- `Time Range Filter`: The search restricts the time range of events to those occurring between the Unix timestamps 1690447949 and 1690450687. These timestamps represent the earliest and latest times in which the events occurred.
- `Process Name Filter`: The search then filters events to only include those where the process_name field is one of a list of specific process names (e.g., `arp.exe`, `chcp.com`, `ipconfig.exe`, etc.) or where the `process_name` field is `cmd.exe` or `powershell.exe` and the process field contains certain substrings. This step is looking for events that involve certain system or network-related commands, as well as events where these commands were run from a Command Prompt or PowerShell session.
- `Statistics`: The stats command is used to aggregate events based on the fields `parent_process`, `parent_process_id`, `dest`, and `user`. For each unique combination of these fields, the search calculates the following statistics:
  - `values(process) as process`: This captures all unique values of the `process field` as a multivalue field named `process`.
  - `min(_time) as _time`: This captures the earliest time (`_time`) that an event occurred within each group.
- `Filtering by Process Count`: The where command is used to filter the results to only include those where the count of the process field is greater than `3`. This step is looking for instances where multiple processes (more than three) were executed by the same parent process.

Detecting Recon By Targeting BloodHound

Timeframe: `earliest=1690195896 latest=1690285475`

Detecting Common User/Domain Recon

```
index=main earliest=1690195896 latest=1690285475 source="WinEventLog:SilkService-Log"
| spath input=Message
| rename XmlEventData.* as *
| table _time, ComputerName, ProcessName, ProcessId, DistinguishedName, SearchFilter
| sort 0 _time
| search SearchFilter="*(samAccountType=805306368)*"
| stats min(_time) as _time, max(_time) as maxTime, count, values(SearchFilter) as SearchFilter by
| where count > 10
| convert ctime(maxTime)
```



Search Breakdown:

- `Filtering by Index and Source`: The search starts by selecting events from the main index where the source is `WinEventLog:SilkService-Log`. This source represents Windows Event Log data gathered by `SilkETW`.
- `Time Range Filter`: The search restricts the time range of events to those occurring between the Unix timestamps 1690195896 and 1690285475. These timestamps represent the earliest and latest times in which the events occurred.
- `Path Extraction`: The `spath` command is used to extract fields from the `Message` field, which likely contains structured data such as `XML` or `JSON`. The `spath` command automatically identifies and extracts fields based on the data structure.
- `Field Renaming`: The `rename` command is used to rename fields that start with `XmlEventData.` to the equivalent field names without the `XmlEventData.` prefix. This is done for easier reference to the fields in later stages of the search.
- `Tabulating Results`: The `table` command is used to display the results in a tabular format with the following columns: `_time`, `ComputerName`, `ProcessName`, `ProcessId`, `DistinguishedName`, and `SearchFilter`. The `table` command only includes these fields in the output.
- `Sorting`: The `sort` command is used to sort the results based on the `_time` field in ascending order (from oldest to newest). The `0` argument means that there is no limit on the number of results to sort.
- `Search Filter`: The search command is used to filter the results to only include events where the `SearchFilter` field contains the string `*(samAccountType=805306368)*`. This step is looking for events related to LDAP queries with a specific filter condition.
- `Statistics`: The `stats` command is used to aggregate events based on the fields `ComputerName`, `ProcessName`, and `ProcessId`. For each unique combination of these fields, the search calculates the following statistics:
  - `min(_time) as _time`: The earliest time (`_time`) that an event occurred within each group.
  - `max(_time) as maxTime`: The latest time (`_time`) that an event occurred within each group.
  - `count`: The number of events within each group.
  - `values(SearchFilter) as SearchFilter`: All unique values of the `SearchFilter` field within each group.
- `Filtering by Event Count`: The `where` command is used to filter the results to only include those where the `count` field is greater than `10`. This step is looking for instances where the same process on the same computer made more than ten search queries with the specified filter condition.
- `Time Conversion`: The `convert` command is used to convert the `maxTime` field from Unix timestamp format to human-readable format (`ctime`).

PROTOCOL

UDP 1337 TCP 443

DOWNLOAD VPN CONNECTION FILE

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 141ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

+ 1 Modify and employ the Splunk search provided at the end of this section on all ingested data (All time) to find all process names that made LDAP queries where the filter includes the string *(samAccountType=805306368)*. Enter the missing process name from the following list as your answer. N/A, Rubeus, SharpHound, mmc, powershell, _

rundll32

Submit

Next

Mark Complete & Next


Next
Mark Complete & Next
Powered by HACKTHEBOX