Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

redcanaryco / atomic-red-team Public Notifications Fork 2.8k Star 9.7k

Code Issues 6 Pull requests 5 Actions Wiki Security Insights

Files

f339e7d

Go to file

- .github
- atomic_red_team
- atomics
  - Indexes
  - T1003.001
  - T1003.002
  - T1003.003
  - T1003.004
  - T1003.005
  - T1003.006
  - T1003.007
  - T1003.008
  - T1003
  - T1006
  - T1007
  - T1010
  - T1012
  - T1014
  - T1016
  - T1018
  - T1020
  - T1021.001
  - T1021.002
  - T1021.003
  - T1021.006
  - T1027.001
  - T1027.002
  - T1027.004
  - T1027
  - T1030
  - T1033
  - T1036.003
  - T1036.004
  - T1036.005
  - T1036.006
  - T1036

atomic-red-team / atomics / T1547.006 / T1547.006.md

Atomic Red Team doc generat... Generated docs from job=generate-d... 819934c · 2 years ago History

Preview Code Blame 73 lines (44 loc) · 4.13 KB Raw

T1547.006 - Kernel Modules and Extensions

Description from ATT&CK

> Adversaries may modify the kernel to automatically execute programs on system boot. Loadable Kernel Modules (LKMs) are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system.(Citation: Linux Kernel Programming)
> When used maliciously, LKMs can be a type of kernel-mode Rootkit that run with the highest operating system privilege (Ring 0).(Citation: Linux Kernel Module Programming Guide) Common features of LKM based rootkits include: hiding itself, selective hiding of files, processes and network activity, as well as log tampering, providing authenticated backdoors, and enabling root access to non-privileged users.(Citation: iDefense Rootkit Overview)
>
> Kernel extensions, also called kext, are used in macOS to load functionality onto a system similar to LKMs for Linux. Since the kernel is responsible for enforcing security and the kernel extensions run as apart of the kernel, kexts are not governed by macOS security policies. Kexts are loaded and unloaded through `kextload` and `kextunload` commands. Kexts need to be signed with a developer ID that is granted privileges by Apple allowing it to sign Kernel extensions. Developers without these privileges may still sign kexts but they will not load unless SIP is disabled. If SIP is enabled, the kext signature is verified before being added to the AuxKC.(Citation: System and kernel extensions in macOS)
>
> Since macOS Catalina 10.15, kernel extensions have been deprecated in favor of System Extensions. However, kexts are still allowed as "Legacy System Extensions" since there is no System Extension for Kernel Programming Interfaces.(Citation: Apple Kernel Extension Deprecation)
>
> Adversaries can use LKMs and kexts to conduct Persistence and/or Privilege Escalation on a system. Examples have been found in the wild, and there are some relevant open source projects as well.(Citation: Volatility Phalanx2)(Citation: CrowdStrike Linux Rootkit)(Citation: GitHub Reptile)(Citation: GitHub Diamorphine)(Citation: RSAC 2015 San Francisco Patrick Wardle)(Citation: Synack Secure Kernel Extension Broken)(Citation: Securelist Ventir)(Citation: Trend Micro Skidmap)

Atomic Tests

- Atomic Test #1 - Linux - Load Kernel Module via insmod

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

Atomic Test #1 - Linux - Load Kernel Module via insmod

This test uses the insmod command to load a kernel module for Linux.

Supported Platforms: Linux

auto_generated_guid: 687dcb93-9656-4853-9c36-9977315e9d23

Inputs:

Name	Description	Type	Default Value
module_name	Name of the kernel module name.	String	T1547006
module_path	Folder used to store the module.	Path	/tmp/T1547.006/T1547006.ko
temp_folder	Temp folder used to compile the code.	Path	/tmp/T1547.006
module_source_path	Path to download Gsecdump binary file	Url	PathToAtomicsFolder/T1547.006/src

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

```
sudo insmod #{module_path}
```

Cleanup Commands:

```
sudo rmmod #{module_name}
[ -f #{temp_folder}/safe_to_delete ] && rm -rf #{temp_folder}
```

Dependencies: Run with `bash`!

Description: The kernel module must exist on disk at specified location

Check Prereq Commands:

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

Get Prereq Commands:

```
if [ ! -d #{temp_folder} ]; then mkdir #{temp_folder}; touch #{temp_fold
cp #{module_source_path}/* #{temp_folder}/
cd #{temp_folder}; make
if [ ! -f #{module_path} ]; then mv #{temp_folder}/#{module_name}.ko #{m
```