NSudo | System Administration Toolkit

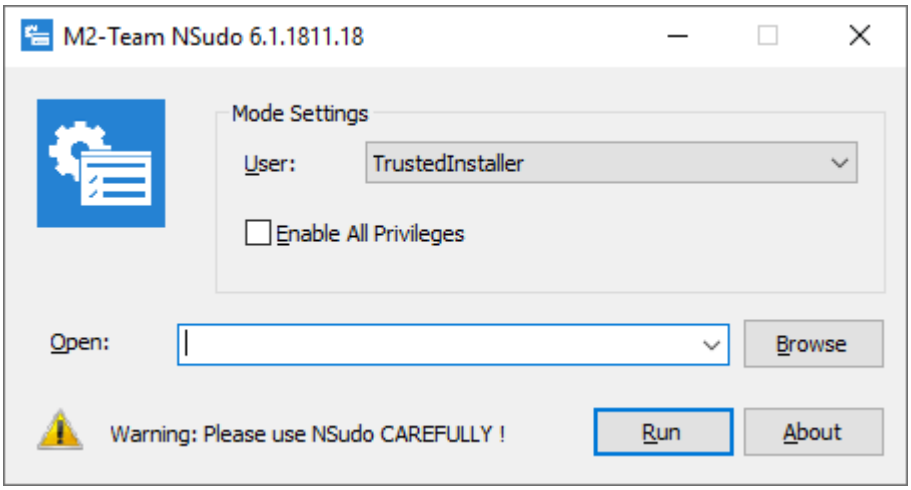

Table of Contents

About NSudo

Features

- Distributed under the MIT License.
- Provide the x86, x86-64, ARM64 binaries.
- Support Windows Vista and later.
- Using VC-LTL and libkcrt from Chuyu Team for smaller binary size.
- Using C++17, but only use core language features in the most cases.
- NSudo Launcher
    - Launch programs with TrustedInstaller access token.
    - Launch programs with System access token.
    - Launch programs with current user access token.
        - Note: If the User Account Control has not been disabled, the privilege of this mode is similar to the standard user.
    - Launch programs with elevated current user access token.
        - Note: The privilege of this mode is similar to the elevated user.
    - Launch programs with current process access token.
        - Note: The privilege of this mode is similar to the elevated user.
    - Launch programs with current process access token with the LUA restriction.
        - Note: The privilege of this mode is similar to the standard user. And the implementation is similar to the iertutil.dll's from the Internet Explorer.
    - Support launching programs with the specified privileges option.
        - Note: "Enable All Privileges" and "Disable All Privileges".
    - Support launching programs with the specified mandatory level (or integrity level) option.
        - Note: "System", "High", "Medium" and "Low".
    - Support launching programs with the specified process priority option.
        - Note: "Idle", "BelowNormal", "Normal", "AboveNormal", "High" and "RealTime".
    - Support launching programs with the specified window mode option.
        - Note: "Show", "Hide", "Maximize" and "Minimize".
    - Support waiting for the created process to end before exiting.
    - Support launching programs with the specified current directory.
    - Support launching programs with the current console window.
    - Support shortcut list.
        - Note: You can custom it via editing NSudo.json.
    - Support multiple command line style.
    - Support multi-languages.
        - Note: Chinese Simplified, Chinese Traditional, English, French, German, Italian and Spanish.
    - Full High DPI Support.
        - Note: As good as the implementation from Windows Shell (conhost.exe), with the full Per-Monitor DPI-Aware support under Windows 10 Build 10240 and later version and full System DPI-Aware support under Windows Vista to Windows 8.1.
    - Full accessibility support.
        - Note: You can use NSudo Launcher with Windows Narrator smoothly.
    - High performance.

        - Note: Because it don't need to create the windows service and the windows service process.
    - Provide C APIs and .Net Core bindings for developers.
- NSudo Devil Mode
    - The most elegant solution for developers who want to bypass the file and registry access checks for the process with the Administrator privilege.
    - Hook APIs via the Microsoft Detours library for the maximum compatibility.
    - The binary only depend on the exported named APIs from ntdll.dll.

System requirements

- Supported OS Version: Windows NT 6.0 or later
- Supported CPU Architecture: x86, x86-64(AMD64), ARM64

Prototype

NSudo Launcher is based on SuperCMD by Raymai97. Visit here for more information about SuperCMD.

NSudo Devil Mode is based on the Dism++ God Mode, with new features and several improvements.

Third-party projects which uses NSudo

- MSMG ToolKit
- Sledgehammer (WUMT Wrapper Script)
- Dism++

Third-party introduction

- HowToDoNinja: https://howtodoninja.com/how-to/nsudo-run-programs-with-full-admin-privileges-windows/
- MajorGeeks: https://www.majorgeeks.com/files/details/nsudo.html
- softpedia.com: https://www.softpedia.com/get/Tweak/System-Tweak/NSudo.shtml
- TrishTech.com: https://www.trishtech.com/2018/11/nsudo-run-programs-with-full-privileges-in-windows/
- Wilders Security Forums: https://www.wilderssecurity.com/threads/396818

Become the sponsor of NSudo

Patreon: https://www.patreon.com/MouriNaruto

爱发电: https://afdian.net/@MouriNaruto

- If I have reached the "$1,000 per month" goal, I will release 2 major releases in a year.
- If I have reached the "$2,000 per month" goal, I will obtain an extended validation (EV) code signing certificate and use it for signing the projects binaries.

Thanks for your support.

Kenji Mouri

Support

Contact

- E-mail: Mouri_Naruto@Outlook.com

Community

- GitHub Issues
- My Digital Life
- QQ Group

Usage

Download NSudo

Binaries

- Current Release
- All Releases
- AppVeyor CI

Source Code

- GitHub
- Gitee

NSudo Installer (Unofficial)

- Source Code
- Current Release

Chocolatey (Unofficial)

> choco install nsudo

scoop (Unofficial)

```
scoop bucket add extras
scoop install nsudo
```

Third-party download site

- MajorGeeks
- softpedia.com

NSudo Launcher

Quick Start

Please go to the [CPU Architecture] folder and click NSudo.exe. Follow the prompts. For example, if you want to use 64-bit NSudo on your Intel or AMD device, you need to go to the x64 folder and click NSudoG.exe

Command Line

```
Format: NSudoL [ Options and parameters ] Command line or ShortCut Command

Options:
```

```
-U:[ Option ] Create a process with specified user option.
Available options:
    T TrustedInstaller
    S System
    C Current User
    E Current User (Elevated)
    P Current Process
    D Current Process (Drop right)
PS: This is a mandatory parameter.

-P:[ Option ] Create a process with specified privilege option.
Available options:
    E Enable All Privileges
    D Disable All Privileges
PS: If you want to use the default privileges to create a process, please do
not include the "-P" parameter.

-M:[ Option ] Create a process with specified Integrity Level option.
Available options:
    S System
    H High
    M Medium
    L Low
PS: If you want to use the default Integrity Level to create a process, please
do not include the "-M" parameter.

-Priority:[ Option ] Create a process with specified process priority option.
Available options:
    Idle
    BelowNormal
    Normal
    AboveNormal
    High
    RealTime
PS: If you want to use the default Process Priority to create a process, please
do not include the "-Priority" parameter.

-ShowWindowMode:[ Option ] Create a process with specified window mode option.
Available options:
    Show
    Hide
    Maximize
    Minimize
PS: If you want to use the default window mode to create a process, please do
not include the "-ShowWindowMode" parameter.

-Wait Make NSudo Launcher wait for the created process to end before exiting.
PS: If you don't want to wait, please do not include the "-Wait" parameter.

-CurrentDirectory:[ DirectoryPath ] Set the current directory for the process.
PS: If you want to use the NSudo Launcher's current directory, please do not
include the "-CurrentDirectory" parameter.
```

```
-UseCurrentConsole Create a process with the current console window.
PS: If you want to create a process with the new console window, please do not
include the "-UseCurrentConsole" parameter.

-Version Show version information of NSudo Launcher.

-? Show this content.
-H Show this content.
-Help Show this content.

Please use https://github.com/Thdub/NSudo_Installer for context menu management.

PS:
    1. All NSudo Launcher command arguments is case-insensitive.
    2. You can use the "/" or "--" override "-" and use the "=" override ":" in
       the command line parameters.  For example, "/U:T" and "-U=T" are
       equivalent.
    3. To ensure the best experience, NSudoLC does not support context menu.

Example:
    If you want to run Command Prompt with TrustedInstaller, enable all
    privileges and the default Integrity Level.
        NSudoL -U:T -P:E cmd
```

Example: If you want to run Command Prompt with TrustedInstaller, enable all privileges and the default Integrity Level:

> NSudo -U:T -P:E cmd

Starting from NSudo 5.0.1708.16, the command line nested quotes is supported. For example:

> NSudo -U:T cmd /c "dir "C:\Program Files" & pause"

Shortcut List

You can edit NSudo.json to custom the Shortcut list, here are the demo (NSudo.json in the NSudo.exe's folder):

```
{
  "ShortCutList_V2": {

    "Command Prompt": "cmd",
    "PowerShell": "powershell",
    "PowerShell ISE": "powershell_ise",
    "Edit Hosts": "notepad %windir%\\System32\\Drivers\\etc\\hosts"
  }
}
```

NSudo Devil Mode

NSudo Devil Mode is a elegant solution for developers who want to bypass the file and registry access checks. It hooks some file and registry Windows NT kernel system calls via Microsoft Detours, so developers only need to load NSudoDM.dll into their apps before enjoy it.

NSudo Devil Mode only needs Administrator privilege. So developers need to run their apps as Administrator if they want to use it.

As the creator of NSudo project, I think NSudo Devil Mode may replace tools similar as NSudo in the most cases. NSudo 8.0 will support run apps as NSudo Devil Mode, and Dism++ God Mode will be refactored with NSudo Devil Mode. So NSudo will be more professional in the future because I don't want to make NSudo is replaced by NSudo Devil Mode, lol.

Why I create the NSudo Devil Mode

NSudo Shared Library is hard for developers to integrate it because it expose a lot of details about Windows security model and looks like a low level library. I don't think only providing NSudo Shared Library is good for developers who want to bypass the file and registry access checks. So I have created the NSudo Devil Mode.

Origin of NSudo Devil Mode

NSudo Devil Mode is based on the Dism++ God Mode or call it "Dism++ 春哥附体" via Chinese. I have refactored the implementations of Dism++ God Mode, add some new features and make its source code available in NSudo's GitHub repository.

The list of hooked Windows NT kernel system calls

Name	Origin
NtCreateKey	Dism++ God Mode.
NtCreateKeyTransacted	NSudo Devil Mode.
NtOpenKey	Dism++ God Mode. Extended in NSudo Devil Mode.
NtOpenKeyTransacted	NSudo Devil Mode.
NtOpenKeyEx	Dism++ God Mode.
NtOpenKeyTransactedEx	NSudo Devil Mode.
NtCreateFile	Dism++ God Mode.
NtOpenFile	Dism++ God Mode.

How to use NSudo Devil Mode

You can enable it via LoadLibrary and disable it via FreeLibrary. Here is a demo code.

```
using System;
using System.IO;
using System.Runtime.InteropServices;

namespace Demo
```

```
{
    class Program
    {
        [DllImport("kernel32.dll", CharSet = CharSet.Unicode)]
        static extern IntPtr LoadLibrary(string lpLibFileName);

        [DllImport("kernel32.dll", SetLastError = true)]
        [return: MarshalAs(UnmanagedType.Bool)]
        static extern bool FreeLibrary(IntPtr hLibModule);

        static void Main(string[] args)
        {
            IntPtr NSudoDevilModeModuleHandle = LoadLibrary(
                @"E:\GitHub\M2Team\NSudo\Output\Release\x64\NSudoDevilMode.dll");

            {
                DirectoryInfo Folder = new DirectoryInfo(
                @"C:\System Volume Information");

                foreach (FileInfo File in Folder.GetFiles())
                {
                    Console.WriteLine(File.FullName);
                }
            }

            FreeLibrary(NSudoDevilModeModuleHandle);

            {
                DirectoryInfo Folder = new DirectoryInfo(
                @"C:\System Volume Information");

                foreach (FileInfo File in Folder.GetFiles())
                {
                    Console.WriteLine(File.FullName);
                }
            }

            Console.ReadKey();
        }
    }
}
```

NSudo 恶魔模式的技术内幕

启用 SeBackupPrivilege 和 SeRestorePrivilege 是前提条件, 但是你也需要在创建文件 或注册表句柄的时候传入对应的选项, 否则是不生效的。

首先说明一点, 那就是 Windows 内核当发现调用者上下文为 SYSTEM 令牌的时候, 据 Microsoft 文档描述是为了提升 Windows 的性能会自动忽略掉大部分访问检查, 毕竟很多 Windows 系统关键组件运行在 SYSTEM 令牌上下文下面, 对于 Windows 用户模式而言, SYSTEM 令牌是至高无上的, 所以访问检查没必要做, 做了也提升不了安全性反而降低了 效率。所以这也是为什么除了 SYSTEM 令牌上下文外的其他令牌都需要启用相关特权 + 创建文件和注册表句柄的 API 传入对应选项才能忽略掉相关访问检查。

我用一个最简单的例子来说明减少不需要的内核级访问检查的好处, 那就是在 Windows AppContainer 下运行的代码, 由于会多出一个额外的内核级访问检查 (用 IDA 分析 ntoskrnl.exe, 然后用 F5 查看相关函数可以发现, 其实就是多出了一个分支和寥寥数行 实现), 大概会比在 AppContainer 外运行会损失 15% 的性能 (这也可以说明越底层的 实现越需要重视性能问题)。Windows AppContainer 是 Windows 8 开始提供的用户模式 沙盒, 主要用在商店应用和浏览器的沙盒上面。

Windows 的大部分内部使用了创建文件和注册表句柄的 API 并没有传入对应的选项, 于是 就出现了普通管理员下即使开启了这两个特权有些目录照样还是无法进行增删查改。而 NSudo 恶魔模式通过 Inline Hook 对 Windows 用户模式的系统调用层进行挂钩以 智能传入相关选项, 这也是 NSudo 恶魔模式能在非 SYSTEM 的但拥有这两个特权的 令牌上下文下绕过文件和注册表访问判断的缘由。

Windows 用户模式系统调用层指的是 ntdll.dll 导出的前缀为 Nt 或 Zw 的 API, Windows 用户模式下的 API 最终全会调用这部分以通过软中断陷阱门或者系统调用指令 进入内核模式完成最终操作。

智能, 指的是只有当前进程令牌上下文能够启用 SeBackupPrivilege 和 SeRestorePrivilege 的时候, 才会传入对应选项。毕竟如果这两个特权没有开启的话, 传入了相关选项是会返回错误的, 这也是为什么 Windows 相关实现并没有传入的原因。

当然 NSudo 恶魔模式为了对调用者更加透明和符合最小权限原则, 在初始化的时候首先会 创建一份当前进程令牌的模拟令牌副本, 然后对该副本开启这两个特权。在 Hook 中, 会先备份当前线程上下文的令牌, 接着替换成模拟令牌副本 (或者用 Microsoft 文档的 称法是模拟令牌上下文), 传入相关选项调用原 API 后再恢复为原来线程上下文的令牌。 (实现细节请参考在 NSudo 代码仓库的 NSudo 恶魔模式的源代码)

我说的有些啰嗦, 请见谅, 希望对你有帮助。

NSudo Shared Library

NSudoCreateProcess function

Creates a new process and its primary thread.

C/C++ prototype

```
EXTERN_C HRESULT WINAPI NSudoCreateProcess(
    _In_ NSUDO_USER_MODE_TYPE UserModeType,
    _In_ NSUDO_PRIVILEGES_MODE_TYPE PrivilegesModeType,
    _In_ NSUDO_MANDATORY_LABEL_TYPE MandatoryLabelType,
    _In_ NSUDO_PROCESS_PRIORITY_CLASS_TYPE ProcessPriorityClassType,
    _In_ NSUDO_SHOW_WINDOW_MODE_TYPE ShowWindowModeType,
    _In_ DWORD WaitInterval,
    _In_ BOOL CreateNewConsole,
    _In_ LPCWSTR CommandLine,
    _In_opt_ LPCWSTR CurrentDirectory);
```

UserModeType parameter

A value from the NSUDO_USER_MODE_TYPE enumerated type that identifies the user mode.

```
typedef enum class _NSUDO_USER_MODE_TYPE
{
    DEFAULT,
    TRUSTED_INSTALLER,
    SYSTEM,
    CURRENT_USER,
    CURRENT_PROCESS,
    CURRENT_PROCESS_DROP_RIGHT
} NSUDO_USER_MODE_TYPE, *PNSUDO_USER_MODE_TYPE;
```

PrivilegesModeType parameter

A value from the NSUDO_PRIVILEGES_MODE_TYPE enumerated type that identifies the privileges mode.

```
typedef enum class _NSUDO_PRIVILEGES_MODE_TYPE
{
    DEFAULT,
    ENABLE_ALL_PRIVILEGES,
    DISABLE_ALL_PRIVILEGES
} NSUDO_PRIVILEGES_MODE_TYPE, *PNSUDO_PRIVILEGES_MODE_TYPE;
```

MandatoryLabelType parameter

A value from the NSUDO_MANDATORY_LABEL_TYPE enumerated type that identifies the mandatory label.

```
typedef enum class _NSUDO_MANDATORY_LABEL_TYPE
{
    UNTRUSTED,
    LOW,
    MEDIUM,
    MEDIUM_PLUS,
    HIGH,
    SYSTEM,
    PROTECTED_PROCESS,
} NSUDO_MANDATORY_LABEL_TYPE, *PNSUDO_MANDATORY_LABEL_TYPE;
```

ProcessPriorityClassType parameter

A value from the NSUDO_PROCESS_PRIORITY_CLASS_TYPE enumerated type that identifies the process priority class.

```
typedef enum class _NSUDO_PROCESS_PRIORITY_CLASS_TYPE
{
    IDLE,
    BELOW_NORMAL,
```

```
    NORMAL,
    ABOVE_NORMAL,
    HIGH,
    REALTIME,
} NSUDO_PROCESS_PRIORITY_CLASS_TYPE, *PNSUDO_PROCESS_PRIORITY_CLASS_TYPE;
```

ShowWindowModeType parameter

A value from the NSUDO_SHOW_WINDOW_MODE_TYPE enumerated type that identifies the ShowWindow mode.

```
typedef enum class _NSUDO_SHOW_WINDOW_MODE_TYPE
{
    DEFAULT,
    SHOW,
    HIDE,
    MAXIMIZE,
    MINIMIZE,
} NSUDO_SHOW_WINDOW_MODE_TYPE, *PNSUDO_SHOW_WINDOW_MODE_TYPE;
```

WaitInterval parameter

The time-out interval for waiting the process, in milliseconds.

CreateNewConsole parameter

If this parameter is TRUE, the new process has a new console, instead of inheriting its parent's console (the default).

CommandLine parameter

The command line to be executed. The maximum length of this string is 32K characters, the module name portion of CommandLine is limited to MAX_PATH characters.

CurrentDirectory parameter

The full path to the current directory for the process. The string can also specify a UNC path. If this parameter is nullptr, the new process will the same current drive and directory as the calling process. (This feature is provided primarily for shells that need to start an application and specify its initial drive and working directory.)

Return value

HRESULT. If the function succeeds, the return value is S_OK.

C# API

Load the M2.NSudo assembly to your project, you will know the usage.

Example

```
using System;

namespace M2.NSudo.Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            NSudoInstance instance = new NSudoInstance();

            instance.CreateProcess(
                NSUDO_USER_MODE_TYPE.TRUSTED_INSTALLER,
                NSUDO_PRIVILEGES_MODE_TYPE.ENABLE_ALL_PRIVILEGES,
                NSUDO_MANDATORY_LABEL_TYPE.SYSTEM,
                NSUDO_PROCESS_PRIORITY_CLASS_TYPE.NORMAL,
                NSUDO_SHOW_WINDOW_MODE_TYPE.DEFAULT,
                0,
                true,
                "cmd",
                null);

            Console.ReadKey();
        }
    }
}
```

Release Notes

NSudo 8.2

- Add the Current User (Elevated) mode support. (Advised by xspeed1989.)
- Fix the blocking bug when using NSudo under Windows Service context. (Thanks to xspeed1989.)
- Improve several implementations.
- Fix the issue that the UI is Chinese when NSudo is running under an unsupported language setting. (Thanks to rlesch.)(#56)
- Update to the latest Mile.Cpp packages.
    - Update Mile.Project to the latest Mile.Project.VisualStudio.
    - Merge Mile.Windows.TrustedLibraryLoader and Mile.Platform.Windows to the latest Mile.Library.
    - Update to the latest VC-LTL.
- Update Windows Template Library (WTL) to 10.0.10320 Release.
- Remove ARM32 support.
    - Reason: https://forums.mydigitallife.net/threads/59268/page-28#post-1660432
- Make several improvements in the documentation.
    - Improve the website. (Contributed by 青春永不落幕.)
    - Improve the Gitee experience.
    - Use GitHub Actions to deploy the website.
- Add German Language. (Contributed by Steve.)
- Remove some experiment implementations, including NSudo Sweeper.
- Add logging support.

NSudo 8.0 Update 1 (8.0.1)

- Update Italian translation. (Contributed by garf02.)
- Use VC-LTL NuGet package edition instead of standalone edition. (Thanks to mingkuang.)
- Create NSudo Sweeper (experiment).
- Use Windows Template Library (WTL) to build the UI.
- Improve the website. (Contributed by 青春永不落幕.)
- Reorganize the whole project.
- Add Mile, Mile.Project, MINT to the project for making NSudo more modularize.
- Improve the AppVeyor and GitHub Action CI support. (Thanks to mingkuang.)
- Add build all targets script.
- Update VC-LTL to 4.1.1-Beta7.
- Fix the crash bug under the Windows 10 Build 21277. (Thanks to jgtoy.)
- Update to .NET 5.0 for NSudo .NET Wrapper.
- Add user manual.

NSudo 8.0

- Reduce the binary size.
    - Use FILE instead of std::ifstream.
    - Use new compiler options.
    - Merge NSudo to NSudoG.

    - Optimize the icon resource.
    - Use jsmn instead of JSON for Modern C++.
- Rename NSudo itself to NSudo Launcher. (NSudoLG.exe and NSudoLC.exe)
- Remove context menu support because you can use https://github.com/Thdub/NSudo_Installer for better experience.
- Add Itlian Language. (Contributed by garf02.)
- Add Spanish Language. (Contributed by Miguel Obando.)
- Use the Semantic Versioning format.
- Improve several implementations and documents.
- Add implementations for developers.
    - Add NSudo Shared Library with C/C++ and .Net interoperability support.
    - Add NSudo Devil Mode (NSudoDM).
    - Add Mouri Internal Library Essentials (Mile).
- Noticeable things about compiling this project.
    - Update to Visual Studio 2019.
    - Update to the newest Windows 10 SDK.
    - Improve AppVeyor CI and GitHub Actions CI support. (Thanks to Margen67.)

NSudo 6.2.1812.31

- Add French translations. (Contributed by Thomas Dubreuil.)
- Use JSON for Modern C++ instead of RapidJSON to conform to C++17.
- Improve the GUI experience. (Thanks to Lenny.)
- Fix context menu bug. (Thanks to Thomas Dubreuil and 龍魂.)
- Fix the command line parser bug. (Thanks to wzzw.)
- Add Traditional Chinese translations. (Contributed by Luo Yufan.)

NSudo 6.1.1811.18

- Merge NSudoC and NSudoG projects to NSudo project.
- Add VC-LTL 4.0 or later support for NSudo release configuration for ARM and ARM64 and drop the earlier version of VC-LTL support. (Huge thanks to mingkuang.)
- Add the following options in the command line usage.
    - CurrentDirectory (Suggested by testtest322.)
    - Help
    - H
    - Priority (Suggested by testtest322.)
    - ShowWindowMode (Suggested by testtest322.)
    - UseCurrentConsole
    - Version
    - Wait (Suggested by testtest322, wzzw and Domagoj Smolčić)
- Remove some undocumented command line usage.
- Improve several implementations.
    - Refactoring the command line parser.
    - Introduce the new frontend of process creation.
    - Using ATL to implement the main window.
    - Fix the context menu bug. (Thanks to Thomas Dubreuil.)
- Update copyright of license.

- Remove donation link in the documents.

NSudo 6.0.1804.5

- Fix a bug which can cause crash on Windows Vista and Server 2008. (Thanks to hydra79545.)
- Share source code with M2-Team UWP projects. (For more information, please read "https://github.com/Project-Nagisa/Nagisa/blob/master/Changelog.md")
- Remove useless implementations.
- Improve the implementation for NSudoStartService function.
- Use RapidJSON instead of JSON for Modern C++ to reduce the binary size.

NSudo 6.0.1802.2 v2

- Fix always opens a command prompt window after click the run button. (Thanks to AeonX.)

NSudo 6.0.1802.2

- Fix several bugs and improve several implementations.
- Add two standalone executable files used in different situations.
    - NSudoC.exe
        - The pure command line version and subsystem setting is "Console".
        - Work well in the console, but it has a black console window if you call it in the non-Console processes.
        - To ensure the best experience, NSudoC does not support context menu.
    - NSudoG.exe
        - The pure command line version and subsystem setting is "Windows"
        - It can run silently, without a black console window.
- NSudo will show the message via the M2MessageDialog instead of TaskDialog.
    - Reasons
        - NSudo can provide more detail infomation when error because of it have vertical scroll bar.
        - You can copy the content in the message dialog.
        - Support using by the Windows Narrator, so you can use CapsLock+H to read the content by the Windows Narrator.
        - The font size is larger than the TaskDialog.
    - Features of M2MessageDialog
        - Fully support Per-Monitor DPI Aware in Windows 10 Build 10240 or later.
        - Fully support Windows Narrator.
        - You can use the vertical scroll bar and copy the content.
        - The font size is larger than the TaskDialog.
        - You can press Enter to close the message dialog.
    - If you want to use the M2MessageDialog in your project, please download these files in https://github.com/M2Team/NSudo/tree/master/NSudoSDK
        - M2DPIScaling.cpp
        - M2DPIScaling.h
        - M2MessageDialog.cpp
        - M2MessageDialog.h
        - M2MessageDialogResource.h

        - M2MessageDialogResource.rc
- Remove Traditional Chinese and Japanese translation because the translation is out of date and I don't know how to use.
- Update JSON for Modern C++ to 3.0.1.
- Context Menu
    - Add multilingual descriptions.
    - Add "Enable All Privileges" options for all item in the context menu.
- Update the command line help and documents.

NSudo 6.0.1801.19

- Fix the NSudoDuplicateSessionToken function definition bug. (Thanks to mingkuang.)
- Fix bugs that cannot enable full privileges under the graphical interface. (Thanks to abbodi1406.)
- Fix bugs that cannot use static compile mode when using release configuration for x86 and x86-64 (AMD64) without VC-LTL.

NSudo 5.3.1801.11

- Fix a potential bug when NSudo obtaining the System Token. (Thanks to mingkuang.)
- Provide VC-LTL support for NSudo release configuration for x86 and x86-64 (AMD64). (Thanks to mingkuang.)
    - PS: Compiling NSudo with VC-LTL can reduce the NSudo's binary size.
- Maintaining documents.

NSudo 5.2 (5.2.1709.8 - 5.2.1710.26)

- 整理代码, 修复若干 Bugs
- 更新文档, 增加英文自述
- 添加对 ARM 和 ARM64 平台的支持 (感谢 fcharlie)
- 优化命令行解析
- 添加右键菜单支持
    - 使用 /Install 或 -Install 参数添加右键菜单 (命令行参数大小写不敏感)
    - 使用 /Uninstall 或 -Uninstall 参数移除右键菜单 (命令行参数大小写不敏感)

NSudo 5.1 (5.0.1708.9 - 5.1.1708.19)

- 修复批处理调用 NSudo 后批处理变量不生效的问题 (感谢 半员外)
- 令 NSudo 在带有命令行的状态下也能自动请求管理员权限 (感谢 鸢一雨音)
- 更换新图标, 顺便解决在 Windows Vista 之前版本系统上不显示 NSudo 图标的问题 (PS: NSudo 最低要求依旧是 Windows Vista)
- 改进命令行解析 (感谢 鸢一雨音)
- 更新源代码许可的版权 (对说辞进行了优化) 和更新感谢名单 (新增人士)

NSudo 5.0 (4.4.1705.28 - 5.0.1707.31)

- 使用新的获取会话 ID 方法解决在 Server 系统的远程桌面会话上使用 NSudo 运行应用可能无 法显示界面的问题 (感谢 sebus)
- 更新文档和许可协议以符合实际情况
- 移除 VC-LTL (由 fcharlie 建议), 理由如下:
    - 虽然二进制大小增加 80KB, 但源代码大小缩小 57.6MB

    - 源代码大小缩小后, NSudo 的云编译速度大幅提升
    - 可以少屏蔽大量编译警告
- 使用 NSudoSDK 项目代替 M2-SDK 项目
- 改进版本定义头文件
- 编译器启用 SDL 检查、调整编译输出目录和更新 CI 编译配置文件
- 调整并优化代码(感谢 fcharlie 的建议)
- .gitignore 文件更新(由 fcharlie 实现)
- 完全使用 MSDN 文档化 API 实现 NSudoAPI.h 以方便人们调用
- 与 Nagisa 项目共用 m2base.h
- 整理屏蔽的警告, 该版本 NSudo 屏蔽了以下警告实现 /W4 /WX 编译
    - C4505 未引用的本地函数已移除(等级 4)
- NSudo 快捷列表文件格式从 ini 迁移到 json 并更新列表内容
- 进程创建时添加环境块以改善兼容性
- 把 Windows XP 控件支持声明和 Per-Monitor DPI Aware V1 支持移入清单文件
- 在清单文件添加兼容性 GUID 定义和 Per-Monitor DPI Aware V2 支持
- 修复当未在浏览窗口选择文件的情况下命令行文本框出现""的问题

NSudo 4.4.1705.19

- 适配最新版 M2-SDK
- 适配最新版 VC-LTL
- 修改编译选项
- 使用 git 子模块机制 (由 myfreeer 实现)
- 配置 AppVeyor (由 myfreeer 提供灵感)
- 开始使用 AppVeyor 自动编译
- 更新 M2-SDK 和 VC-LTL 子模块
- 命令行解析从 main 函数拆分
- 修复升级 VC-LTL 后出现的编译警告 (有空会 pull fix 到 VC-LTL)
- 版本号重新由自己而不是 CI 编译服务控制
- 整理解决方案布局

NSudo 4.3.1703.25

- 32 位版本取消对 SSE 和 SSE2 指令集的依赖 (为了保证完美的兼容性)
- 移除 NTIShell, NSudo.AppContainer, MiniFM 子项目
- NSudoSDK 完全被 M2-SDK 和 M2.NSudo.h 替代
- 关于界面布局调整
- 子系统设置调整为 Windows 子系统 (为了不再弹出黑框)
- 优化代码, 减少全局变量
- System 令牌副本创建函数移除会话 ID 参数 (因为现实情况只能使用当前会话 ID)
- 使用旧版应用调用方式 (即使用 cmd, 解决无法调用带参数应用的问题)
- 优化在 UI 自动化工具 (例如讲述人等读屏软件) 上的使用体验
- "运行" 按钮被设为默认按钮以提升使用体验
- 优化多语言资源以减小体积
- 修复 UI 标题栏没有图标的问题
- 为 UI 增加最小化按钮
- 修复数个库函数返回值 Bug
- 修复数个命令行解析 Bug

- 修复 UI 图标的 DPI 缩放问题
- 开始使用 Visual Studio 2017 编译
- 移除 NSudo-GUI 项目
- 代码不再包含 M2-SDK 和 VC-LTL 的内容, 需要单独从 github 克隆

NSudo 4.2

- 引入新 NSudoSDK API 并且对已有 NSudoSDK API 进行改善
- 优化代码, 以减少 Windows API 调用次数
- 修复不带任何参数情况下可能的奔溃问题
- 修复控制台部分不能在非管理员权限显示命令行帮助的问题
- 基于 ShellExecute 自建调用宿主, 以去除对 cmd.exe 的依赖
- 引入 NTIShell (相当于 NSudo 1.0) 重制版, 作为 NSudoSDK 的一个示例
- 更改 MiniFM 图标

NSudo 4.1

- 修复命令行使用-U:D 导致程序奔溃的问题
- 更正命令行的 NSudoC 残余描述 (感谢 NotePad)
- 支持文件拖拽 (感谢 NotePad)

NSudo 4.0

- 重写代码, 提供 NSudoSDK, 使代码容易使用在其他项目上
- 命令行下新增"/"前缀参数支持,例如: NSudo /U:T /P:E cmd (感谢 th1r5bvn23)
- 支持默认参数, 即以 TrustedInstaller 令牌且开启全部特权运行 (感谢 老九)
- 在默认快捷命令列表加入 host 编辑
- 增加 NSudo 和 MiniFM 的 Per-Monitor DPI Aware 支持
- 采用 VC-LTL 大幅度减小程序体积 (感谢 mingkuang)
- 更改图标 (感谢 20011010wo)
- 精简并优化主界面 (感谢 kCaRhC 卡壳, さくら)
- 使用 TaskDialog 替代 MessageBox
- 对关于界面进行调整, 并在关于界面加入命令行帮助
- 修复弹出文件不存在的问题
- 修复命令行解析的一个潜在 Bug
- 缓解 NSudo 图形界面的空格问题 (浏览功能自动给命令行加引号)
- 消除在编译时的警告(/Wall 和/WX 两个参数同时使用)

NSudo 2016.1

- 修复 TrustedInstaller 下运行程序界面不显示问题 (感谢 abbodi1406)
- 修复命令行解析的漏洞和 UI 错误 (感谢 imadlatch)
- 整理代码, 提升可读性
- 当前目录设为 NSudo 所在目录 (未来会更加灵活)
- ShortCut 实现无限项目
- 新增简易文件管理器小工具 (感谢 20011010wo)

NSudo 2016

- 支持多语言 (程序内含简中, 繁中, 英文, 日文)

- 命令行处理重写
- 实现代码全部重构; 效率更高

NSudo 3.2 Fix1

- 优化程序逻辑; 减少无用代码
- 命令行版和图形版二合一

NSudo 3.2

- 修复无法使用带有空格的路径的问题
- NSudo 和 NSudoC 单文件化
- 增加 NSudo.bat 方便新手准确调用与电脑架构相符的 NSudo 版本
- NSudoSDK 增加静态库 (用 NSudo SDK 开发的工具可以实现单文件)
- 编译平台采用 Visual Studio 2015 + Windows 10 SDK

NSudo 3.1 Debug

- 修复 UI 的 ComboBox 不能输入太长文字的问题
- 修复某些情况下不能使用的问题 (由于开发机 Windows10 的 Bug 而导致误认为那种方式可 行)
- 增加真正的令牌降权 (除了 cmd 会误显示管理员外; 其他的会将其看作普通用户)
- 增加命令行版本
- 增加常用列表自定义功能

NSudo 3.0 R2

- 修复不能打开其他被系统关联文件的 Bug
- SDK 的头文件改进: 增加#pragma comment(lib,"NSudoAPI.lib")

NSudo 3.0

- 支持外部应用调用 (很抱歉让一些人等太久)
- 增加了常用调用列表 (暂时不支持自定义; 未来 3.1 会加入)
- 加入了降权功能 (当然, 是完美降权到 UAC 未提权前。当然原理不是用获取 explorer 令牌 和创建计划任务)
- 支持对权限令牌的自定义
- 界面的完全重构 (相对于 2.x 来说)
- 代码优化 (相对于 NSudo 3.0 M1 来说)
- 加入 NSudo SDK
- 原生 64 位版本
- 实现了调用外部程序无视 WOW64 重定向的方法 (NSudoCreateProcess)
- WinPE 支持 (虽然没起多大作用)

NSudo 2.1

- 实现自动开启所有权限 Token
- 对 cmd 的调用使用绝对路径, 估计可以避免一些不必要的 Bug
- 优化程序代码

NSudo 2.0

- 代码全部使用 C++ Win32 SDK 重写 (程序从 692KB 缩小到 92KB)
- 提供获取权限的选项
- 提供命令行参数模式
- 更换了图标

NSudo 1.2

- 未公开发布 (估计还是在修复 SessionID 问题)

NSudo 1.1

- 修复 SessionID 问题
- 32 位和 64 位版本合体 (根据架构确定运行那个架构的命令提示符, 采用 SysNative 目录 (64 位 Vista 开始有的重定向) 调用 64 位 cmd)

NTIShell 1.0

- 根据 raymai97 的超级命令提示符制作的第一个版本