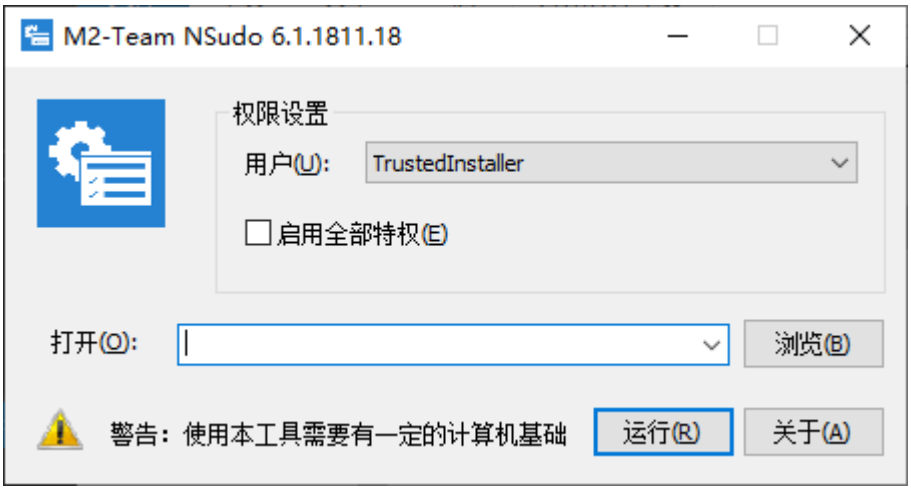
NSudo - 系统管理工具包



目录

关于 NSudo

特性列表

- 以 MIT 许可证发行
- 提供 x86, x86-64, ARM, ARM64 二进制
- 支持 Windows Vista 及之后版本
- 使用初雨团队的 VC-LTL 和 libkcrt 以获取更小的二进制体积
- 使用 C++17，但在大部分情况下只使用核心语言特性
- NSudo Launcher
    - 以 TrustedInstaller 访问令牌运行程序
    - 以 System 访问令牌运行程序
    - 以当前用户的访问令牌运行程序
        - 注：如果用户账户控制即 UAC 没有被禁用，则该模式的权限与标准用户等价
    - 以当前进程的访问令牌运行程序
        - 注：该模式的权限与提升后的用户等价
    - 以当前进程 LUA 模式的访问令牌运行程序
        - 注：该模式的权限与标准用户等价且该实现和 Internet Explorer 浏览器中的 iertutil.dll 中的对应实现一致
    - 支持以指定的特权设置运行程序
        - 注：启用全部特权, 禁用所有特权
    - 支持以指定的完整性级别运行程序
        - 注：系统、高、中、低
    - 支持以指定的进程优先级运行程序
        - 注：低、低于正常、正常、高于正常、高、实时
    - 支持以指定的窗口模式运行程序
        - 注：显示窗口、隐藏窗口、最大化、最小化
    - 支持进程创建后并等待其运行结束
    - 支持以指定的当前目录运行程序
    - 支持在当前控制台窗口下运行程序
    - 支持快捷方式列表
        - 注：你可以通过编辑 NSudo.json 的方式定制
    - 支持多种命令行风格
    - 多语言支持
        - 注：简体中文、繁体中文、英语、法语、意大利语、西班牙语
    - 完整的高 DPI 支持
        - 注：和 Windows 外壳 (conhost.exe) 的实现一样完美，在 Windows 10 Build 10240 及之后版本有完整的 Per-Monitor DPI-Aware 支持和在 Windows Vista 到 Windows 8.1 之间的版本有完整的 System DPI-Aware 支持
    - 完整的无障碍访问支持
        - 注：你可以在 Windows 讲述人下顺畅的使用 NSudo Launcher
    - 高性能
        - 注：因为其实现不需要创建 Windows 服务和 Windows 服务进程
    - 为开发者提供 C APIs 和 .Net Core 互操作支持
- NSudo 恶魔模式

    - 对于希望无视管理员权限的进程下的文件和注册表访问判断的开发者而言是最优雅的 解决方案
    - 使用 Microsoft Detours 库对 API 进行挂钩以保证最大兼容
    - 其二进制仅依赖了 ntdll.dll 的以函数名导出的 API

系统要求

- 支持的系统版本：Windows NT 6.0 及之后版本
- 支持的处理器架构：x86, x86-64(AMD64), ARM, ARM64

项目原型

注意：NSudo 基于 raymai97 的超级命令提示符，请参阅 这里 以获取更多关于 超级命令提示符的信息。

使用了 NSudo 的第三方项目

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

第三方介绍

- 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

成为 NSudo 的赞助者

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

爱发电: https://afdian.net/@MouriNaruto

- 如果我达到了每个月 1000 美元的目标，我将会每年更新两个大版本。
- 如果我达到了每个月 2000 美元的目标，我会为 NSudo 二进制添加 EV 代码签名证书。

感谢支持。

毛利

获得支持

联系方式

- 邮箱：Mouri_Naruto@Outlook.com

社区

- GitHub Issues

- My Digital Life
- QQ 群

使用方法

下载 NSudo

二进制

- 当前版本
- 所有版本
- AppVeyor CI

源代码

- GitHub
- 码云

NSudo Installer (非官方)

- 源代码
- 当前版本

Chocolatey (非官方)

> choco install nsudo

scoop (非官方)

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

第三方下载站

- MajorGeeks
- softpedia.com

NSudo Launcher

快速上手

请打开【CPU 架构】目录，然后双击 NSudo.exe. 根据提示操作即可。例如，如果你想在 你的 Intel 或 AMD 设备上使用 64 位 NSudo，你首先需要打开的是 x64 目录，然后双击 NSudoG.exe。

命令行选项

```
格式: NSudo [ 选项与参数 ] 命令行或常用任务名

选项:

-U:[ 选项 ] 以指定用户选项创建进程。
可用选项:
    T TrustedInstaller
    S System
    C 当前用户
    P 当前进程
    D 当前进程 (降权)
PS: 这是一个必须被包含的参数。

-P:[ 选项 ] 以指定特权选项创建进程。
可用选项:
    E 启用全部特权
    D 禁用所有特权
PS: 如果想以默认特权选项创建进程的话，请不要包含“-P”参数。

-M:[ 选项 ] 以指定完整性选项创建进程。
可用选项:
    S 系统
    H 高
    M 中
    L 低
PS: 如果想以默认完整性选项创建进程的话，请不要包含“-M”参数。

-Priority:[ 选项 ] 以指定进程优先级选项创建进程。
可用选项:
    Idle 低
    BelowNormal 低于正常
    Normal 正常
    AboveNormal 高于正常
    High 高
    RealTime 实时
PS: 如果想以默认进程优先级选项创建进程的话，请不要包含“-Priority”参数。

-ShowWindowMode:[ 选项 ] 以指定窗口模式选项创建进程。
可用选项:
    Show 显示窗口
    Hide 隐藏窗口
    Maximize 最大化
```

```
    Minimize 最小化
PS：如果想以默认窗口模式选项创建进程的话，请不要包含“-ShowWindowMode”参数。

-Wait 令 NSudo 等待创建的进程结束后再退出。
PS：如果不想等待，请不要包含“-Wait”参数。

-CurrentDirectory:[ 目录路径 ] 设置进程的的当前目录。
PS：如果你想用 NSudo 的当前目录，请不要包含“-CurrentDirectory”参数。

-UseCurrentConsole 使用当前控制台窗口创建进程。
PS：如果你想在新控制台窗口创建进程，请不要包含“-UseCurrentConsole”参数。

-Version 显示 NSudo 版本信息。

-? 显示该内容。
-H 显示该内容。
-Help 显示该内容。

上下文菜单管理请使用 https://github.com/Thdub/NSudo_Installer。

PS:
    1. 所有的NSudo命令行参数不区分大小写。
    1. 可以在命令行参数中使用 "/" 或 "--" 代替 "-" 和使用 "=" 代替 ":"。例如
       "/U:T" 和 "-U=T" 是等价的。
    1. 为了保证最佳体验，NSudoC不支持上下文菜单。

例子:
    以TrustedInstaller权限，启用所有特权，完整性默认运行命令提示符
        NSudo -U:T -P:E cmd
```

例子：以 TrustedInstaller 权限，启用所有特权，完整性默认运行命令提示符

> NSudo -U:T -P:E cmd

从 NSudo 5.0.1708.16 开始命令行支持嵌套引号，例如：

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

常用列表

关于常用列表的自定义,可以使用记事本等工具编辑 NSudo.json。你可以照着示例的做法添 加你的自定义项目：

```
{
  "ShortCutList_V2": {
    "命令提示符": "cmd",
    "PowerShell": "powershell",
    "PowerShell ISE": "powershell_ise",
    "Hosts编辑": "notepad %windir%\\System32\\Drivers\\etc\\hosts"
  }
}
```

NSudo 恶魔模式

NSudo 恶魔模式 (NSudo Devil Mode) 是为想无视文件和注册表操作权限检查的开发者 量身定做的一个用起来还算优雅的解决方案。

其原理是使用开源的 Microsoft Detours 库对 Windows NT 内核的文件和注册表相关的 系统调用进行 Inline Hook 以传入选项让开发者基本不用修改自己的实现也能充分的利用 管理员权限所提供的特权，这也使得开发者只需要把 NSudo 恶魔模式的动态链接库加载入 自己的以管理员身份运行的应用进程的地址空间内即可启用 NSudo 恶魔模式。

由于 NSudo 恶魔模式可以在大部分情况下替代类似 NSudo 的工具，于是 NSudo 未来的 功能会变得更加专业向。毕竟作为 NSudo 的作者的我可不希望 NSudo 就这么轻易地被 替代掉。当然，由于 NSudo 恶魔模式属于 Dism++ 春哥附体的后续版本 (毕竟我也是 Dism++ 的其中一位开发者，这么说还是有依据的)，于是未来 Dism++ 的春哥附体的实现 会被替换成 NSudo 恶魔模式以帮助我更好的重构 Dism++ 的实现。当然，NSudo 也会支持 以恶魔模式运行应用。

NSudo 恶魔模式的起源、命名和意义

正如上文所说，NSudo 恶魔模式属于 Dism++ 春哥附体的后续版本，而且你也能在本文中 了解 NSudo 恶魔模式和 Dism++ 春哥附体的区别。

命名为 NSudo 恶魔模式的灵感来源是《入间同学入魔了》的被蛋爷改造过的拥有四个档位 的入间手中的"恶食戒指"。

最开始听到 MSMG Toolkit 的作者希望我能提供 NSudo 的 SDK 方便他进行二次开发时， 我试着做了基于 COM 接口的 NSudo Shared Library 即 NSudoAPI，但是由于 NSudoAPI 暴露的细节太多，如果不是对 Windows 安全特性足够了解的开发者是很难驾驭的。于是 我觉得得换个方向，于是就做了 NSudo 恶魔模式。

NSudo 恶魔模式挂钩的 Windows NT 内核系统调用列表

名称	起源
NtCreateKey	Dism++ 春哥附体
NtCreateKeyTransacted	NSudo 恶魔模式
NtOpenKey	Dism++ 春哥附体，并增强了效果
NtOpenKeyTransacted	NSudo 恶魔模式
NtOpenKeyEx	Dism++ 春哥附体
NtOpenKeyTransactedEx	NSudo 恶魔模式
NtCreateFile	Dism++ 春哥附体
NtOpenFile	Dism++ 春哥附体

如何使用 NSudo 恶魔模式

调用 LoadLibrary 加载 NSudo 恶魔模式的动态链接库以启用 NSudo 恶魔模式，调用 FreeLibrary 释放 NSudo 恶魔模式的动态链接库的 HMODULE 句柄即可禁用 NSudo 恶魔模式。

当然，你的应用需要在管理员权限下运行，相对于原本要求 SYSTEM 和 TrustedInstaller 权限的情况下其实好了不少。

注：如果你有本事把 NSudo 恶魔模式远程注入到以管理员或者更高权限的进程内（譬如 7-Zip），也能为该进程赋能（无视文件和注册表的权限）。

下面提供一个使用 C# 编写的测试用例。（遍历 C:\System Volume Information 目录 的内容，当启用 NSudo 恶魔模式的情况下可以正常显示，禁用后会抛出文件夹拒绝访问 的异常。）

```
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();
        }
```

```
        }
    }
```

License

The offical NSudo repository (not including third-party libraries) is **only** distributed under the MIT License today because we want to give the **maximum respect** to every NSudo users and we also hope every people treat others kindly.

Mouri_Naruto's libraries which used in the official NSudo repository

Name	Project website
Mouri Internal Library Essentials	https://github.com/MouriNaruto/Mile

M2-Team's libraries which used in the official NSudo repository

Name	Project website
M2-Team Common Library	https://github.com/M2Team/M2TeamCommonLibrary

Chuyu Team's libraries which used in the official NSudo repository

Name	Project website
libkcrt	https://github.com/Chuyu-Team/libkcrt
Mouri's Internal NT API Collections	https://github.com/Chuyu-Team/MINT

Name	Project website
VC-LTL	https://github.com/Chuyu-Team/VC-LTL

3rd-party libraries which used in the official NSudo repository

Name	Project website
JSMN	https://github.com/zserge/jsmn
Microsoft Research Detours Package	https://github.com/microsoft/Detours
Windows Template Library	https://sourceforge.net/projects/wtl

相关人士

- 本列表按字母顺序排列。

创立者

- Mouri_Naruto (https://github.com/MouriNaruto)

原型作者

- raymai97 (https://github.com/Raymai97)

贡献者

- 20011010wo (https://github.com/yangrq)
- Bill (https://github.com/bianyifan)
- Eugene Wang J.y (https://github.com/ewjy)
- Force.Charlie-I (https://github.com/fcharlie)
- garf02 (https://github.com/garf02)
- laosb (https://github.com/laosb)
- 罗宇凡 Luo Yufan (https://github.com/njlyf2011)
- Margen67 (https://github.com/Margen67)
- May_magic (https://github.com/873578156)
- Microsoft_Mars
- Miguel Obando (https://github.com/obando777)
- mingkuang (https://github.com/mingkuang-Chuyu)
- myfreeer (https://github.com/myfreeer)
- 青春永不落幕 (https://github.com/qcyblm)
- Thomas Dubreuil (https://github.com/Thdub)

赞助者

- 安磊磊
- boyangpangzi
- cjy__05
- dfdc5
- mhxkx
- NotePad
- tangmigold
- wondersnefu
- xy137425740
- 龍魂
- 月光光

反馈者

- 4071E95D-A09B-4AA3-8008

- abbodi1406
- AeonX
- Domagoj Smolčić
- hydra79545
- imadlatch
- jgtoy
- kCaRhC 卡壳
- Lenny
- NotePad
- sebus
- testtest322
- th1r5bvn23
- 老九
- 龍魂
- 半员外
- 鸢一雨音 (https://github.com/TobiichiAmane)
- さくら

特别感谢

- 高坂穗乃果 (因为知晓了她的事迹，使我没有放弃对 NSudo 的开发)

发行日志

NSudo 8.0 Update 1 (8.0.1)

- 更新意大利语翻译（由 garf02 贡献）
- 使用 NuGet 包版本 VC-LTL 替代独立版 VC-LTL（感谢 mingkuang）
- 创建 NSudo Sweeper 项目（实验性）
- 使用 Windows Template Library (WTL) 构建界面
- 改善项目网站实现（由 青春永不落幕 贡献）
- 整理整个项目
- 添加 Mile、Mile.Project、MINT 子项目使 NSudo 实现更加模块化
- 改善 AppVeyor 和 GitHub Action 连续集成的支持（感谢 mingkuang）
- 添加一键编译脚本
- 升级 VC-LTL 到 4.1.1-Beta7
- 修复在 Windows 10 Build 21277 下崩溃的问题（感谢 jgtoy）
- NSudo .NET 互操作库采用 .NET 5.0 编译

NSudo 8.0

- 精简二进制体积
  - 使用 FILE 而不是 std::ifstream
  - 使用新的编译器选项
  - NSudo 合并入 NSudoG
  - 优化图标资源
  - 使用 jsmn 而不是 JSON for Modern C++.
- NSudo 更名为 NSudo Launcher（NSudoLG.exe 和 NSudoLC.exe）
- 移除上下文支持，因为 https://github.com/Thdub/NSudo_Installer 体验更好
- 添加意大利语翻译（由 garf02 贡献）
- 添加西班牙语翻译（由 Miguel Obando 贡献）
- 遵循语义化版本规范
- 改善数个代码和文档方面的实现
- 添加为开发者设计的相关设施
  - 添加 NSudo Shared Library 且带有 C/C++ 和 .Net 互操作支持
  - 添加 NSudo Devil Mode (NSudoDM).
  - 添加 Mouri Internal Library Essentials (Mile).
- 编译本项目需要注意的变更事项
  - Visual Studio 已经升级到 2019
  - Windows 10 SDK 已经升级到最新
  - 改善对 AppVeyor CI 和 GitHub Actions CI 的支持（感谢 Margen67）

NSudo 6.2.1812.31

- 添加法语翻译（由 Thomas Dubreuil 贡献）
- 使用 JSON for Modern C++ 替代 RapidJSON 以符合 C++17 规范
- 改善图形界面用户体验（感谢 Lenny）
- 修复上下文菜单 Bug（感谢 Thomas Dubreuil 和 龍魂）

- 修复命令行解析器 Bug（感谢 wzzw）
- 添加繁体中文翻译（由 罗宇凡 贡献）

NSudo 6.1.1811.18

- 把 NSudoC 与 NSudoG 合并入 NSudo
- 为 ARM 和 ARM64 的 Release 二进制编译配置添加 VC-LTL 支持，并移除 VC-LTL 4.0 之前版本的支持（大力感谢 mingkuang）
- 添加以下新的命令行选项
    - CurrentDirectory（由 testtest322 建议）
    - Help
    - H
    - Priority（由 testtest322 建议）
    - ShowWindowMode（由 testtest322 建议）
    - UseCurrentConsole
    - Version
    - Wait（由 testtest322、wzzw 和 Domagoj Smolčić 建议）
- 移除一些未文档化的命令行使用方式
- 改进数个实现
    - 重构命令行解析器
    - 引入新式创建进程前端
    - 使用 ATL 实现主窗口
    - 修复上下文菜单 Bug（感谢 Thomas Dubreuil）
- 更新许可的版权所有者
- 在文档移除捐赠链接

NSudo 6.0.1804.5

- 修复在 Windows Vista 和 Server 2008 下崩溃的问题（感谢 hydra79545）
- 与 M2-Team UWP 项目共享实现（详情请阅读 “https://github.com/Project-Nagisa/Nagisa/blob/master/Changelog.md”）
- 移除无用实现
- 改进 NSudoStartService 函数的实现
- 使用 RapidJSON 替代 JSON for Modern C++ 以减小二进制大小

NSudo 6.0.1802.2 v2

- 修复点击运行按钮只弹出命令提示符的问题。（感谢 AeonX）

NSudo 6.0.1802.2

- 修复多个 Bug 和改善多个实现
- 增加两个独立的可执行文件用于不同情况
    - NSudoC.exe
        - 纯命令行版本，子系统设置为“控制台”
        - 在控制台下运行良好，但是在非控制台进程调用会出现黑色控制台窗口
        - 为了保证最佳体验，NSudoC 不支持上下文菜单
    - NSudoG.exe
        - 纯命令行版本，子系统设置为“Windows”

        - 可以静默运行，没有黑色控制台窗口
- NSudo 将通过 M2MessageDialog 而不是 TaskDialog 显示信息
    - 理由
        - 因为可以使用纵向滚动条，NSudo 可以在出错时提供更加详细的内容
        - 你可以复制对话框里的内容
        - 支持 Windows 讲述人，于是可以使用 CapsLock+H 让讲述人读取对话框内容
        - 比 TaskDialog 的字体更大
    - M2MessageDialog 特性
        - 在 Windows 10 Build 10240 或之后版本完全支持 Per-Monitor DPI Aware
        - 完全支持 Windows 讲述人
        - 你可以使用纵向滚动条并且可以复制里面的内容
        - 比 TaskDialog 的字体更大
        - 你可以按 Enter 键关闭对话框
    - 如果你想在你的项目使用 M2MessageDialog，请从此处下载以下文件：
      https://github.com/M2Team/NSudo/tree/master/NSudoSDK
        - M2DPIScaling.cpp
        - M2DPIScaling.h
        - M2MessageDialog.cpp
        - M2MessageDialog.h
        - M2MessageDialogResource.h
        - M2MessageDialogResource.rc
- 移除繁体中文和日语的翻译，因为翻译内容已经过时而且我不懂怎么用
- 升级 JSON for Modern C++ 到 3.0.1
- 右键菜单
    - 增加多语言描述
    - 为所有项目添加“开启全部特权”选项
- 更新命令行帮助和文档

NSudo 6.0.1801.19

- 修复 NSudoDuplicateSessionToken 函数定义 Bug（感谢 mingkuang）
- 修复在图形界面下无法启用全部特权的 Bug（感谢 abbodi1406）
- 修复没有 VC-LTL 时 x86 和 x86-64(AMD64)的 Release 编译配置未采用静态编译的 Bug

NSudo 5.3.1801.11

- 修复获取 System 令牌函数的一个潜在 Bug（感谢 mingkuang）
- 对 x86 和 x86-64(AMD64)的 Release 编译配置提供 VC-LTL 库支持（感谢 mingkuang）
    - PS：把 NSudo 和 VC-LTL 一起使用可以减小 NSudo 二进制的体积
- 整理文档

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