

# r77 Rootkit

## 技术文档



r77 版本  
发布日期

1.2.2  
31.08.2021

作者  
网站  
GitHub

字节码77  
[bytecode77.com/r77-rootkit](https://bytecode77.com/r77-rootkit)  
[github.com/bytecode77/r77-rootkit](https://github.com/bytecode77/r77-rootkit)



# 目录

1	介绍	
1.1	支持的平台	
1.2	兼容性	
1.3	已测试的应用程序	
1.4	依赖项和要求	5
1.4.1	提升的特权	
2	Rootkit	
2.1	Rootkit DLL	
2.2	安装人员	
2.3	卸载程序	
2.4	r77 服务	
2.4.1	无文件启动	
2.5	隐藏实体	
2.5.1	文件系统	
2.5.2	流程	
2.5.3	登记处	
2.5.4	TCP 和 UDP 连接	11
2.6	隐藏前缀	
2.7	配置系统	12
2.7.1	进程 ID	
2.7.2	进程名称	13
2.7.3	路径	
2.7.4	服务名称	13
2.7.5	本地 TCP 端口	
2.7.6	远程 TCP 端口	
2.7.7	UDP 端口	
2.7.8	启动路径	
2.8	自定义启动文件	14
2.9	枚举与访问	14
3	测试环境	
3.1	测试控制台	
3.2	示例.exe	
4	实施细则	18
4.1	r77 标头	
4.2	编译时间常数	19
4.3	子进程挂钩	20
4.4	Hooked API	
4.4.1	NtQuerySystemInformation	20
4.4.2	NtResumeThread	
4.4.3	NtQueryDirectoryFile	21



4.4.4	NtQueryDirectoryFileEx .....	21
4.4.5	NtEnumerateKey .....	21
4.4.6	EnumServiceGroupW .....	21
4.4.7	EnumServicesStatusExW .....	21
4.4.8	NtEnumerateValueKey .....	21
4.4.9	NtDeviceIoControlFile .....	21
4.5	AV 规避技术 .....	22
4.5.1	AMSI 旁路 .....	22
4.5.2	DLL 解钩 .....	22
5	集成最佳实践 .....	23
5.1	包括 Install.exe .....	23
5.2	直接安装安装 .....	23
6	已知的问题 .....	26
7	待办事项清单 .....	26
8	错误报告 .....	26
9	更改日志 .....	27



# 1 简介

r77 Rootkit 是一个无文件环 3 Rootkit。它的主要目的是隐藏某些实体，例如文件、目录、进程等。

此外，rootkit 附带一个开箱即用的安装程序，用于处理进程注入和持久性。安装是完全无文件的，这意味着不需要将文件写入磁盘。r77 仅依赖于内存中的操作，并在重新启动后保留在系统上。

对于 r77 的部署，只需要一个只需要执行一次的可执行文件。

本文档面向 r77 的集成商和旨在修改 r77 代码的开发人员。

## 1.1 支持的平台

支持 Windows 10 和 Windows 7，包括 x64 和 x86 版本。在整个产品中，操作系统位数都被考虑在内。当文档提到 x64 和 x86 区别时，这仅适用于 64 位版本。在 32 位 Windows 上，仅安装 32 位组件。

支持的操作系统基于市场份额而不是官方支持微软。

操作系统	x64	x86	市场份额 *
视窗 10	支持的	支持的	58%
Windows 7的	支持的	支持的	25%
视窗 8.1	不支持	不支持	3%
视窗 8	不支持	不支持	< 1%

\* 市场份额统计数据来自 2021 年 2 月的 netmarketshare。

r77 在发布之前已在所有支持的操作系统上进行了测试。

## 1.2 兼容性

一般而言，Rootkit 旨在适用于任何程序，而不仅仅是特定的应用程序，例如 Explorer.exe 和 TaskMgr.exe。r77 钩子函数ntdll.dll, 这是ring 3中可用的最低层。因此，任何程序都兼容r77，包括将来开发的程序。

## 1.3 测试应用

有一组应用程序用于测试每个模块。但是，r77 应该同样适用于任何其他应用程序。

测试中使用的应用：



- Windows 任务管理器 \*
- 程资源管理器
- 进程黑客
- Windows资源管理器
- Windows 注册表编辑器
- Services.msc
- 视图
- 当前端口
- 命令行工具
  - 目录
  - netstat.exe \*

\* 有关所列应用程序的已知问题，请参阅第 6 节。

要报告有关行为不正确的应用程序的错误，无论它们是否在测试应用程序列表中，请转到第 8 部分。

## 1.4 依赖和要求

除了操作系统本身和初始安装后已经存在的工具之外，r77 没有任何依赖项。二进制文件是用 C++ 编写的，并用 /MT 编译。

但是，无文件启动机制需要 PowerShell 和 .NET Framework。这两个依赖项都存在于 Windows 7 和 Windows 10 的全新安装中。

.NET Framework 通常存在这个问题，其中 .NET 2.0-3.5 和 .NET 4.0-4.8 是两个不同的 CLR。这意味着面向 .NET 3.5 的 .NET 可执行文件不会运行，仅当 .NET 4.x 已安装——当仅安装 .NET 3.5 时，面向 .NET 4.x 的 .NET 可执行文件不会运行。然而，这是**这不是问题** 对于 r77 stager。

在 Windows 7 上，默认安装 .NET 3.5，在 Windows 10 上，不安装 .NET 3.5，而是安装 4.x。如第 2.4.1 节所述，从 PowerShell 在内存中执行 C# 二进制文件时，目标版本无关紧要。无文件 stager 的目标框架设置为 .NET 3.5，以避免任何与 .NET 4.x 不兼容的代码。但是，如果 .NET 3.5 或者 .NET 4.x 已安装。

因此，始终满足此要求。r77 是**不是** 一个 “.NET rootkit”，因为只有启动代码需要 .NET，而 rootkit 本身完全是用 C++ 编写的。

### 1.4.1 提升权限

具有持久性的完整安装需要提升的权限。使用漏洞利用或 UAC 绕过技术提升权限不是该项目的一部分。

当使用以中等 IL 运行的测试控制台时，可以将 r77 DLL 注入到具有中等 IL 的进程中，但不能注入到提升的进程中。这足以进行一些测试，但是当没有注入提升的进程时，完全安装是没有意义的。



## 2 根工具包

### 2.1 Rootkit DLL

r77 Rootkit 是一个 DLL 文件（r77-x86.dll 和 r77-x64.dll）编译好的，分别用于 32 位和 64 位进程。一旦注入到一个进程中，这个进程就不会显示隐藏的实体。

r77 实现了反射 DLL 注入。任何时候都不需要将 DLL 写入磁盘。相反，文件被写入远程进程内存，并且反射DllMain 调用导出以最终加载 DLL 并调用 DLL 主。因此，该 DLL 未列在 PEB 中。

将 DLL 注入已注入的进程没有任何影响。主目录将检测到这一点并返回错误的卸载自己。

### 2.2 安装程序

安装程序将 r77 注入每个正在运行的进程并将 rootkit 持久化在系统。从现在开始，新进程在运行任何自己的指令之前被注入。这是通过挂钩进程创建来实现的。r77 设置为在重新启动后启动并在第一个用户登录之前注入所有进程。

安装程序两者都有 r77-x86.dll 和 r77-x64.dll 包含在其 PE 资源中。没有必要将 DLL 与它一起部署。这是一个单文件部署。安装程序也可以使用进程挖空来执行，以避免在部署期间将安装程序写入磁盘。

执行时安装程序第二次安装 r77 后，r77 服务进程被终止并重新创建。这是受支持的行为，也是将 r77 升级到当前版本的正确方法。已经注入的进程将不是分离并重新注入当前版本的 rootkit DLL。为此，请使用

卸载.exe。

有关如何将安装程序集成到您自己的项目中的详细信息，请查看第 5 节。

### 2.3 卸载程序

要从系统中完全删除 r77，请运行以下步骤：卸载.exe。它将卸载 r77

1. 删除 \$77stager 注册表中的值。删除定时任务。
- 2.
3. 终止 r77 服务。
4. 从所有注入的进程中分离 r77 DLL。
5. 如果操作系统是 64 位操作系统，则需要再次执行上述所有步骤，但要在 64 位可执行文件中执行。为此，将具有随机文件名的可执行文件放到临时目录中，然后执行并删除。这个可执行文件被嵌入到 PE 资源中

卸载.exe。

6. 删除 \$77配置 注册表中的键。



执行 卸载程序 第二次没有效果。但是，如果上述任何一个步骤失败，它将清理剩余的剩菜。

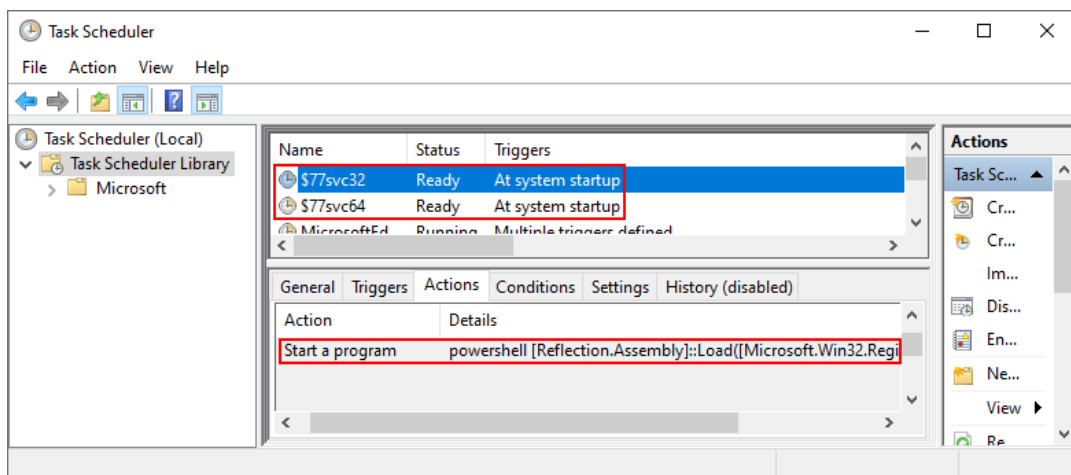
## 2.4 r77 服务

什么时候安装程序执行后，r77 服务就建立并启动了。r77 服务是无文件，这意味着安装程序不会将任何文件写入磁盘。

需要两个单独的 r77 服务进程来注入 32 位和 64 位进程。r77 服务的主要目的是在 r77 服务启动时注入所有正在运行的进程，以及注入稍后创建的进程。

### 2.4.1 无文件启动

**阶段1：** 安装程序为 32 位和 64 位 r77 服务创建两个计划任务。计划任务确实需要一个名为 \$ 的文件 77svc32.job 和 77svc64.job 存储，这是无文件概念的唯一例外。但是，一旦 rootkit 运行，计划任务也会通过前缀隐藏。

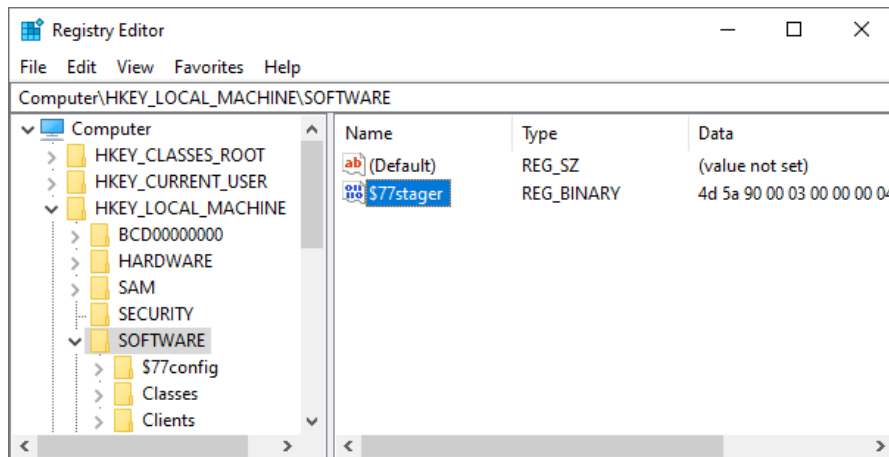


计划任务确实 **不是** 从磁盘启动 r77 服务可执行文件。相反，它开始 PowerShell.exe 在系统启动时使用以下命令行：

```
[Reflection.Assembly]::Load([Microsoft.Win32.Registry]::LocalMachine.OpenSubkey('SOFTWARE').GetValue('$77stager')).EntryPoint.Invoke($Null,$Null)
```

该命令是内联的，不需要 .ps1 脚本。在这里，利用 PowerShell 的 .NET Framework 功能从注册表加载 C# 可执行文件并在内存中执行它。为了这，Assembly.Load().EntryPoint.Invoke() 用来。

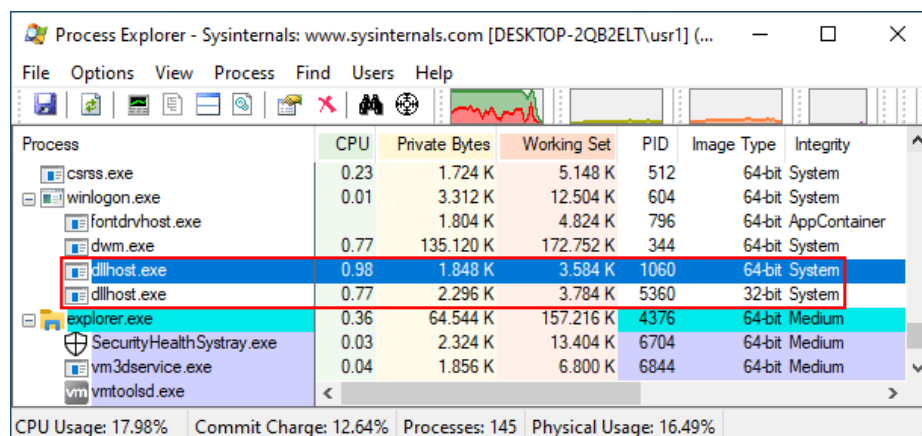
此外，内联脚本必须绕过 AMSI 以逃避 AV 检测（请参阅第 4.5.1 节）。



**第二阶段：** 执行的 C# 二进制文件是 stager。它将使用进程挖空创建 r77 服务进程。

r77 服务是分别以 32 位和 64 位编译的本机可执行文件。父进程被欺骗并设置为登录程序 为了额外的默默无闻。另外，这两个进程被ID隐藏，在任务管理器中不可见。

由于计划任务在SYSTEM账户下启动PowerShell，r77服务也在SYSTEM账户下运行。因此，它可以用系统IL注入进程，受保护的进程除外，例如服务.exe。



**重要的：** 写入文件系统的唯一项目是作业文件（\$77svc32.job 和 \$77svc64.job）和注册表值 \$77stager 带有 stager 可执行文件。没有 EXE 或 DLL 文件直接存储在文件系统中。甚至 安装程序 可以使用执行处理空洞以完全无文件的方式部署 r77。这一点非常重要，因为rootkit 安装程序或包含的DLL 文件可能会被AV 检测到并被删除。

**第 3 阶段：** 两个 r77 服务进程现在都在运行。执行以下操作：

1. 进程ID存储在配置系统中以隐藏进程。因为进程是使用进程挖空创建的，所以它们不能有 \$77 字首。
2. 注入所有正在运行的进程。
3. 创建一个命名管道来处理新创建的子进程的注入。
4. 除了子进程挂钩之外，一个子程序每 100 毫秒检查一次新创建的进程。这是因为有些进程无法注入，但仍会创建子进程





过程。这尤其适用于服务.exe, 这是一个受保护的过程。

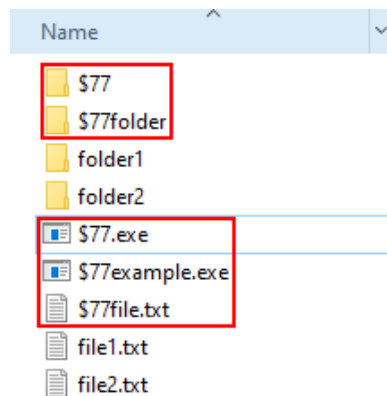
## 2.5 隐藏实体

通过前缀、特定条件或配置隐藏以下实体系统：

实体	隐 按前缀	隐 按条件	隐 按配置
文件	是的		隐藏路径
目录	是的		隐藏路径
命名管道	是的		隐藏路径
计划任务	是的		
过程	是的		隐藏的PID 隐藏的进程名称
CPU使用率		隐藏进程的CPU使用率	
注册表项	是的		
注册表值	是的		
服务	是的		隐藏的服务名称
TCP 连接		TCP 连接 隐藏进程	的 隐藏的本地 TCP 端口 隐藏 的远程 TCP 端口
UDP 连接		UDP 连接 隐藏进程	的 隐藏的 UDP 端口

### 2.5.1 文件系统

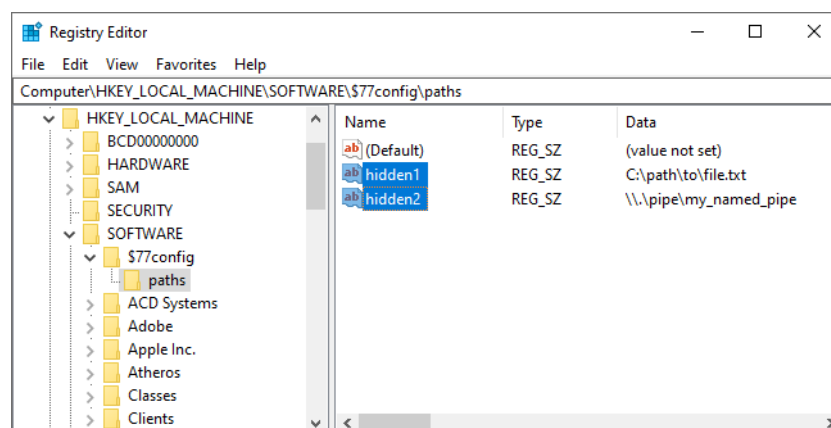
在文件系统上，所有带有前缀的目录和文件都是隐藏的。



这还包括：

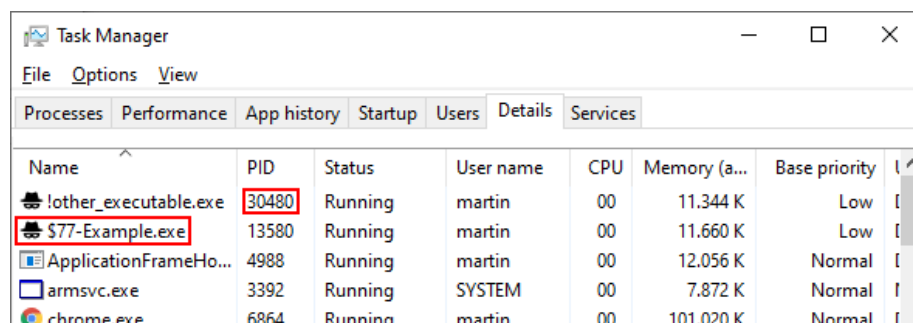
- 文件和目录连接
- 命名管道
- 计划任务（。工作 带有前缀的文件被隐藏；定时任务是隐藏的，当在枚举定时任务的服务中注入r77时，不mmc.exe。）

此外，配置系统可以隐藏单个文件、目录和命名管道。为此，需要将完整路径写入配置系统：



## 2.5.2 流程

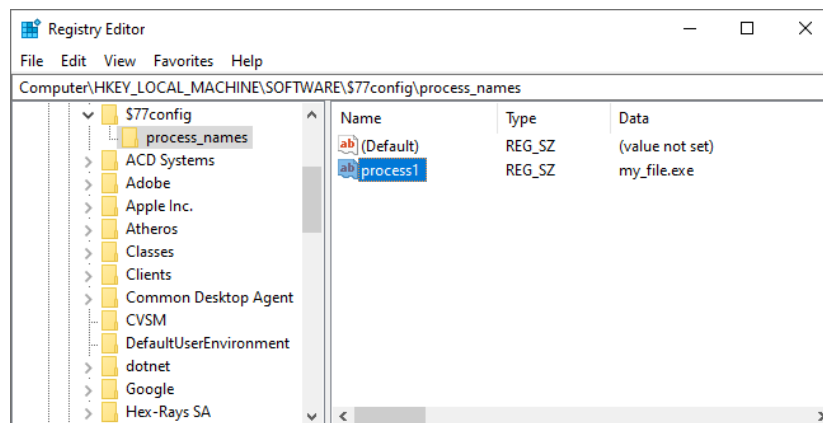
文件名以前缀开头的可执行文件的进程被隐藏。



此外，可以将单个进程 ID 写入配置系统。这些进程 ID 由 r77 获取，实际上，进程被 ID 隐藏。对于磁盘上的文件，首选的方式是通过前缀隐藏进程和可执行文件。对于无法更改文件名的内存中创建的进程，通过 ID 隐藏进程是两个选项之一。

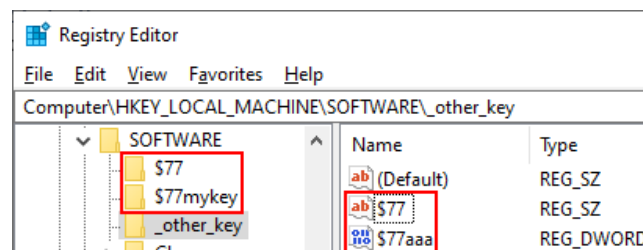


或者，也可以使用配置系统通过特定名称隐藏进程：



### 2.5.3 注册表

注册表项和值由前缀隐藏。



### 2.5.4 TCP & UDP 连接

TCP 和 UDP 连接基于以下任一情况隐藏：特定条件：

- 该进程被前缀隐藏。
- 该进程被 ID 隐藏。
- 该进程按名称隐藏。

或者具体的配置：

- 在配置系统中可以找到 TCP 或 TCPv6 连接的本地或远程端口。
- UDP 或 UDPv6 连接的端口在配置系统中找到。UDP 连接没有远程端口。

Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port
wininit.exe	592	TCP	0.0.0.0	49665	0.0.0.0	0
services.exe	740	TCPv6	[0:0:0:0:0:0:0:0]	49671	[0:0:0:0:0:0:0:0]	0
services.exe	740	TCP	0.0.0.0	49671	0.0.0.0	0
lsass.exe	748	TCPv6	[0:0:0:0:0:0:0:0]	49664	[0:0:0:0:0:0:0:0]	0
lsass.exe	748	TCP	0.0.0.0	49664	0.0.0.0	0
svchost.exe	996	TCPv6	[0:0:0:0:0:0:0:0]	135	[0:0:0:0:0:0:0:0]	0
svchost.exe	996	TCP	0.0.0.0	135	0.0.0.0	0
svchost.exe	1060	UDPv6	[0:0:0:0:0:0:0:0]	80	*	*
svchost.exe	1060	UDP	0.0.0.0	80	*	*
svchost.exe	1060	TCPv6	[0:0:0:0:0:0:0:0]	80	[0:0:0:0:0:0:0:0]	0
svchost.exe	1060	TCP	45.136.30.249	80	222.186.129.68	53874



要隐藏传出 TCP 连接，请将远程端口写入配置系统。例如，隐藏远程 TCP 端口 443 会隐藏由 Web 浏览器等创建的所有 HTTPS 连接。

要隐藏 TCP 侦听器，请将本地端口写入配置系统。

## 2.6 隐藏前缀

# 定义 `HIDE_PREFIX L"$77"`

hide 前缀是一个编译时变量。它指定隐藏的名称的开头。这适用于进程的文件名、文件和目录以及第 2.5 节中描述的所有其他实体。如果文档中提到了字符串 \$77、它指的是在整个代码中使用的编译时变量。这是更改前缀的唯一地方。

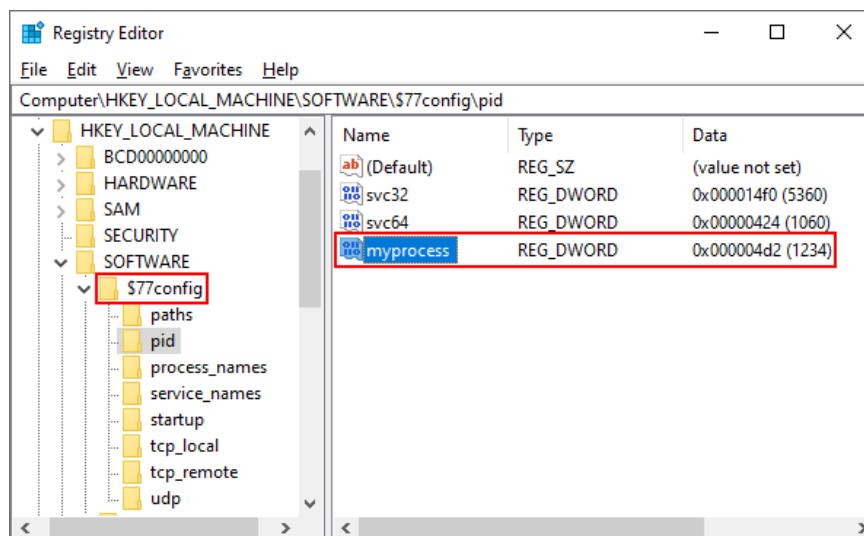
此外，此更改必须应用于

`GlobalAssemblyInfo.cs`。

不能用 r77 注入被前缀隐藏的进程。这主目录 r77 的将返回 错误的， 如果进程以前缀开头。

## 2.7 配置系统

这 配置 系统 是 存储 在 这 登记处 在下面 `HKEY_LOCAL_MACHINE\SOFTWARE\$77config`。此注册表项的 DACL 设置为允许对所有用户进行完全访问。



r77 每 1000 毫秒将配置读取到一个包含以下信息的结构中：

- 隐藏进程 ID 数组 隐藏进程名称数
- 组 隐藏路径数组
- 
- 隐藏服务名称数组 隐藏本地 TCP 端
- 口数组 隐藏远程 TCP 端口数组 隐
- 藏 UDP 端口数组
- 
- 启动路径数组



此数据用于根据自定义配置隐藏实体。任何进程都可以写入配置系统并执行**不是**需要提升的权限。

值的名称通常被忽略。价值 `$77config\pid\svc32` 和 `$77config\pid\svc64`, 但是, 保留给 r77 服务, 不应修改。它们是在 r77 服务启动时自动创建的。

**笔记:** 用 **具体的** 值名称 – **不要** 使用随机值名称。每次创建一个新的值名称时, 列表会随着时间的推移变得很长, 最终会减慢计算机的速度。

### 2.7.1 进程 ID

子键 `$77config\pid` 包含要隐藏进程 ID 的 DWORD 值。此外, 来自隐藏进程的网络连接也被隐藏。可以使用测试控制台测试此功能。

### 2.7.2 进程名称

子键 `$77config\process_names` 包含 REG\_SZ 值以及要隐藏的进程的文件名。此外, 来自隐藏进程的网络连接也被隐藏。

**笔记:** 仅当文件名不能带有前缀时, 才建议按 ID 或名称而不是按前缀隐藏进程。流程挖空尤其如此。该前缀还可以防止进程被 rootkit 注入。

### 2.7.3 路径

子键 `$77config\路径` 包含 REG\_SZ 值以及要隐藏的文件、目录、连接或命名管道的完整路径。例子:

- `C:\path\to\file.txt`
- `\\.\管道\my_named_pipe`

### 2.7.4 服务名称

子键 `$77config\service_names` 包含 REG\_SZ 值以及要隐藏的服务名称。根据此列表检查服务的名称和显示名称。

### 2.7.5 本地 TCP 端口

子键 `$77config\tcp_local` 包含要隐藏本地 TCP 端口的 DWORD 值。

### 2.7.6 远程 TCP 端口

子键 `$77config\tcp_remote` 包含要隐藏远程 TCP 端口的 DWORD 值。

### 2.7.7 UDP 端口

子键 `$77config\udp` 包含要隐藏 UDP 端口的 DWORD 值。



### 2.7.8 启动路径

子键 `$77config\启动` 包含 `REG_SZ` 值以及 `r77` 服务启动时应执行的文件路径。这会在 Windows 启动时和任何用户登录之前发生。这些文件（通常是可执行文件）在 `SYSTEM` 帐户下启动。

## 2.8 自定义启动文件

如第 2.7.8 节所述，可以将启动文件（通常是可执行文件）的路径写入注册表。`r77` 服务将使用外壳执行在系统启动时。

**问题：** 如果您为启动设置了隐藏文件，例如使用 `HKCU\...\运行键`，Windows 找不到该文件（因为它是隐藏的），因此它不会启动。

**解决方案：** `r77` 负责启动隐藏文件。这有几个优点：

1. 您的文件将在具有系统完整性的 `SYSTEM` 帐户下启动。
2. 您的文件将在第一个用户登录之前启动。
3. 您可以添加文件以非提升权限启动，它们将在系统完整性下启动。

如果您希望您的进程在特定用户帐户下运行，则必须执行模拟。如果您需要访问用户的桌面，这是必需的。

**笔记：** 只需将文件添加到 `$77config\启动`，它不是隐式隐藏的。相同的规则适用：文件必须具有前缀，否则必须被配置系统隐藏。如果你希望文件不被 `r77` 注入，那么将 `helper` 签名写入可执行文件将避免注入（见 4.1 节）。

## 2.9 枚举与访问

`r77` 中的“隐藏”意味着从枚举中删除隐藏的实体。如果用户知道文件名，则仍然可以直接访问文件——或者如果知道进程 ID，则可以打开进程。


打开文件/进程/等的函数。没有被钩住，也不会返回“未找到错误”来进一步伪装隐藏的实体。一般假设是不会猜测隐藏实体的名称。

主要原因是，目前 `r77` 没有其他方法可以自我维护。例如，如果隐藏密钥完全无法访问，则 `r77` 无法从配置系统读取。


这个问题可能会在 `r77` 的未来版本中解决。

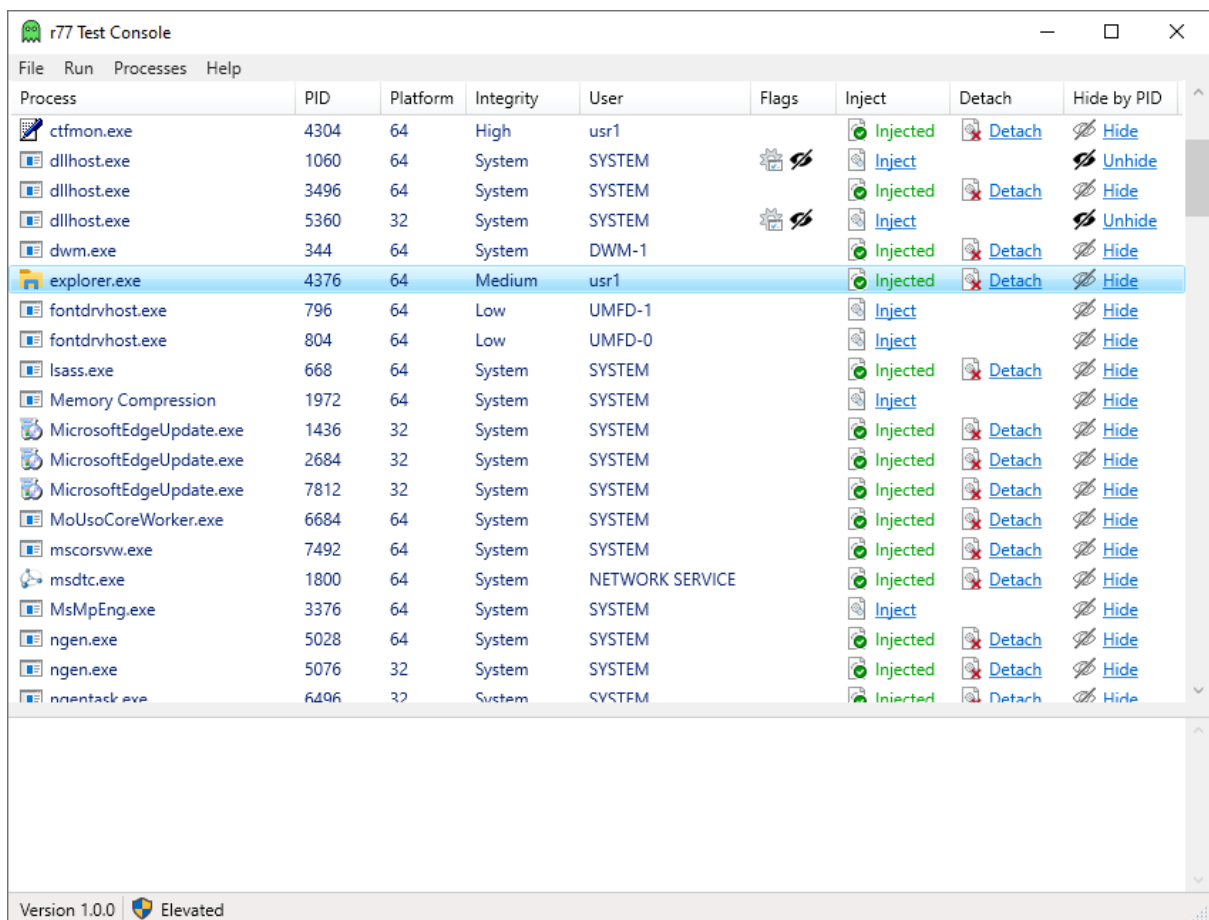


### 3 测试环境

部署 r77 的标准单个文件是  安装.exe。另外，测试环境可用于将 r77 注入单个进程或从单个进程中分离 r77。这在 r77 的开发过程中尤为重要。此外，它是在集成期间测试 r77 的有用工具。


#### 3.1 测试控制台

 测试控制台 可用于将 r77 注入单个进程或从单个进程中分离 r77。进程列表没有被 r77 过滤，因为 r77 没有注入测试控制台。



进程列表显示，哪些进程被注入。“Hide”-Link 可用于将特定进程 ID 写入配置系统。

“标志” 列显示特定进程的标志。

标志含义	
	该进程为r77服务进程。不能用 r77 注入。



	<p>该进程是一个 r77 辅助进程。不能用 r77 注入。</p> <p>测试控制台 是一个辅助进程，以及 Helper32.exe 和 Helper64.exe，由测试控制台调用。</p> <p>还， 安装程序和 卸载程序 是辅助进程。这是为了避免 r77 被注入其中。</p>
	<p>进程 ID 可在配置系统中找到。任务管理器不显示此过程。这包括默认的 r77 服务并且可以扩展到其他进程。 卸载程序 删除隐藏进程 ID 的列表。</p>

测试控制台是用 C# 编写的。这是包含文件的列表：

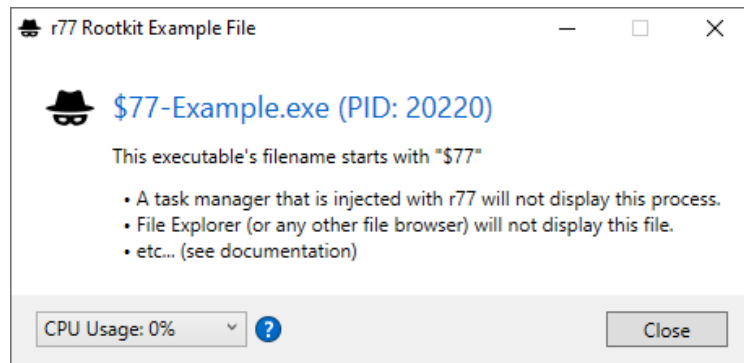
文件	目的
BytecodeApi*.dll	的依赖关系 测试控制台.exe。
Helper32.exe Helper64.exe	<p>测试控制台使用这些命令行可执行文件来：</p> <ul style="list-style-type: none"><li>• 获取所有进程的列表。由于Test Console中的进程列表包含 r77相关信息，因此枚举进程的可执行文件的位数需要与被枚举进程的位数相匹配。每次刷新进程列表时，都会调用两个可执行文件并解析它们的输出以填充进程列表。</li><li>• 注入特定进程或注入所有进程。</li><li>• 与特定进程分离或与所有进程分离。</li></ul>
r77-x86.dll r77-x64.dll	rootkit DLL 包含在以下资源中 安装程序 允许完全内存注入。然而， Helper32.exe 和 Helper64.exe 正在从磁盘加载这些 DLL 文件。

以上所有文件仅供测试控制台使用，不属于 r77 交付的一部分。

### 3.2 示例.exe

\$77-Example.exe 对于测试任务管理器和文件查看器很有用。要对进程隐藏执行快速测试，请启动此可执行文件，然后使用测试控制台将 r77 注入任务管理器。该进程在注入的任务管理器中不再可见。要隐藏文件，请使用测试控制台注入资源管理器。





此可执行文件不实现除 MessageBox 之外的任何内容。通过将其文件名重命名为以前缀开头，任何可执行文件都可用于测试目的。

要模拟 CPU 使用率，请更改“CPU 使用率”组合框中的值。如果此进程隐藏在任务管理器中，则 CPU 使用率将添加到系统空闲进程中。此外，处理器使用图将得到更正\*。

\* 请查看有关处理器使用图问题的第 6 节。



## 4 实施细节

除了上述部分中提到的内容之外，本节还描述了实现细节。

### 4.1 r77 标题

要将进程标记为注入或 r77 服务等，使用“r77 标头”。

进程内存中最适合将 r77 标头写入的位置是 DOS 存根。因为它不用于任何事情，所以如果在此处覆盖字节，则该过程不会发生故障：



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00000000h:	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	; MZ .....ÿÿ..
00000010h:	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	; .....@.....
00000020h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	; .....
00000030h:	00	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	; .....€...
00000040h:	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	; ..º..".í!..lí!Th
00000050h:	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	; is program canno
00000060h:	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	; t be run in DOS
00000070h:	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	; mode...\$. ....
00000080h:	50	45	00	00	4C	01	03	00	B0	8E	FC	5F	00	00	00	00	; PE..L...°žü_....
00000090h:	00	00	00	00	E0	00	22	20	0B	01	30	00	00	7E	04	00	; ....à." ..0..~..
000000a0h:	00	06	00	00	00	00	00	00	8A	9D	04	00	00	20	00	00	; .....š ... ..
000000b0h:	00	A0	04	00	00	00	00	10	00	20	00	00	00	02	00	00	; . ....
000000c0h:	04	00	00	00	00	00	00	06	00	00	00	00	00	00	00	00	; .....
000000d0h:	00	E0	04	00	00	02	00	00	00	00	00	03	00	60	85	00	; .à.....`...

请注意，这是可执行文件的内存映射的视图，并且 **不是** 文件的视图。



一个进程可能包含以下任一标题。签名被写入 DOS 存根的前两个字节。任何以下数据都写在签名之后。

签名和默认值	描述
R77_签名 = 0x7277	<p>当r77 DLL被注入时，R77_签名 写入主模块的 DOS 存根。</p> <p>这样，测试控制台可以检测是否注入了进程。</p> <p>如果将 r77 注入进程，但 r77 签名已经存在，主目录 返回 错误的 避免双重注射。</p> <p>后 R77_签名，一种 功能 指针 到 Rootkit::分离 被储存了。调用它将优雅地从进程中分离 r77。这由测试控制台使用，并且 卸载.exe。</p> <p>从进程中分离 r77 时，通过将 DOS 存根恢复到其原始状态来删除 r77 标头。</p>



<b>R77_SERVICE_SIGNATURE</b> = 0x7273	<p>r77 服务进程需要可识别安装程序和  卸载.exe。这个签名是写入可执行文件在 <b>编译时间</b> 避免在写入签名之前将 r77 注入服务进程。</p> <p>如果将r77注入r77服务进程，主目录 也返回 错误的。</p>
<b>R77_HELPER_SIGNATURE</b> = 0x7268	<p>帮助程序签名在编译时写入帮助程序文件。这些包括测试环境的任何可执行文件，例如作为  测试控制台.exe。</p> <p>如果将 r77 注入到辅助进程中，主目录 也返回 错误的。</p> <p>如果您不希望将 r77 注入到您自己的文件中，您可以在编译时将帮助程序签名写入您自己的文件中。</p>

## 4.2 编译时间常数

两个文件中都定义了编译时常量。必须在两个文  r77api.h 和  GlobalAssemblyInfo.cs。件中应用更改。

r77 签名已在第 4.1 节中提到。更改这些时，可以将 r77 自定义为与公开可用的 r77 二进制文件不同且无法检测到。

这是可以修改的附加常量列表：

常量和默认值	描述
<b>HIDE_PREFIX</b> = "\$77"	隐藏实体的前缀。
<b>R77_SERVICE_NAME32</b> = HIDE_PREFIX + "svc32"  <b>R77_SERVICE_NAME64</b> = HIDE_PREFIX + "svc64"	启动 r77 服务进程的计划任务的名称。对于计划任务，一个 .工作 文件已创建，因此前缀很重要。
<b>CHILD_PROCESS_PIPE_NAME32</b> = "\\.\pipe\" + HIDE_PREFIX + "childproc32"  <b>CHILD_PROCESS_PIPE_NAME64</b> = "\\.\pipe\" + HIDE_PREFIX + "childproc64"	用于子进程挂钩请求的命名管道的名称。



## 4.3 子进程挂钩

当 r77 服务启动时，所有当前运行的进程都会被注入。稍后产生的进程也必须被注入。有两个概念可以实现这一点：

**1.) 挂钩进程的创建：** 创建进程时始终调用的函数是 NtResumeThread。当进程A创建进程B时，该函数在进程B完全初始化后被进程A调用，但仍然挂起。在这个函数调用之后，进程B的创建就完成了。

所以，恢复线程被钩住并且 r77 被注入到新进程中 **前**

实际上调用了这个函数。不幸的是，32 位进程可以产生 64 位子进程，反之亦然。如前所述，32 位可执行文件无法将 64 位 DLL 注入 64 位进程。为了解决这个问题，r77 服务提供了一个命名管道来处理注入请求。当向 r77 服务发送新的进程 ID 时，该服务会注入进程并返回确认。收到服务确认后，注入完成，恢复线程可以调用。根据创建的子进程的位数，必须选择 32 位或 64 位 r77 服务来发送请求。

这样，进程在执行任何自己的指令之前会被注入 r77。这非常重要，因为有些程序（例如 RegEdit）初始化很快，然后在启动后不久执行枚举并显示结果。rootkit 必须在一开始就注入！

但是，不幸的是，并非所有进程都可以注入。Windows 10 保护某些进程不被访问，例如服务.exe。因此，单独依赖子进程挂钩将导致在没有注入 r77 的情况下产生服务，以及其他进程。这是当 **概念 2** 发挥作用。

**2.) 定期挂钩新进程：** 每 100 毫秒，检索一个正在运行的进程列表。该列表中的任何新进程都将被注入。这样，子进程例程遗漏的进程仍会被注入。但是，有长达 100 毫秒的延迟，其中 r77 未在该进程中运行。

如前所述，进程的双重注入没有负面影响。它是受支持的行为。

## 4.4 挂钩 API

Detours 是用于挂钩函数的挂钩库 ntdll.dll。这个 DLL 被加载到操作系统上的每个进程中。它是所有系统调用的包装器，使其成为环 3 中可用的最低层。内核32.dll 或其他库和框架最终会调用 dll 文件 职能。无法直接挂钩系统调用。这是对 ring 3 Rootkit 的常见限制。

隐藏服务异常需要挂钩 advapi32.dll 和 sechost.dll  
反而。请阅读第 4.4.7 节了解为什么这是一项要求。

以下章节描述了每个挂钩的函数。

### 4.4.1 NtQuerySystemInformation

此函数用于枚举正在运行的进程并检索 CPU 使用率。



#### 4.4.2 NtResumeThread

该函数用于注入已创建的子进程，而新进程仍处于挂起状态。只有在注入完成后，才会真正调用该函数。

**笔记：** 挂钩 创建进程 API 不是一个好的选择，因为它一次性创建和启动一个进程。此外，它是一个高级 API，其中有几个。将许多相似的功能挂钩，只是为了完成一项任务，这将是糟糕的设计。

#### 4.4.3 NtQueryDirectoryFile

此函数枚举文件、目录、连接点和命名管道。

#### 4.4.4 NtQueryDirectoryFileEx

这个功能非常类似于 Nt 查询目录文件 它也必须被钩住。实现大致相同。

这 目录 命令输入 命令行工具 使用此功能而不是 NtQueryDirectory 文件。

#### 4.4.5 NtEnumerateKey

此函数用于枚举注册表项。调用者指定一个键的索引来检索它。要隐藏注册表项，必须更正索引。因此，必须再次枚举键以找到正确的“新”索引。

#### 4.4.6 枚举服务组W

该函数用于枚举服务。

#### 4.4.7 枚举服务状态ExW

这个功能类似于 EnumServiceGroupW。

**笔记：** 这两个函数都通过 RPC 与 services.exe 通信以检索服务列表。一个钩子dll文件 不会有任何影响，因为只有 services.exe 使用 ntdll 函数。因此，更高级别的 DLL 的advapi32.dll 和 sechost.dll 被迷住了。

**笔记：** 对于服务隐藏，只挂钩了 Unicode 函数，因为枚举服务状态ExA，等似乎没有被任何应用程序使用。只是缺乏实际应用程序来实际测试 ANSI 类似物。

#### 4.4.8 NtEnumerateValueKey

此函数用于枚举注册表值。这个钩子的实现非常类似于NtEnumerateKey。

#### 4.4.9 NtDeviceIoControlFile

此函数用于使用 IOCTL 访问驱动程序。

如果驱动程序是\设备\Nsi 和 IOCTL 是 0x12001b，调用者请求所有 TCP 和 UDP 连接的列表。


要隐藏一行，所有后续行都需要向上移动一格，并且需要减少总计数。




## 4.5 AV 规避技术

已经实施了几种逃避 AV/EDR 检测的技术。

### 4.5.1 AMSI 绕过

这  舞台表演者 是 开始 经过 电源外壳。 这 电源外壳 脚本 用途 `Assembly.Load().EntryPoint.Invoke()` 从注册表加载 C# stager 可执行文件并调用它。但是，AMSI 是在 Powershell 和 .NET Framework 本身中实现的。一个电话程序集 `.Load()` 将触发 AMSI 并将可执行文件发送到 AV 进行分析。要绕过这一点，必须在整个 Powershell 进程中禁用 AMSI。

功能 `amsi.dll!AmsiScanBuffer` 必须修补以始终返回

`AMSI_RESULT_CLEAN`。在  安装.cpp，Powershell 启动脚本的组成是包含执行此补丁的代码。安装 r77 时会动态混淆此代码。

**笔记：** Powershell 代码不得包含任何 添加类型 带有 C# 代码的 cmdlet。它会调用 `csc.exe` (.NET 编译器)，这会将 C# DLL 放到磁盘上。

**笔记：** 只有在 Windows 10 上才需要绕过 AMSI，因为它在 Windows 7 中不受支持。

### 4.5.2 DLL 解钩


许多 EDR 解决方案在 `ntdll.dll`，有时在 `内核32.dll`。这些钩子监控 API 调用，尤其是代码注入、进程挖空等所需的调用。为了逃避进程挖空的检测，需要删除 EDR 钩子。



必须在以下位置执行脱钩：

- **舞台：** Powershell 调用的 C# 可执行文件使用进程挖空创建 r77 服务。为了逃避进程空洞的检测，需要移除 EDR 钩子。
- **r77 服务：** 因为 r77 服务会注入所有正在运行的进程，所以这里也必须删除 EDR 钩子。

删除 EDR 钩子是通过加载一个新的副本来实现的 `dll` 文件 从磁盘并用原始解开的文件内容替换当前加载的 `ntdll` 模块的 `.text` 部分。

EDR 挂钩通常是一个 `jmp` 几个可疑的 `ntdll` 函数开头的指令。这些钩子很容易删除，因为它们只存在于用户模式中。EDR 通常不实现内核模式挂钩。

**笔记：** 如果您正在调用  安装程序 使用流程挖空，执行的流程 进程挖空也应该解开 `ntdll.dll`。您可以使用该功能 `UnhookDll()` 作为实现参考。看：

 `r77api.h` 和  `解钩.cs` 对于 C# 和 C++ 执行。



## 5 个集成最佳实践

将 r77 包含到现有项目中很简单，可以通过多种方式完成。

### 5.1 包含Install.exe

包括安装程序并在安装项目时执行它是首选方式。支持在已安装 r77 时执行安装程序。它不会更新已经注入的进程；然而，新的进程被注入了新版本的 rootkit DLL。

安装程序可以写入磁盘并执行，也可以使用进程挖空生成。流程挖空的实现需要用你的项目语言来编写。如果你的项目是用C#写的，stager的流程挖空实现可以作为源码参考。否则，您必须自己编写。需要执行 32 位进程挖空，因为安装程序是本机 32 位可执行文件。

在内存中执行安装程序是值得的，因为将文件放在磁盘上可能会触发 AV 检测。

如果在使用进程挖空运行时遇到AV检测问题

安装.exe,

您应该在执行进程挖空之前执行 DLL 解钩。您有责任逃避 AV/EDR 检测以成功运行安装程序。执行进程挖空时如何实现DLL unhook见4.5.2节。

### 5.2 直接安装

如果您想更好地控制执行流程，另一种选择是将安装程序直接实现到您的项目代码中。为此，行为

安装.cpp 必须

被复制：

1. 安装Stager.exe 需要包含在您的项目资源中。
2. 安装Stager.exe 必须同时写入 32 位和 64 位注册表项。
3. 两个定时任务都需要创建和启动。

您的实现不需要本机可执行文件。确保您的代码正确处理 Windows x86/x64 差异。对于每个 r77 更新，构建解决方案并采用

安装Stager.exe 来自安装程序项目资源的文件。确保检查代码中的更改

安装.cpp 并根据需要在您的项目中实施它们。


安装程序本身的源代码相当短并且有注释。

**笔记：** 这仅适用于高级用户。通常，包括第 5.1 节就足够了。 安装程序 如中所述



## 6 已知问题

为确保质量和兼容性，r77 已通过多个操作系统版本和知名应用程序进行测试。这是已知和需要在未来版本中解决的所有问题的列表。

问题
不能注入沙盒进程。沙箱由 LOW 或 UNTRUSTED 的完整性级别定义。典型的应用程序是 Web 浏览器和文档阅读器的沙盒进程。注入沙盒进程会导致进程崩溃，因此，r77 目前不会注入沙盒。
据报道，注入关键进程（例如 smss.exe、csrss.exe 或 wininit.exe）会导致问题。确切原因未知，因此只有在未标记为关键的情况下才会注入进程。
注入 r77 时，某些 Visual Studio 二进制文件（例如 MSBuild.exe）无法正常工作。
<p>从 Windows 8 开始，进程可以处于挂起状态。这与 Windows 应用程序尤其相关。一旦主窗口最小化，Windows 应用程序就会暂停。</p> <p>注入这些进程会导致奇怪的行为，例如在进程恢复之前注入没有完成。此外，当进程处于挂起状态时，分离此类进程不起作用。卸载 r77 时，挂起的进程不会分离，可能需要重新启动才能从注入的 DLL 中清除 Windows 应用程序。这需要妥善处理。然而，这是一个小问题，不会导致不稳定或故障。</p>
<p>隐藏 CPU 使用率只能部分起作用。隐藏进程的 CPU 使用率正确累积到系统空闲进程。</p> <p>但是，显示 CPU 使用率的任务管理器的图形要么保持不变，要么不显示正确的值。</p> <p>详细：</p> <ul style="list-style-type: none"><li>• TaskMgr 和 perfmon 图根本没有得到纠正。</li><li>• ProcessHacker：显示单个 CPU 内核的图表中有峰值，因为该算法当前假设 CPU 使用率在所有逻辑处理器之间平均分配。它需要同时计算每个进程和每个内核，但尚未实现。</li></ul> <p>用  \$77-Example.exe 测试 CPU 使用率隐藏。</p>
隐藏 TCP 和 UDP 行当前在 netstat 命令中不起作用 cmd.exe 需要确定 netstat 使用的确切机制来挂钩适当的函数。





## 7 待办事项清单

以下功能已列入即将发布的版本：

- 隐藏任何进程的 PEB 中具有前缀的已加载 DLL。
- 隐藏 GPU 使用情况
- 实施一种机制来提升使用中等 IL 运行的进程的权限。
- 禁止查询/打开/删除/等。对隐藏实体的操作（参见部分 2.9）。



## 8 错误报告


请随时报告不在已知问题列表中的任何错误。要么在[GitHub](#) 存储库或访问 [bytecode77.com/contact](https://bytecode77.com/contact) 取得联系。

仅针对受支持的操作系统考虑错误报告（请参阅第 1.1 节）。

### 范围




- 注入 r77 时，进程崩溃或行为异常。
- 由于文档中未说明的原因，无法注入进程。
- 即使注入了 r77，隐藏的实体也是可见的。任何应用程序都在范围内，而不仅仅是明确测试的应用程序列表。
- r77 服务在系统启动时无法启动。r77 服务崩溃。
- 安装失败。
- 卸载失败或没有完全删除 r77。
- 测试控制台或一般测试环境中的错误。
- 任何上面没有明确提到但构成错误的东西。

### 超出范围

- 未清零的内存区域。示例：枚举被挂钩，删除一项并减少计数。任何超出新数组边界的东西都不会被清零，因为预计调用者不会读取缓冲区之外的内容来查找隐藏实体。
- 臭名昭著的钩子函数参数会导致 r77 DLL 以正常使用不会发生的方式崩溃或故障。DLL 的渗透和模糊测试不在范围内。但是，诸如缺少对参数的 NULL 检查之类的编码错误也在范围内。
- 使用调试器揭示 rootkit 超出了范围。使用内核模式调试完全超出了范围。r77 向普通计算机用户和中级计算机用户隐藏实体，而不是渗透测试者。
- 内核代码超出范围，因为 r77 是 ring 3 rootkit。它不会对内核或内核模式驱动程序隐藏任何内容。
- 由特定的 AV 供应商检测；r77 的设计非常规避。无文件概念是使其成为可能的基石。但是，r77 是一个开源项目，AV 供应商最终会检测到 rootkit。为特定的 AV 供应商实施规避是一项艰巨的任务，一旦 AV 更新了检测程序，rootkit 最终将再次被检测到。如果你想要它是 FUD，那么你必须自己做修改。无法实现规避的唯一情况是，如果 r77 驻留在磁盘上，则情况并非如此。扫描时间检测
-  安装.exe。这个文件应该使用 process 执行挖空如第 5 节所述。以无文件方式运行安装程序是您的工作。



## 9 变更日志

版本	释放	变化
0.6.0	17.12.2017	测试版
1.0.0	21.02.2021	<b>初始发行</b> <ul style="list-style-type: none"> <li>• 完全重写</li> <li>• 解决了 Beta 版使用 Detours 而不是 MinHook 的所有问题</li> <li>• 实现正确的和开箱即用的安装和持久性</li> <li>• rootkit 是无文件测试</li> <li>• 框架</li> <li>• .....还有更多</li> </ul>
1.0.1	05.03.2021	<ul style="list-style-type: none"> <li>• 错误修复：注入关键进程（例如 smss.exe、csrss.exe 或 wininit.exe）时崩溃。解决方案：不要注入关键进程。</li> </ul>
1.1.0	11.04.2021	<ul style="list-style-type: none"> <li>• 按名称隐藏进程</li> <li>• 按完整路径隐藏文件、目录、连接和命名管道</li> </ul>
1.2.0	18.04.2021	<ul style="list-style-type: none"> <li>• 按前缀和名称隐藏服务。</li> <li>• <b>突破性变化：</b> 配置系统现在在 HKEY_LOCAL_MACHINE\SOFTWARE\77config 只。DACL 设置为允许对所有用户进行完全访问。钥匙 HKEY_CURRENT_USER\Software\77config 不再考虑。进行此架构更改是因为某些进程（例如，NETWORK SERVICE 帐户下的进程）没有对 HKU 注册表配置单元的读取访问权限。的帮助文件</li> <li>•  测试控制台 组合成 一组文件：  Helper32.exe 和  Helper64.exe。</li> </ul>
1.2.1	20.06.2021	<p>已经实施了几种 AV 规避技术（参见第 4.5 节）。</p> <ul style="list-style-type: none"> <li>• 通过覆盖绕过 Powershell 启动的 AMSI 检测 amsi.dll! AmsiScanBuffer 与 退。</li> <li>• 解开 dll 文件和 内核32.dll 在 stager 和 r77 服务中逃避 EDR 检测。</li> <li>• 钩 sechost.dll 代替 api-ms-*.dll。</li> </ul>
1.2.2	31.08.2021	<ul style="list-style-type: none"> <li>• 自定义启动文件（见 2.8 节），解决 Windows 在启动时不考虑隐藏文件的问题，因为 Windows 无法“看到”这些文件。</li> </ul>