Geekbang. 极客邦科技 InfoQ

QCon 全球软件开发大会 【北京站】2016

沙箱逃逸：来自安全软件的鼎力相助

丁川达
腾讯玄武实验室

Intemational Software Development Conference

关于演讲者

- 曾经从事软件开发工作
  - 基于 Chromium 的浏览器开发，移动浏览器开发
- 2014 年加入腾讯
- 玄武实验室专注于研究真实世界的安全问题

安全软件指的是……

- 名字里带“安全”字样的
  - Internet Security
  - Total Security
  - Maximum Security
  - …
- 有显著安全影响的软件

在AV-Test列出的杀毒软件里...

55%

有漏洞

数据来自 2015 年 12 月 Home User Product Test 中列出的产品

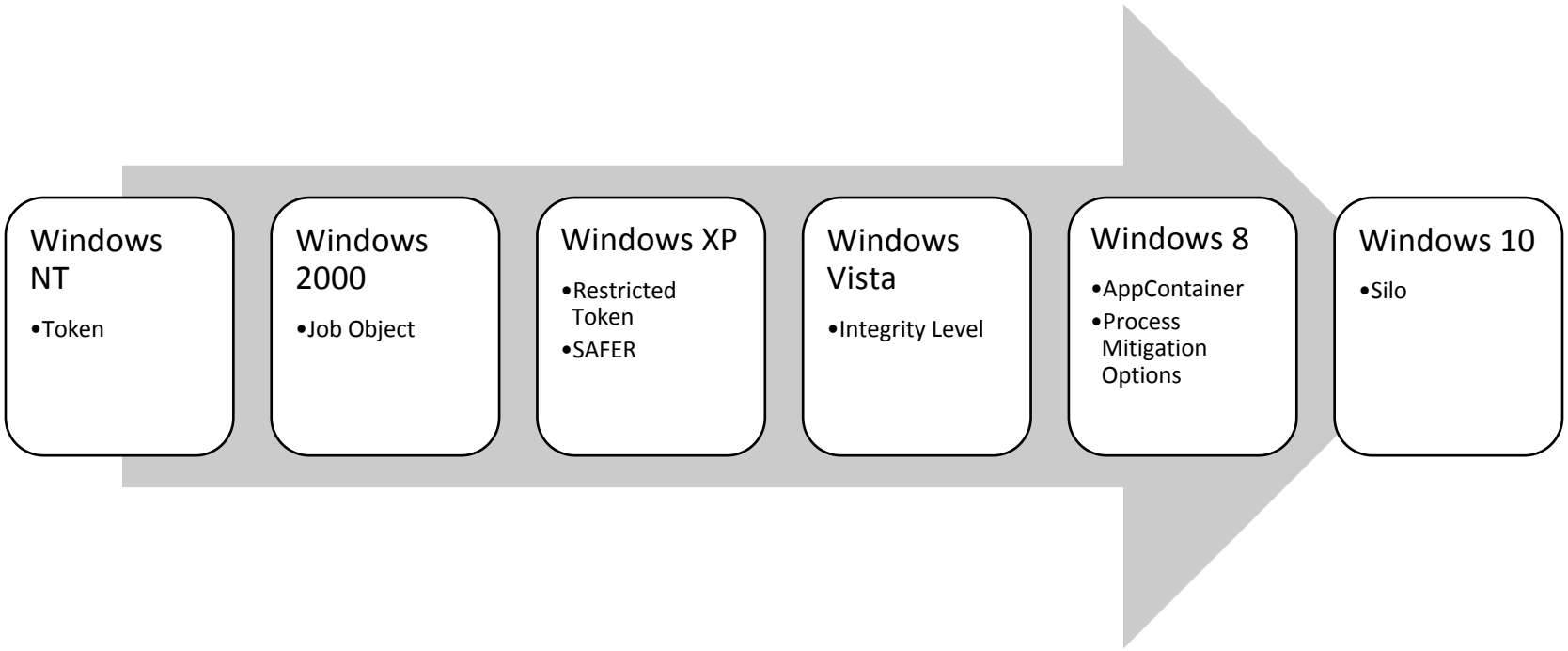
Windows 上的沙箱

- 早期原型可能来自 HIPS 软件
  - 例如 *System Safety Monitor*（还有人记得么？）
- 两大流派
  - 基于内核钩子，例如 *Sandboxie*
  - 应用层沙箱，依赖于系统访问控制

Windows 上的应用层沙箱

- 最早见于 *Internet Explorer 7* 保护模式
- **David LeBlanc** 在 *Practical Windows Sandboxing* 中进行了详细介绍
- 后来被 *Google Chrome* 使用
- 现在在 *Microsoft Office*, *Adobe Reader* 等软件中大量使用

沙箱相关的 API 支持


Windows NT
•Token
Windows 2000
•Job Object
Windows XP
•Restricted Token
•SAFER
Windows Vista
•Integrity Level
Windows 8
•AppContainer
•Process Mitigation Options
Windows 10
•Silo

一个常见的应用层沙箱实现

- Restricted Token
  - 限制对安全对象的访问
- Job Object
  - 限制进程数，Win32k 相关的功能
- 独立的桌面、窗口站
  - 限制窗口消息、窗口钩子、键盘钩子
- Low Integrity Level
  - 又一层对安全对象的访问限制
  - 给新创建的对象加上标签
- AppContainer
  - 限制对象名称空间
  - 限制网络访问

各种作死姿势

花样作死

“我们需要聊聊”式作死

- 高权限组件和低权限组件
- 它们需要交流信息
- 但却不知道正确的方式
- 作死

BitDefender Total Security

- 几乎所有内核对象都设置了 **ALL APPLICATION PACKAGES** ACE
- 命名管道 **\\.\pipe\VSSERV** 没有任何安全措施

```
struct message {
   int unknown[3];
   int control_code;
   int total_length;
   int unknown[2];
   int path_length;
   char path[1];
}
```

- 发送控制码 **0x17a5** 即可以当前用户身份启动任意进程
- CVE-2015-8693
- *BitDefender* 送给我们一年的激活码（多谢！）

Comodo Internet Security

- `\Comodo\GeekBuddy\launcher.exe`
- **`launcher.exe`** <路径> <参数>
- 高权限 `launcher_service` 的 RPC 客户端
- 有数字签名验证
- 一个程序即可绕过
  - `\Comodo\GeekBuddy\unit.exe`
  - 从命令行路径加载 DLL
- CVE-2015-8690

Total Defense Internet Security

- 本地 COM 服务器 `ccprovsp.exe`
- 可通过 `CoCreateInstance {AACF4A1C-BC69-4359-9518-DF3F77E462BF}` 启动
- IID `{5CF74415-61B8-4EF3-95E0-23576BEB31FC}`
- 控制码 `0B7C2CB3-E910-4672-9BDF-7E636333DC47`
- 传入控制码和文件路径即可以 SYSTEM 用户身份启动任意进程
- CVE-2015-8695

“这个我也会”式作死

- 绕过系统默认的安全机制
- 自己实现各种安全检查
- 作死

Internet Explorer “权限提升策略”

- 控制 EPM 沙箱的权限提升
- {HKLM,HKCU}\SOFTWARE\Microsoft\Internet Explorer\Low Rights\ElevationPolicy
  - {GUID}
    - AppName
    - AppPath
    - Policy
      - 0 – 禁用
      - 1 – 低
      - 2 – 中，带提示
      - **3 – 中，静默**

Avast! Internet Security

- `\AVAST Software\Avast\aswWrcIELoader32.exe`
- **`aswWrcIELoader32.exe`** `<DLL 目录> <DLL 文件名> <窗口句柄>`
- 静默提升到中完整性级别
- 然而这个文件没有 AppContainer ACE
- 在 EPM 中 `CreateProcess` 会失败，拒绝访问
- 没关系，我们可以直接调用 `IEUserBroker`
- 这个功能可以向目标窗口所在进程注入 DLL
- CVE-2015-8692

Panda Internet Security

\Pandasecuritytb\dtuser.exe

- 静默提升到中完整性级别
- 执行任意命令

  dtuser.exe runappasadmin calc.exe
- 复制任意文件

  dtuser.exe copyfile <源> <目标>

HP Support Assistant

HPSAObjectMetrics.exe

- 静默提升到中完整性级别

HPSAObjectMetrics.exe

HPSAObjectMetrics:messageid=11&context=0&value=1&launch=C:\Windows\system32\calc.exe&params=123

“我是上帝”式作死

- 驱动程序是安全软件中权限最高的组件
- 但常常很不重视安全问题
- 多数安全软件创建的设备对象都设置了 **Everyone** (!) 可访问
- 当然他们还会把 IO 控制码设为 **FILE_ANY_ACCESS**

Kaspersky Internet Security

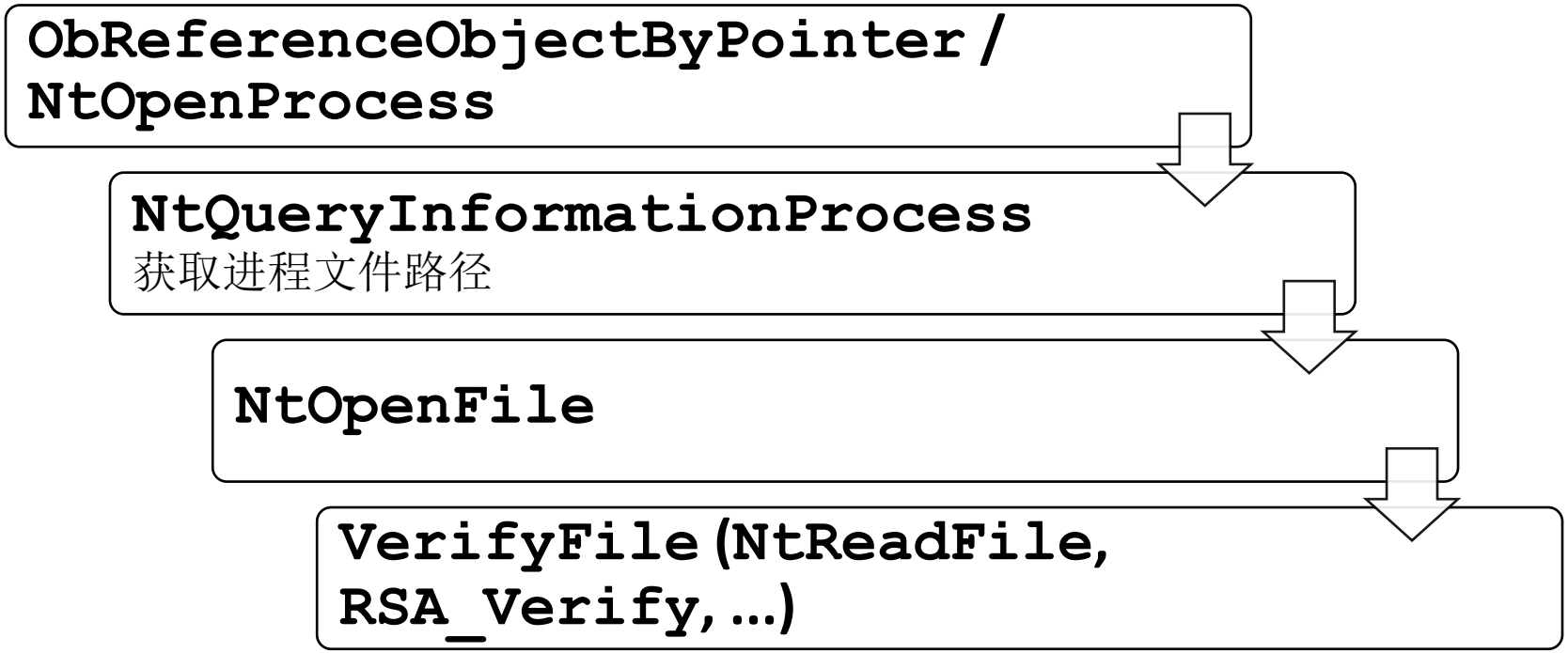
- `Kldisk.sys` 创建了设备 **`\Device\kldiskctl`**
- IO 控制码 `0x8123e040` / `0x8123e04c`
- 可在任意位置写入文件（不能覆盖）
  - 就算是 `System32` 也可以写入
  - DLL 劫持系统程序（如 `wmiprise.exe`），然后让 `services.exe` 执行
- CVE-2015-8691
- AVG 和 Avira 存在几乎一样的问题
  - **`\Device\avgidshrhr`**, **`\Device\avipbb`**
  - CVE-2015-8696, CVE-2015-8689

“但我们有安全措施呀！”

- 许多的安全软件都喜欢在 **IRP_MJ_CREATE** 的处理函数中自己做安全检查

```
void HandleIrpMajorCreate(PIRP Irp) {
    if (!VerifyProcess(IoCurrentProcess()) {
        Irp->IoStatus = STATUS_ACCESS_DENIED;
        IoCompleteRequest(Irp);
    }
    // 开始执行高权限代码
}
```

安全软件如何信任进程？


ObReferenceObjectByPointer/
NtOpenProcess
NtQueryInformationProcess
获取进程文件路径
NtOpenFile
VerifyFile(NtReadFile,
RSA_Verify,...)

系统如何知道进程文件路径？

- EPROCESS->SectionObject->Segment->ControlArea->FilePointer
- **ObQueryNameString**
- FileObject->QueryNameProcedure
- **IopQueryName**
  - 往下层文件系统发送 IO 请求包（IRP）
- 非常耗时，因此结果会被缓存

又见条件竞争

- Windows 允许用 **`MoveFileEx`** 移动一个正在运行程序的文件
- 只需用一个已签名的文件替换即可绕过
- 有些服务也用类似的方式做验证，同样可绕过

Lenovo System Update

- 创建没有安全设置的命名管道
- 在管道连接时验证客户端进程文件签名
  - 可用条件竞争绕过
- 可以以 SYSTEM 用户身份运行 `regedit.exe`
  - 但是命令行参数可以任意指定！
- 用 IFEO 劫持 `regedit.exe` 即可拿到 SYSTEM
- 而且它还在用户可写的位置创建临时文件
  - 然后就把临时文件导入到注册表里！
- 利用条件竞争可以写入任意注册表位置

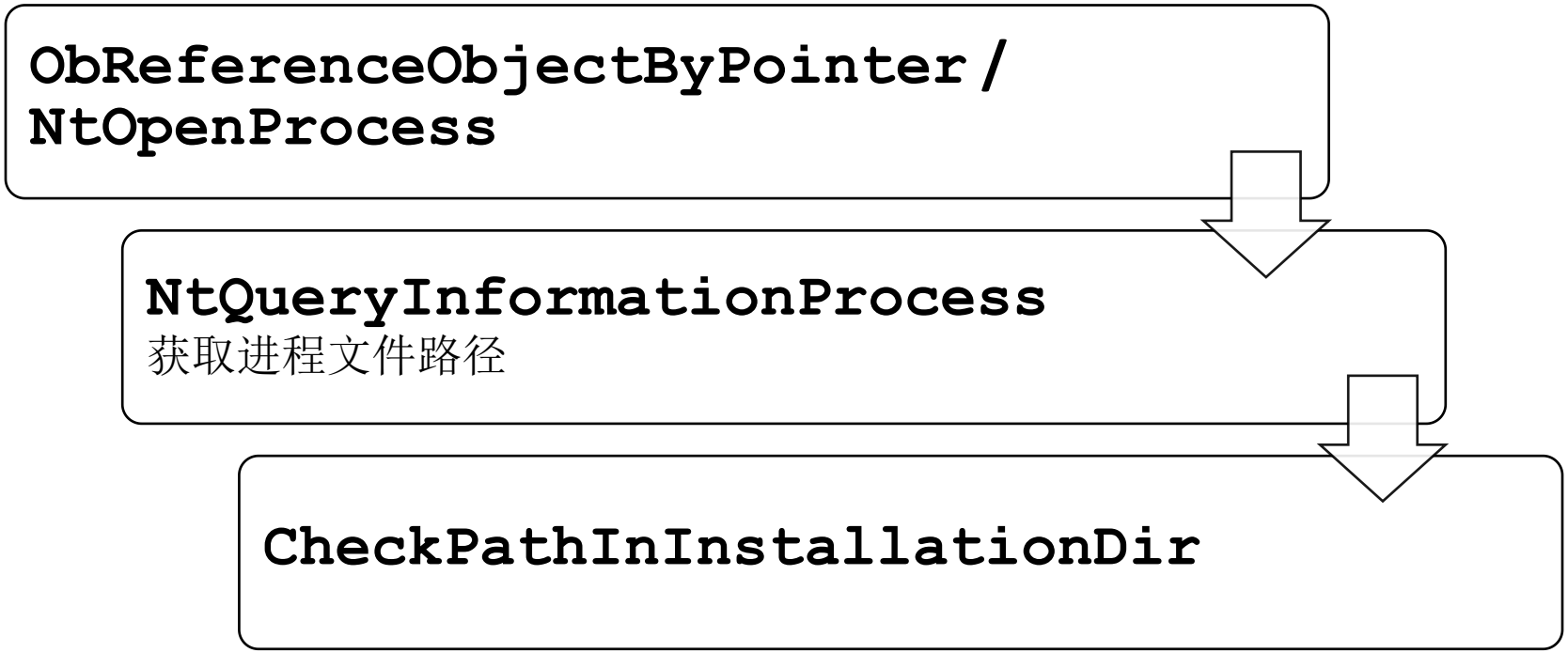
Lenovo System Update

- 它还可以安装任意 INF 文件，没有检查！
  - 可以用来加载任意驱动程序
- 它还监听 `0.0.0.0:20050`，并实现了一个控制服务
- **`DownloadBean`** 和 **`LaunchIE`** 下载并以当前用户身份执行任意程序
  - Trustwave 独立发现了这个问题，他们说是本地提权，实际上是远程代码执行
- CVE-2015-7333 / CVE-2015-7334 / CVE-2015-7335 / CVE-2015-7336

较好（但无文档）的做法

- 查询无缓存的文件路径
  - **NtQueryVirtualMemory**
    - MemoryMappedFilenameInformation
- 打开文件，至少需要 FILE_EXECUTE | SYNCHRONIZE
- **NtQueryInformationProcess**
  - ProcessImageFileMapping
- STATUS_SUCCUSS 说明是正确的文件，STATUS_UNSUCCESSFUL 说明不是
- 重复，直到拿到正确的文件句柄
- 使用这个句柄做验证操作

安全软件如何信任路径？


ObReferenceObjectByPointer/
NtOpenProcess
NtQueryInformationProcess
获取进程文件路径
CheckPathInInstallationDir

哦，你有进程保护？

- 攻击者可以从安装目录启动可信进程
- 不能 **`OpenProcess`**
  - 被驱动程序保护了
- 但是，**`CreateProcess`** 会返回一个完全控制句柄，还记得么？
- 轻松注入可信进程，与服务和驱动程序通讯
- 可用于 AVG 和 Avira

最好（且有文档）的做法

- 别通过检测路径就信任某个程序
- 正确使用访问控制机制隔离权限边界
- 不要自己发明一些安全检查

为什么会出这种问题？

为什么总是出这种问题？

真相

- 安全软件公司里并非所有的开发者都是安全专家
- 其中有许多普通的开发者犯着常见的错误
- 但他们在犯错时有最高的权限

“随便搜一下”

- 开发者遇到问题时，首先求助于搜索引擎
- 常见的结果通常来自 Stack Overflow、Code Project 等
- 特定的错误码可能会搜到个人博客上的解决方案
- 示例代码基本上从不考虑安全性问题

关键问题

- 文档和示例通常跟不上新系统发布的节奏
- 微软提供的示例通常不完整或不能在新系统上使用
- 开发者找不到正确的解决方案时，就会拼凑一些代码，自己造轮子
- 你可以说他们没有遵循 SDL，但代价是由 Windows 生态系统中的所有人来付的

微软的示例代码

“…改成下面这样就好啦，加一个 untrusted integrity ACE
D:(A;;GA;;;WD)(A;;GA;;;AN)S:(ML;;NW;;;S-1-16-0)…”

《操作互斥体时的拒绝访问错误》
Windows SDK 技术支持团队博客

神奇的字符串——是什么意思？

D:(A;;GA;;;WD)(A;;GA;;;AN)S:(ML;;NW;;;S-1-16-0)

Everyone

GENERIC_ALL

Anonymous Logon

GENERIC_ALL

Untrusted Integrity Level

No Write Up

微软的示例代码

“客户想要我们提供给 Everyone 用户组和 Anonymous 用户组添加完全控制权限的示例代码，这简直太简单了。”

```
_tcscat(szStringSecurityDis2,TEXT("D:(A;;GA;;;WD)(A;;GA;;;AN)"));
```

《网络程序操作命名管道时的拒绝访问错误》

Windows SDK 技术支持团队博客

开发者才是安全机制的破坏者

保护措施	“解决方案”
访问控制列表	空 ACL
Integrity Level	加上 Untrusted IL ACE
AppContainer	加上 ALL APPLICATION PACKAGES ACE
沙箱	加入沙箱白名单

开发者满意度很重要

- 给出完整可用的示例
- 经常更新，让它们在新系统上可用
- 做点搜索引擎优化

自动发现沙箱攻击面

想法

- 多合一的通用沙箱测试工具
- 即使在最恶劣的环境下也可以运行
  - 例如 Chrome 沙箱
- 可适应不同版本的 Windows
  - Windows 7 – Windows 10 TH2
- 输出系统对象访问权限的详细报告
  - 报告可以机读就更好了

Chrome 沙箱

- 无论权限设置如何，都无法打开任何文件
- **nt!ObOpenObjectByName -> nt!ObpLookupObjectName**
- **nt!ObpCheckTraverseAccess**
- 沙箱内没有 **\Global??** 的 **TRAVERSE** 权限
  - **\Global??** 是全局设备映射表
- DOS 设备名实际上是 **\Global??** 下的符号链接
  - **\Global??\C:** -> **\Device\HarddiskVolume1**

以往的相关项目

- **Stephen A. Ridley** 的 *SandKit*
  - Python
- **James Forshaw** 的 *Sandbox Attack Surface Analysis Tools*
  - C++ & C#
  - 一组小程序的集合

测试方法

- 审核对象的 ACL
  - 快，可靠，有文档的方法
  - 要是 Broker 进程代理了请求怎么办？
  - 要是目标进程内有令牌和句柄泄露怎么办？
- 进程令牌模拟
  - 文档较少的方法
  - 依然是在不同的执行环境里
- 进程注入
  - “微软概不负责”的方法
  - 测试在目标进程内完成

如何枚举系统内的所有对象？

- 进程、线程、Job
- 文件、目录、设备
- 命名管道、ALPC 端口
- 符号链接、Section
- 注册表键
- 会话、窗口站、桌面、窗口
- 事件、互斥体、信号量
- …

如何枚举系统内的所有对象？

- 思考：Windows 自己是如何管理这些对象的？
  - 对象管理器
- 对象树遍历
- 从根目录开始
  - 也就是 \ 对象目录

如何测试对象访问权限？

- 多数内核对象的系统调用接口都是一样的

```
NTSTATUS NtOpen*(PHANDLE ObjectHandle,
                 ACCESS_MASK AccessMask,
                 POBJECT_ATTRIBUTES
ObjectAttributes)
```

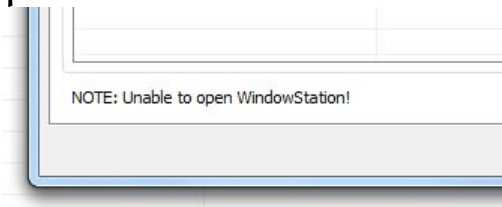
- 一个通用的分发函数搞定大多数内核对象
  - **NtGenericOpen**
- 其余的用代理函数处理
- 用一个数组存储对象类型到函数指针的映射
- 函数调用自动重定向到对应的 NT 系统调用

AppContainer 怎么搞？

- *AppContainer* 限制了 TRAVERSE 权限
- 没有 TRAVERSE 就无法枚举子对象
- 用命名管道从另一个进程获取对象全路径
- 将测试结果写回辅助进程

窗口站和桌面

- **OpenDesktopW** & **OpenWindowStationW**
- 对象根目录不一样
  - **User32!UserClientDllInitialize**
  - **WinStation** 目录会创建在会话根目录下
  - *Windows Object Explorer 64* 就挂在这了
- NT USER API
  - **NtUserOpenDesktop** & **NtUserOpenWindowStation**
  - 未导出函数
  - 系统调用索引号每个版本都会变



窗口站和桌面

- 用一个小型反汇编器搜索内存代码

```
__stdcall OpenWindowStationW(x, x, x) proc near
...
                call    ds:RtlInitUnicodeString(x,x)
...
                call    CommonOpenWindowStation(x,x,x)
                leave
                retn    0Ch
__stdcall CommonOpenWindowStation(x, x, x) proc near
...
                call    ds:NtOpenDirectoryObject(x,x,x)
...
                call    NtUserOpenWindowStation(x,x)
```

One More Thing…

厂商需要改变

- 找安全接口人比找漏洞难多了
- 拒绝提供漏洞修复的任何进度
- 甚至直接忽略漏洞报告

我们的团队

这次演讲内容的贡献者

刘科、王文群、王连赢、霍志鹏、韦伟

特别感谢

- 于旸 (@tombkeeper)
  - 组建了超赞的团队
- James Forshaw (@tiraniddo)
  - 关于 Windows 安全的精彩文章
- Tavis Ormandy (@taviso)
  - 慷慨提供了多个厂商的联系方式

提问？

xlab.tencent.com

@腾讯玄武实验室

Geekbang 极客邦科技 InfoQ

THANKS!