Users' Secure-But-Man-in-the-Machine: Exploiting Ill-Secured Communication Inside the Computer

概述：(S) 应用开发人员对于本地攻击者和进程间通信的态度是模棱两可的。(T) 作者希望揭示计算机内部的无特权的授权用户可进行的攻击。(A) 通过对三现实应用如 PM, Hardware tokens Backends with HTTP API 的 case study 揭示未授权的访问，client/server impersonation 的问题 (R) 揭示了漏洞，讨论了漏洞通常是由于设计不规范或编写不当而来。最终提供了缓解措施。

优点：①指出了许多安全软件如 PM 存在内部的攻击，一个未广泛关注的内容

②为未来 IPC 安全应用提供了建议。

问题：①文中提出了 IPC 的攻击需在正常的 server 连接前完成连接，此时间差的开销应考虑，不能超出攻击获取的收益

②可以进一步探索 PM 在本地的通信方式进行讨论。

③本文未给出实际的攻击实现，应提供真实的攻击测试，做成功率及开销的评估。

▲背景介绍

1) OS 提供了许多 inter-process communication (IPC) 机制，软件可通过 IPC 使得前端与后端通信，它们分别是同一 PC 的不同进程。

2) 先前研究多集中于来自未授权用户和 Internet 的外部威胁，本文关注 IPC 的安全性。

而实际许多 PC 支持多个授权访问，本文研究来自授权 insiders 的威胁 (非特权用户)

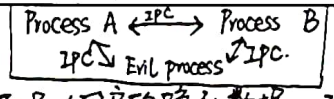
3) 应用开发者对本地攻击者和 IPC 通道的安全性的态度比较模糊。

①这类威胁未得到充分的考虑，安全专家认为防御本地攻击者是无用的

②应用实现通常会尝试认证或加密通信，但很少以同样谨慎的态度对待物理网络通信

▲攻击者的能力和攻击场景。

1) 考虑 multi-user computers



2) 攻击者是无特权的用户，尝试窃取另一个用户的隐私数据。攻击者的进程在后台运行且与受体的进程属于不同的登录会话。称为攻击为：man in the machine (MitMa)

需有帐户以此能力

3) 攻击场景 (个人 PC 不支持多用户同时使用，但允许 fast user switching)

①Console login: macOS 和 Linux 可以保证用户登出时进程仍运行，而 Windows 用户登出进程终止

②guest account: 可以通过 fast user switching 使恶意进程在后台运行

③SSH: Windows 和 macOS 以及 Linux 均提供了 SSH 的功能

④remote desktop: 可以在 Windows 中通过类似 fast user-switching 的方式运行进程。

(但本文未在 Windows server 做实证研究)

▲PC 内部 Client-server 通信 (本文关注 server 进程或设备监听来自 client 进程的连接的 IPC 方法)

1) Network sockets: 广泛应用于 CS 架构，S 在特定的 IP 和端口监听 C 的请求，可以同/异主机通信

▲C 和 S 的进程应在应用层相互认证，防止未授权的进程窃听通信，同主机通信会更加安全

①Attack vectors

· 恶意进程可以连接本地任意服务的端口，可完成 client impersonation；通过 port jacking，可在合法进程前完成 server impersonation.

在本地攻击时，由于 server 和攻击不可为同一端口号，可通过 port agility 选择下一个正确口号完成认证。

2) Windows named pipes. (以 Windows 为例，位于命名管道文件系统的根目录)

①枚举到 \\.\pipe\ 下，每个系统用户 (包括 guest) 均有访问权。

命名管道可由任意进程创建，可定义 security descriptor 描述可连接实例的最大数目，包含一个访问控制列表 DACL. (默认除创建者和管理员只有读权限)

若有同名的命名管道存在，则只有有 FILE_CREATE_PIPE_INSTANCE 的进程可创建新的实例，进程可通过设置 FILE_FLAG_FIRST_PIPE_INSTANCE 确保创建第一个程序。

②Attack vectors.

攻击者的恶意进程可以直接连接到管道，并伪装成合法客户端

攻击者可以通过创建一个管道实例来 hijack the pipe name 进而成为管道对象的拥有者进而伪装成 server，进而可以让一些实体创建新的管道实例，进而成为中间人。

(实际上要求 server 或 client 未执行严格的检查)。

3) Hardware security tokens (包括 keyboard, pointing devices 以及 hardware security token)

①Linux 中当前登录用户对 USB HID 有读写的权限，而 Windows 则可以有多个用户有权限。

②Attack vectors

在 Windows 中攻击者可以访问其它用户的 USB HID，故只能取决于软件或硬件的应用级的安全机制。

4) Safe IPC Methods.

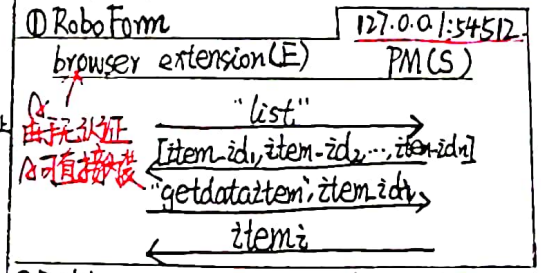
上述的攻击通常需要创建进程的时间差，而 anonymous pipes, sockets pairs 由于 IPC 通道两端同时创建，可避免上述攻击 (但此类管道只用于 related 进程)

macOS 中的基于 Mach kernel 的 Mach IPC methods 也可抵御上述攻击。

▲Case study: Password Manager (研究最两个不同部件的 PM: stand alone app & browser extension) 尽管考虑了浏览器扩展和 browser 之间通信的认证和加密，但未完全保护所有的通信

1) PMs with network sockets (在 OS 与浏览器中可移植且与 web API 兼容)。

①RoboForm



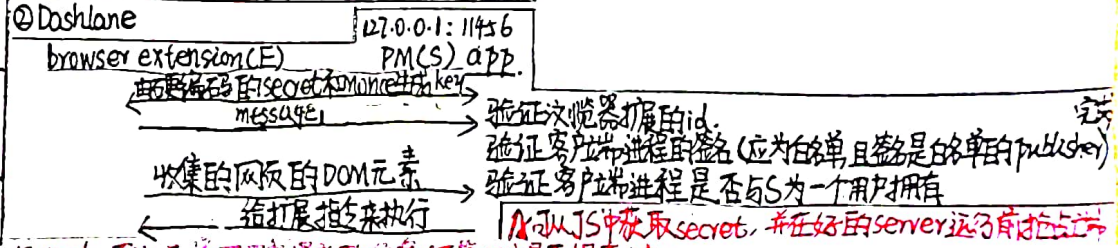
由于认证

可直接获取

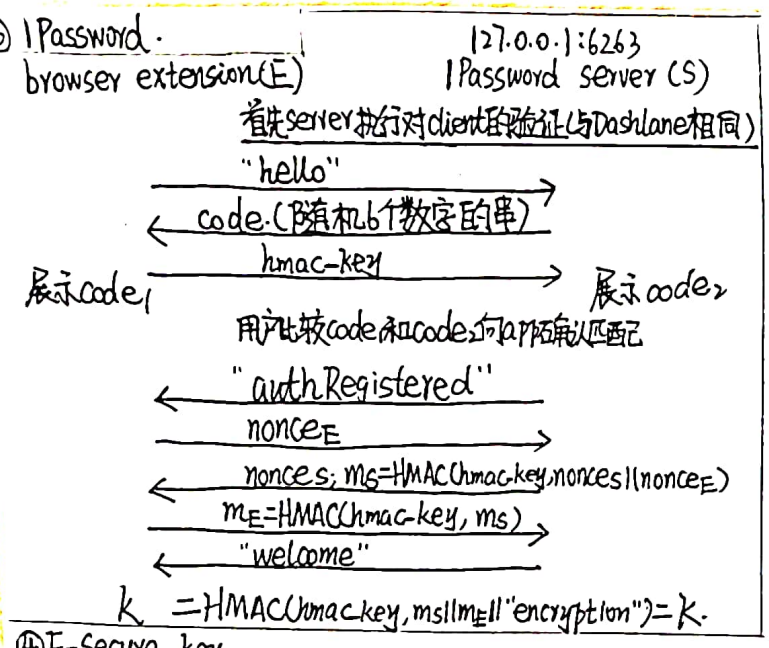
使用 HTTP, 无认证

伪装后可得到所有隐私信息

②Dashlane



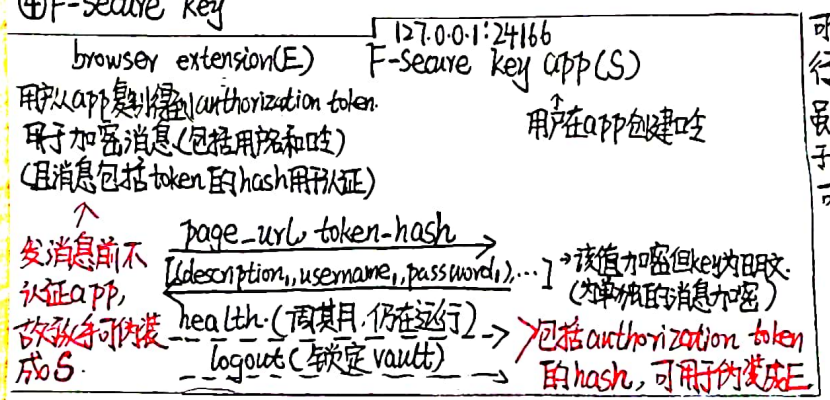
③ 1Password.



由于用户只在app端确认app是否相等,各自的进程可以跳过确认过程.

▲app可以向extension发指令,例如collectDocuments可收集用户访问或提交的内容

④F-Secure key



可伪装成双方故攻击可执行replay攻击.

虽然攻击无法解密,但由于无完整性保证,攻击者可以修改其中的条目.

2> Managers with native messaging (为network socket和named pipes提供替代方案)

▲用于browser extension和native code通信,Windows中使用随机名称的named pipe通信; Linux和macOS中使用匿名管道.

NMH由PM注册,可设置哪个browser可访问.

(作为browser的子进程)

▲可用于创建只使用web browser和browser extension访问的PM.

①Password Boss (使用native messaging和named pipes)

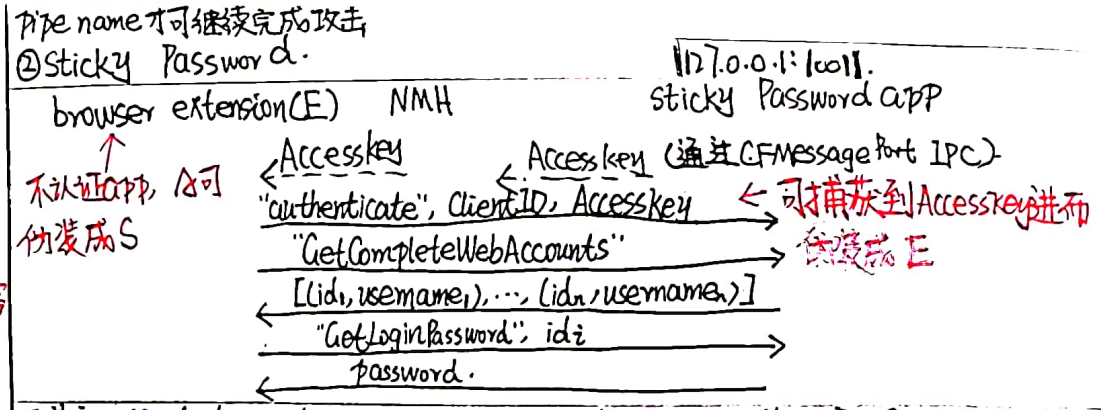
消息通过明文发送,且未尝试ACL之外的认证方式.管道名称固定,至多连60个实例

则所有认证(指ACL)的用户均可完成攻击,可得知所有的通信消息

只有guest access的攻击无法完成攻击,因为命名管道需认证用户可访问故攻击需首先hijack

pipe name才可继续完成攻击

②Sticky Password.



→其它case study: Hardware tokens: ①U2F security key. ②Fujitsu DigiSign

Software back-ends with HTTP API: ①Blizzard ②Transmission ③Spotify

Other C-S applications: ①MySQL ②keybase

▲缓解机制(提供概念上的说明,未提供全部的技术细节.

1>Spatial and temporal separation of user sessions: 用户会话在空间

2>Access control: 访问控制.

3>Attack detection: 采用具体的攻击检测方法

4>Cryptographic protection: 采用密码学层面的保护

5>Architectual changes to software: 更改软件的架构

▲讨论和未来工作

1>描述的漏洞基本由设计不小心或编写不当的软件造成,但采用IPC的软件也有安全的实践(Dropbox, SpiderOak, Box),它们采用了健壮的安全机制如DACL,严格检查参与通信的实体的身份及标识.

2>IPC问题的普遍原因:

①开发人员仍倾向于将本地视为可信环境.

②安全IPC的实践没有归档记录,开发者可能不知道威胁和相应的解决方案.

③进一步的研究

i)研究服务器软件中类似的MitMa漏洞

ii).同一用户的应用程序之间的攻击.