资源代价与安全算力

打造一款诊断型个人防火墙

安天 | 技术委员会

CONTENTS

目 录

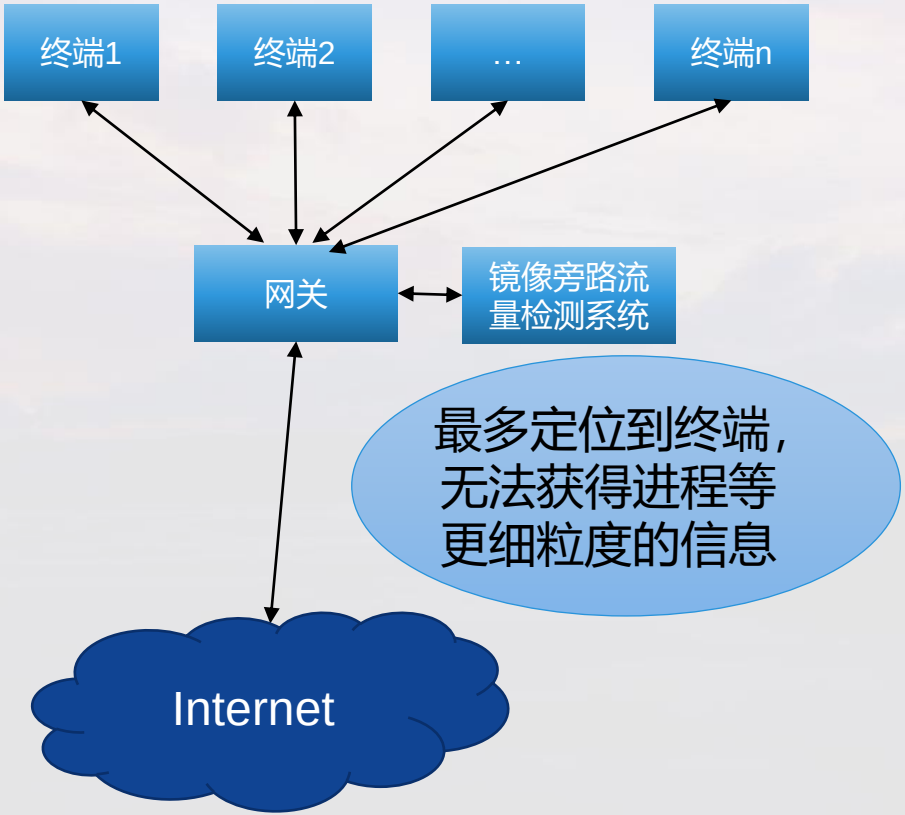
00 场景

进程联网，数据进出，

攻击载荷，取证追溯

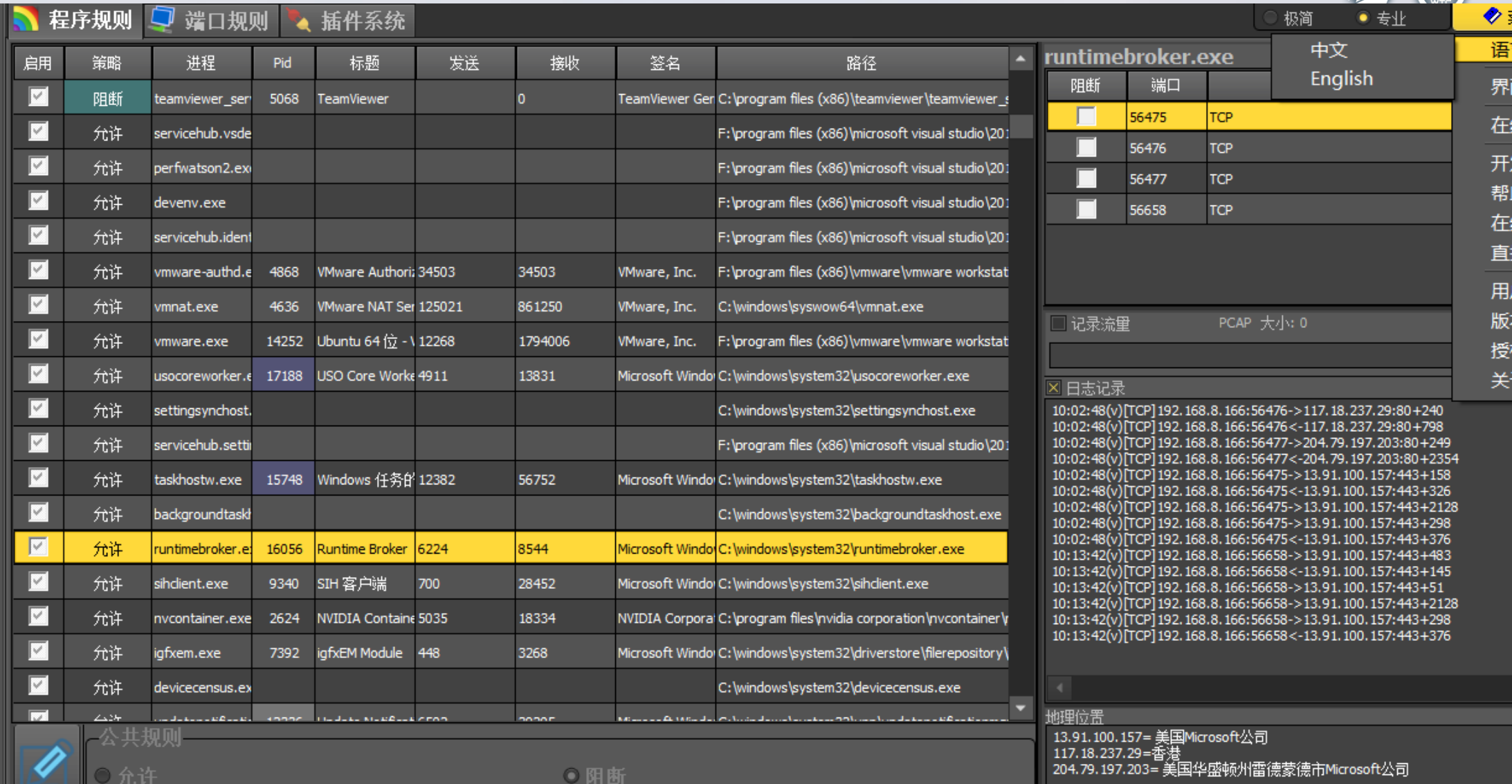
0-场景

- 面临问题



哪些进程在悄悄联网，连到了哪里？发了什么内容出去？有没有人扫描我？有没有内网沦陷机在横向移动？能否捕获黑客的攻击载荷？能否顺利取证、追溯。

0-场景 围绕用户进程和流量做出设计

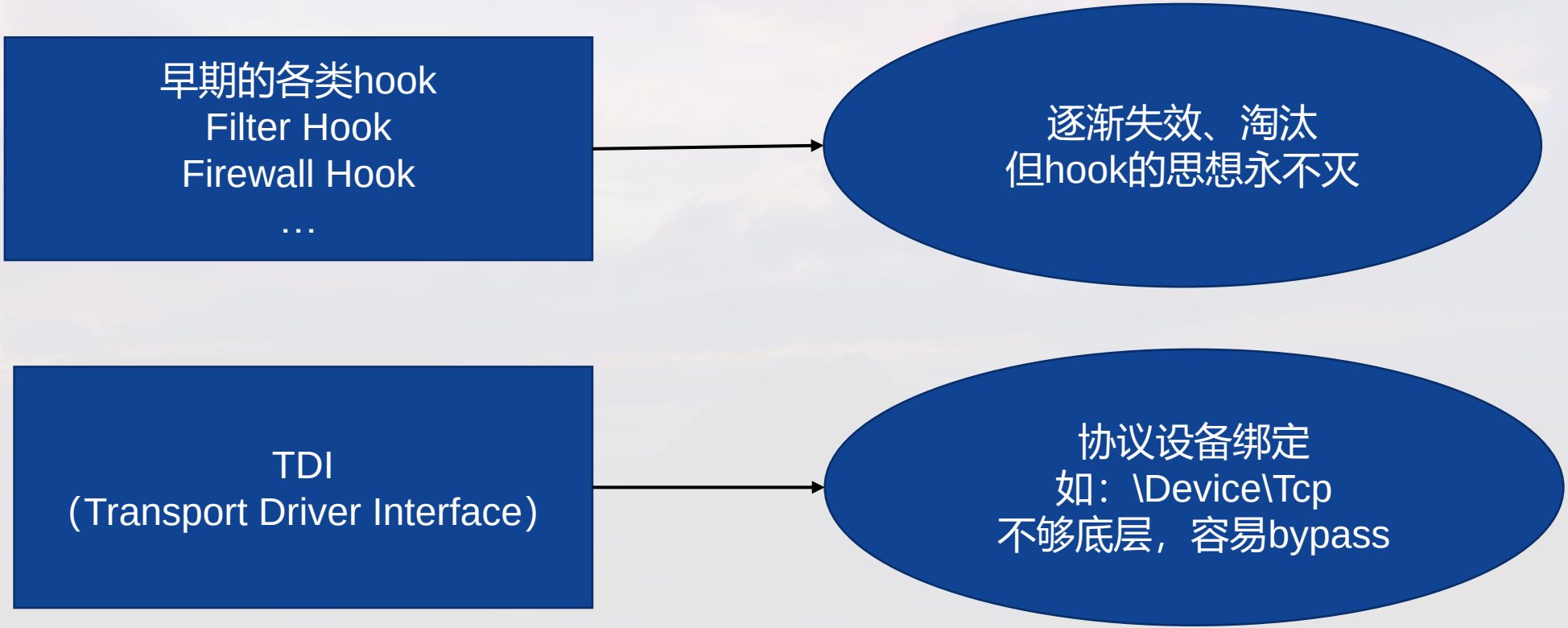


01 内核中的流量线索

获取对应进程信息以及端口列表

1-内核中的流量线索

- 个人防火墙技术演进史



1-内核中的流量线索

■ 个人防火墙技术演进史

NDIS
(Network Driver Interface Specification)

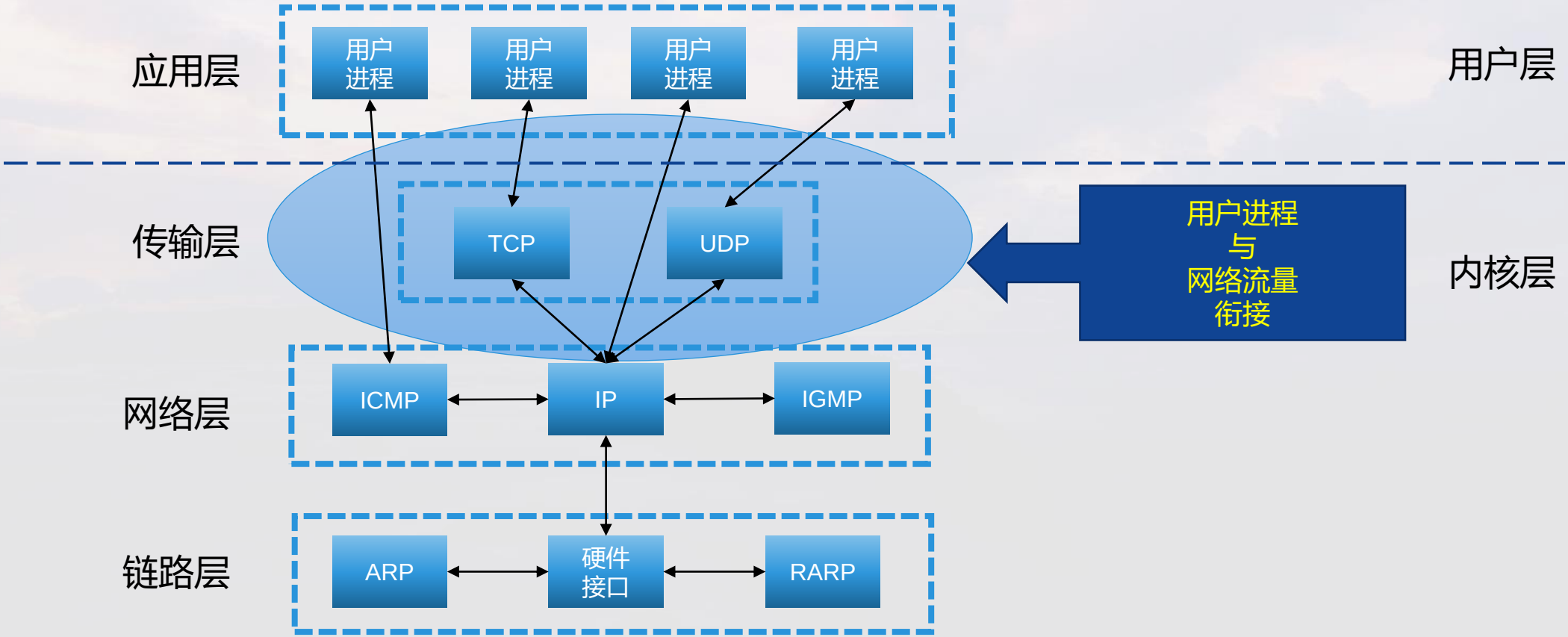
→ 1.协议驱动：插入自定协议。
2.小端口驱动：虚拟设备。
3.中间层驱动：本质依然是Hook。
都无法拿进程信息。如：WireShark就是用的NDIS驱动。

WFP
(Windows Filter Platform)

→ 目前微软推荐的过滤框架，后面详细说明。

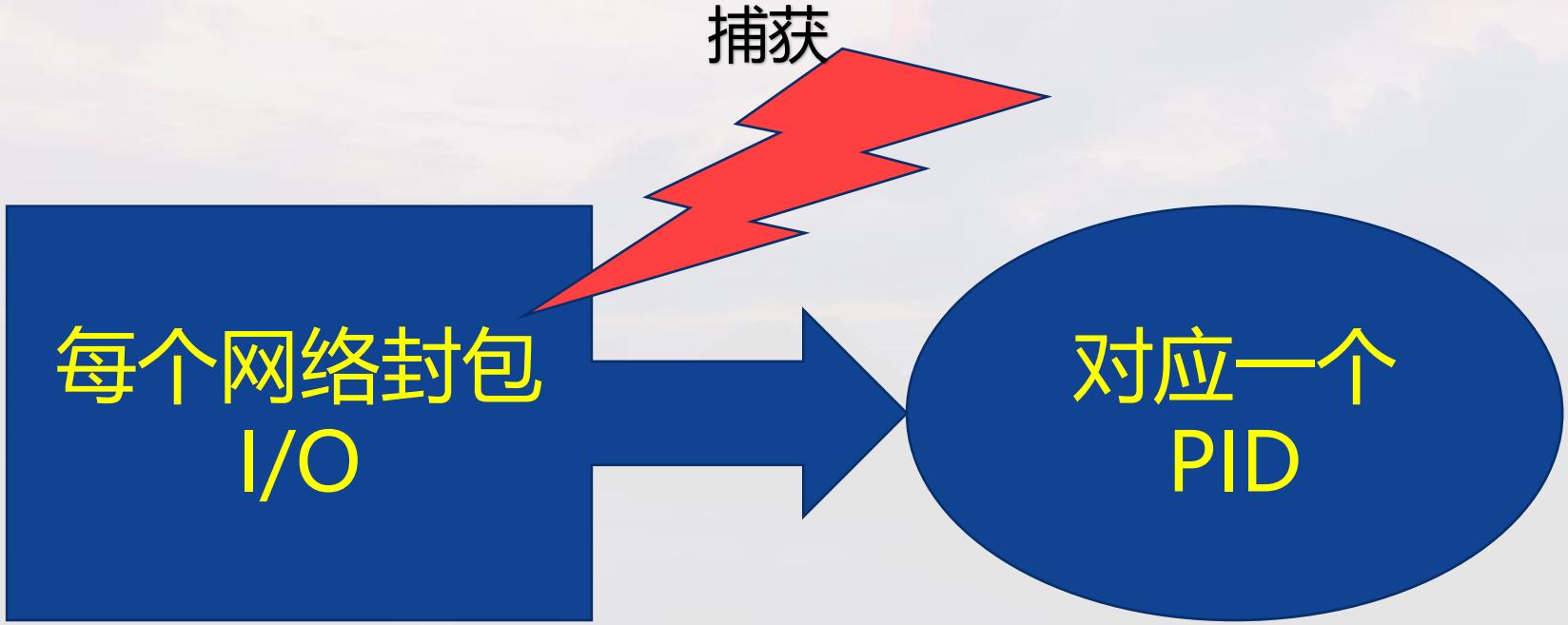
1-内核中的流量线索

- Internet所基于的TCP/IP协议模型，各个层次在操作系统中的分布



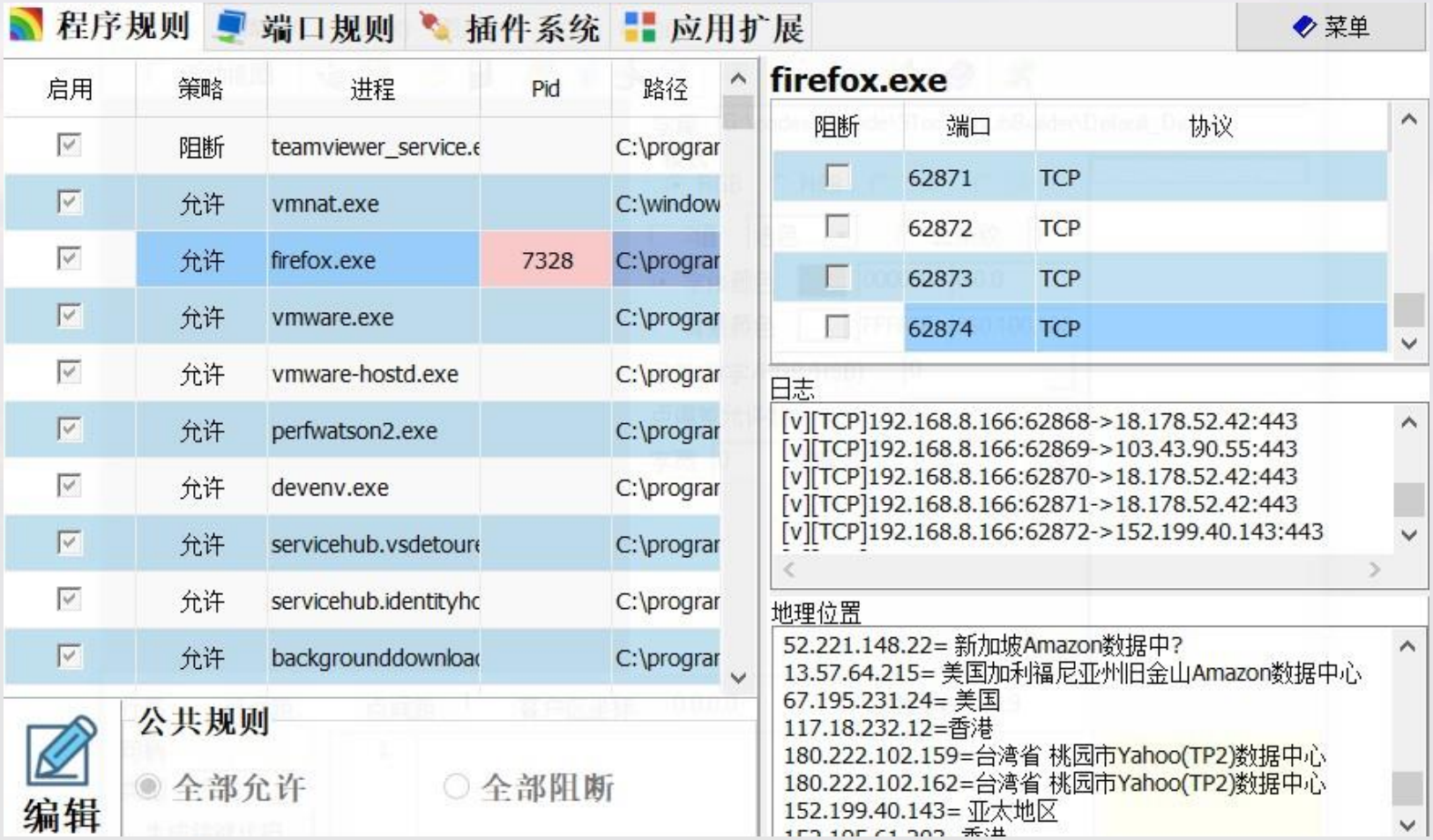
1-内核中的流量线索

- 合适的机制



1-内核中的流量线索

- 再看设计，进程对应的部分端口操作



禁止感染端口，
开放正常端口。
杀毒不停机。

1-内核中的流量线索

- 由PID引出更多维度的信息，作为检测向量，接下来就可以多维联动了

02 实时的诊断及多维联动

扫描感知，溯源，情报，监测，诱捕，取证

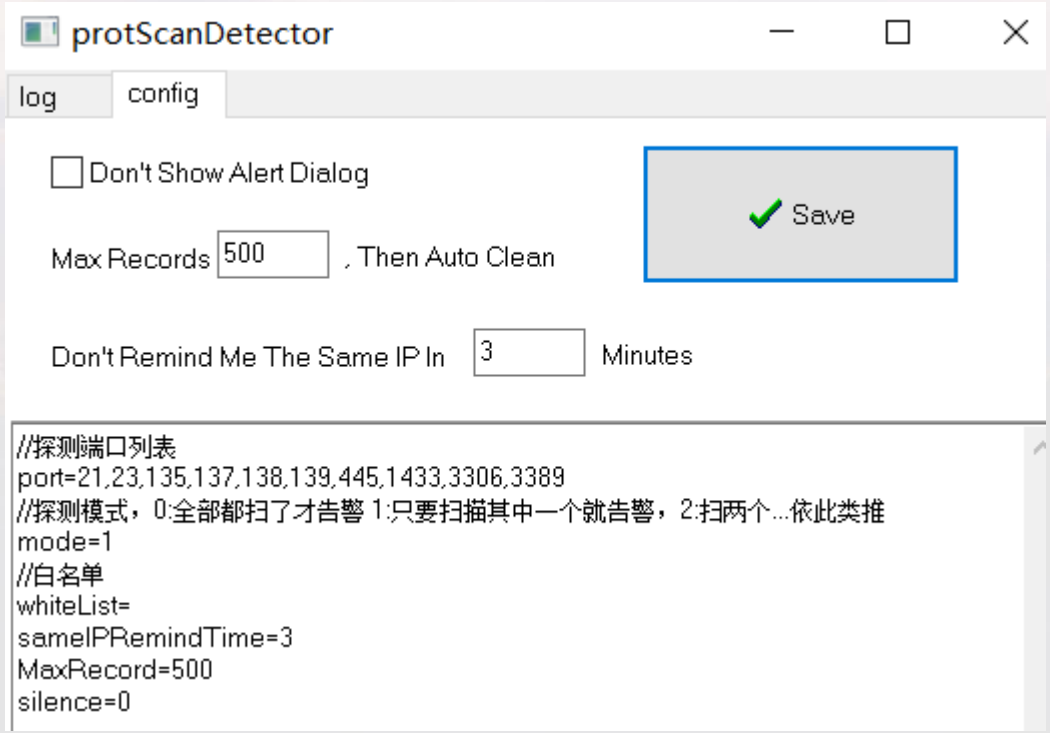
2-实时的诊断及多维联动

- 联动的形式：所有外部联动基于插件扩展，保持低耦合，易维护

启用	名称	作者	详细描述
☐	logManager	Simpower	日志记录插件，同时提供全日志查询图形界面
☐	logManger	Simpower	日志记录插件，同时提供全日志查询图形界面 ..
☐	advClean	Simpower	清理弹窗
☑	geolocation	Simpower	在线获取地理位置,用的人太多会Ddos封IP，造成查询失效，需要时启用，否则禁用
☑	PortScanDetect	Simpower	端口扫描探测插件，如果被人扫描，则告警！
☐	pythonEngin	Simpower	支持用log.py(Python,例子是python2.7语法)脚本来获取日志并进行处理，需要安装32位python环境，
☐	PluginTemp	Simpower	插件模板生成（C++语法）
☐	HoneyPot	Simpower	小蜜，可以打开端口，并将端口的流量转发致安全大脑(或蜜罐)进行分析。
☐	PASUnion	Simpower	流量分析联动插件，可以将流量信息上报至安天PAS或PTD，也可通过cfg.txt联动其他中控或分析系
☐	ThreatIntelligen	gzy2001	威胁情报联动
☐	UserCounter	Simpower	用户体验计划，统计本软件功能使用频率，启用本插件可以帮助我们做好产品！

2-实时的诊断及多维联动

■ 端口扫描感知插件的图形界面 及 告警窗口



根据远端尝试接入的连接信息(如：敏感端口列表，连接频率)，判断自己是否被扫描

2-实时的诊断及多维联动

■ 端口扫描感知的配置脚本设计

```
//探测端口列表
port=21,23,135,137,138,139,445,1433,3306,3389
//探测模式，0:全部都扫了才告警 1:只要扫描其中一个就告警，2:扫两个...依此类推
mode=1
//白名单
whiteList=
sameIPRemindTime=3
MaxRecord=500
silence=0
```

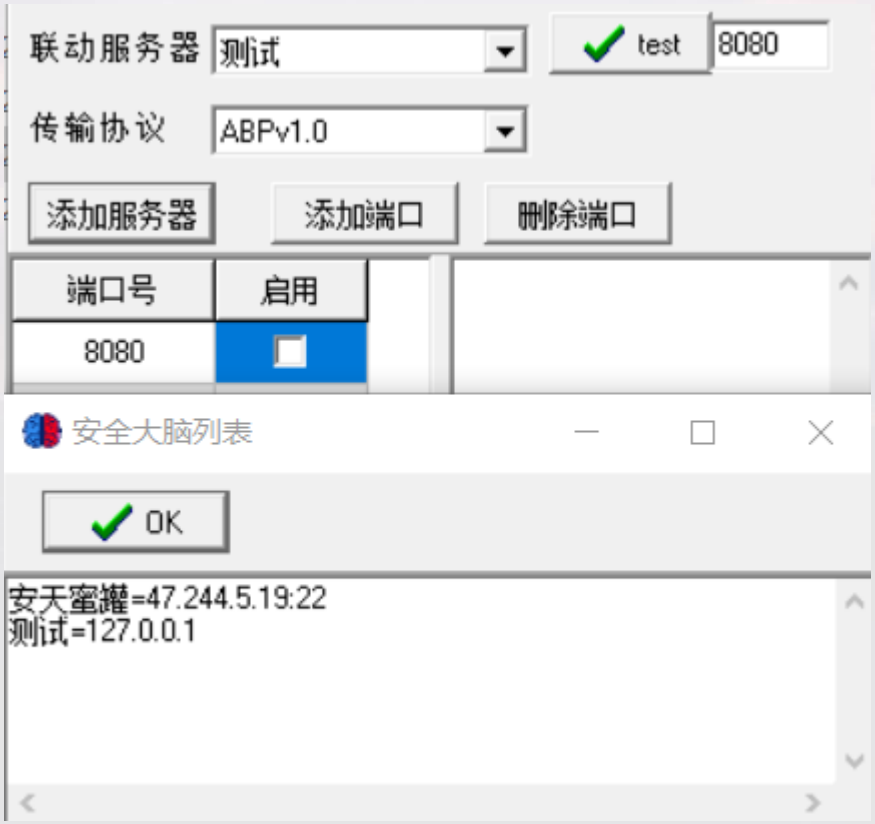
2-实时的诊断及多维联动

- 端口扫描感知的日志记录

远端IP　　扫描时间，被扫描的本机端口
192.168.8.200=2022-01-08 19:10:24,137
……

2-实时的诊断及多维联动

■ 蜜罐联动



通过模拟端口，将首包发送至蜜罐，用于诊断是否自己正在被攻击，通过自定义的协议建立代理，将攻击方数据透明转发至蜜罐，进行下一步的追踪捕获等

2-实时的诊断及多维联动

■ 流量检测系统联动，如:ptd，pas等

配置文件：cfg.txt

ip=127.0.0.1
port=8091

与流量检测系统的联动，
使得流量检测系统能够定位到异常流量
所**对应的样本**，方便取证。

■ 报文标准:

- 五元组+程序路径+时间(精确到毫秒)。
- 五元组对应程序路径如果没有变动，则只发一次，如果后面对应关系有了变化，则再次发送。

这样设计的好处是尽可能减少额外流量，每个应用只需要多出几十个字节即可完成联动。

2-实时的诊断及多维联动

- 威胁情报联动：

根据IP，c2等

- 与赛博超脑联动：

通过获取关键网络封包(如：首包)，并将其发送至赛博超脑，用于分析，并传回结果，决定是否封锁异常进程。

……

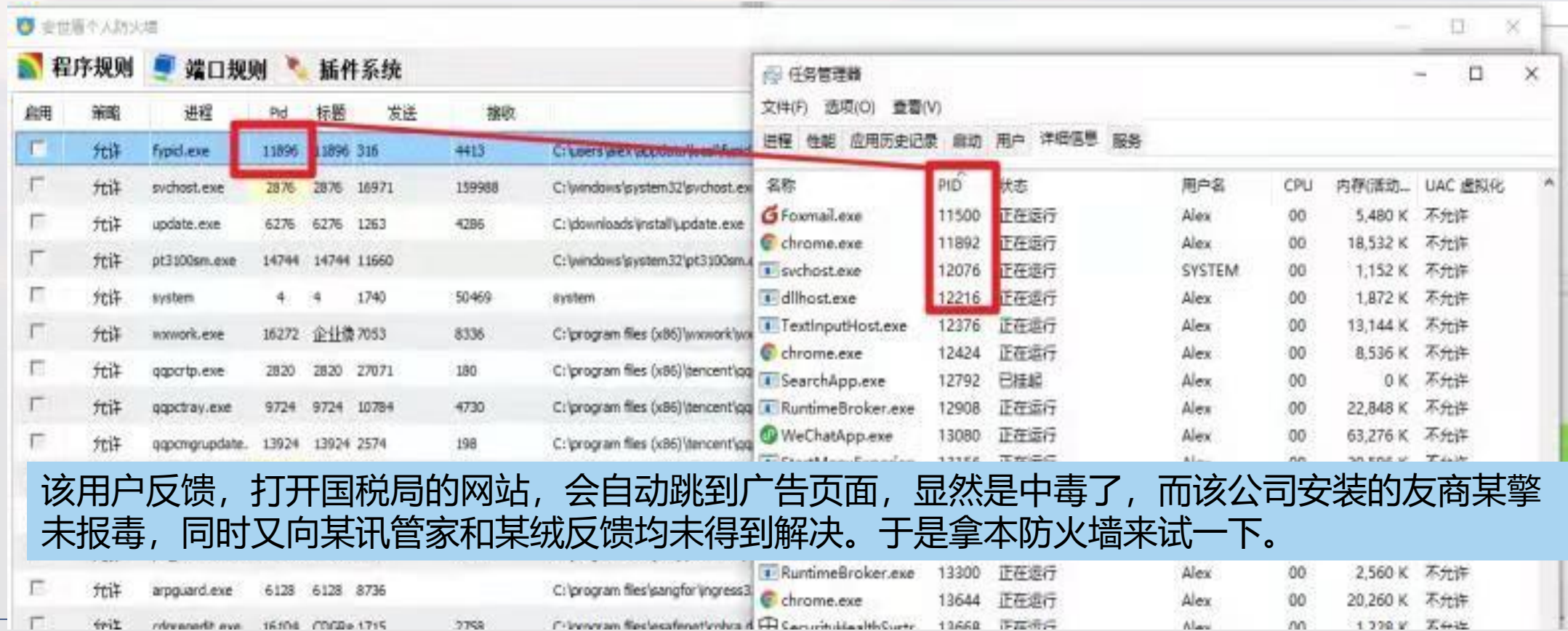
03 实际案例与插件体系

用户实践截图、代码演示

3-实际案例与插件体系

■ 某电力行业国企发现病毒并清理

来自用户的截图：任务管理器看不到该进程ID，而防火墙却看到ID11896正在联网，发送和接收数据。



该用户反馈，打开国税局的网站，会自动跳到广告页面，显然是中毒了，而该公司安装的友商某擎未报毒，同时又向某讯管家和某绒反馈均未得到解决。于是拿本防火墙来试一下。

3-实际案例与插件体系

■ 某电力行业国企发现病毒并清理

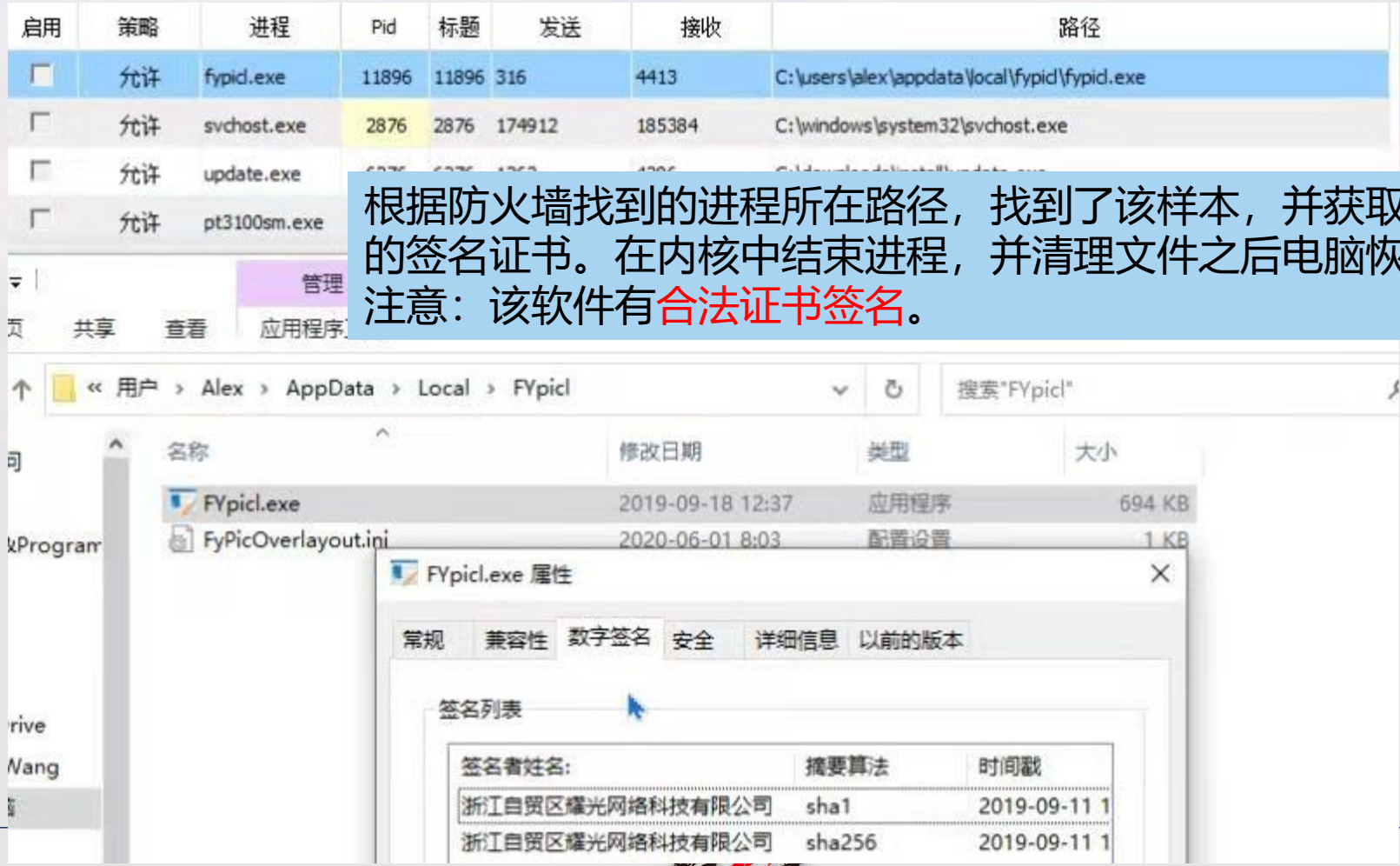
- 用户试图通过命令行结束该隐藏进程，然而却提示没有找到进程11896。

- 用户在某绒论坛的反馈（未得到解决）：

  http://bbs.huorong.cn/forum.php?mod=viewthread&tid=69678&highlight=%E7%96%91%E4%BC%BC&mobile=2

3-实际案例与插件体系

■ 某电力行业国企发现病毒并清理

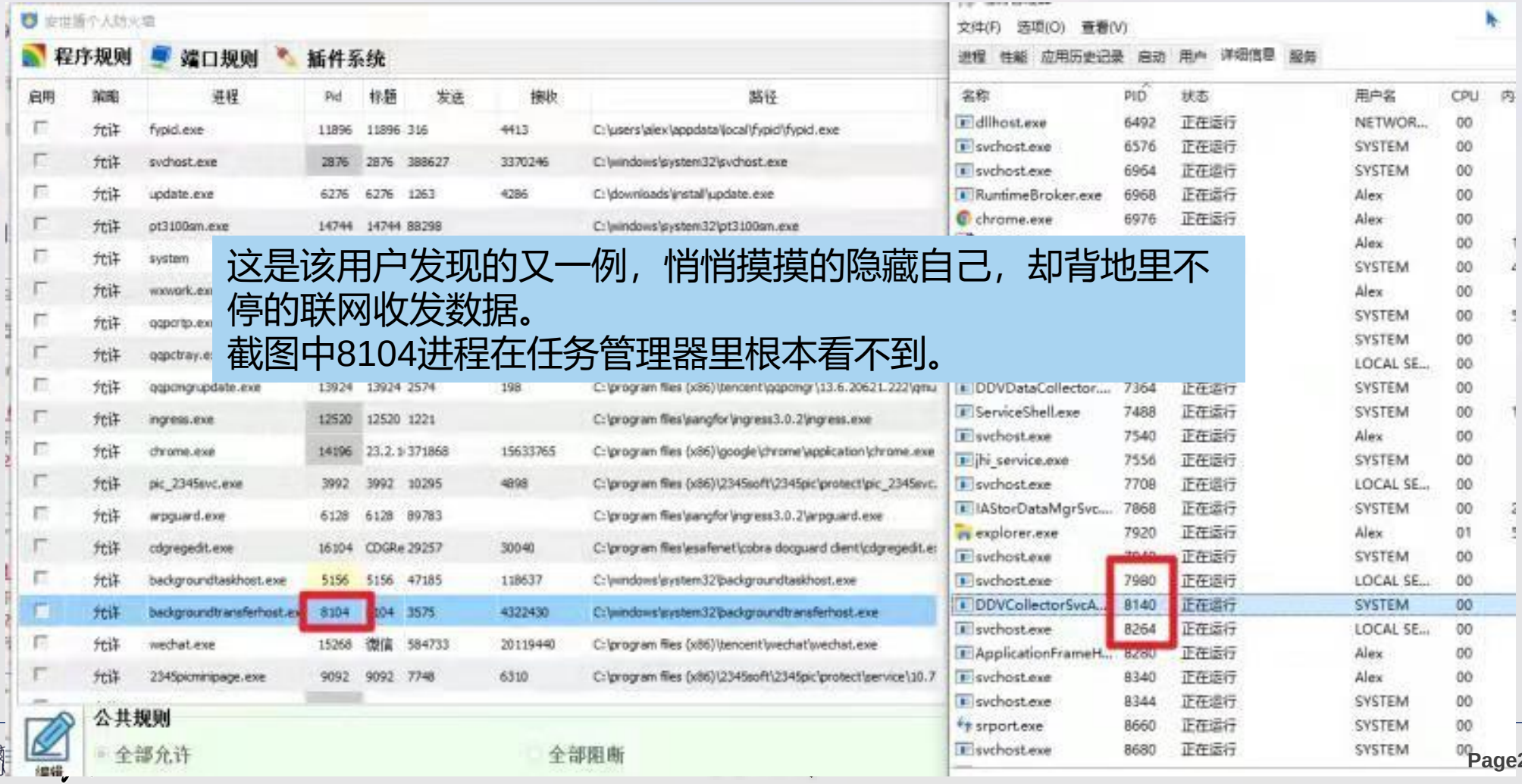


根据防火墙找到的进程所在路径，找到了该样本，并获取了该样本的签名证书。在内核中结束进程，并清理文件之后电脑恢复正常。

注意：该软件有合法证书签名。

3-实际案例与插件体系

- 某电力行业国企发现病毒并清理



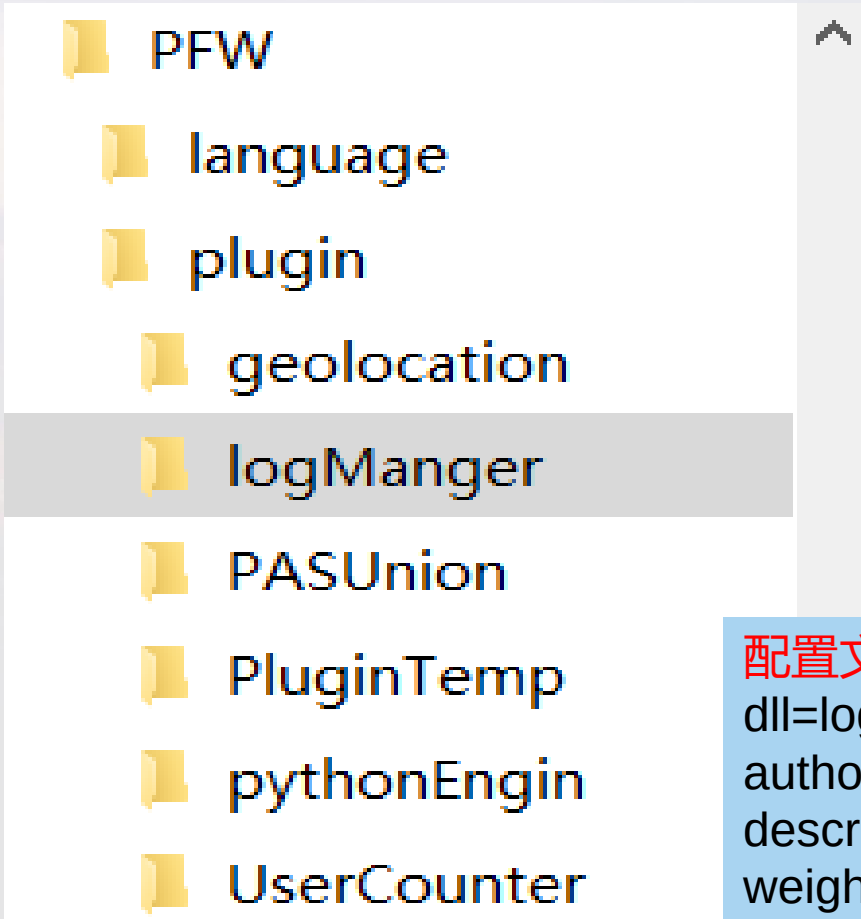
这是该用户发现的又一例，悄悄摸摸的隐藏自己，却背地里不停的联网收发数据。
截图中8104进程在任务管理器里根本看不到。

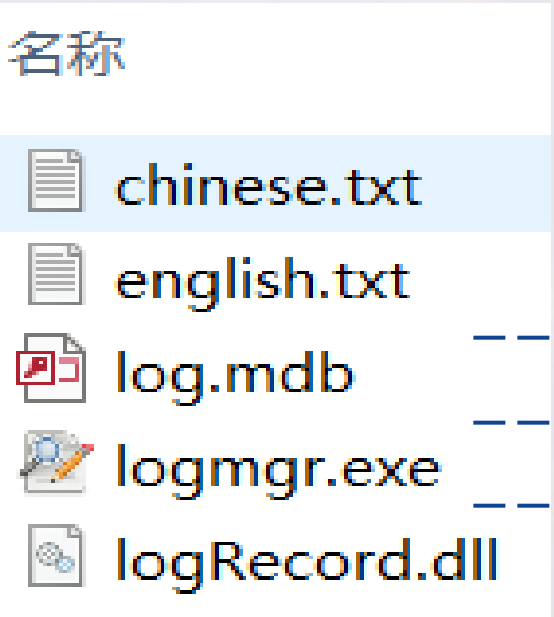
3-实际案例与插件体系

- 对Python的支持

3-实际案例与插件体系

- 插件的目录结构





配置文件

插件所需的其他文件

图形界面

插件主程序

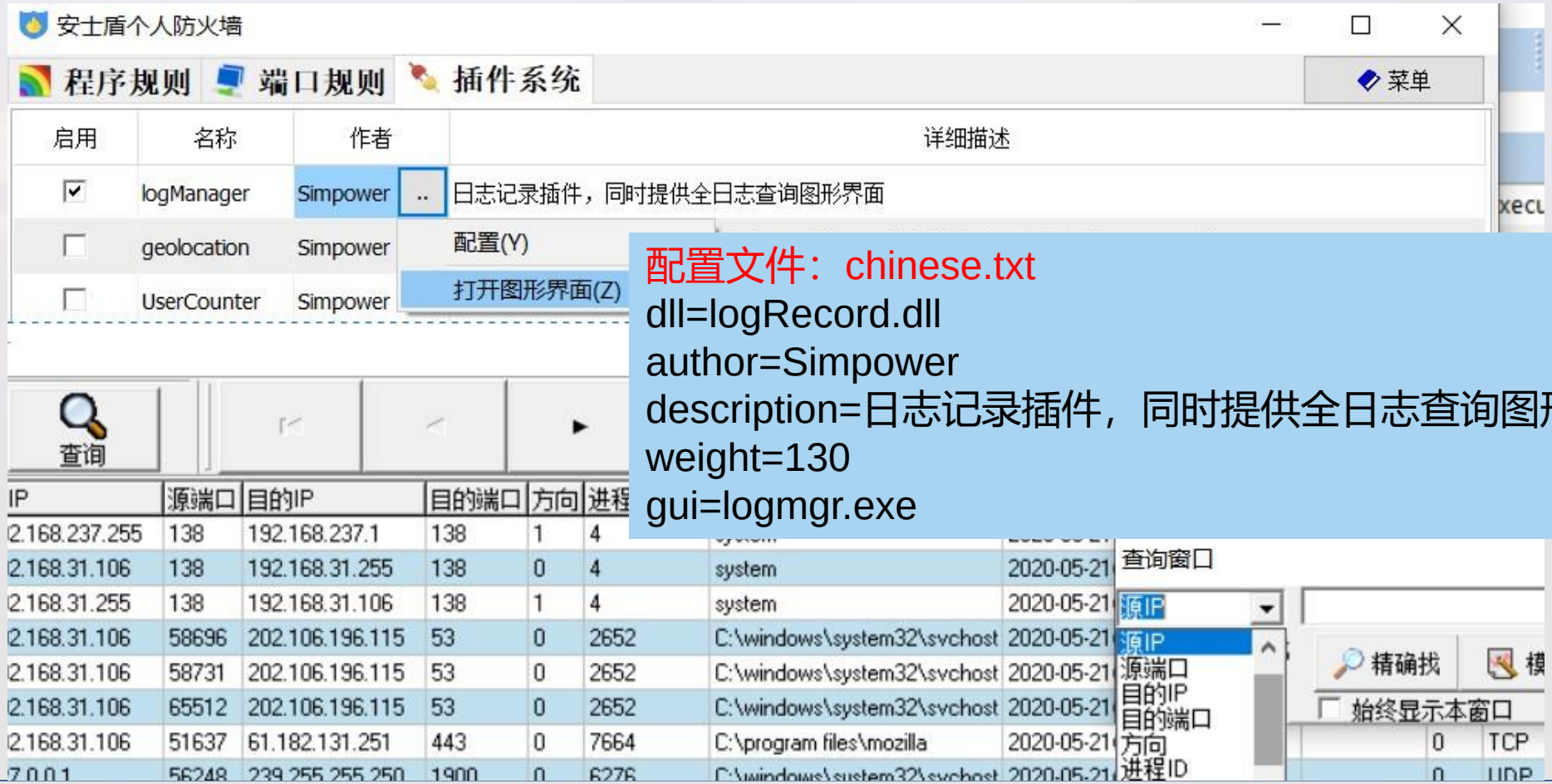
配置文件：chinese.txt

```
dll=logRecord.dll
author=Simpower
description=日志记录插件，同时提供全日志查询图形界面
weight=130
gui=logmgr.exe
```

权重越大优先级越高

3-实际案例与插件体系

- 插件在界面上的显示情况



配置文件：chinese.txt

```
dll=logRecord.dll
author=Simpower
description=日志记录插件，同时提供全日志查询图形界面
weight=130
gui=logmgr.exe
```

3-实际案例与插件体系

■ 三个开放接口，给开发者，用户可以按需扩展

- #define DLL_EXPORT __declspec(dllexport)

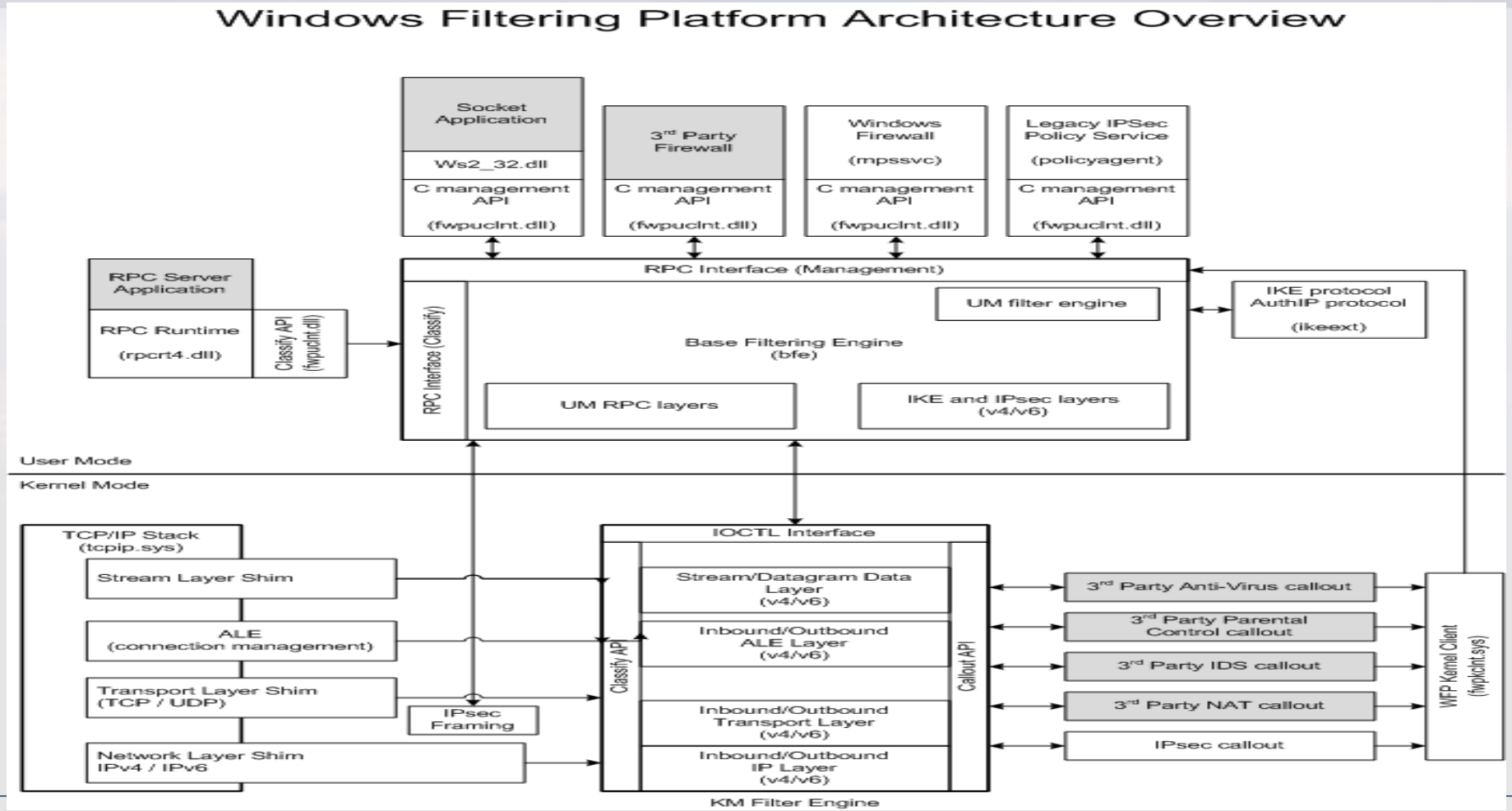
- void DLL_EXPORT __stdcall init(int tid);//初始化，启用插件时调用，主线程执行
- void DLL_EXPORT __stdcall uninit(int tid);//反初始化，取消插件时调用，主线程执行
- void DLL_EXPORT __stdcall recvLog(int tid, const LPCSTR log);//接收网络日志，独立线程

04 由现有系统架构带来的展望

Wfp,反-反沙箱，引导程序维护，链接与加载，沙丁鱼系列

4-由现有系统架构带来的展望

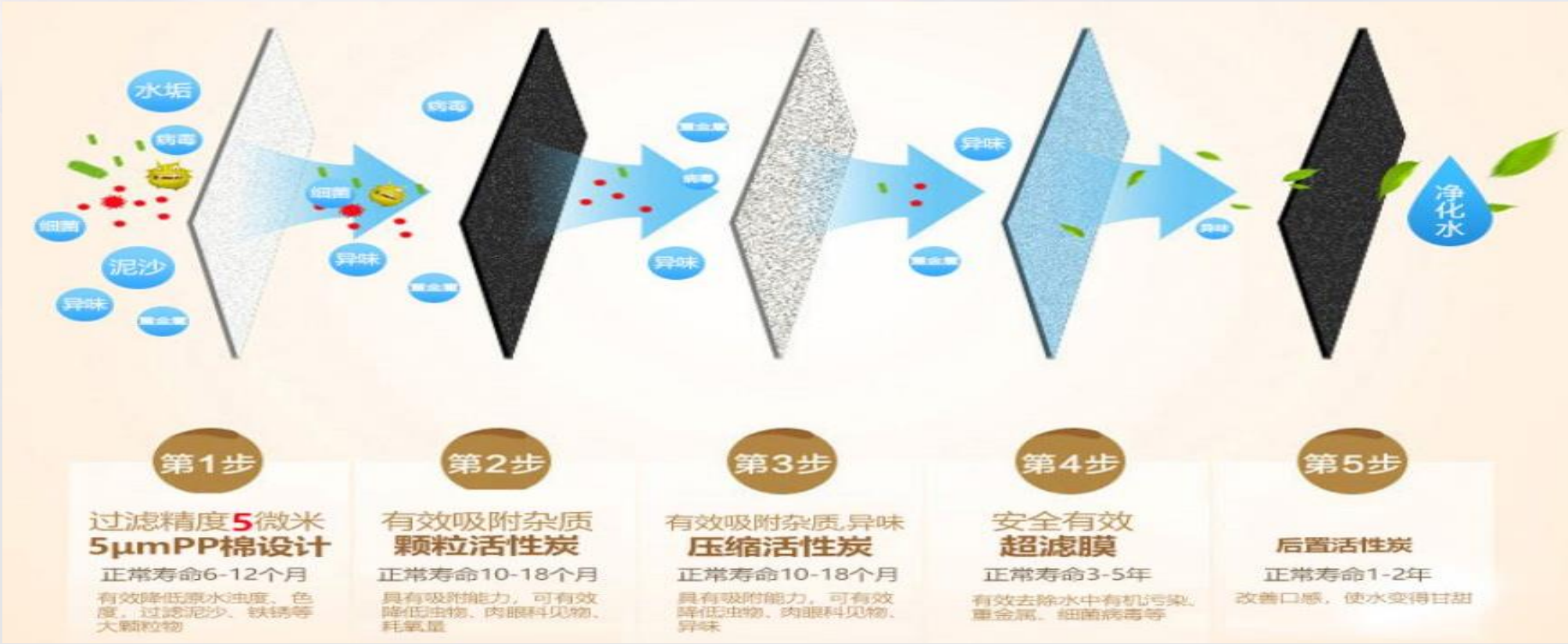
- WFP简介



复杂度增加，往往带来更大攻击面

4-由现有系统架构带来的展望

- 过滤驱动的核心，shims（垫片）和 Callout（呼出接口，我觉得用注册回调函数更贴切）



4-由现有系统架构带来的展望

■ 于是取其精华，统一接口，打造跨平台的内核级安全框架

好处：

1.摆脱现有框架(恶意代码也能用)，我们要绕到更底层，实现更深度的检测。

2.一套程序直接用在windows/linux/Android/鸿蒙，跨平台，低维护成本。

3. 加固操作系统内核，贯穿引导到结束整个生命周期，对抗一些APT，rootkit，如暗云3之类的恶意代码。

4.关于我们操作系统缺芯少魂的问题，通过核心组件的实现，形成若干突破点之一。

4-由现有系统架构带来的展望

- 面临的问题。

内核保护机制：

Patch guard，hyper guard等

改内核蓝屏

4-由现有系统架构带来的展望

- 解决方法：

◆ 引导系统介入，一开始就取得CPU的全部资源，调优配置，可在ring -1进行监控，同时能够反-反沙箱，反-反取证。

打造引导程序链接器：vc->coff->bootloader，不同FS调整偏移

植入引导加载器：载入自定义FS，自定UEFI安全模块等

4-由现有系统架构带来的展望

■ 思考：

一些底层的东西也少有人做，比如编译器链接器，当今商用IDE所生成的可执行文件不断膨胀，一旦出问题将影响成千上万的软件。

我们的链接器将vc单独编译的多个obj链接出的文件会比自带链接出的小很多，需要进一步研究它多出来的是什么。

4-由现有系统架构带来的展望

■ 沙丁鱼系列-研究过程中产生的工具 STDN.TECH下载

除了防火墙之外，一系列小工具，简单介绍

- ActionScope(取可执行文件的行为)
- certiScope(提取可执行文件中的资源证书等)
- VirtualMatrix(一系列虚拟化工具集)
- FileRescuer(恢复误删除的文件和目录等)
- MMWarpIn(直接读写物理内存，从内核投放代码到进程等)

4-由现有系统架构带来的展望

■ 沙丁鱼系列-研究过程中产生的工具 STDN.TECH下载

ActionScope(取可执行文件的行为)

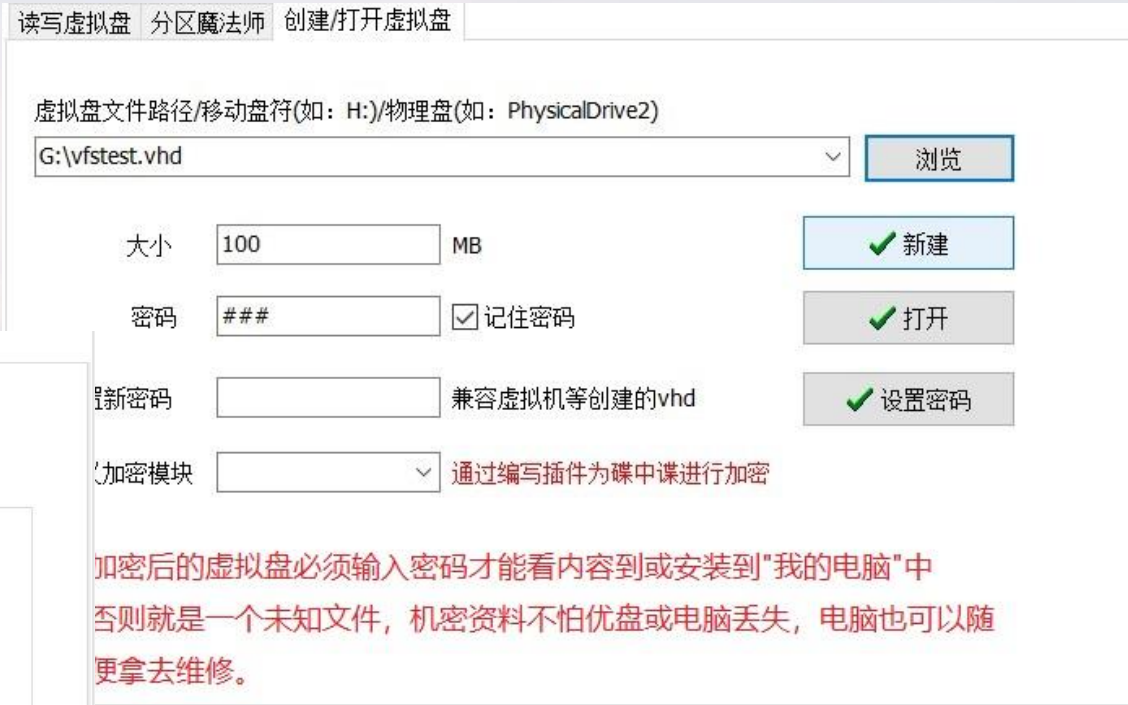
ID	CNName	Details	LEVEL	returnAddr	ACTION_DESCRIPTION	ApiName	pro
19	创建窗口	{"dwExStyle":"0","lpClassName":"0x76c650e4","lpWindowNa	LEVEL_1	0x76c7a216	CREATE_WINDOW	CreateWindowExW	not
20	创建窗口	{"dwExStyle":"907644","lpClassName":"0x76c66560","lpWinc	LEVEL_1	0x76c8303f	CREATE_WINDOW	CreateWindowExW	not
21	获取系统信息（处理器版本、处理器类型等）	{"lpSystemInfo":"905708"}	LEVEL_1	0x6c94e8dc	HOSTINFO.SYSTEMINFO	GetSystemInfo	not
22	获取系统信息（处理器版本、处理器类型等）	{"lpSystemInfo":"899216"}	LEVEL_1	0x6c94e81d	HOSTINFO.SYSTEMINFO	GetSystemInfo	not
23	获取当前用户的界面使用的语言	""	LEVEL_0	0x0ff3a942	HOSTINFO.UI_LANGUAGE	GetUserDefaultUILanguage	not
24	获取当前用户的界面使用的语言	""	LEVEL_0	0x76f90fe8	HOSTINFO.UI_LANGUAGE	GetUserDefaultUILanguage	not
25	获取系统信息（处理器版本、处理器类型等）	{"lpSystemInfo":"1769335772"}	LEVEL_1	0x6958df7b	HOSTINFO.SYSTEMINFO	GetSystemInfo	not
26	获取系统信息（处理器版本、处理器类型等）	{"lpSystemInfo":"263575852"}	LEVEL_1	0x710519a2	HOSTINFO.SYSTEMINFO	GetSystemInfo	not
27	获取当前用户的界面使用的语言	""	LEVEL_0	0x76f8d3da	HOSTINFO.UI_LANGUAGE	GetUserDefaultUILanguage	not
28	堆喷射	action:heapSpray	LEVEL_5	0x0a808944	EXPLOIT.HEAPSPRAY	exploit	not
29	获取系统版本	{"lpVersionInformation":"313306696"}	LEVEL_1	0x129b21d0	HOSTINFO.VERSION	GetVersionExW	not
30	获取系统信息（处理器版本、处理器类型等）	{"lpSystemInfo":"310948980"}	LEVEL_1	0x53add781	HOSTINFO.SYSTEMINFO	GetSystemInfo	not
31	获取系统版本	{"lpVersionInformation":"310949060"}	LEVEL_1	0x53add7a5	HOSTINFO.VERSION	GetVersionExW	not
32	获取系统信息（处理器版本、处理器类型等）	{"lpSystemInfo":"310944176"}	LEVEL_1	0x53add781	HOSTINFO.SYSTEMINFO	GetSystemInfo	not
33	获取系统版本	{"lpVersionInformation":"310944256"}	LEVEL_1	0x53add7a5	HOSTINFO.VERSION	GetVersionExW	not
34	堆喷射	action:heapSpray	LEVEL_5	0x0a80ab64	EXPLOIT.HEAPSPRAY	exploit	not
35	堆喷射	action:heapSpray	LEVEL_5	0x0a80b344	EXPLOIT.HEAPSPRAY	exploit	not
36	堆喷射	action:heapSpray	LEVEL_5	0x0a80b654	EXPLOIT.HEAPSPRAY	exploit	not
37	堆喷射	action:heapSpray	LEVEL_5	0x0a80bf14	EXPLOIT.HEAPSPRAY	exploit	not
38	获取系统版本	{"lpVersionInformation":"310953224"}	LEVEL_1	0x0b81c9b7	HOSTINFO.VERSION	GetVersionExW	not
39	创建窗口	{"dwExStyle":"0","lpClassName":"0x76c650e4","lpWindowNa	LEVEL_1	0x76c7a216	CREATE_WINDOW	CreateWindowExW	not
40	创建窗口	{"dwExStyle":"907644","lpClassName":"0x76c66560","lpWinc	LEVEL_1	0x76c8303f	CREATE_WINDOW	CreateWindowExW	not
41	创建窗口	{"dwExStyle":"0","lpClassName":"0x76c650e4","lpWindowNa	LEVEL_1	0x76c7a216	CREATE_WINDOW	CreateWindowExW	not
42	创建窗口	{"dwExStyle":"907644","lpClassName":"0x76c66560","lpWinc	LEVEL_1	0x76c8303f	CREATE_WINDOW	CreateWindowExW	not

4-由现有系统架构带来的展望

■ 沙丁鱼系列-研究过程中产生的工具 STDN.TECH下载

4-由现有系统架构带来的展望

■ 沙丁鱼系列-研究过程中产生的工具 STDN.TECH下载



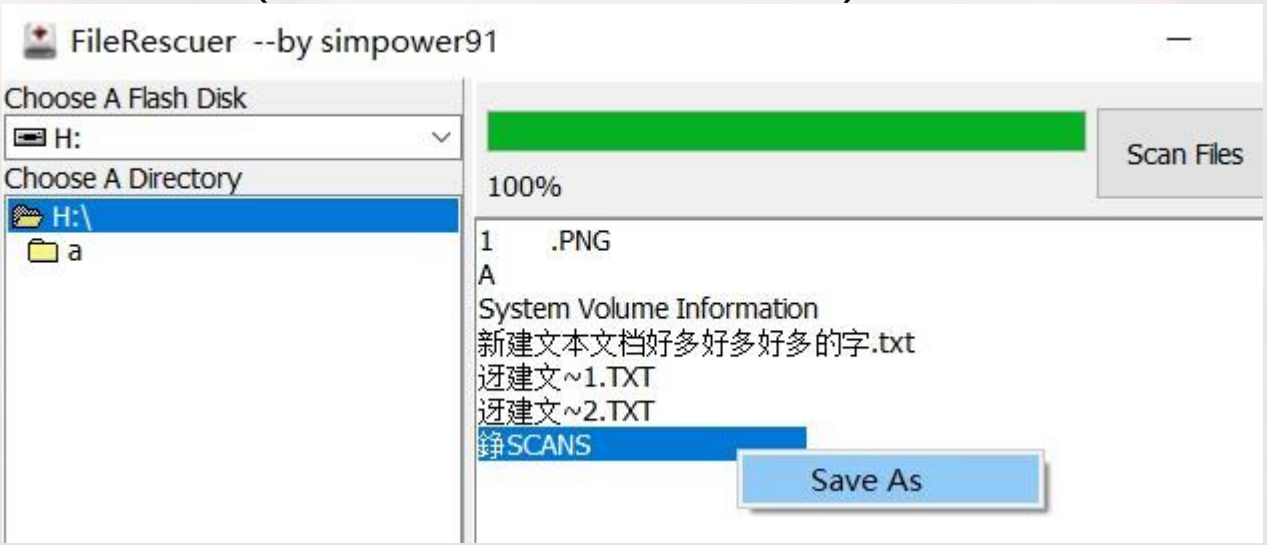
VirtualMatrix(一系列虚拟化工具集)

分区之间有一些空隙，文件系统不可见，可以藏rootkit进去。这个可以用于取证。

4-由现有系统架构带来的展望

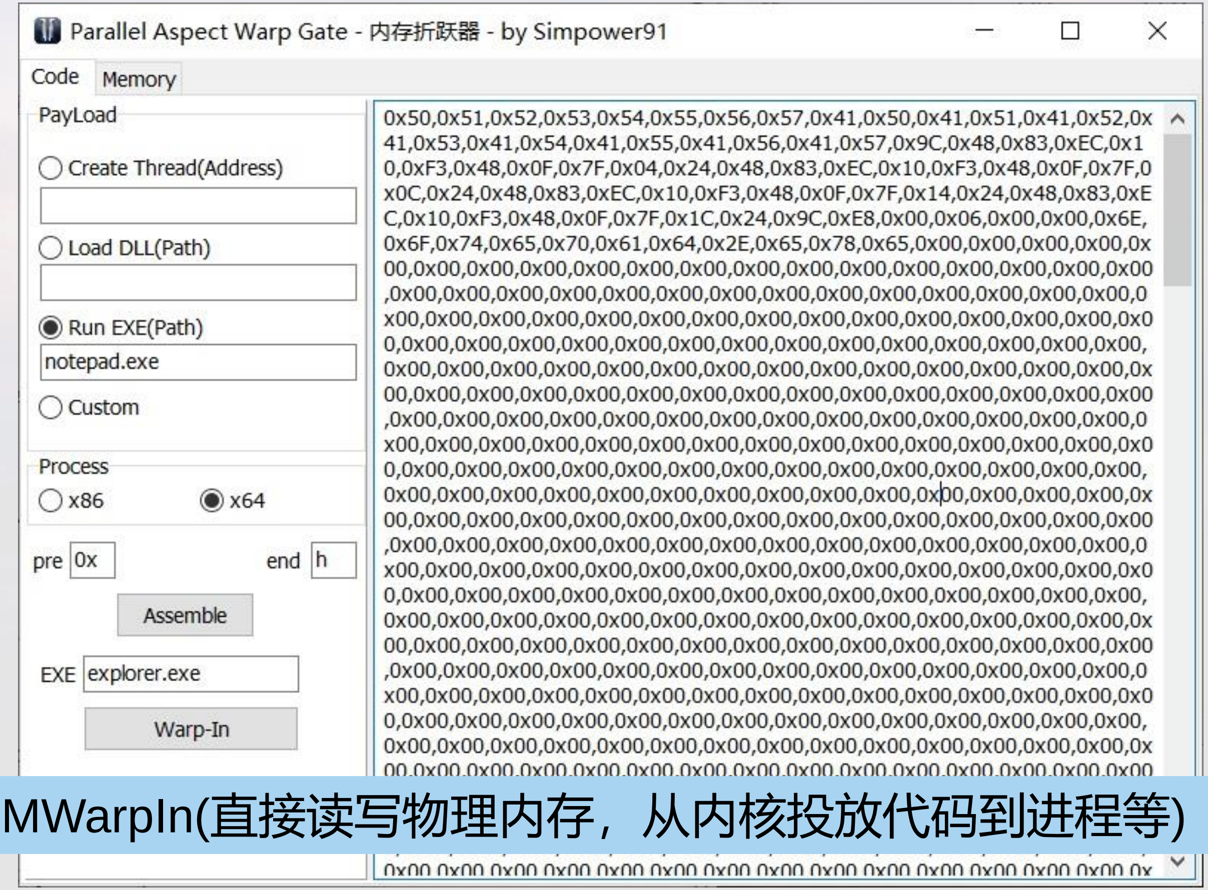
■ 沙丁鱼系列-研究过程中产生的工具 STDN.TECH下载

FileRescuer(恢复误删除的文件和目录等)



4-由现有系统架构带来的展望

- 沙丁鱼系列-研究过程中产生的工具 STDN.TECH下载



MMWarpIn(直接读写物理内存，从内核投放代码到进程等)

谢谢大家

安天冬训营 wtc.antiy.cn