



01 回顾：攻防往事 03 对抗精准检测

02 内存检测策略
剖析 04 对抗持续检测

CONTENTS

/01

回顾：攻防往事



Q: BYOVD技术已经成熟且武器化, 且EDR检测能力逐渐完善, 为什么还要研究内存检测逃逸相关的技术?

意义何在?



深信服深蓝攻防实验室
SANGFOR DEEP BLUE

ADCONF

高压环境下的EDR致盲目的不再止步于
“一键卸载 安全卫士/电脑管家”



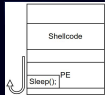
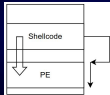
- 有以下几种场景是不能完全依赖BYOVD的, 我们的木马要被迫与完全体EDR共存。
 - 加载驱动的黑白名单。
 - EDR的R3/R0组件中有类暗桩的存在。
 - 与EDR控制端通信的模块直接做在了R0里面, 卸载驱动等同于直接切断通信。
 - 部分场景下, 根本没有系统的高权限且无法提权。
 - 机器重启后, 被致盲的EDR会恢复正常, 权限维持的木马必须有一定存活能力。
 -



- 两个阶段

- 木马Loader载入Shellcode并释放植入体的整个初始化过程需要逃逸EDR的检测;
- 植入体线程在“运行——休眠”的过程中长期与EDR共存而不被检测到。

回顾：一个标准Loader的工作原理



- R3: 借助API HOOK拦截敏感函数调用, 跟踪参数和返回值。

- R0: 利用底层组件监控敏感调用。

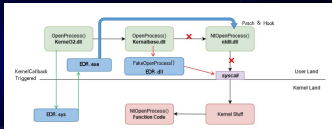
- 内核回调-Windows / 内核探针 (Kprobes) -Linux

- 内核钩子

- ETW

- 硬件辅助

-



/02

内存检测策略剖析

内存检测

- EDR的内存取证功能可以收集和分析内存中的数据，重建事件的时间线和上下文，包括分析内存中的残留数据、运行时状态和其他关键信息。

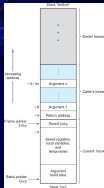
- 关键词1 - 栈回溯：

- 32位下可以很方便地找到每一个函数的返回地址和栈帧。
- 64位默认开启了FPO优化，不保存RBP到栈上。

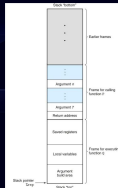
- 关键词2 - “随机”内存扫描。

- 标记的RWX内存页、高熵内存页
- 由栈回溯发现的可疑地址或调用链触发
- 线程休眠时期

-



```
push    ebp
mov     ebp, esp
sub     esp, 30h
call   ds:GetTickCount
mov     ecx, 36AAh
```



```
sub     rsp, 68h
call   cci__lap_GetTickCount
mov     ecx, 36AAh
xor     edx, edx
mov     r8d, 5Ch ; '\
div     ecx
```


两类扫描策略偏好

策略① - 偏向精准检测

- - R3 Hook, 初筛敏感API调用;
- - 检测SYSCALL;
- - 触发规则就进行栈回溯;
- - 重点扫描可疑地址对应的内存。

策略② - 偏向持续检测

- - 监控线程状态 (包含线程的堆栈、运行状态等);
- - 对私有内存页进行扫描 (通常在线程休眠时);
- - 搜索高熵区域、RWX等区域, 重点标记;
- - 对重点标记区提升扫描频率或重点监控该区域的读写、访问行为, 直到探测到植入体相关特征。

/03

对抗方案

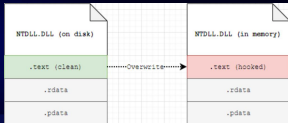


- 绕过R3 Hook检测
 - SYSCALL
 - Unhook
- 混淆栈回溯检测
 - 栈帧伪造

- Unhook

- 方法1: 将磁盘上“干净”的dll映射到当前进程中, 读取.text节并覆盖被hook的dll的.text节。
- 方法2: 创建一个白名单进程, 读取其未被hook的dll, 覆盖当前进程中dll的.text节。
- 方法3: 没有白名单程序的情况下, 在新进程启动加载完成dll时将其挂起, 保留其中“干净”的dll快照, 覆盖当前进程dll的.text节。

-



Unhook_Test.exe - PID: 7528 - 模块: ntdll.dll - 线程: 主线程 5952 - x64dbg [管理员]

文件(F) 视图(V) 调试(D) 跟踪(M) 插件(P) 收藏夹(I) 选项(O) 帮助(H) Mar 8 2024 (TitanEngine)

CPV 日志 笔记 断点 内存布局 调用堆栈 寄存器

2543F955	CD 2E	int 2E
2543F957	C3	ret
2543F958	0F1F8400 00000000	nop dword ptr ds:[rax+rax],eax
2543F960	E9 52001E9C	jmp 7FF92543F963
2543F961	CC	int3
2543F966	CC	int3
2543F967	CC	int3
2543F968	F60425 0803FE7F 01	test byte ptr ds:[7FF92543F968],1
2543F970	75 03	jne ntdll.7FF92543F975
2543F972	0F05	syscall
2543F974	C3	ret
2543F975	CD 2E	int 2E
2543F977	C3	ret
2543F978	0F1F8400 00000000	nop dword ptr ds:[rax+rax],eax
2543F980	4C:88D1	mov r30,rcx
2543F983	8B 19000000	mov eax,19
2543F988	F60425 0803FE7F 01	test byte ptr ds:[7FF92543F988],1
2543F990	75 03	jne ntdll.7FF92543F995
2543F992	0F05	syscall

Unhook_Test.exe - PID: 7528 - 模块: ntdll.dll - 线程: 主线程 5952 - x64dbg [管理员]

文件(F) 视图(V) 调试(D) 跟踪(M) 插件(P) 收藏夹(I) 选项(O) 帮助(H) Mar 8 2024 (TitanEngine)

CPV 日志 笔记 断点 内存布局 调用堆栈 寄存器

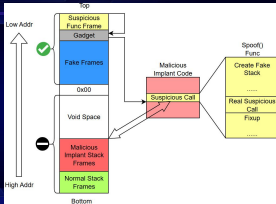
2543F955	CD 2E	int 2E
2543F957	C3	ret
2543F958	0F1F8400 00000000	nop dword ptr ds:[rax+rax],eax
2543F960	4C:88D1	mov r30,rcx
2543F963	8B 19000000	mov eax,19
2543F968	F60425 0803FE7F 01	test byte ptr ds:[7FF92543F968],1
2543F970	75 03	jne ntdll.7FF92543F975
2543F972	0F05	syscall
2543F974	C3	ret
2543F975	CD 2E	int 2E
2543F977	C3	ret
2543F978	0F1F8400 00000000	nop dword ptr ds:[rax+rax],eax
2543F980	4C:88D1	mov r30,rcx
2543F983	8B 19000000	mov eax,19
2543F988	F60425 0803FE7F 01	test byte ptr ds:[7FF92543F988],1
2543F990	75 03	jne ntdll.7FF92543F995
2543F992	0F05	syscall



策略①-对抗方案

• 栈帧伪造

- 1. 重写函数;
- 2. 隐藏原始栈帧;
- 3. 部署一个假栈;
- 4. 部署一个Gadget Frame;
- 5. SYSCALL;
- 6. 利用Gadget修复栈帧并返回。



未伪造栈帧效果

```
Stack - thread 44260
Name
0  opend-shellcode-hidden-tool.exe!main+0x2df
1  opend-shellcode-hidden-tool.exe!invoke_main+0x29
2  opend-shellcode-hidden-tool.exe!__scrt_common_main_ash...
3  opend-shellcode-hidden-tool.exe!__scrt_common_main+0xe
4  opend-shellcode-hidden-tool.exe!mainCRTStartup+0xe
5  kernel32.dll!BaseThreadInitThunk+0xd
6  ntdll.dll!UserThreadStart+0x28
```

进入
NtAllocateVirtualMemory()

```
Stack - thread 44260
Name
0  ntdll.dll!NtAllocateVirtualMemory
1  kernel32.dll!VirtualAlloc+0x48
2  opend-shellcode-hidden-tool.exe!main+0x2e0
3  opend-shellcode-hidden-tool.exe!invoke_main+0x29
4  opend-shellcode-hidden-tool.exe!__scrt_common_main_ash...
5  opend-shellcode-hidden-tool.exe!__scrt_common_main+0xe
6  opend-shellcode-hidden-tool.exe!mainCRTStartup+0xe
7  kernel32.dll!BaseThreadInitThunk+0xd
8  ntdll.dll!UserThreadStart+0x28
```

```
00007FF8296304C3 mov     eax, 18h
00007FF8296304C8 test   byte ptr [7FFE0308h], 1
00007FF8296304D0 jne    00007FF8296304D5  [用时: 1ms]
00007FF8296304D2 syscall
00007FF8296304D4 ret
00007FF8296304D5 int    2Eh
00007FF8296304D7 ret
00007FF8296304D8 nop    dword ptr [rax+rax]
00007FF8296304E0 mov    r10, rcx
00007FF8296304E3 mov    eax, 19h
00007FF8296304E8 test   byte ptr [7FFE0308h], 1
```

栈帧伪造效果

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
4 StackPool.swampool+0x00
5 StackPool.swampool+0x00
6 StackPool.swampool+0x00
```

进入Spooof()

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
4 StackPool.swampool+0x00
5 StackPool.swampool+0x00
6 StackPool.swampool+0x00
```

抬栈

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
```

PUSH 0

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
```

伪造假栈+Gadget

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
```

跳转回原位

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
4 StackPool.swampool+0x00
5 StackPool.swampool+0x00
6 StackPool.swampool+0x00
```

返回Gadget, 跳转并修复栈帧

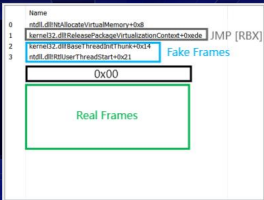
```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
```

```
00007FF8296304E8 nop dword ptr [rax+rax]
00007FF8296304E9 mov r10,rcx
00007FF8296304EA mov eax,15h
00007FF8296304EB test byte ptr [7FF8296304EB],1 已启用JOP -> jmp
00007FF8296304EC jnz 00007FF8296304E6
00007FF8296304ED syscall
```

真实函数调用

```
Stack - thread 47E76
Name
0 StackPool.swampool+0x00
1 StackPool.swampool+0x00
2 StackPool.swampool+0x00
3 StackPool.swampool+0x00
```

```
128 mov r10,rcx
129 mov rax,[rdi + 72]
130
131 jmp r11 已启用JOP -> jmp
```





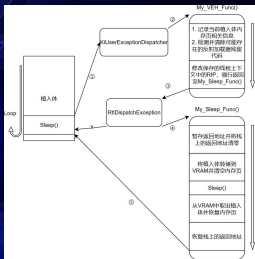
- 防止反射加载器特征被扫描识别
 - 降低探测到植入体休眠指令相关返回地址的概率
 - 植入体“隐身”
-
- 参考案例-1: <https://github.com/mgeeky/ThreadStackSpoofers/tree/master>; 其利用hook Sleep()截断栈帧, 以对抗线程休眠期间的栈回溯探测。
 - Inline Hook ✘
 - 参考案例-2: <https://github.com/vxunderground/VXUG-Papers/blob/main/GpuMemoryAbuse.cpp>; 利用CUDA将内存转储至VRAM。
 - CUDA API, 跨平台 ✘

- “无内存特征” Hook
 - 给Sleep()设置硬件断点，等待植入体执行后调用；
 - 自定义一个VEH处理函数并设置最高优先级，承接硬件断点；
 - 准备一个MySleep()方法，实现核心功能。
- VRAM内存转储
 - 对于没有安装任何GPU驱动程序的环境，如无显卡的服务器或未配置虚拟显卡的虚拟机，要安装CPU设备的专用OpenCL Runtime（msi安装包可使用命令一键安装，无需GUI操作）；
 - 在Loader中利用OpenCL的SDK创建一个全局对象，用来存储OpenCL缓冲区句柄和其他临时数据。

VRAM内存转储

工作原理

- 0. 准备工作：创建全局OpenCL内存对象，并设置Sleep()的硬件断点，等待程序调用Sleep()；
- 1. 在自定义VEH中记录当前植入体内存页相关信息并将反射加载器残留内存页释放；
- 2. 修改VEH暂存的线程上下文结构体中的RIP，引导其返回到自定义Sleep()方法；
- 3. 在自定义Sleep()方法中将可疑内存页写入OpenCL内存对象的缓冲区，通过OpenCL库写入VRAM；
- 4. 在自定义Sleep()方法中关闭内存页X权限；
- 5. 在自定义Sleep()方法中暂存返回地址并将真实返回地址覆盖为0x00，截断栈帧；
- 6. 在自定义Sleep()方法中调用真正的Sleep()；
- 7. 在自定义Sleep()方法返回前恢复内存页。
 - 访问OpenCL内存对象的缓冲区，取出之前转储的数据；
 - 重新开启内存页X权限；
 - 恢复暂存的返回地址。
- 8. 正常返回到植入体代码的CALL Sleep()下一条指令位置，进入下一循环周期。



效果对比



深信服深蓝攻防实验室
SANGFOR DEEP BLUE



休眠线程堆栈

General Statistics Performance Threads Tokens Modules Memory Environment Handles GPU Disk and Network Comm...

TD	CPU	Cycle data	Start address
12804			msvcrt!malloc@77e41480
12812			msvcrt!calloc@77e41480
21268			msvcrt!calloc@77e41480
21244			msvcrt!calloc@77e41480
20128			msvcrt!calloc@77e41480
17284			msvcrt!calloc@77e41480
3088			msvcrt!calloc@77e41480
1788			!kernel!_pfn@77d80000
6388			!kernel!_pfn@77d80000
2728			!kernel!_pfn@77d80000
3408			!kernel!_pfn@77d80000
3272			!kernel!_pfn@77d80000
2248			!kernel!_pfn@77d80000
2872			!kernel!_pfn@77d80000
12848			!kernel!_pfn@77d80000
12848			!kernel!_pfn@77d80000

Stack - thread 12848

Name

- !kernel!_pfn@77d80000

Open Refresh Close

线程休眠时内存

General Statistics Performance Threads Tokens Modules Memory Environment Handles GPU Disk and Network Comm...

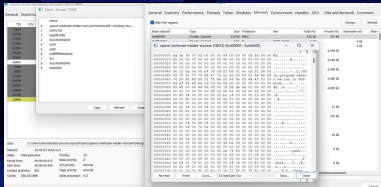
Info flow regions

Base address	Type	Size	Protection	Base	Total KB	Private KB	Shared KB	Share...
80000000	Private - Commit	252 KB	RW		252 KB	252 KB		
8c7f0000	Private - Commit	4 KB	R	8c7f0000	4 KB	4 KB		
8c7f4000	Private - Commit	4 KB	R	8c7f4000	4 KB	4 KB		

C:\Users\administrator\jason...> C:\Users\administrator\jason...

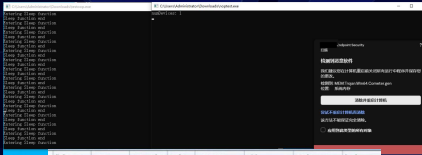
```
numPlatforms: 2
numDevices: 1
0: myExceptionHandler_4 - 0x77f868f2ae78
1: RtlGetLengthWithoutLastFullDllCharacter - 0x77f868687faa
2: RtlFindCharInUnicodeString - 0x77f86862e662
3: KThreadExceptionDispatcher - 0x77f86868a38e
4: Sleep - 0x77f86c158670
5: [Unknown Function] - 0x800765
---Entering hooked Sleep() function---
Hook ret address is: 0x800765
Allocation base address of the page: 0000000000000000
Total page size: 00000
Release original mem page.
```

未使用该方案的休眠堆栈和内存



The screenshot displays the Windows Task Manager Performance tab, specifically the Memory section. The left pane shows system statistics, including a task list for 'svchost.exe' with a memory usage of 1.04 GB. The right pane shows the 'Memory' tab with a 'Memory usage' of 9.1 GB. A 'Memory dump' window is open, showing a hex dump of memory contents, with a 'Total size' of 1.04 GB. The dump shows a large amount of zeroed-out memory, indicating that the memory is not being actively used or is in a dormant state.

Hex address	Type	Size	Position	Hex	Total size	Private size	Shareable size	Other
00000000	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000001	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000002	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000003	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000004	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000005	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000006	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000007	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000008	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000009	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000000A	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000000B	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000000C	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000000D	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000000E	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000000F	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000010	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000011	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000012	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000013	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000014	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000015	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000016	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000017	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000018	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000019	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000001A	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000001B	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000001C	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000001D	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000001E	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000001F	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000020	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000021	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000022	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000023	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000024	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000025	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000026	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000027	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000028	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000029	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000002A	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000002B	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000002C	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000002D	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000002E	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000002F	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000030	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000031	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000032	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000033	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000034	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000035	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000036	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000037	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000038	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000039	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000003A	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000003B	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000003C	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000003D	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000003E	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
0000003F	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB
00000040	Private/Commit	1.04 GB	0000	0000	1.04 GB	1.04 GB	0.00 GB	0.00 GB



listener	user	computer	note	process	pid	arch	last
HTTPS	Adminis...	DESKT...		*testloop.exe	4100	x64	379ms
HTTPS	Adminis...	DESKT...		*noplay.exe	8124	x64	14h

ADCONF



深信服深蓝攻防实验室
SANGFOR DEEP BLUE

主场·见真章

THANKS