Toward Revealing Kernel Malware Behavior in Virtual Execution Environments

Chaoting Xuan, John Copeland, Raheem Beyah
Department of Electric and Computer Engineering
Georgia Institute of Technology

09/25/2009

Motivation

- Rootkits are used to cooperate with other malware to accomplish complicate tasks. It's necessary to dissect rootkits to understand the attacking strategies of hackers as a whole.

- Rootkits becomes complex and multi-task oriented. Some of them are protected by anti-reverse-engineering techniques (code obfuscation and packing).

Goal

Comprehensively Revealing rootkit behavior:

1. What kernel functions have been called by a rootkit?

2. What kernel objects have been visited (read/write) by a rootkit?

Related Work

- Rootkit detection, prevention and analysis.
- Single-purpose rootkit analysis systems: Limbo, Panorama, HookFinder, HookMap and K-Tracer.
- Mulitple-purpose rootkit analysis systems: PoKeR and Rkprofiler.

Scope

- Rkproiler just monitors rootkits whose functionalities are carried out through malicious code.

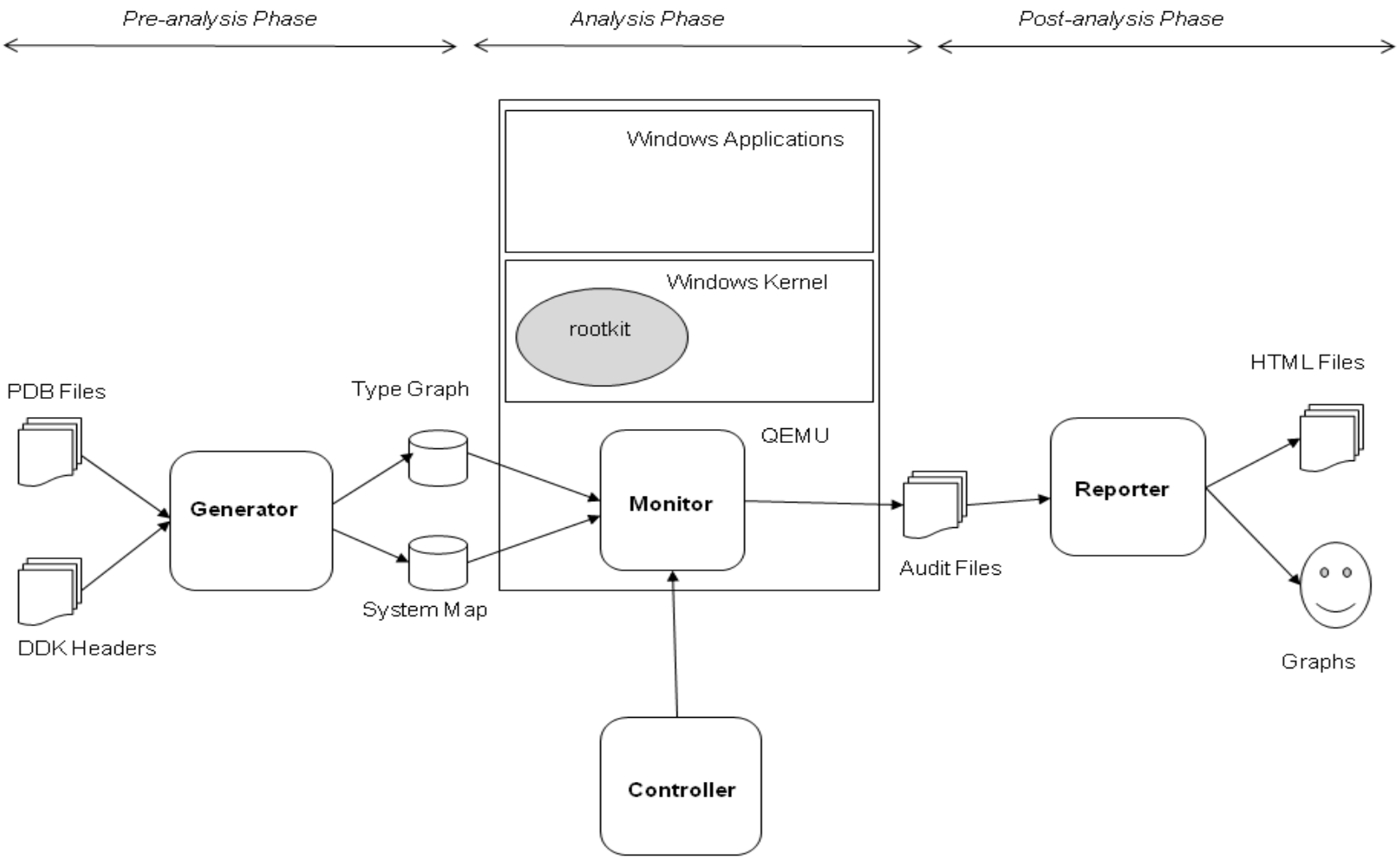
- Others: return-oriented rootkits, hardware-based rootkits like SMM rootkits VMM rootkits.

Challenges

- Identification of malicious kernel code including kernel driver, patches and dynamically generated code.

- Kernel object reverse lookup: given the memory address of a kernel object, identify the corresponding symbols.

Architecture



Changes of Qemu

- Rootkit monitoring takes place at the code translation of Qemu.
- Only malicious kernel instructions are inspected: each translation block of malicious code is changed to single instruction and not cached.
- All other VM instructions are not monitored by Rkprofiler and their code translation processes are not changed.

Malicious Code Identification

- Rule: before loading kernel malware, all kernel code is treated as benign code; after loading kernel malware, newly loaded kernel code is considered malicious.

- Rkprofiler interprets images of benign kernel modules and obtains the relative virtual addresses (RVA) of the code sections. The actual virtual addresses in the RAM are the addition of RVA and base addresses of kernel modules.

Malicious Code Identification (continue)

- A hash table is used to represent the trust code zone (TCZ) that contains the code addresses of all benign module.
- During the analysis, all translation blocks (TB) of kernel code are checked against the TCZ and a TB is malicious if it is not in the TCZ.
- A kernel integrity verifier is built to assure the integrity of TCZ. If a malicious patch is detected TCZ is modified to exclude the patched code.

Function Call Tracking

- Basic idea is to capture the CALL/RET pairs.
- Rkprofiler only monitors the malicious code, so CALL/RET pairs may not be always available.

E2I and I2E Calls

Benign Code (External)

```
M0: Function_A
{
   …
M1:   CALL M3
M2:   XXX
   …
}
```

Malicious Code (Internal)

```
M3: Function_B
{
   …
M4:  RET  M2
}
```

Benign Code (External)

```
M5: Function_C
{
   …
M6:   RET M9
}
```

Malicious Code (Internal)

```
M7: Function_D
{
   …
M8:  CALL  M5
M9:   XXX
   …
}
```

Function Call Tracing (continue)

- We adds two more members to TB data structure to indicate if the last instruction of the TB is CALL or RET and the means to find the corresponding operand (register name or memory address).
- A global pointer is used to always point to the previous translated TB.
- Interrupt gap can break the scheme, but, fortunately, they don't often happen.

Function Parameters

- Rkprofiler monitors the parameters for each identifiable function call, which is necessary for tracking kernel objects.

- The parameters information is acquired by parsing the function declaration in DDK header files and they are stored in an extended system map.

Memory Tagging

- Principle: tracing unknown kernel objects from known kernel objects.
- A memory tag represent one kernel object and contains: tag id, virtual address, type ID, variable name (optional), and parent tag id (optional).
- Tag inferring is a process that a kernel object is derived from another (parent kernel object).
- Two types of kernel object are considered contagious (inferable): pointer (or struct containing pointer) and function.

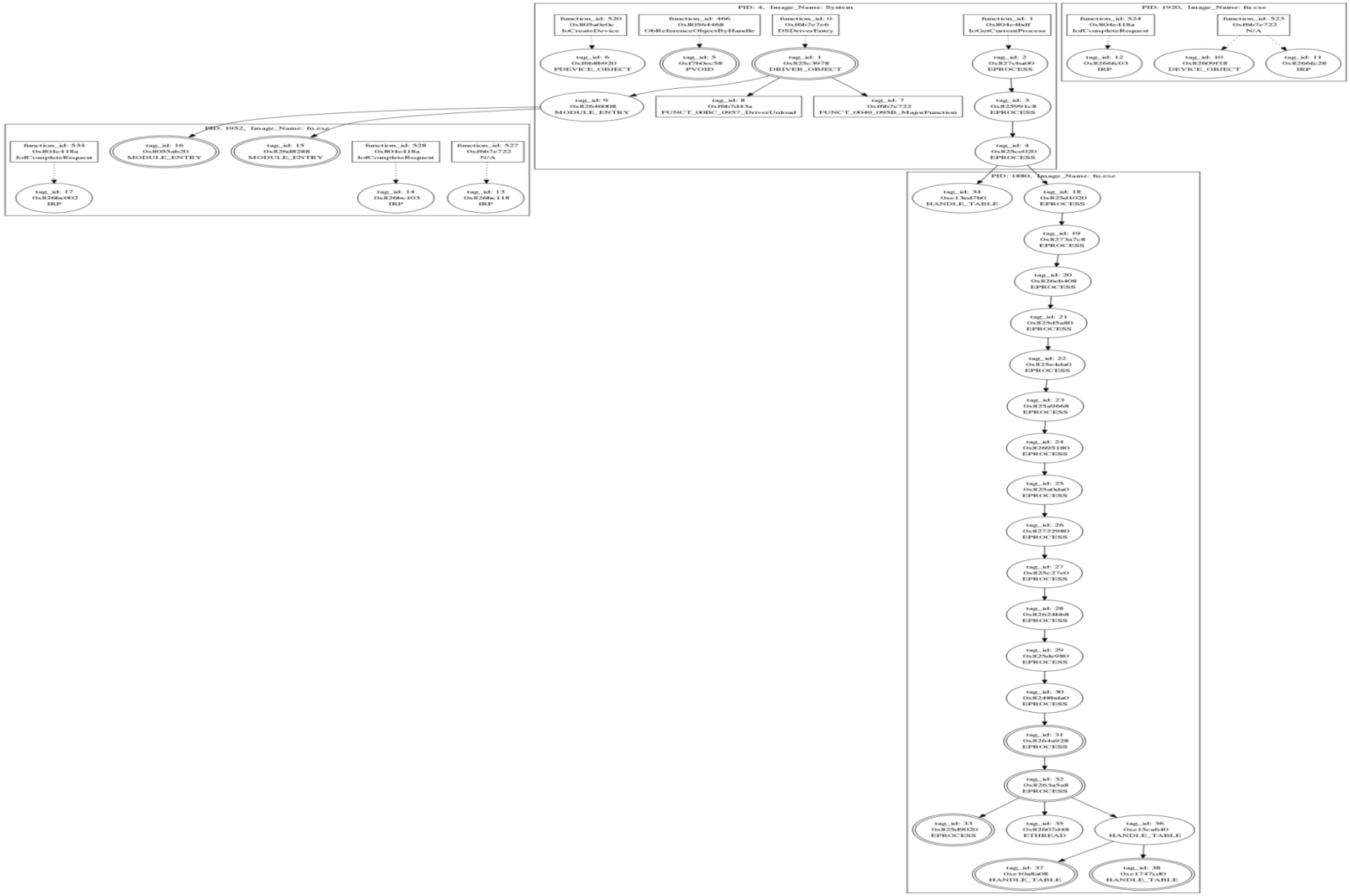
Memory Tagging (Continue)

- Linked list types (SINGLE_LIST_ENTRY and LIST_ENTRY) are annoted, so that Rkprofiler can find the actual kernel objects that they contains.
- Relative pointers are integers and they are also annoted as well.
- Ambiguous data types (generic pointer, union and dynamic array) are manually handled in Rkprofiler. (KOP project provides an automation solution)

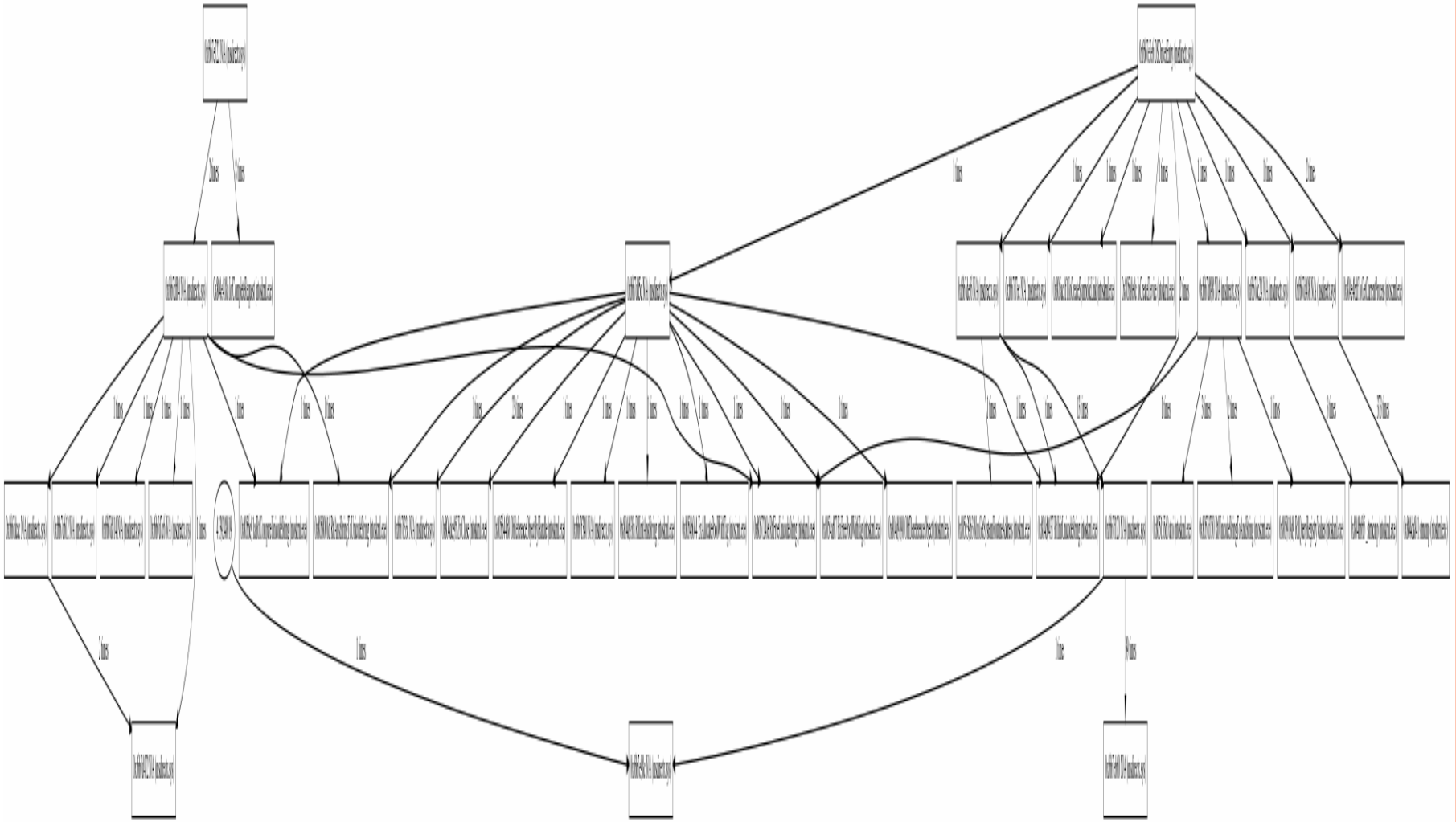
Evaluations

- We study the effectiveness of Rkprofiler on three Windows rootkits: FuTo, TCPIPHOOK and Rustock.B.
- FuTo applies DKOM to hide processes and drivers, and escalate process privileges.
- TCPIPHOOK employs hooking techniques to hide TCP connections from local users.
- Rustock.B is packed and closed-source. Hackers use it for accomplishing multiple tasks: hiding files, inserting spam thread and so on.
- Each test is finished within only a few minutes.

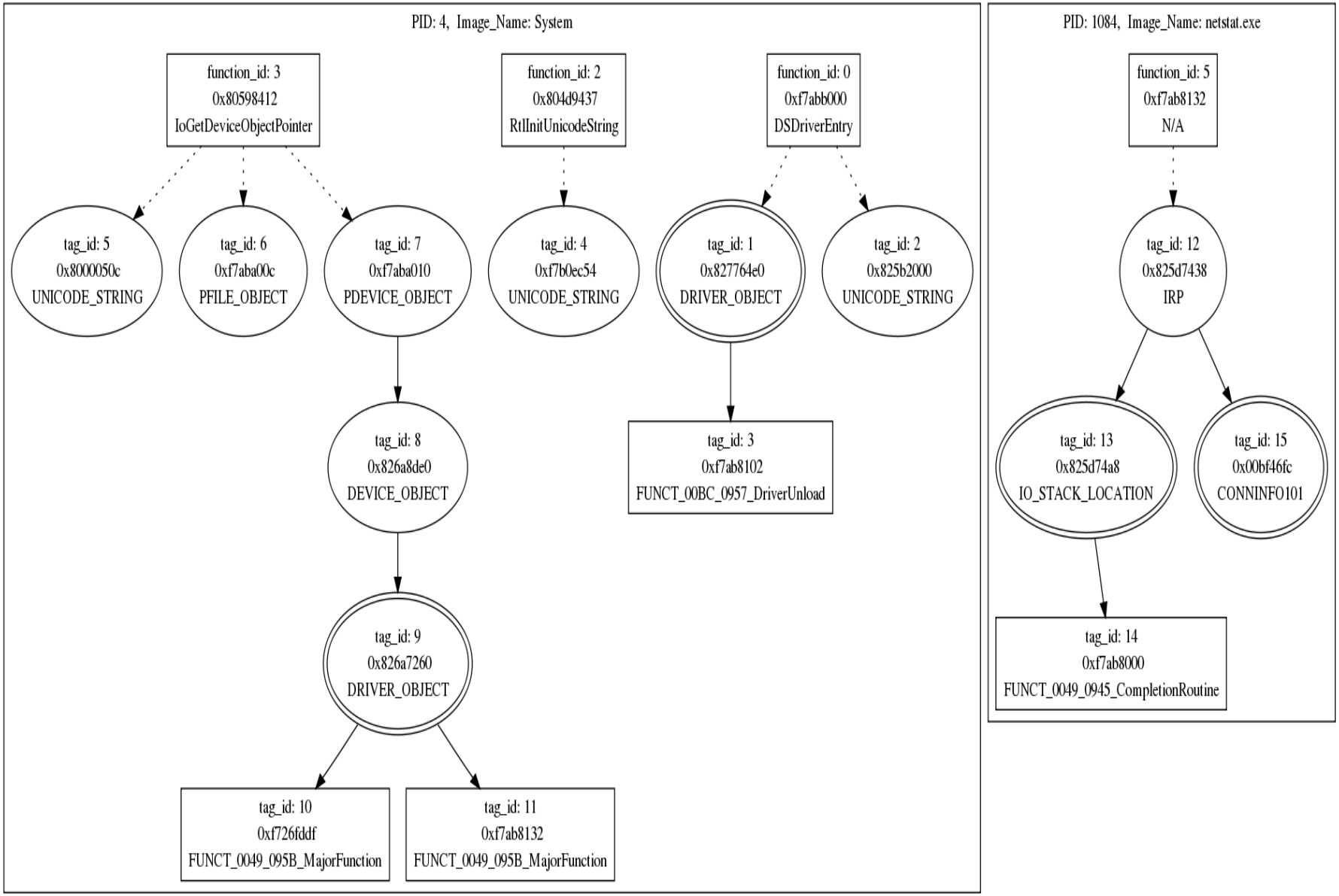
Tag Trace Graph of FuTo



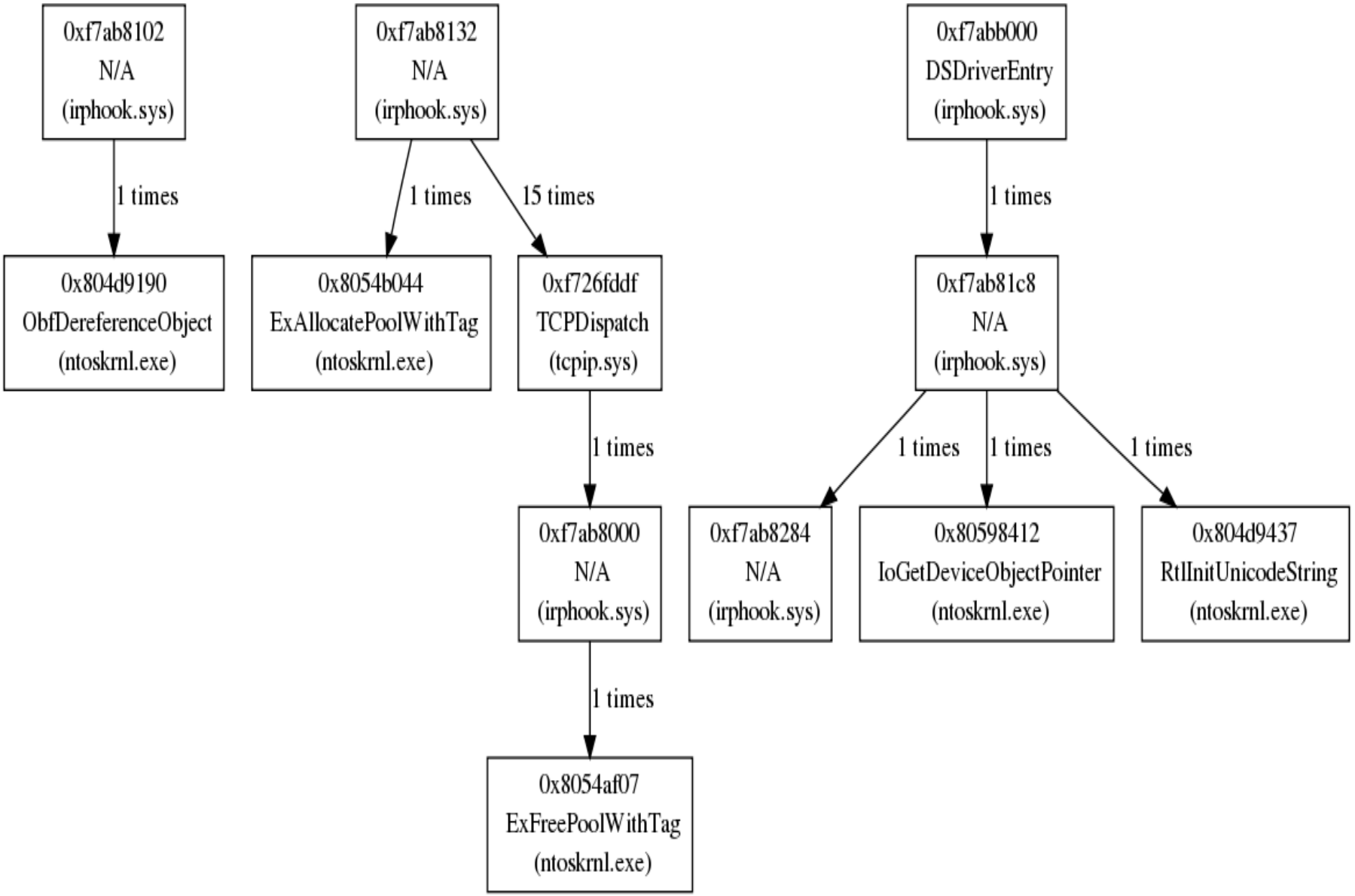
Call Graph of FuTo



Tag Trace Graph of TCPIRPHOOK



Call Graph of TCPIRPHOOK



External Functions and Registries Visited by Rustock.B

External Functions	**ExAllocatePoolWithTag, ExFreePoolWithTag, ExInitializeNPagedLookasideList, IoAllocateMdl, IoGetCurrentProcess, IoGetDeviceObjectPointer, IoGetRelatedDeviceObject, KeClearEvent, KeDelayExecutionThread, KeEnterCriticalRegion, KeInitializeApc, KeInitializeEvent, KeInitializeMutex, KeInitializeSpinLock, KeInsertQueueApc, KeLeaveCriticalRegion, KeWaitForSingleObject, MmBuildMdlForNonPagedPool, MmMapLockedPages, MmProbeAndLockPages, NtSetInformationProcess, ObfDereferenceObject, ObReferenceObjectByHandle, ProbeForRead, PsCreateSystemThread, PsLookupProcessByProcessId, PsLookupThreadByThreadId, RtlInitUnicodeString, _stricmp, _strnicmp, swprintf, wcschr, wcscpy, _wcsicmp, _wcslwr, wcsncpy, _wcsnicmp, wcstombs, ZwClose, ZwCreateEvent, ZwCreateFile, ZwDeleteKey, ZwEnumerateKey, ZwOpenKey, ZwQueryInformationFile, ZwQueryInformationProcess, ZwQuerySystemInformation, ZwReadFile**
Registry Keys	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\pe386 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_pe386

Limitations

- Microsoft doesn't publish all kernel symbols.
- Hackers may obfusticate the function call operations, e.g., replace CALL/RET with JMP/JMP.
- Dynamic objects may be located through brute-force searching and pattern matching.
- Rootkits may detect emulator and change its behavior accordingly.
- Rootkit may directly exploit the vulnerabilities of Emulator and attack the Rkprofiler.

Conclusions

- Rkprofiler is a sandbox-base rootkit analysis system.
- Rkprofiler can identify malicious kernel code regardless of the means of their access to the kernel.
- Rkprofiler reveals the building blocks of rootkit behaviors: function calls, kernel objects being visited, interest hardware accesses.
- Rkprofiler has limitations and need to be improved in the future.

Questions ?