Towards Automatic Inference of Kernel Object Semantics From Binary Code

JAN 8TH, 2016

论文下载：https://www.utdallas.edu/~zxl111930/file/RAID15.pdf

Abstract && Introduction

- 理解内核对象语义是很有意义的，比如可以帮助识别内核函数，这对于VM Introspection、内存取证等等都是有帮助的
- 这篇文章提出了第一个直接从内核二进制自动化获取内核对象语义的系统**Argos**
- 方法基础是Zhiqiang Lin过去工作，即API+系统调用统计
- 工作亮点：显性定义了语义，通过系统调用集合反映

System Overview

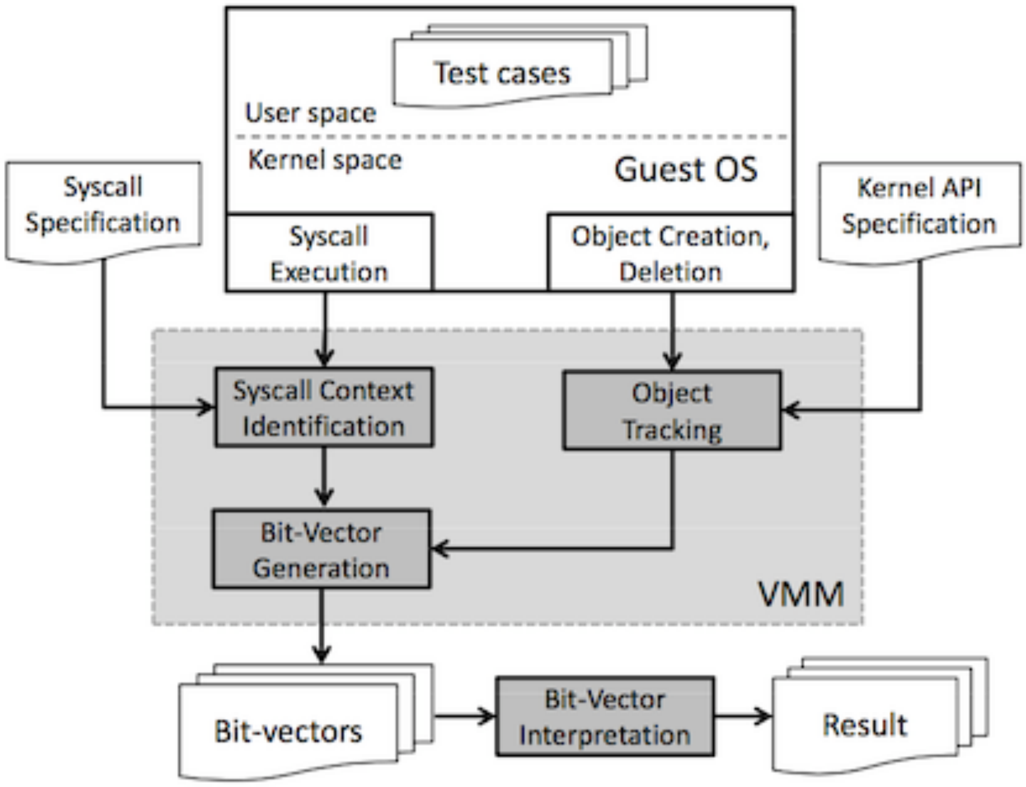


Fig. 1. An Overview of ARGOS.

- 主要提取两类信息，一个是内核对象，一个是系统调用
- 先要知道哪一团数据是内核对象，并要在这些对象当中做区分，因为这些信息都是动态获取的
- 而后根据这些对象相关的系统调用获取语义

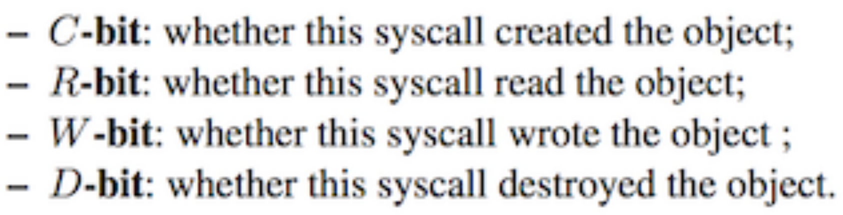
Design and Implementation

Object Tracing

- All objects need to be allocated
- 所以用调用malloc相关allocation函数的指令地址标志每个不同的对象
- 数据表明，80.3%的对象得到了一一对应
- 对象size动态根据指针偏移最大量得到

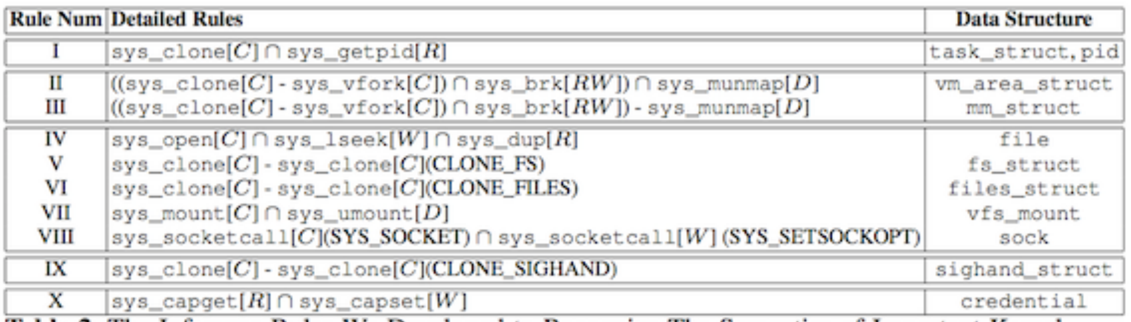
Bit-Vector Generation

- 系统调用追踪是老问题了，这里说明一下语义表示方法
- 每个内核对象都有一个比特向量
- 每4个bit代表一个系统调用：



- *C*-**bit**: whether this syscall created the object;
- *R*-**bit**: whether this syscall read the object;
- *W*-**bit**: whether this syscall wrote the object ;
- *D*-**bit**: whether this syscall destroyed the object.

Evaluation

- 这是一个侧重经验的实验
- 以为是投RAID的文章，重点分析的是安全相关的数据结构（内核对象）
- 

Rule Num	Detailed Rules	Data Structure
I	sys_clone[C] ∩ sys_getpid[R]	task_struct,pid
II	((sys_clone[C] - sys_vfork[C]) ∩ sys_brk[RW]) ∩ sys_munmap[D]	vm_area_struct
III	((sys_clone[C] - sys_vfork[C]) ∩ sys_brk[RW]) - sys_munmap[D]	mm_struct
IV	sys_open[C] ∩ sys_lseek[W] ∩ sys_dup[R]	file
V	sys_clone[C] - sys_clone[C](CLONE_FS)	fs_struct
VI	sys_clone[C] - sys_clone[C](CLONE_FILES)	files_struct
VII	sys_mount[C] ∩ sys_umount[D]	vfs_mount
VIII	sys_socketcall[C](SYS_SOCKET) ∩ sys_socketcall[W] (SYS_SETSOCKOPT)	sock
IX	sys_clone[C] - sys_clone[C](CLONE_SIGHAND)	sighand_struct
X	sys_capget[R] ∩ sys_capset[W]	credential

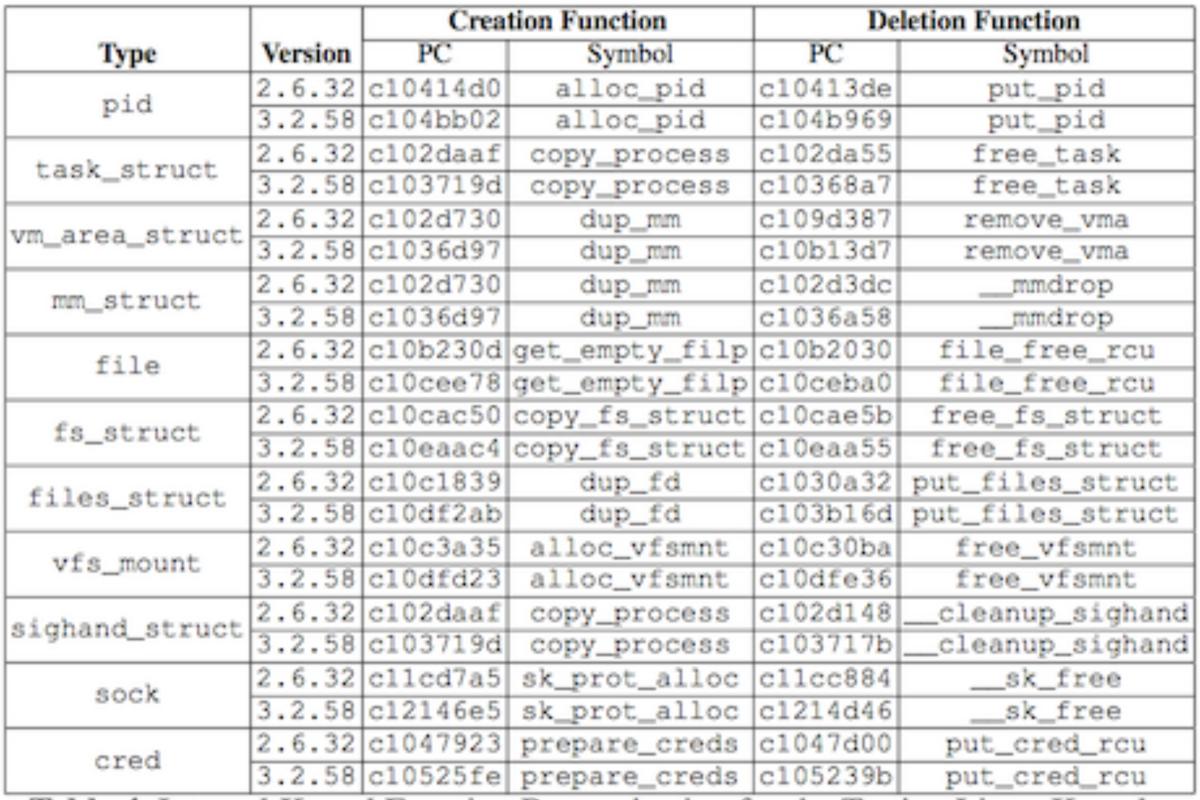
Table 2. The Inference Rules We Developed to Recognize The Semantics of Important Kernel Data Structures.

- 作者经验型地总结了一些规则，当满足时，认定是相关数据结构
- （并没有正式讨论正确率等问题）
- （没有显性解释规则当中-这个符号的含义）
- 以上也有一个规则对应多个结构的情况，之后通过已知的数据结构指向（point-to）关系区分

Application

- 内核函数识别

- 一个简单的启发式规则：the function calls the allocation process for a object, is the creator of the object



Type	Version	Creation Function		Deletion Function	
		PC	Symbol	PC	Symbol
pid	2.6.32	c10414d0	alloc_pid	c10413de	put_pid
	3.2.58	c104bb02	alloc_pid	c104b969	put_pid
task_struct	2.6.32	c102daaf	copy_process	c102da55	free_task
	3.2.58	c103719d	copy_process	c10368a7	free_task
vm_area_struct	2.6.32	c102d730	dup_mm	c109d387	remove_vma
	3.2.58	c1036d97	dup_mm	c10b13d7	remove_vma
mm_struct	2.6.32	c102d730	dup_mm	c102d3dc	__mmdrop
	3.2.58	c1036d97	dup_mm	c1036a58	__mmdrop
file	2.6.32	c10b230d	get_empty_filp	c10b2030	file_free_rcu
	3.2.58	c10cee78	get_empty_filp	c10ceba0	file_free_rcu
fs_struct	2.6.32	c10cac50	copy_fs_struct	c10cae5b	free_fs_struct
	3.2.58	c10eaac4	copy_fs_struct	c10eaa55	free_fs_struct
files_struct	2.6.32	c10c1839	dup_fd	c1030a32	put_files_struct
	3.2.58	c10df2ab	dup_fd	c103b16d	put_files_struct
vfs_mount	2.6.32	c10c3a35	alloc_vfsmnt	c10c30ba	free_vfsmnt
	3.2.58	c10dfd23	alloc_vfsmnt	c10dfe36	free_vfsmnt
sighand_struct	2.6.32	c102daaf	copy_process	c102d148	__cleanup_sighand
	3.2.58	c103719d	copy_process	c103717b	__cleanup_sighand
sock	2.6.32	c11cd7a5	sk_prot_alloc	c11cc884	__sk_free
	3.2.58	c12146e5	sk_prot_alloc	c1214d46	__sk_free
cred	2.6.32	c1047923	prepare_creds	c1047d00	put_cred_rcu
	3.2.58	c10525fe	prepare_creds	c105239b	put_cred_rcu

Table 4. Internal Kernel Function Recognization for the Testing Linux Kernels.

- 已经识别出某个结构，根据已有知识，则可以通过调用allocation过程的函数识别出相应的创建函数