Malware Memory Forensics (MMF) Workshop at ACSAC'14

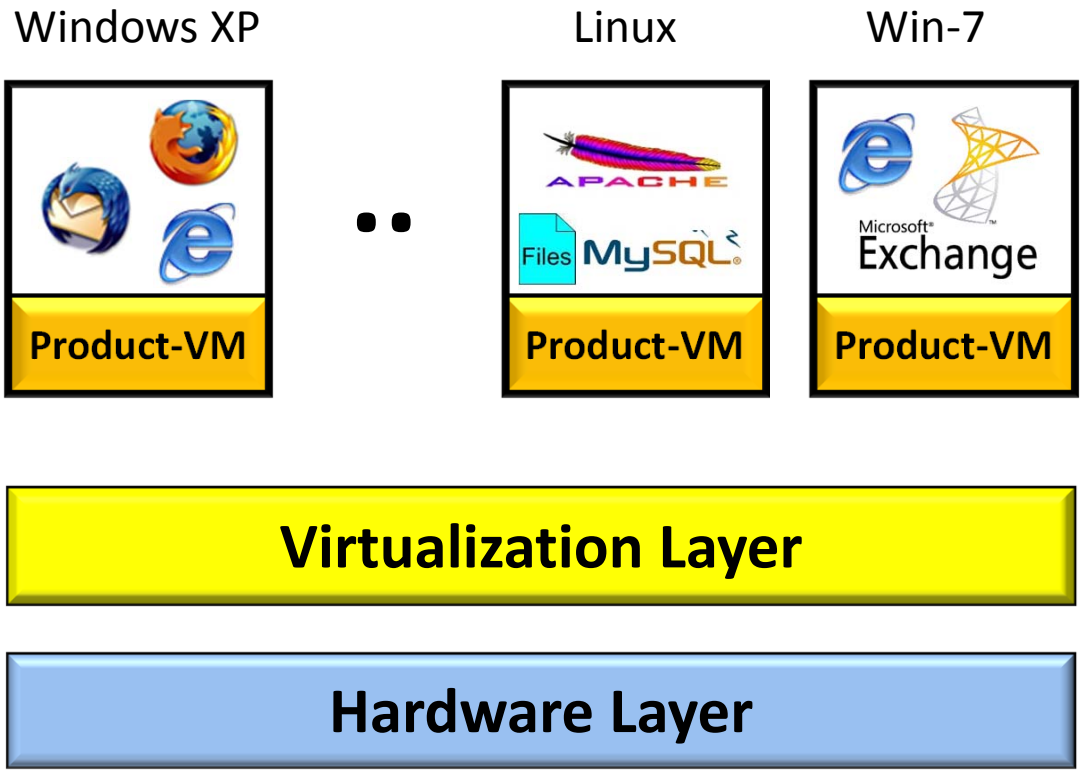
Automating Introspection and Forensics Software Development via Binary Code Reuse

Zhiqiang Lin

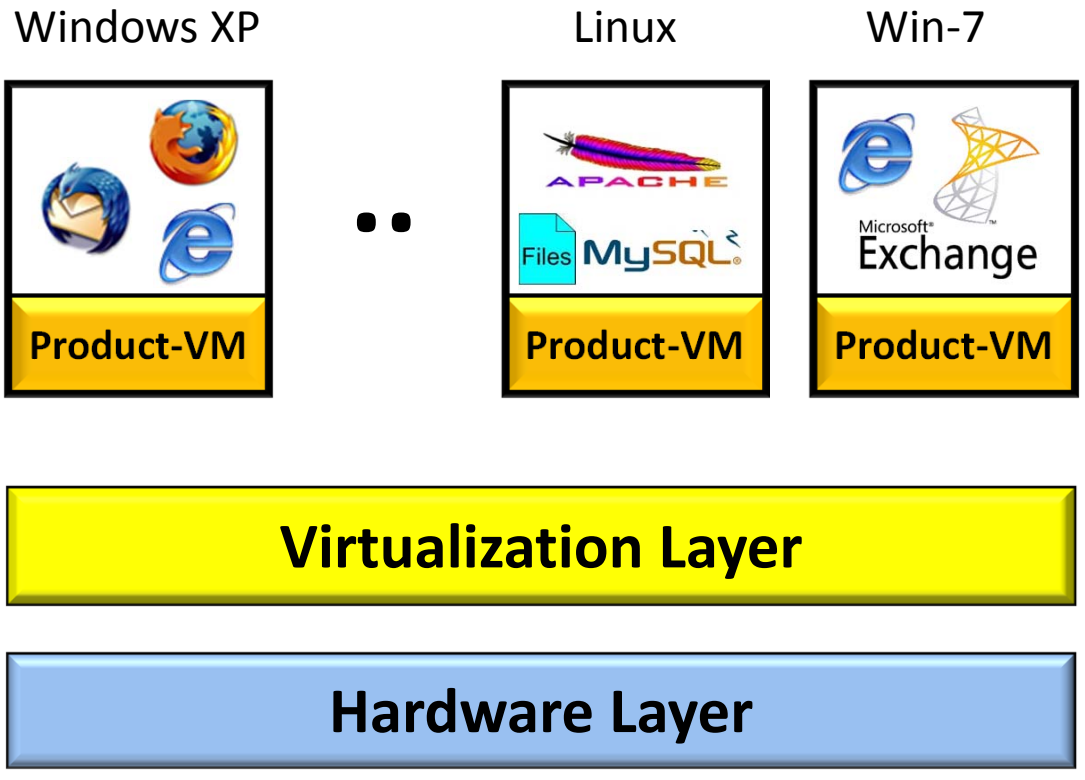
Department of Computer Sciences
The University of Texas at Dallas

December 9^{th}, 2014

Virtualization, Hypervisor, and the Cloud

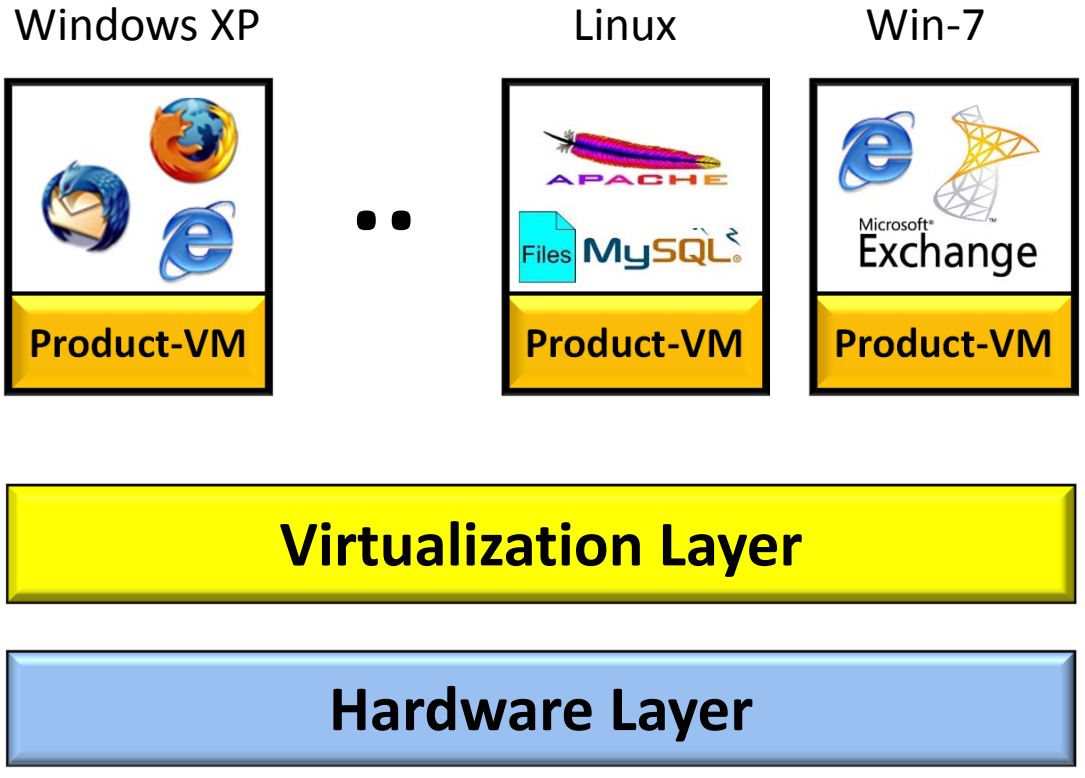


Virtualization, Hypervisor, and the Cloud



Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from **multi-tasking** to **multi-OS**.

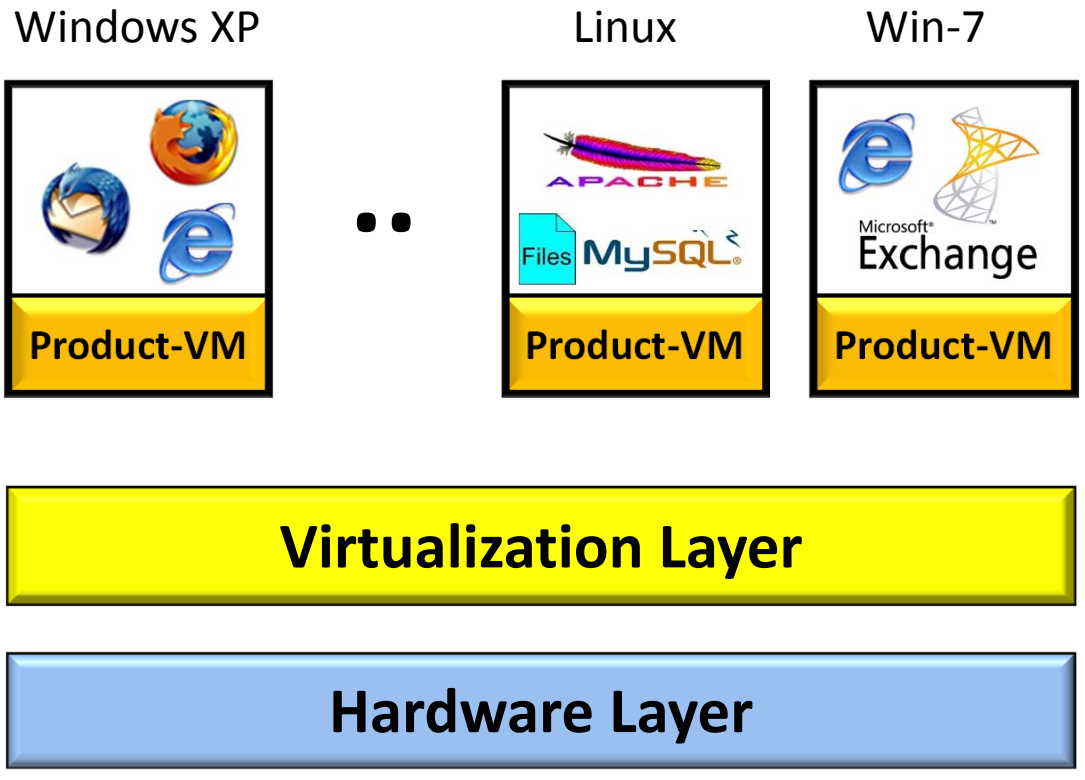
Virtualization, Hypervisor, and the Cloud



Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from **multi-tasking** to **multi-OS**.

Multiplexing, Isolation, Migration, ...

Virtualization, Hypervisor, and the Cloud

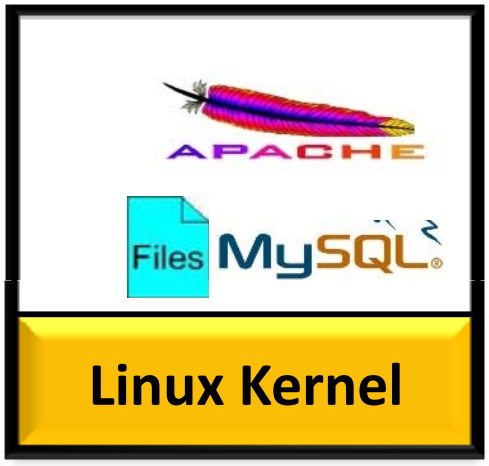


Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from **multi-tasking** to **multi-OS**.

Multiplexing, Isolation, Migration, ...

Xen KVM vmware vSphere Microsoft Hyper-V QEMU

On Anti-Virus Software Evolution



Virtualization Layer

On Anti-Virus Software Evolution

Virtualization Layer

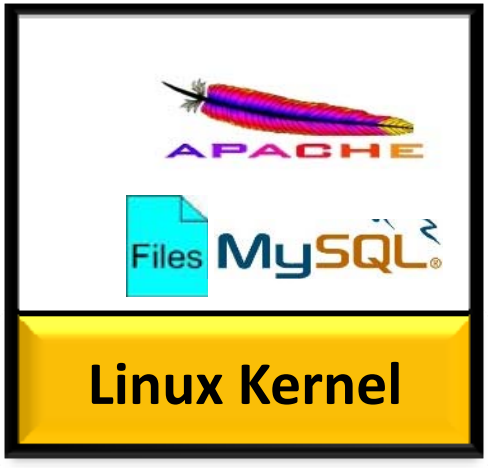
On Anti-Virus Software Evolution

Virtualization Layer

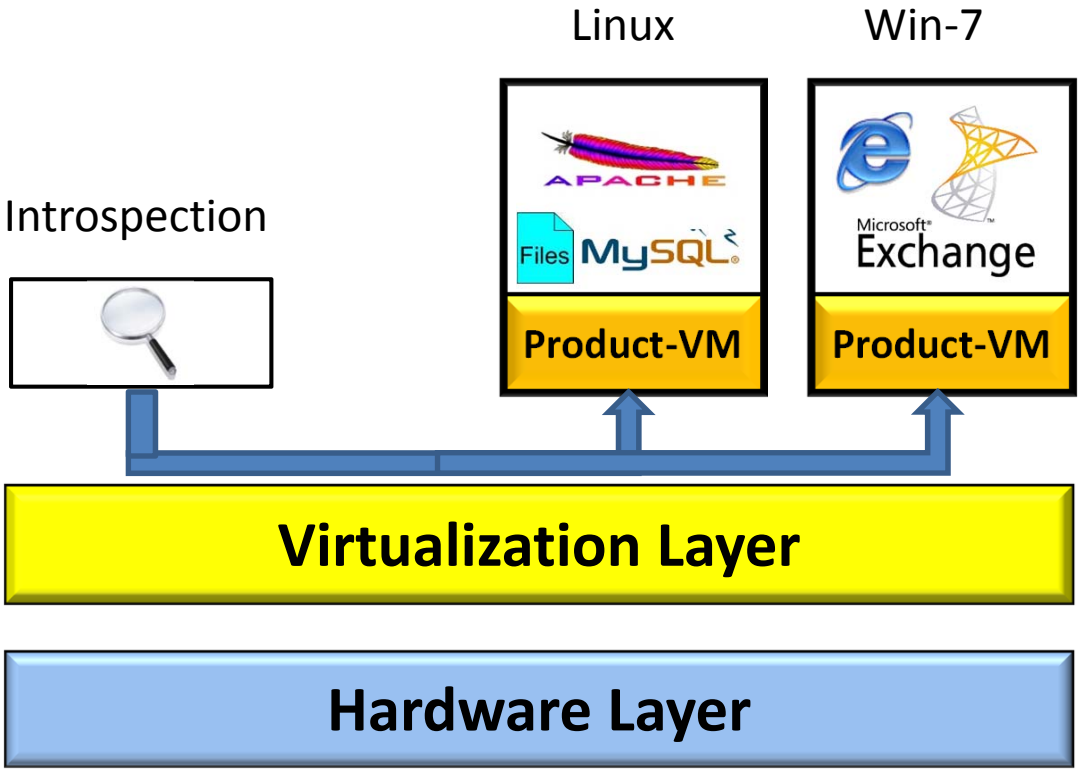
On Anti-Virus Software Evolution

Virtualization Layer

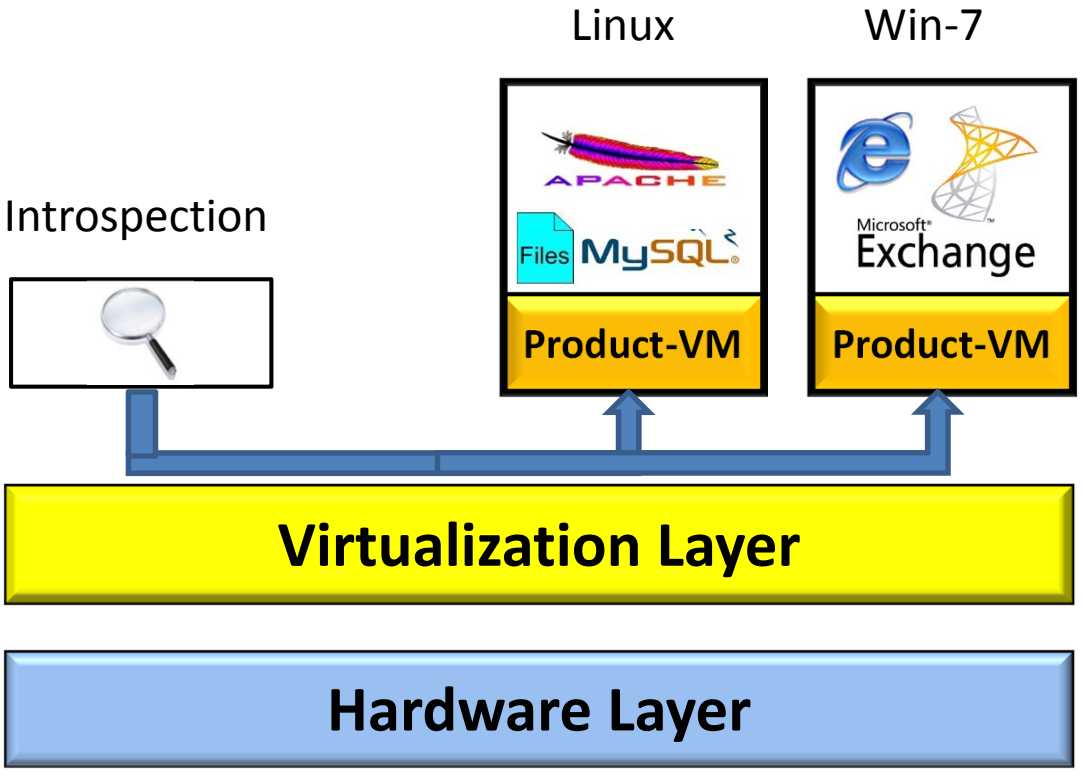
On Anti-Virus Software Evolution



Virtual Machine Introspection (VMI) [Garfinkel et al, NDSS'03]

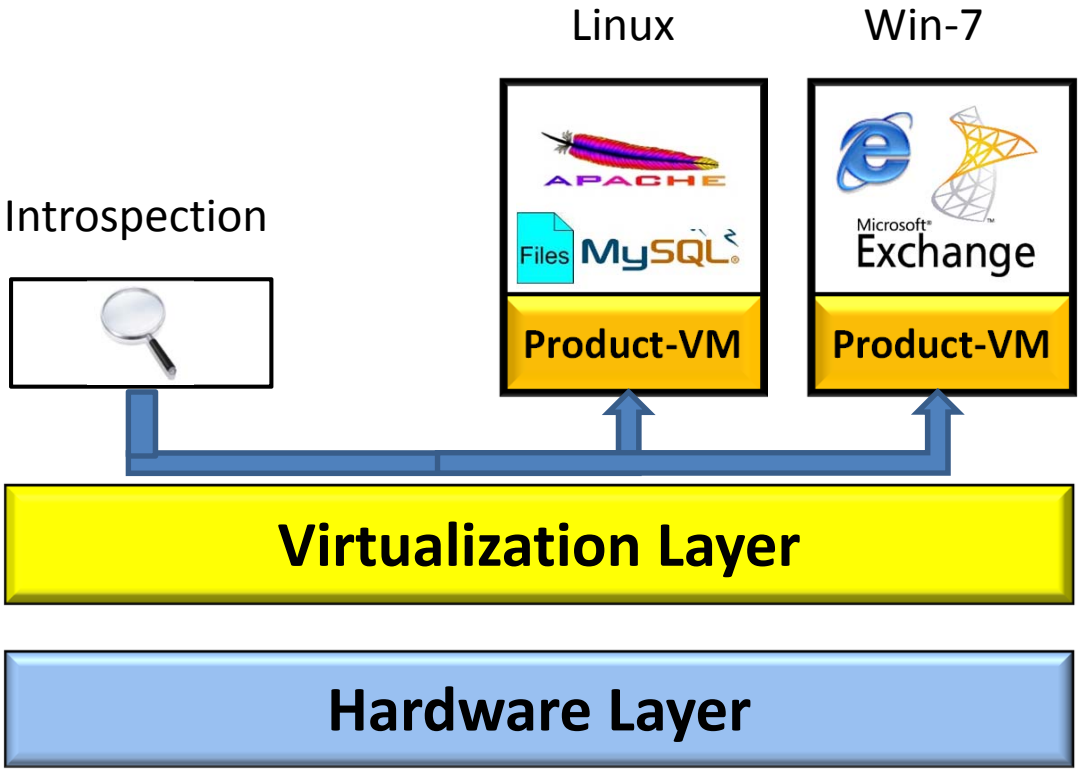


Virtual Machine Introspection (VMI) [Garfinkel et al, NDSS'03]



Using a trusted, dedicated virtualization layer program to monitor the running VMs

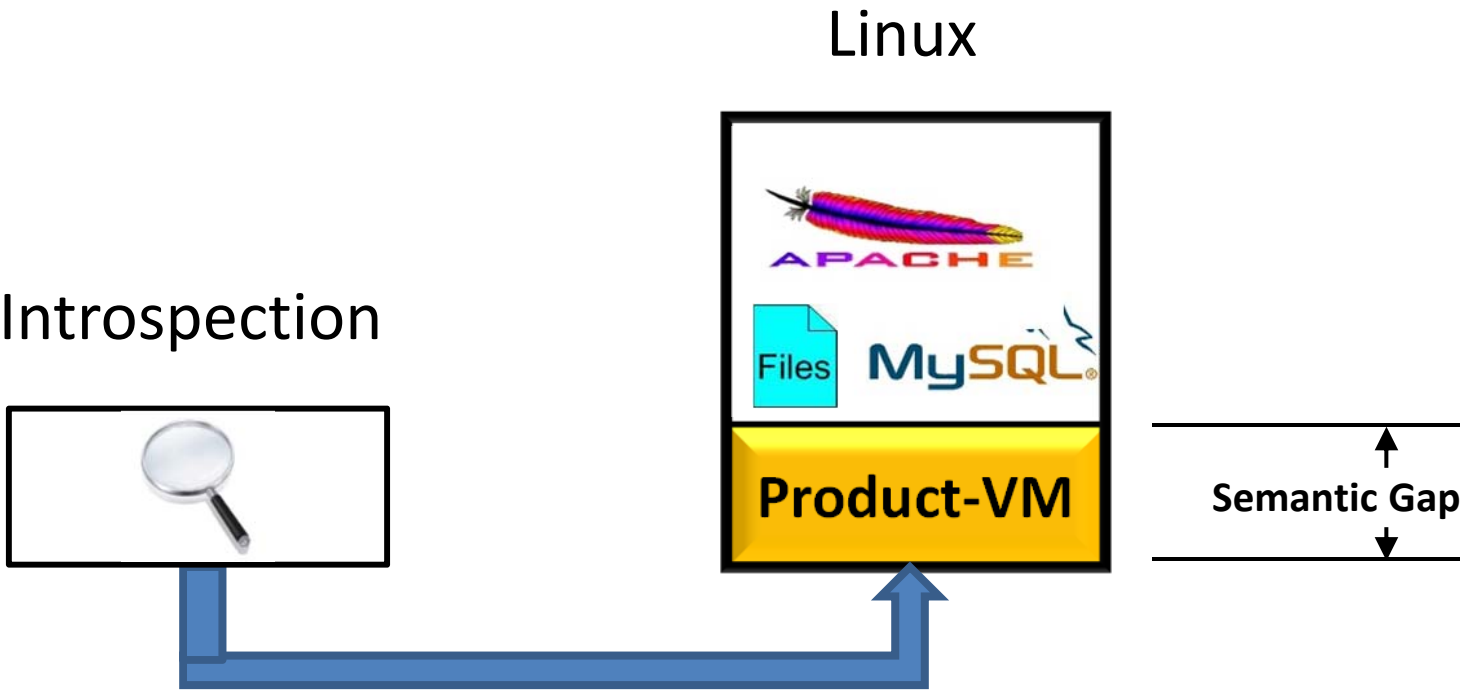
Virtual Machine Introspection (VMI) [Garfinkel et al, NDSS'03]



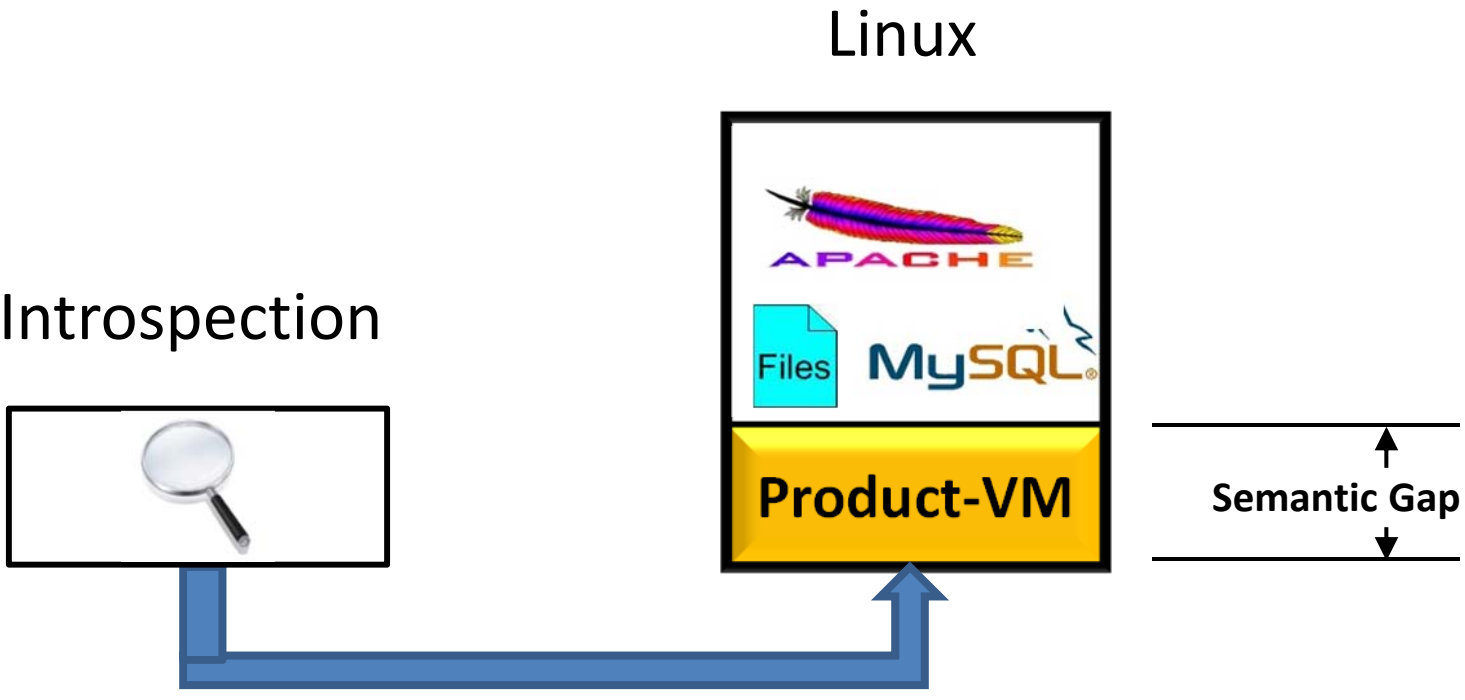
Using a trusted, dedicated virtualization layer program to monitor the running VMs

- Intrusion Detection
- Malware Analysis
- Memory Forensics

The Semantic Gap in Out-of-VM ([Chen and Noble HotOS'01])

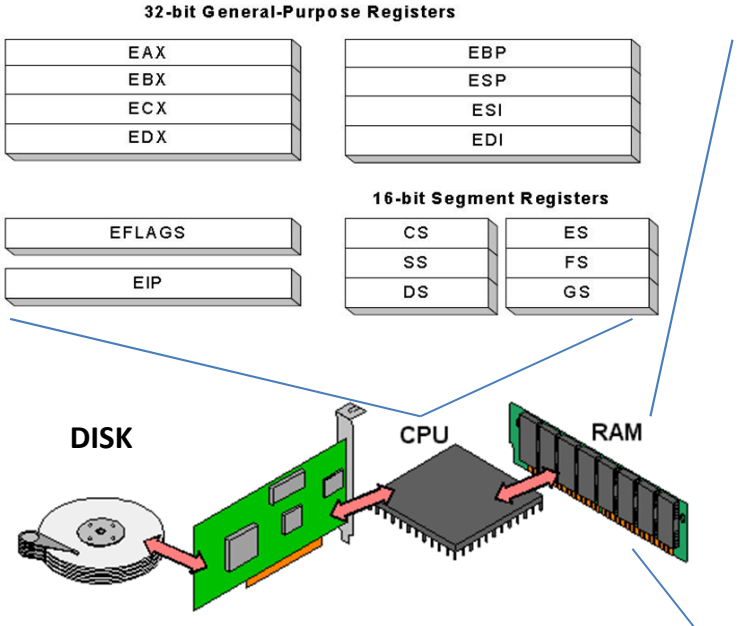


The Semantic Gap in Out-of-VM ([Chen and Noble HotOS'01])



- View exposed by Virtual Machine Monitor is at low-level
- There is no abstraction and no APIs
- Need to reconstruct the guest-OS abstraction

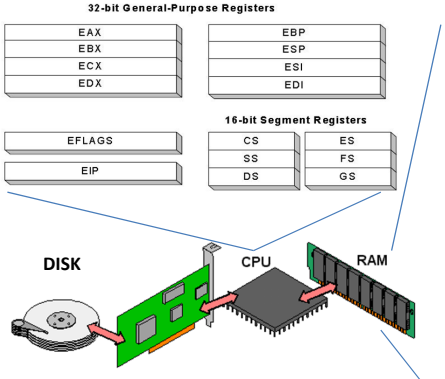
Example: Inspect `pid`s of Guest Memory from VMM



Virtual Machine Monitor Layer

```
…
00001800  eb 40 1b 02 63 74 00 f0  00 00 00 00 00 00 00 00  |.@..ct..........|
00001810  00 00 00 00 80 00 00 00  00 00 00 00 00 00 00 00  |................|
00001820  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001830  00 00 00 00 00 00 00 00  10 76 16 cc 00 00 00 00  |.........v......|
00001840  00 19 66 8c d0 50 b8 08  00 00 00 66 8e d0 53 8b  |..f..P.....f..S.|
00001850  d9 ff 2d 19 02 00 00 0f  20 c0 0f ba f0 1f 0f 22  |..-..... ......"|
00001860  c0 eb 00 b9 80 00 00 c0  0f 32 0f ba f0 08 0f 30  |.........2.....0|
00001870  0f 20 e0 0f ba f0 05 0f  22 e0 60 9c 8b d3 c1 ea  |. ......".`.....|
00001880  04 89 a3 76 02 00 00 0f  01 83 80 02 00 00 0f 01  |...v............|
00001890  8b 88 02 00 00 8b 8b 3c  00 00 00 0b c9 74 12 8b  |.......<.....t..|
000018a0  b3 38 00 00 00 8b fb 81  c7 00 30 00 00 2b f9 f3  |.8........0..+..|
000018b0  a4 0f 01 9b 90 02 00 00  0f 01 93 68 02 00 00 66  |...........h...f|
000018c0  b8 10 00 66 8e d8 66 8e  c0 66 8e d0 66 8e e0 66  |...f..f..f..f..f|
…
*
00100f60  00 00 00 00 00 00 00 00  00 f0 ff 5d 76 e3 f0 2f  |...........]v../|
00100f70  93 c9 a4 1d f9 48 be f8  6c c7 1d 92 4c 1e 6e 35  |.....H..l...L.n5|
00100f80  b4 f8 1b ae f6 69 e8 c0  b7 34 74 a1 4e 5a a7 93  |.....i...4t.NZ..|
00100f90  97 2f f3 47 cf d7 10 df  f0 d6 e3 9b f5 cf a9 23  |./.G...........#|
00100fa0  cd 9f 87 4f 37 7f 1e f1  fe dc 7d b9 f9 f3 7b ef  |...O7.....}...{.|
00100fb0  cf 95 bf 94 3f 8d 63 9a  cc 8a 36 5b 56 7b d2 76  |....?.c...6[V{.v|
00100fc0  b6 d9 ad ee 61 f6 90 a4  2c 2b 54 66 37 de 3d a9  |....a...,+Tf7.=.|
00100fd0  b9 d9 67 37 1e 7a b5 ce  ef 0c 58 ee 4d 30 d0 9b  |..g7.z....X.M0..|
00100fe0  c0 6e bc e7 3d f3 e7 d0  9a bf a4 82 1b c7 9c f1  |.n..=...........|
00100ff0  db 66 2b d8 38 cb 2a 91  80 ad 7d 25 d8 0a e5 db  |.f+.8.*...}%....|
```

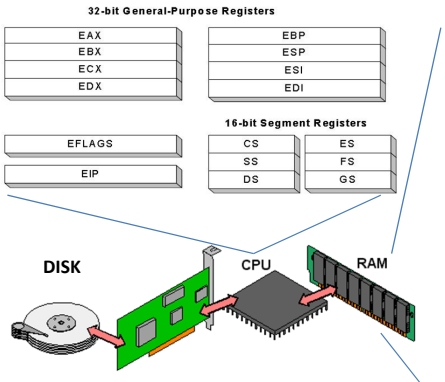
Example: Inspect `pid`s of Guest Memory from VMM



Virtual Machine Monitor Layer

```
…
00001800  eb 40 1b 02 63 74 00 f0  00 00 00 00 00 00 00 00  |.@..ct..........|
00001810  00 00 00 00 80 00 00 00  00 00 00 00 00 00 00 00  |................|
00001820  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001830  00 00 00 00 00 00 00 00  10 76 16 cc 00 00 00 00  |.........v......|
00001840  00 19 66 8c d0 50 b8 08  00 00 00 66 8e d0 53 8b  |..f..P.....f..S.|
00001850  d9 ff 2d 19 02 00 00 0f  20 c0 0f ba f0 1f 0f 22  |..-..... ......"|
00001860  c0 eb 00 b9 80 00 00 c0  0f 32 0f ba f0 08 0f 30  |.........2.....0|
00001870  0f 20 e0 0f ba f0 05 0f  22 e0 60 9c 8b d3 c1 ea  |. ......".`.....|
00001880  04 89 a3 76 02 00 00 0f  01 83 80 02 00 00 0f 01  |...v............|
00001890  8b 88 02 00 00 8b 8b 3c  00 00 00 0b c9 74 12 8b  |.......<.....t..|
000018a0  b3 38 00 00 00 8b fb 81  c7 00 30 00 00 2b f9 f3  |.8........0..+..|
000018b0  a4 0f 01 9b 90 02 00 00  0f 01 93 68 02 00 00 66  |...........h...f|
000018c0  b8 10 00 66 8e d8 66 8e  c0 66 8e d0 66 8e e0 66  |...f..f..f..f..f|
…
*
00100f60  00 00 00 00 00 00 00 00  00 f0 ff 5d 76 e3 f0 2f  |...........]v../|
00100f70  93 c9 a4 1d f9 48 be f8  6c c7 1d 92 4c 1e 6e 35  |.....H..l...L.n5|
00100f80  b4 f8 1b ae f6 69 e8 c0  b7 34 74 a1 4e 5a a7 93  |.....i...4t.NZ..|
00100f90  97 2f f3 47 cf d7 10 df  f0 d6 e3 9b f5 cf a9 23  |./.G...........#|
00100fa0  cd 9f 87 4f 37 7f 1e f1  fe dc 7d b9 f9 f3 7b ef  |...O7.....}...{.|
00100fb0  cf 95 bf 94 3f 8d 63 9a  cc 8a 36 5b 56 7b d2 76  |....?.c...6[V{.v|
00100fc0  b6 d9 ad ee 61 f6 90 a4  2c 2b 54 66 37 de 3d a9  |....a...,+Tf7.=.|
00100fd0  b9 d9 67 37 1e 7a b5 ce  ef 0c 58 ee 4d 30 d0 9b  |..g7.z....X.M0..|
00100fe0  c0 6e bc e7 3d f3 e7 d0  9a bf a4 82 1b c7 9c f1  |.n..=...........|
00100ff0  db 66 2b d8 38 cb 2a 91  80 ad 7d 25 d8 0a e5 db  |.f+.8.*...}%....|
```

Example: Inspect `pid`s of Guest Memory from VMM



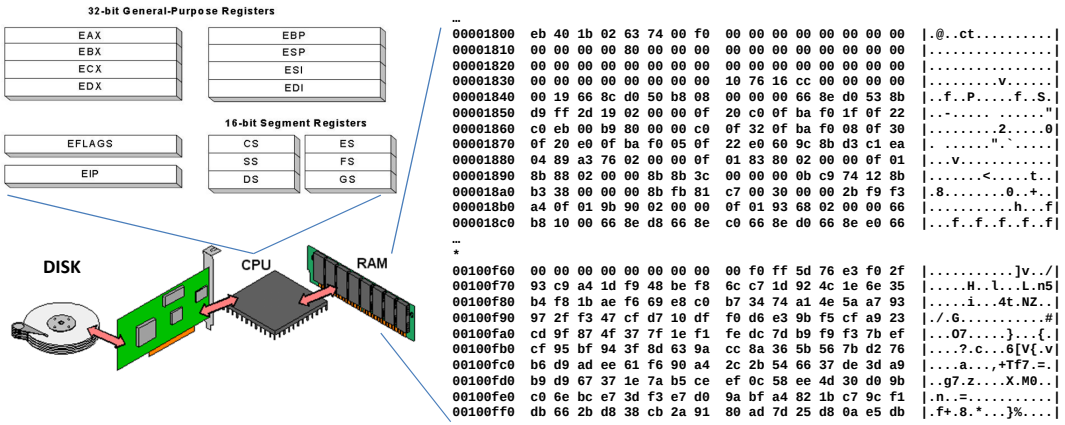
Virtual Machine Monitor Layer

```
…
00001800  eb 40 1b 02 63 74 00 f0  00 00 00 00 00 00 00 00  |.@..ct..........|
00001810  00 00 00 00 80 00 00 00  00 00 00 00 00 00 00 00  |................|
00001820  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001830  00 00 00 00 00 00 00 00  10 76 16 cc 00 00 00 00  |.........v......|
00001840  00 19 66 8c d0 50 b8 08  00 00 00 66 8e d0 53 8b  |..f..P.....f..S.|
00001850  d9 ff 2d 19 02 00 00 0f  20 c0 0f ba f0 1f 0f 22  |..-..... ......"|
00001860  c0 eb 00 b9 80 00 00 c0  0f 32 0f ba f0 08 0f 30  |.........2.....0|
00001870  0f 20 e0 0f ba f0 05 0f  22 e0 60 9c 8b d3 c1 ea  |. ......".`.....|
00001880  04 89 a3 76 02 00 00 0f  01 83 80 02 00 00 0f 01  |...v............|
00001890  8b 88 02 00 00 8b 8b 3c  00 00 00 0b c9 74 12 8b  |.......<.....t..|
000018a0  b3 38 00 00 00 8b fb 81  c7 00 30 00 00 2b f9 f3  |.8........0..+..|
000018b0  a4 0f 01 9b 90 02 00 00  0f 01 93 68 02 00 00 66  |...........h...f|
000018c0  b8 10 00 66 8e d8 66 8e  c0 66 8e d0 66 8e e0 66  |...f..f..f..f..f|
…
*
00100f60  00 00 00 00 00 00 00 00  00 f0 ff 5d 76 e3 f0 2f  |...........]v../|
00100f70  93 c9 a4 1d f9 48 be f8  6c c7 1d 92 4c 1e 6e 35  |.....H..l...L.n5|
00100f80  b4 f8 1b ae f6 69 e8 c0  b7 34 74 a1 4e 5a a7 93  |.....i...4t.NZ..|
00100f90  97 2f f3 47 cf d7 10 df  f0 d6 e3 9b f5 cf a9 23  |./.G...........#|
00100fa0  cd 9f 87 4f 37 7f 1e f1  fe dc 7d b9 f9 f3 7b ef  |...O7.....}...{.|
00100fb0  cf 95 bf 94 3f 8d 63 9a  cc 8a 36 5b 56 7b d2 76  |....?.c...6[V{.v|
00100fc0  b6 d9 ad ee 61 f6 90 a4  2c 2b 54 66 37 de 3d a9  |....a...,+Tf7.=.|
00100fd0  b9 d9 67 37 1e 7a b5 ce  ef 0c 58 ee 4d 30 d0 9b  |..g7.z....X.M0..|
00100fe0  c0 6e bc e7 3d f3 e7 d0  9a bf a4 82 1b c7 9c f1  |.n..=...........|
00100ff0  db 66 2b d8 38 cb 2a 91  80 ad 7d 25 d8 0a e5 db  |.f+.8.*...}%....|
```

In Kernel 2.6.18

```
struct task_struct {
      ...
    [188] pid_t pid;
    [192] pid_t tgid;
      ...
    [356] uid_t uid;
    [360] uid_t euid;
    [364] uid_t suid;
    [368] uid_t fsuid;
    [372] gid_t gid;
    [376] gid_t egid;
    [380] gid_t sgid;
    [384] gid_t fsgid;
    ...
    [428] char comm[16];
    ...
}
SIZE: 1408
```

Example: Inspect `pid`s of Guest Memory from VMM



Virtual Machine Monitor Layer

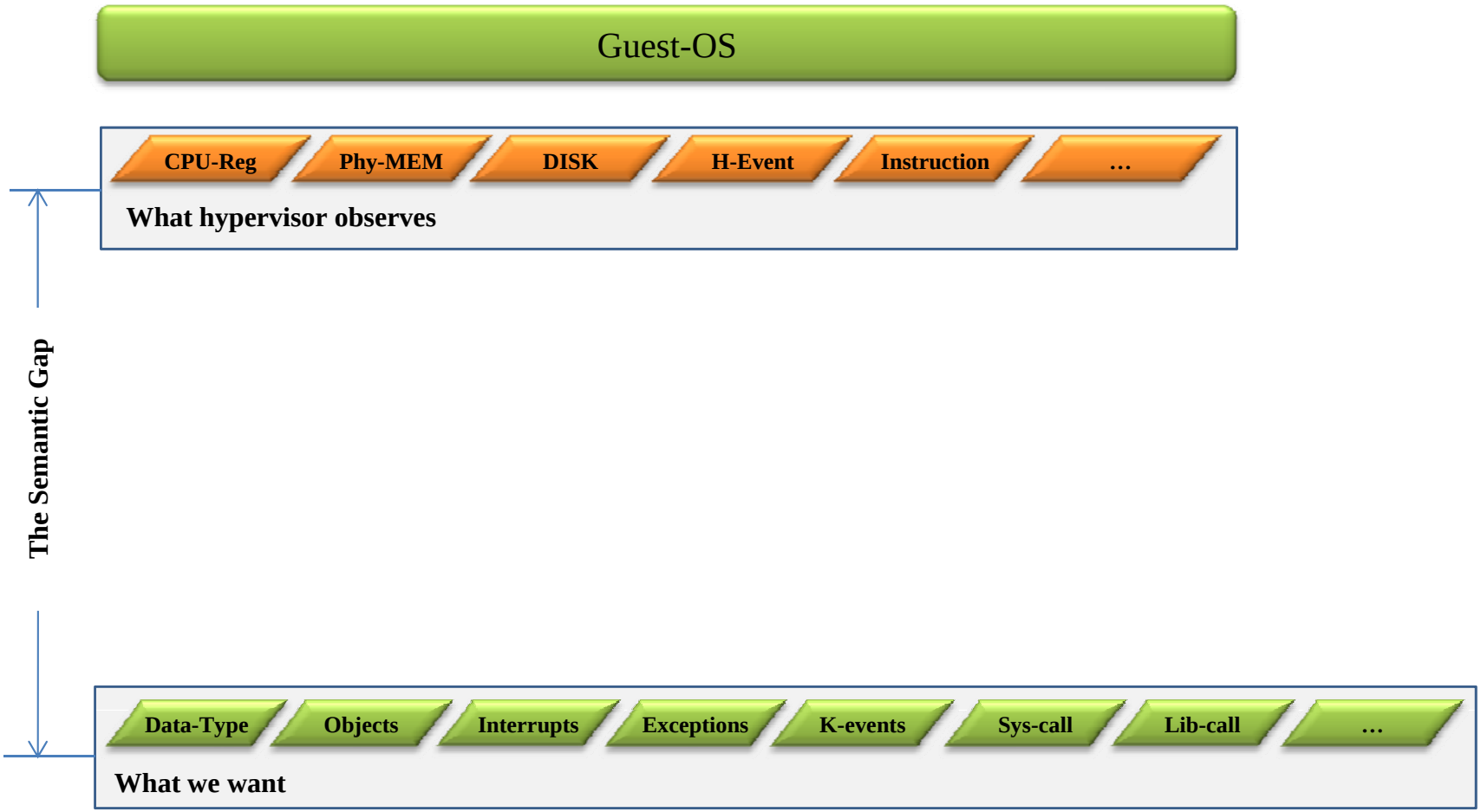
- Kernel specific data structure definition
- Kernel symbols (global variable)
- Virtual to physical (V2P) translation

In Kernel 2.6.18

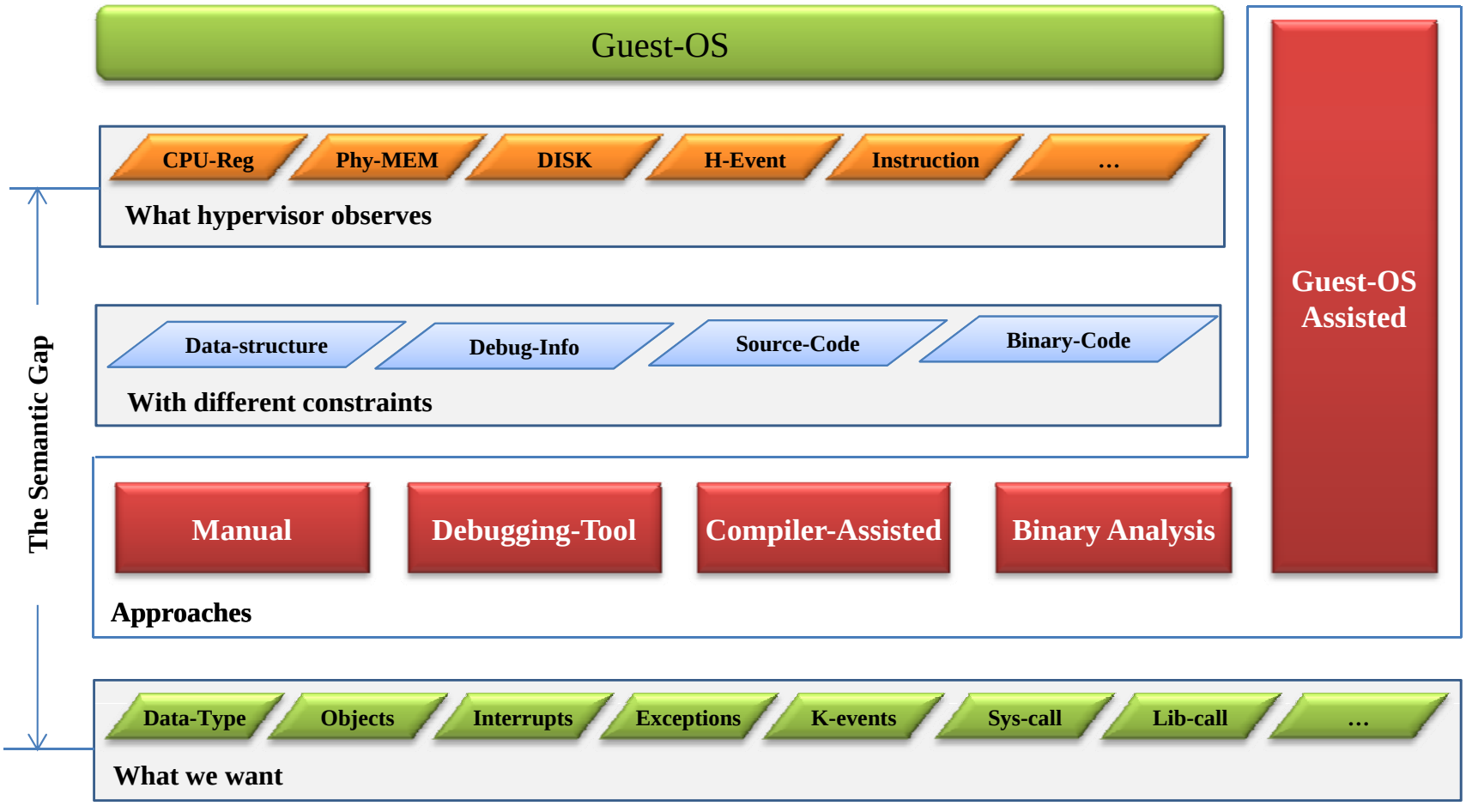
```
struct task_struct {
      ...
    [188] pid_t pid;
    [192] pid_t tgid;
      ...
    [356] uid_t uid;
    [360] uid_t euid;
    [364] uid_t suid;
    [368] uid_t fsuid;
    [372] gid_t gid;
    [376] gid_t egid;
    [380] gid_t sgid;
    [384] gid_t fsgid;
    ...
    [428] char comm[16];
    ...
}
SIZE: 1408
```

How to bridge the semantic gap

How to bridge the semantic gap



How to bridge the semantic gap



State-of-the-art in bridging the Semantic Gap

State-of-the-art in bridging the Semantic Gap

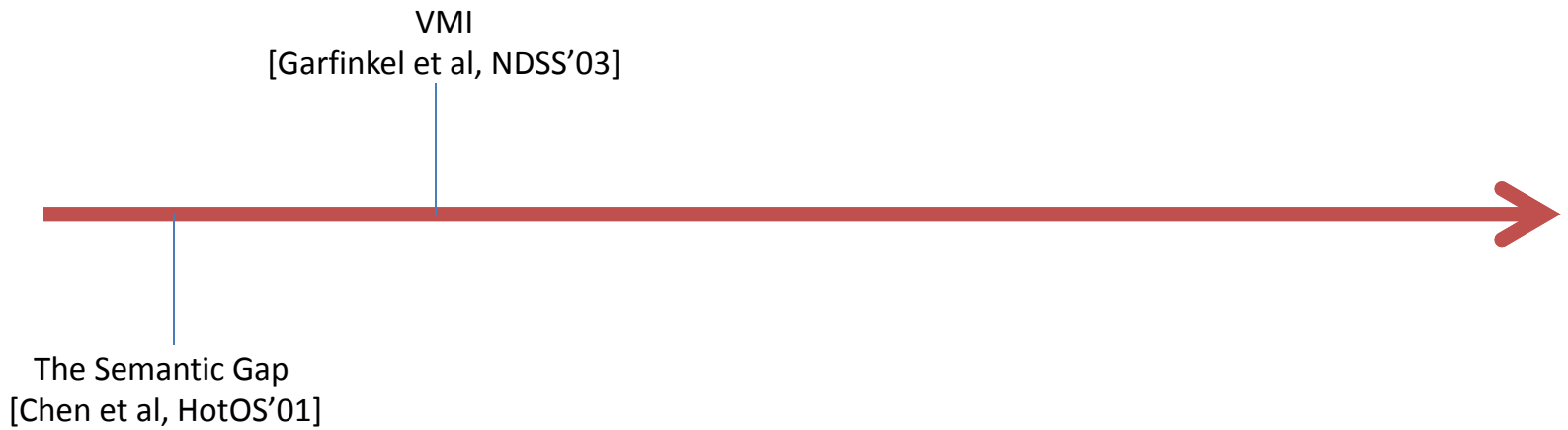
- In HotOS'01, Chen and Noble first raised the semantic gap problem in virtualization

State-of-the-art in bridging the Semantic Gap

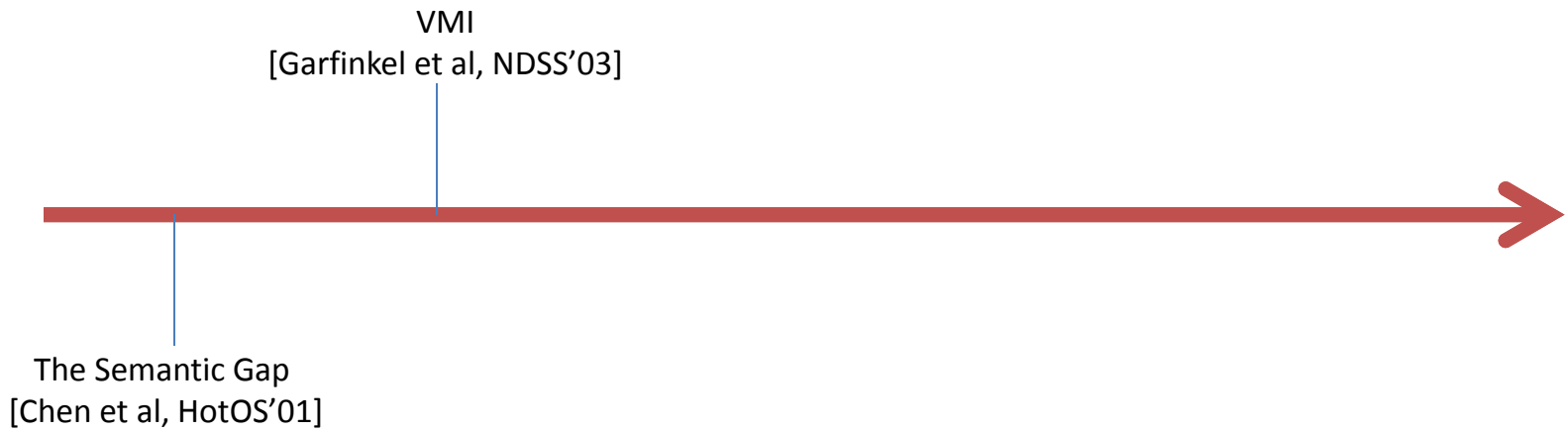
- In HotOS'01, Chen and Noble first raised the semantic gap problem in virtualization

"Services in the VM operate below the abstractions provided by the guest OS ... This can make it difficult to provide services."

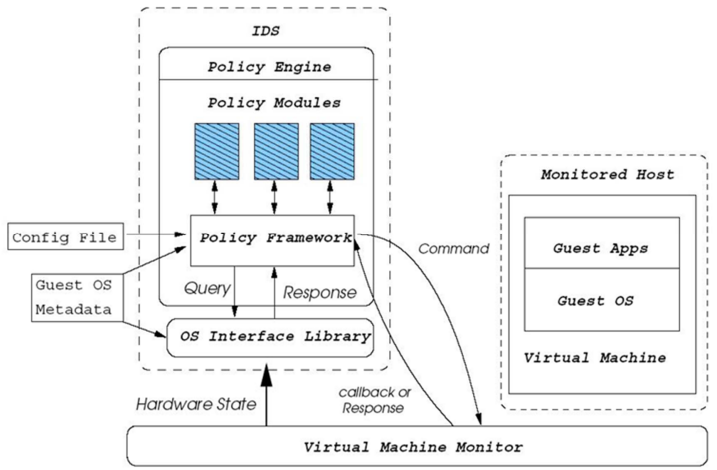
State-of-the-art in bridging the Semantic Gap



State-of-the-art in bridging the Semantic Gap



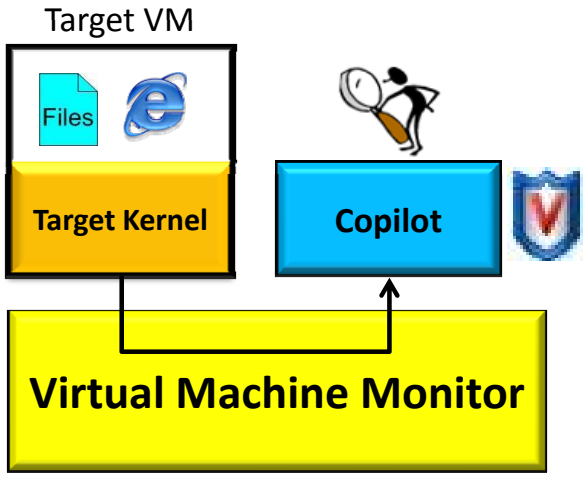
- In NDSS'03, Garfinkel et al. first proposed VMI, demonstrated for IDS
- Introspection routine is based on `crash` utility



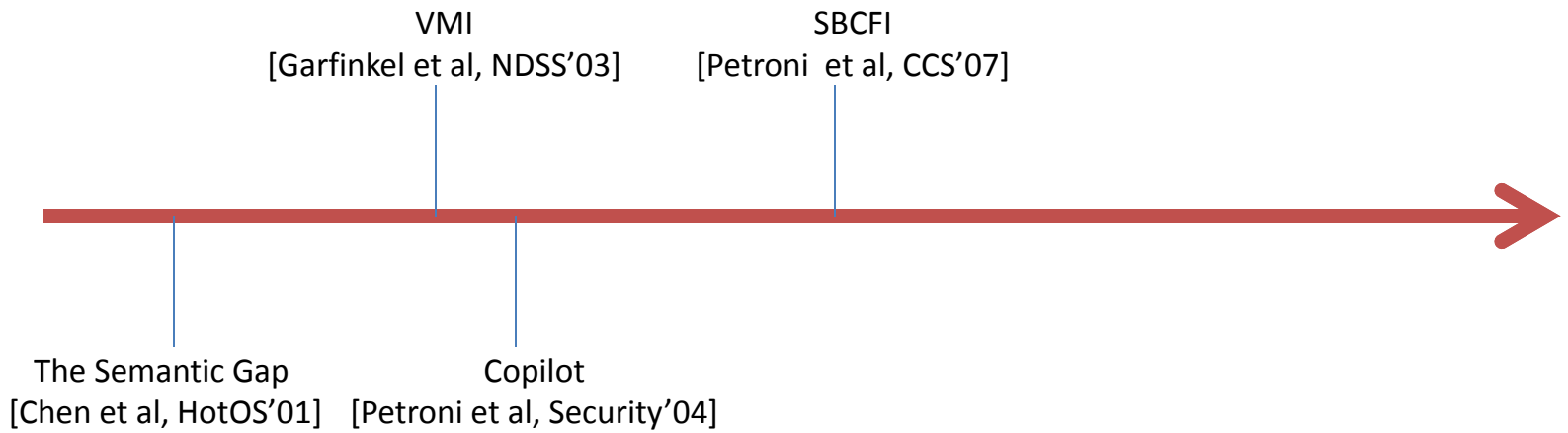
State-of-the-art in bridging the Semantic Gap

State-of-the-art in bridging the Semantic Gap

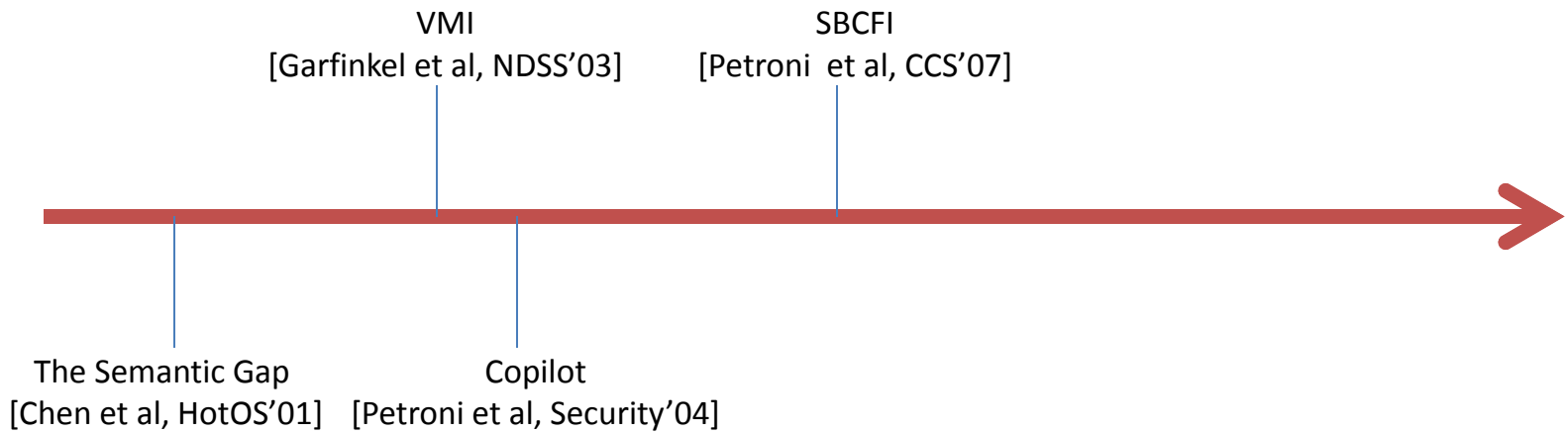
- In USENIX Security'04, Petroni et al. proposed Copilot
- Introspection routine is based on manually created code



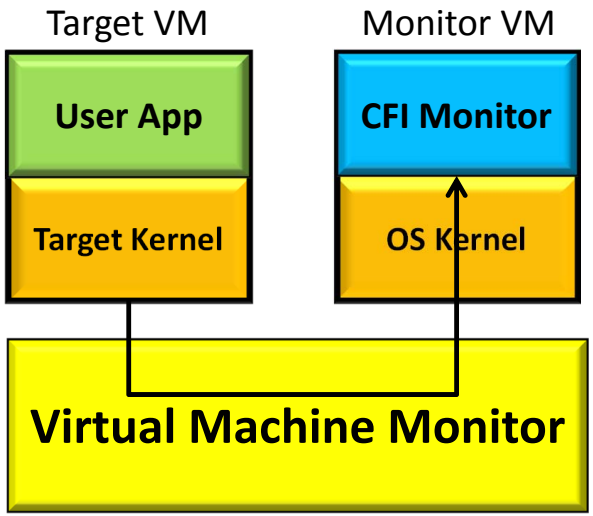
State-of-the-art in bridging the Semantic Gap



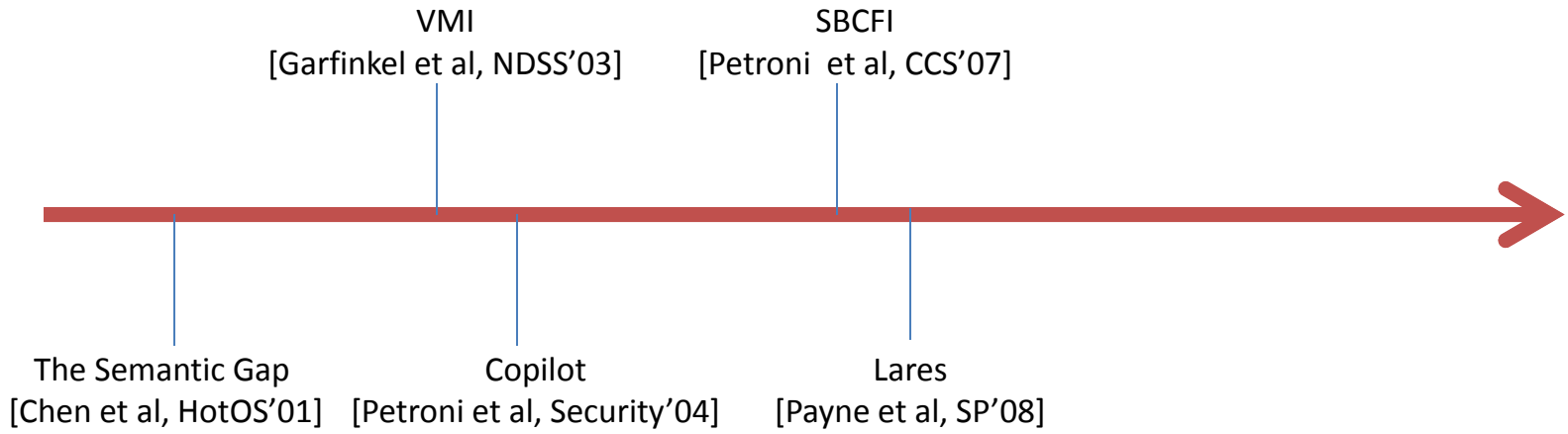
State-of-the-art in bridging the Semantic Gap



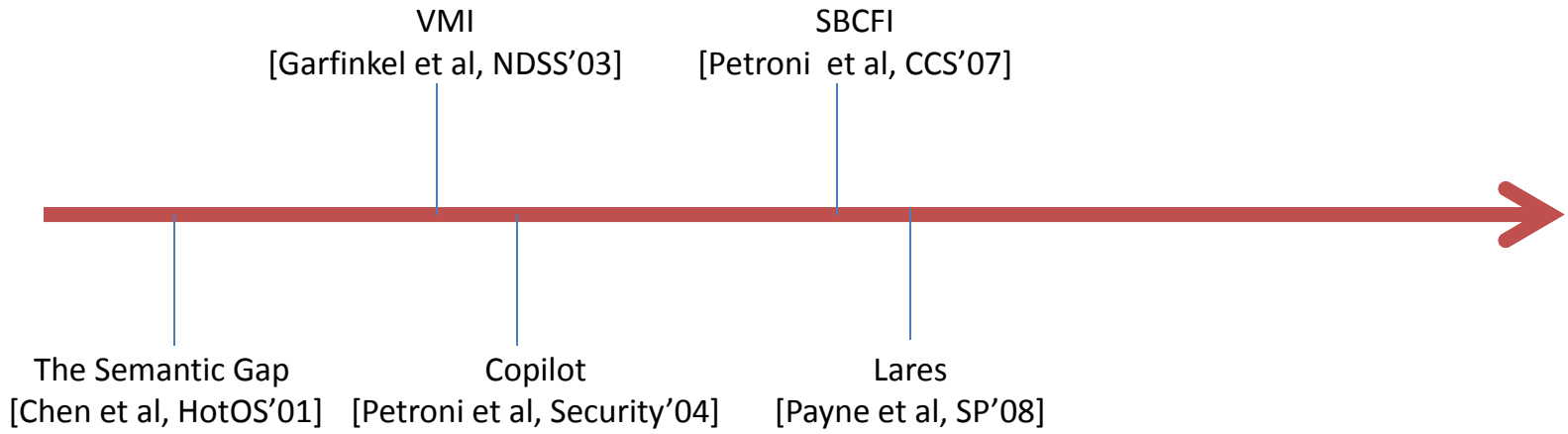
- In CCS'07, Petroni et al. proposed SBCFI
- Introspection routine is based on customized kernel source code



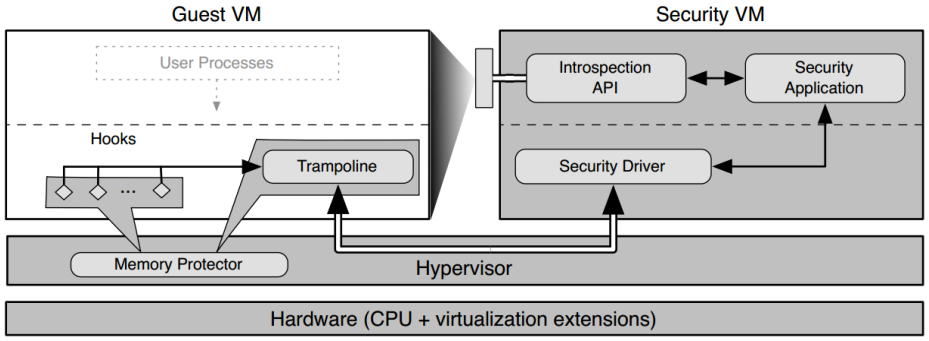
State-of-the-art in bridging the Semantic Gap



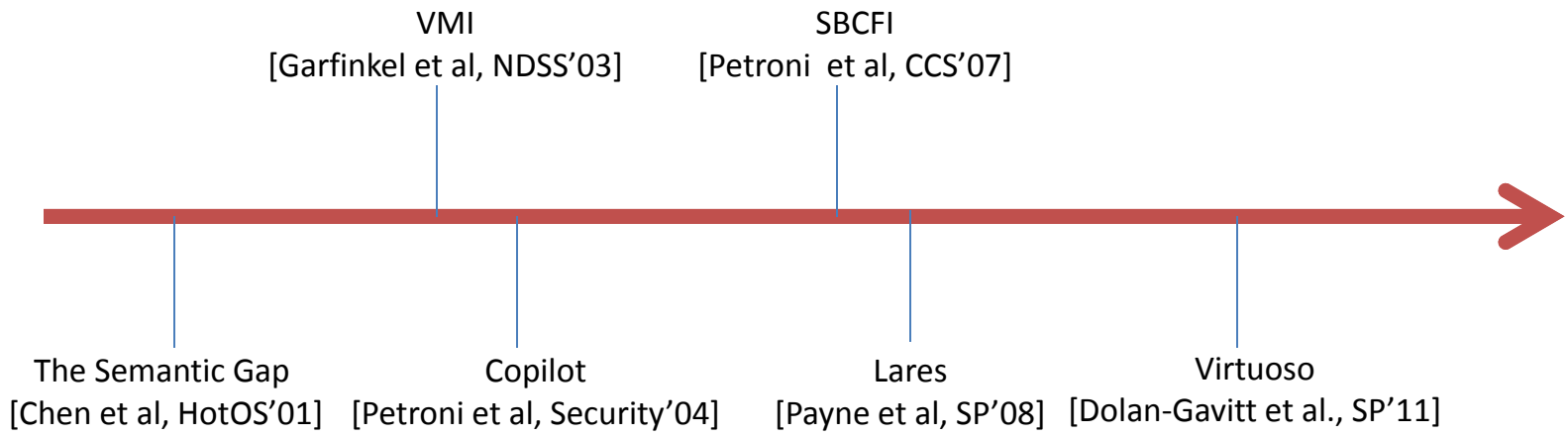
State-of-the-art in bridging the Semantic Gap



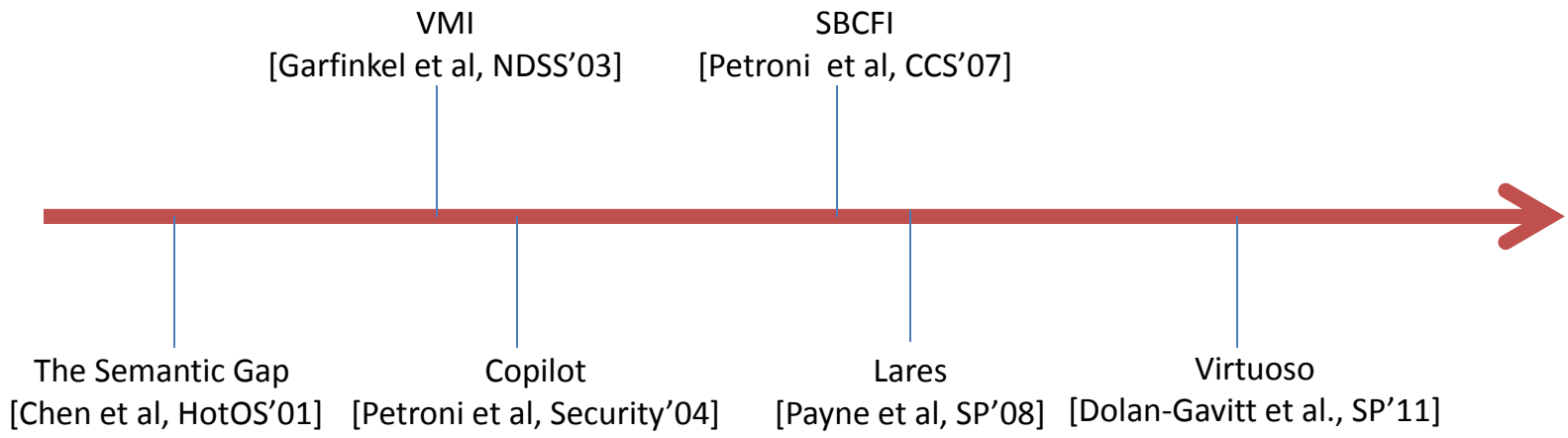
- In SP'08, Payne et al. proposed Lares
- Introspection routine is placed inside the guest OS with special help



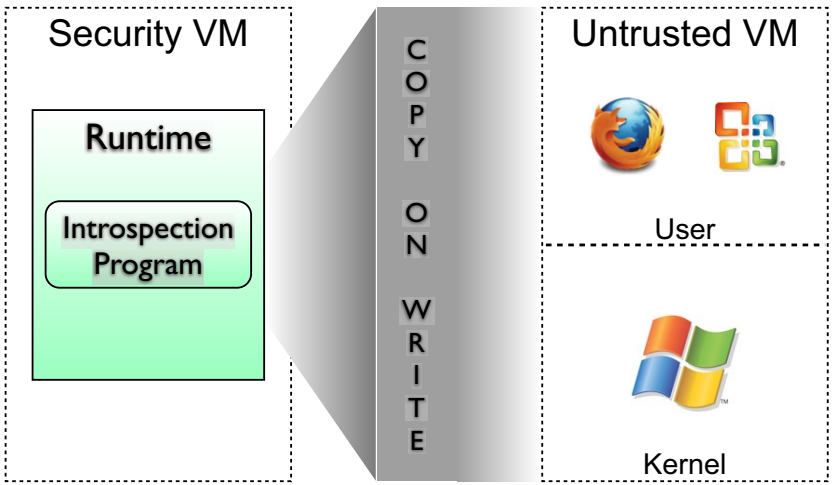
State-of-the-art in bridging the Semantic Gap



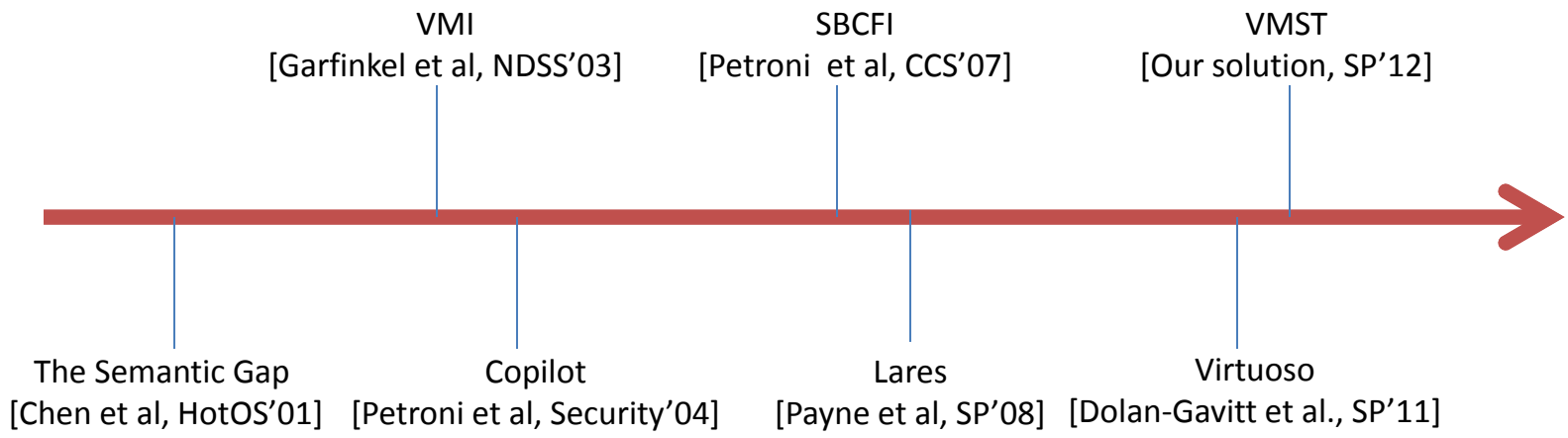
State-of-the-art in bridging the Semantic Gap



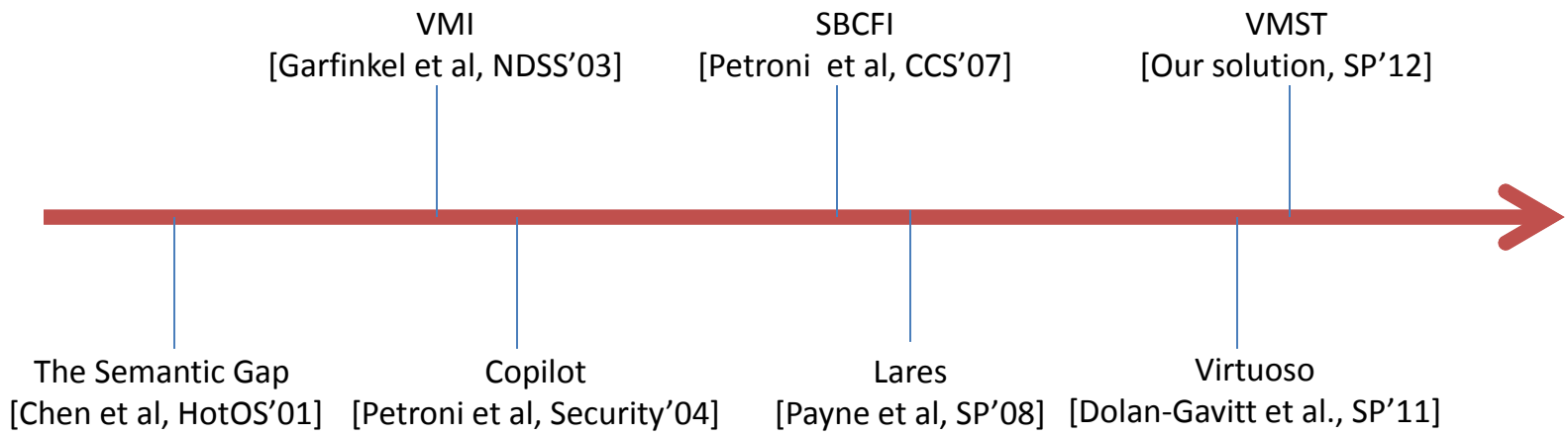
- In SP'11, Dolan-Gavitt et al. proposed Virtuoso
- Introspection routine is based on the trained user level and kernel level code



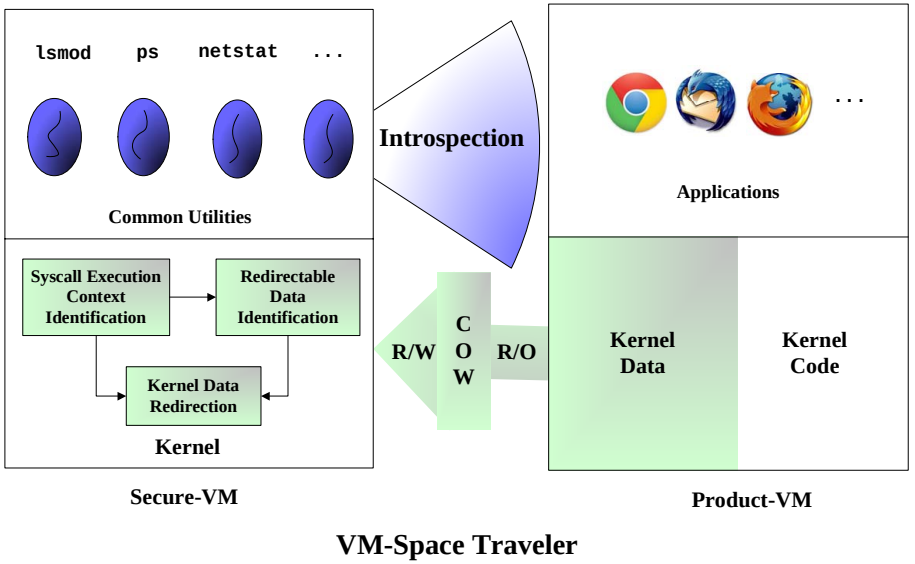
State-of-the-art in bridging the Semantic Gap



State-of-the-art in bridging the Semantic Gap

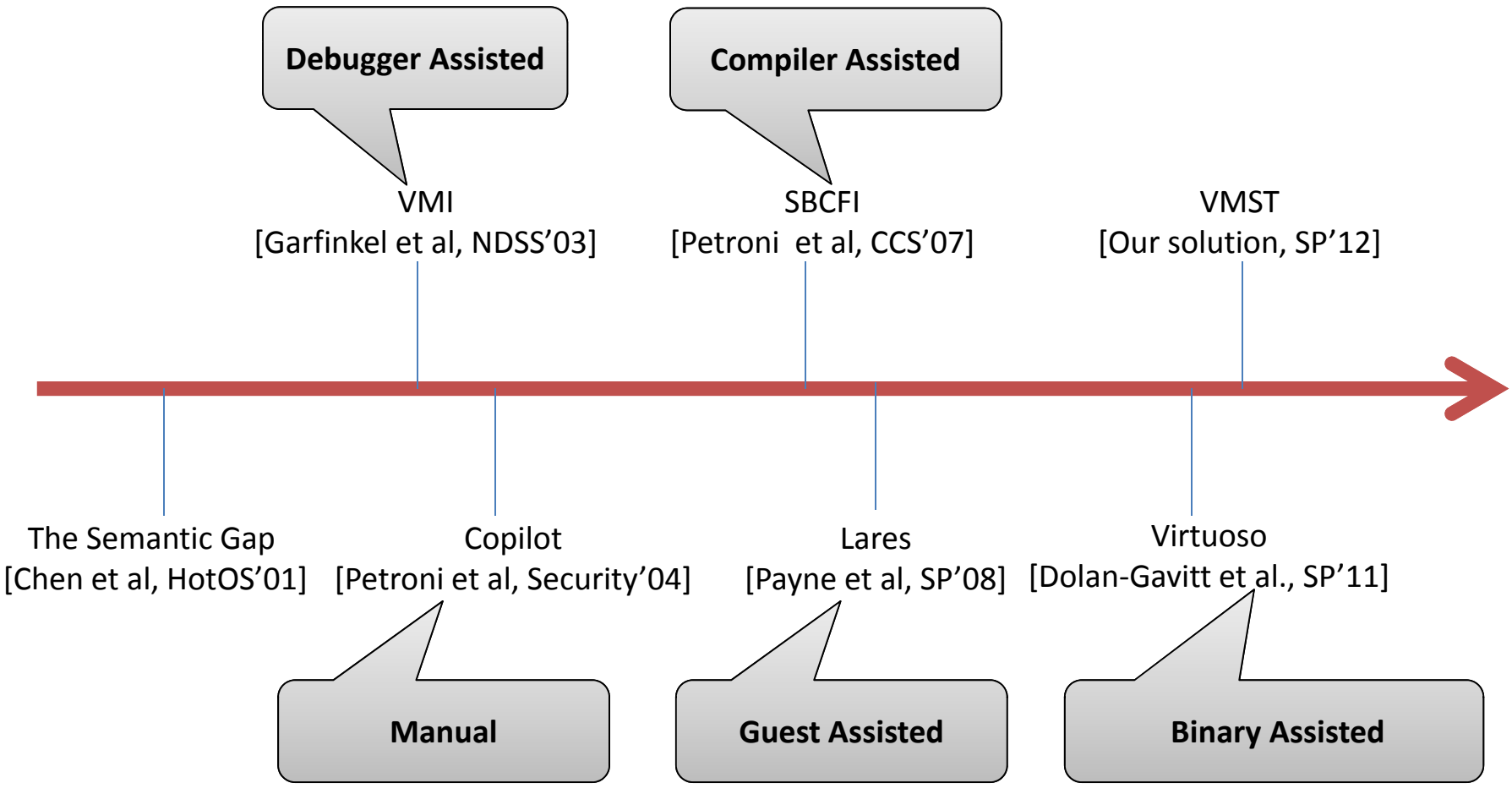


- In SP'12, we propose VM space traveling.
- Introspection routine is automatically generated from the native user level and kernel level code

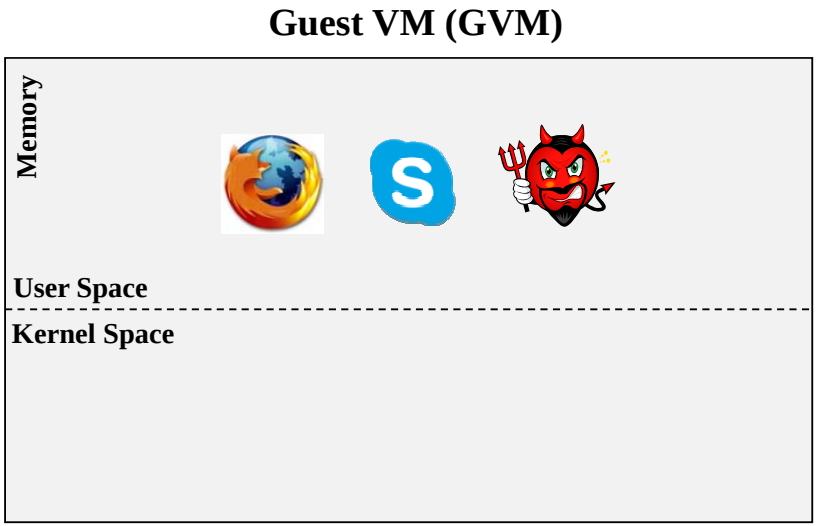


VM-Space Traveler

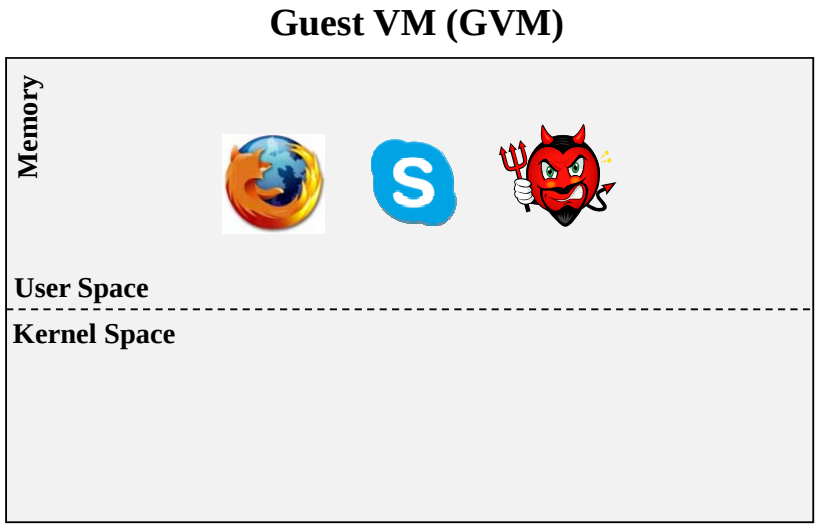
Approach Evolution



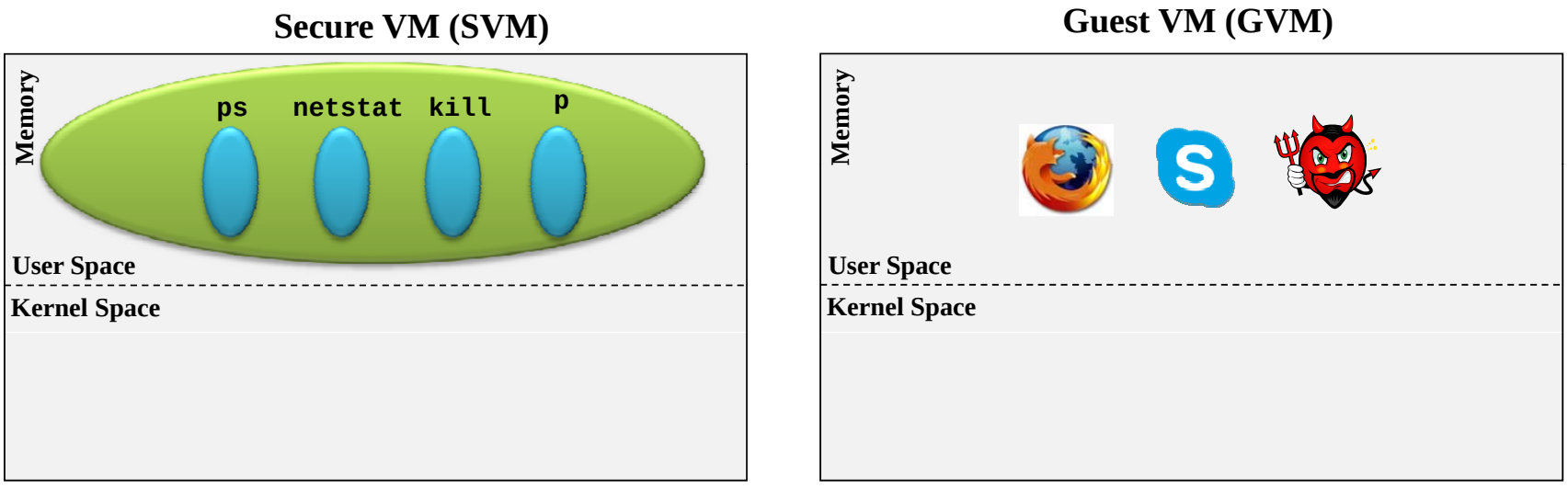
Two **Binary Code Reuse** Based Approaches



Two **Binary Code Reuse** Based Approaches



Two **Binary Code Reuse** Based Approaches



Using a trusted sibling VM to intropsect the running guest VM.

1. Redirect kernel data [SP'12, VEE'13, NDSS'14]
2. Redirect system call execution [USENIX ATC'14]

Approach-I: Redirect Kernel Data [SP'12]

In-VM `getpid` Program

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n",getpid());
7     return 0;
8 }
```

```
 1 execve("./getpid",..)           = 0
 2 brk(0)                    = 0x83b8000
 3 access("/etc/ld.so.nohwcap",.) = -1
...
23 getpid()                        = 13849
...
26 write(1, "pid=13849\n", 10)     = 10
27 exit_group(0)                   = ?
```

Approach-I: Redirect Kernel Data [SP'12]

In-VM `getpid` Program

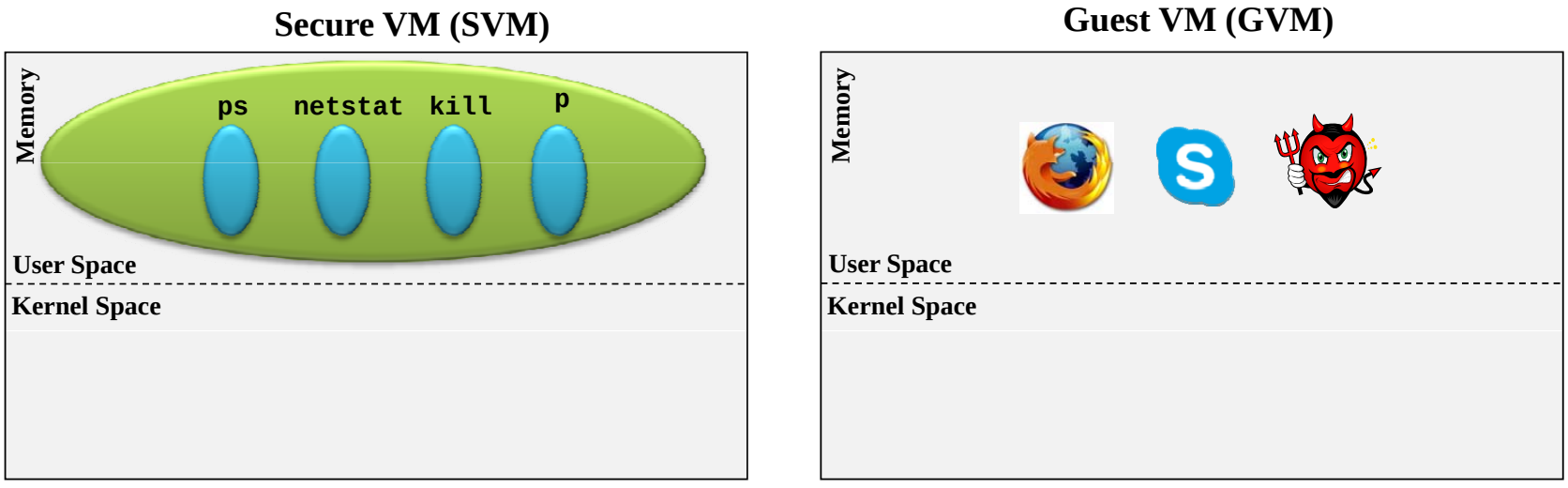
```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n",getpid());
7     return 0;
8 }
```

```
 1 execve("./getpid",..)           = 0
 2 brk(0)                    = 0x83b8000
 3 access("/etc/ld.so.nohwcap",.) = -1
...
23 getpid()                        = 13849
...
26 write(1, "pid=13849\n", 10)     = 10
27 exit_group(0)                   = ?
```

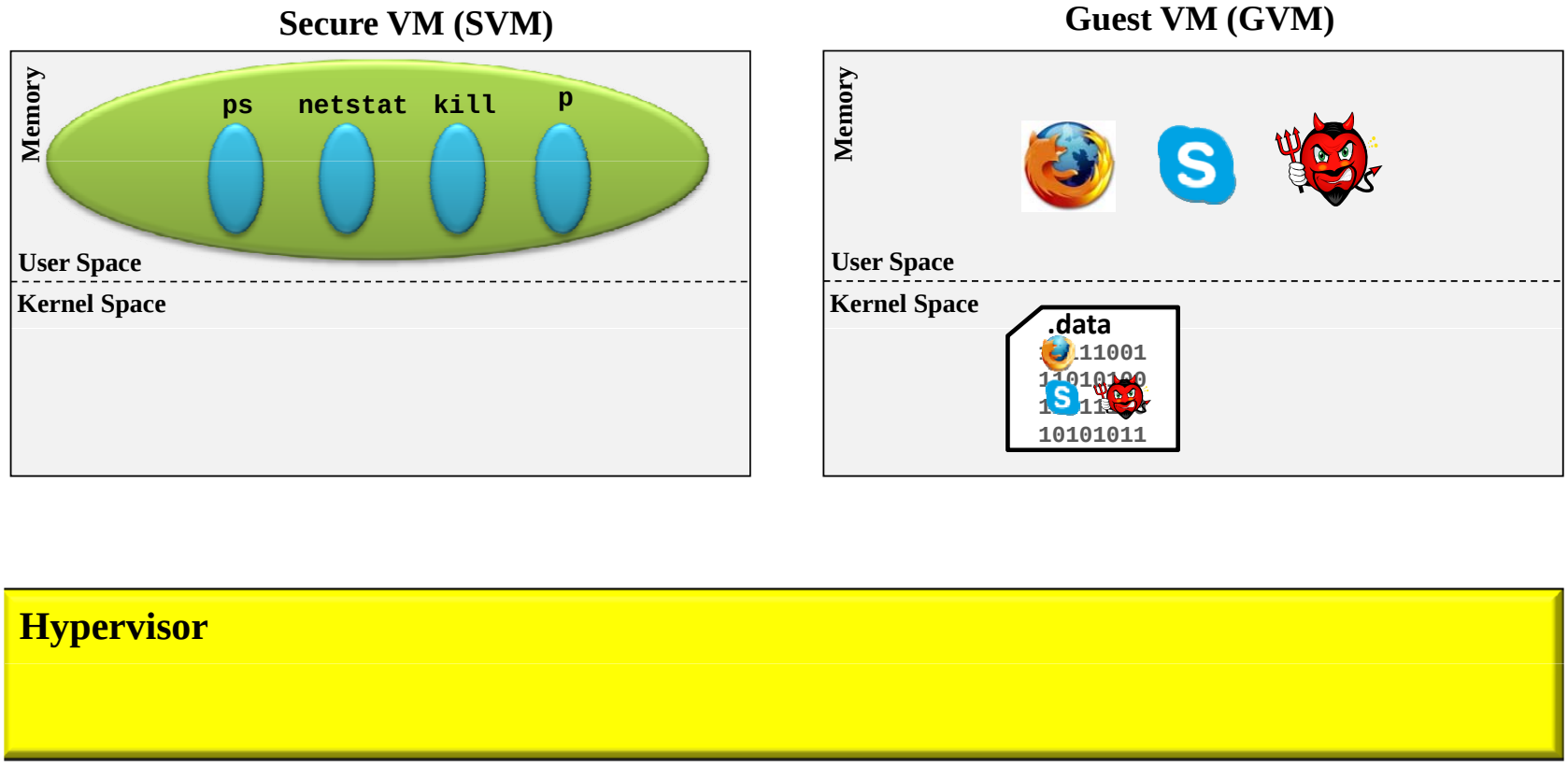
Key Insight

- Reuse the execution context of a native process in SVM.
- When syscall `getpid` executed, redirects the data of interest from the guest VM.

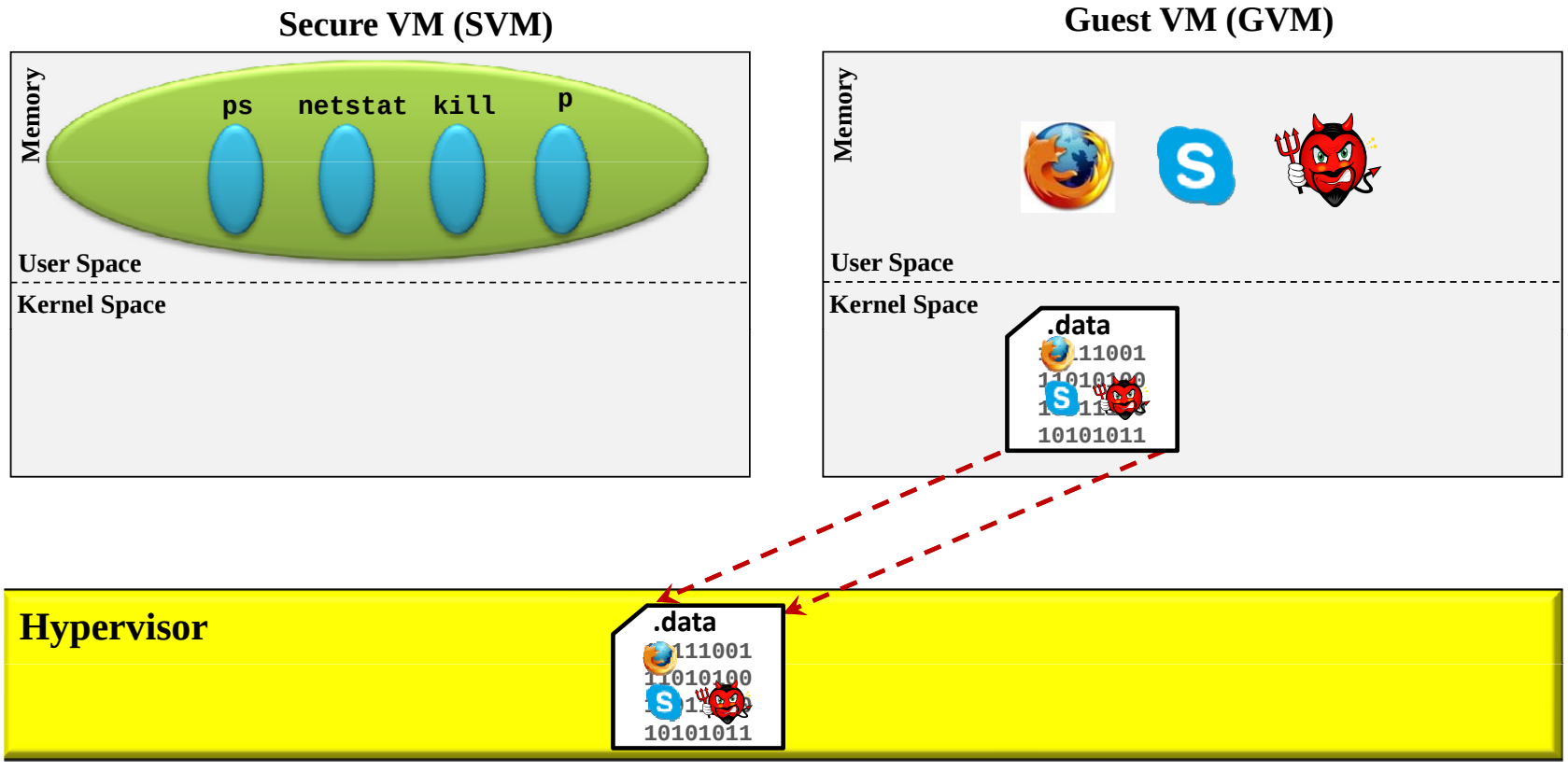
Approach-I: Redirect Kernel Data [SP'12]



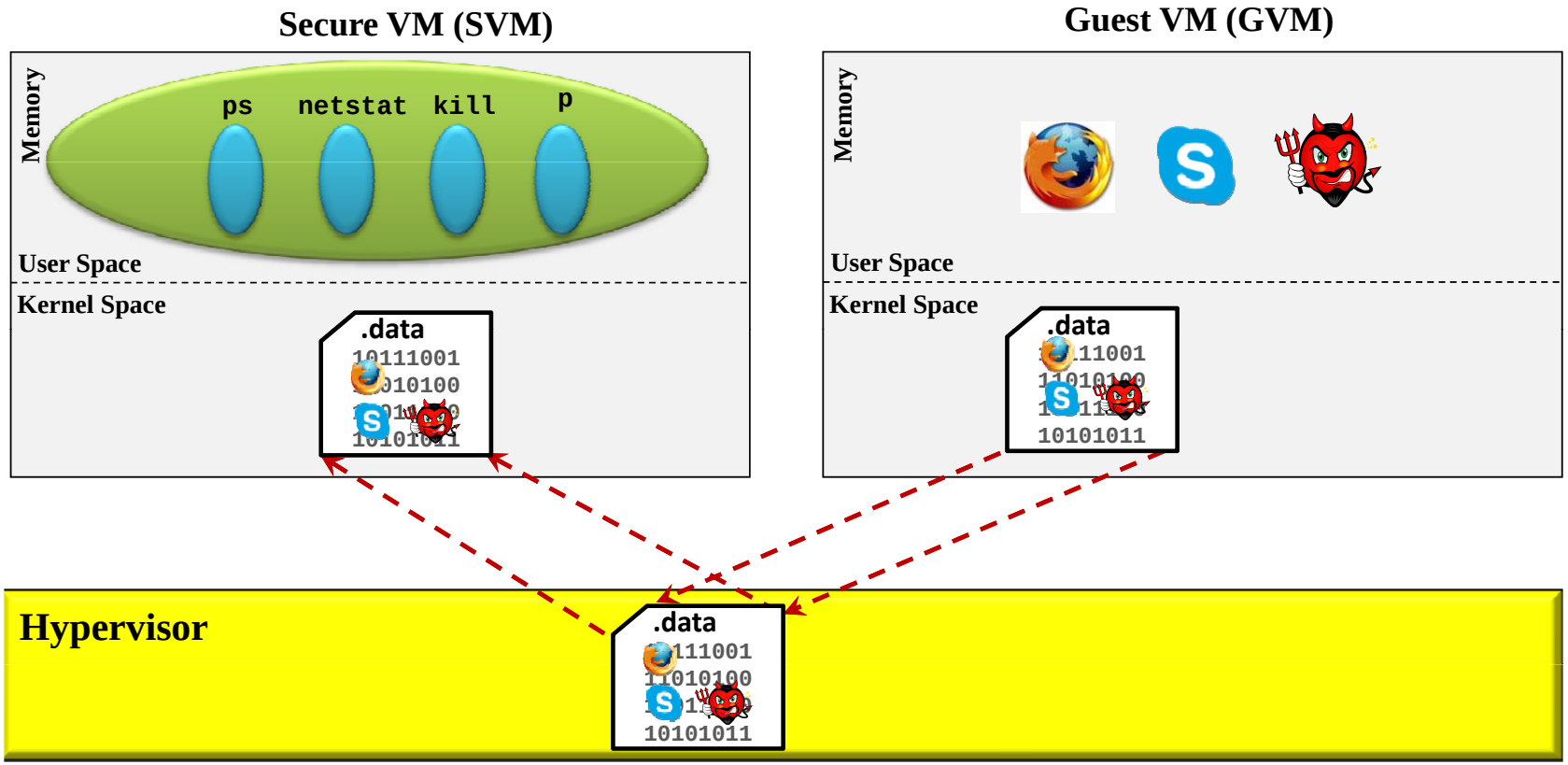
Approach-I: Redirect Kernel Data [SP'12]



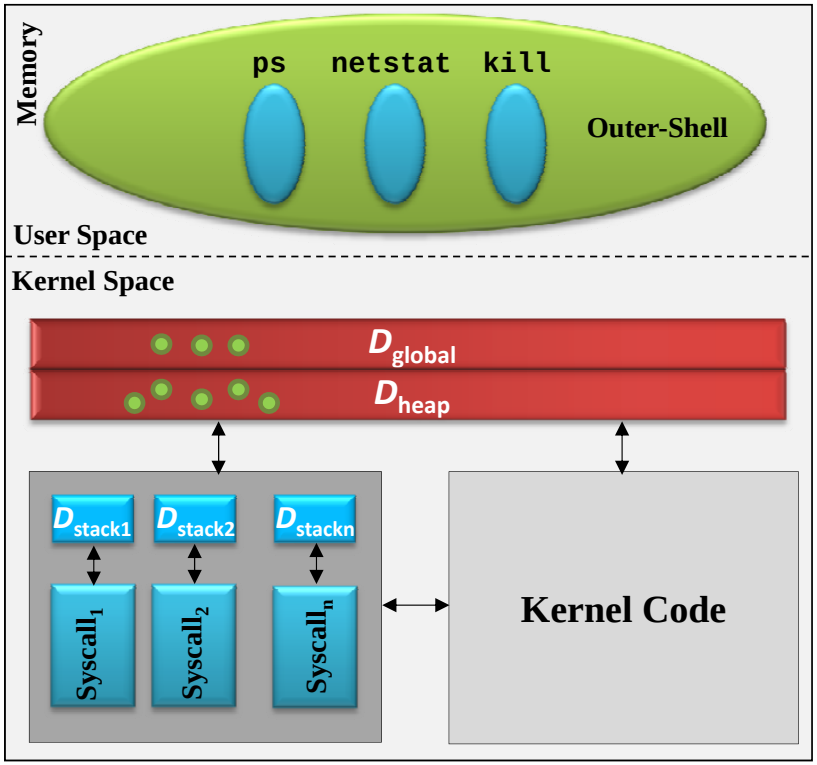
Approach-I: Redirect Kernel Data [SP'12]



Approach-I: Redirect Kernel Data [SP'12]

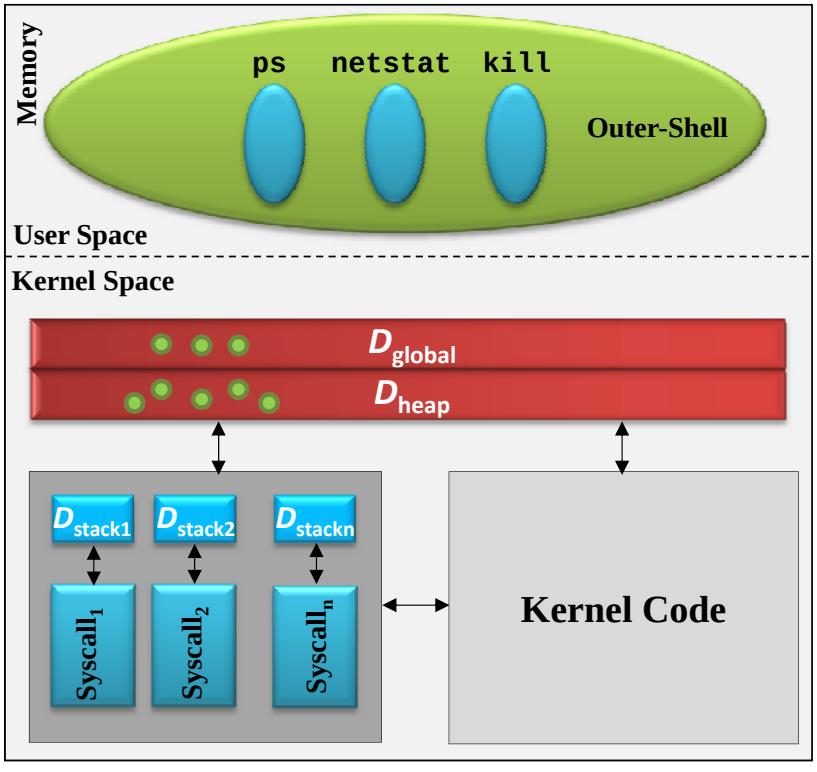


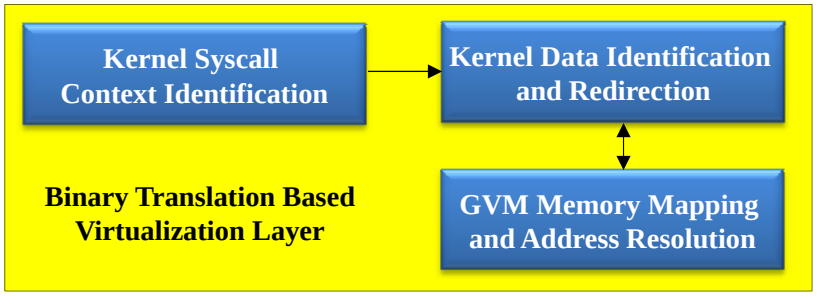
Approach-I: Design & Implementation [SP'12]



Secure VM (SVM)

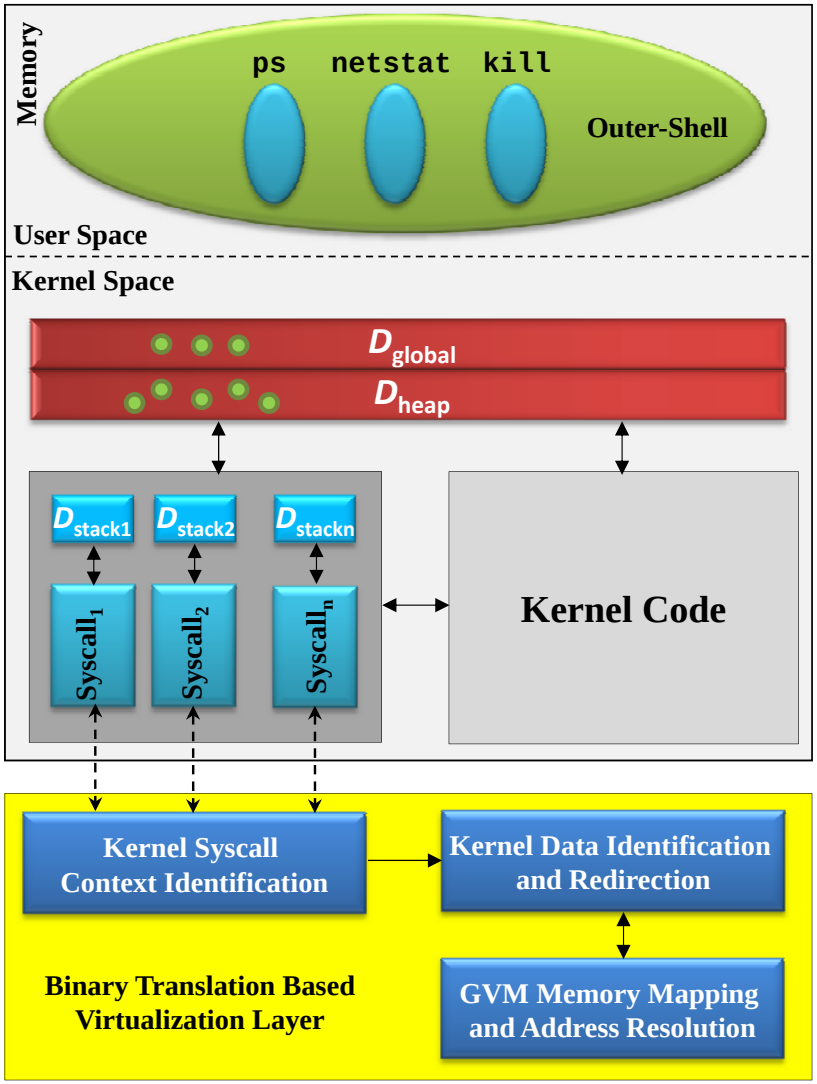
Approach-I: Design & Implementation [SP'12]





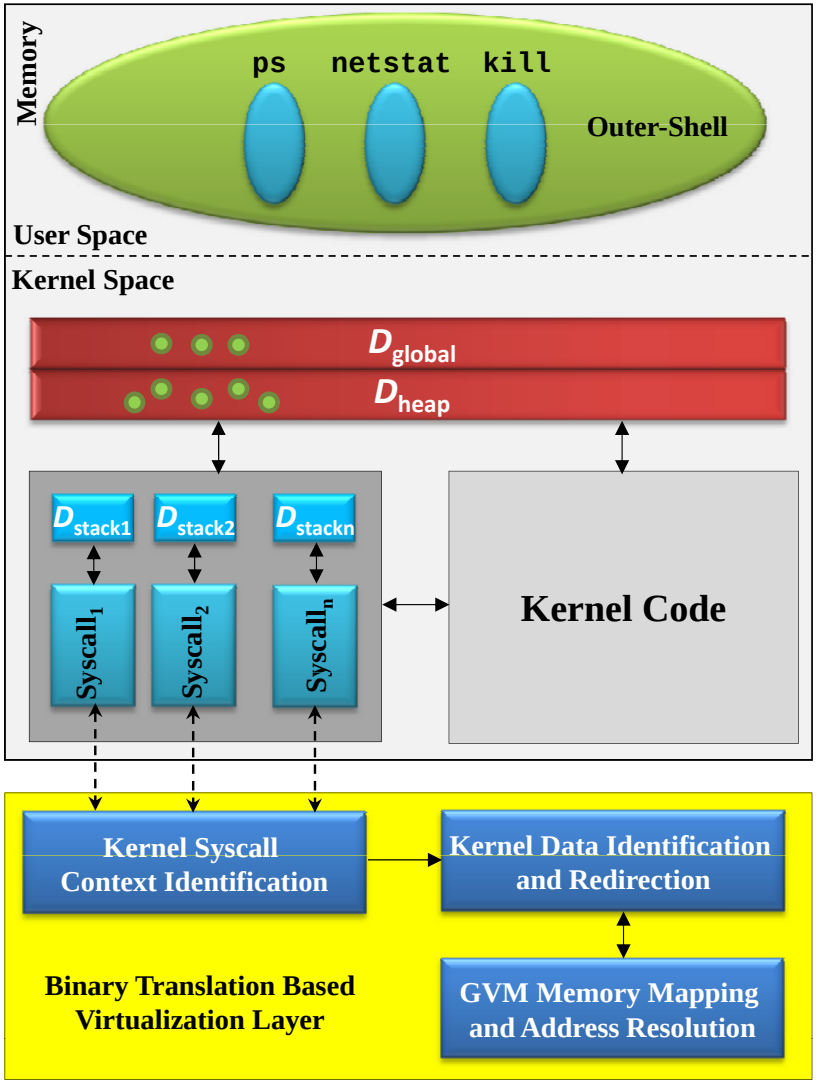
Secure VM (SVM)

Approach-I: Design & Implementation [SP'12]

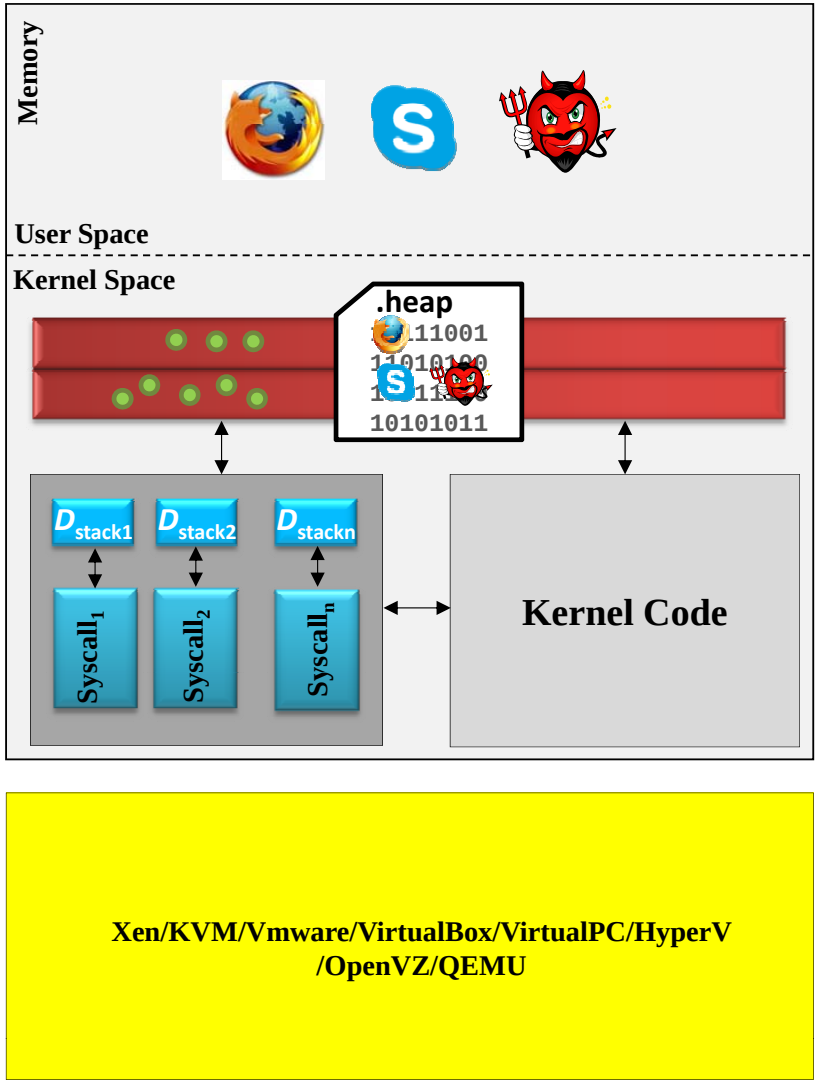


Secure VM (SVM)

Approach-I: Design & Implementation [SP'12]

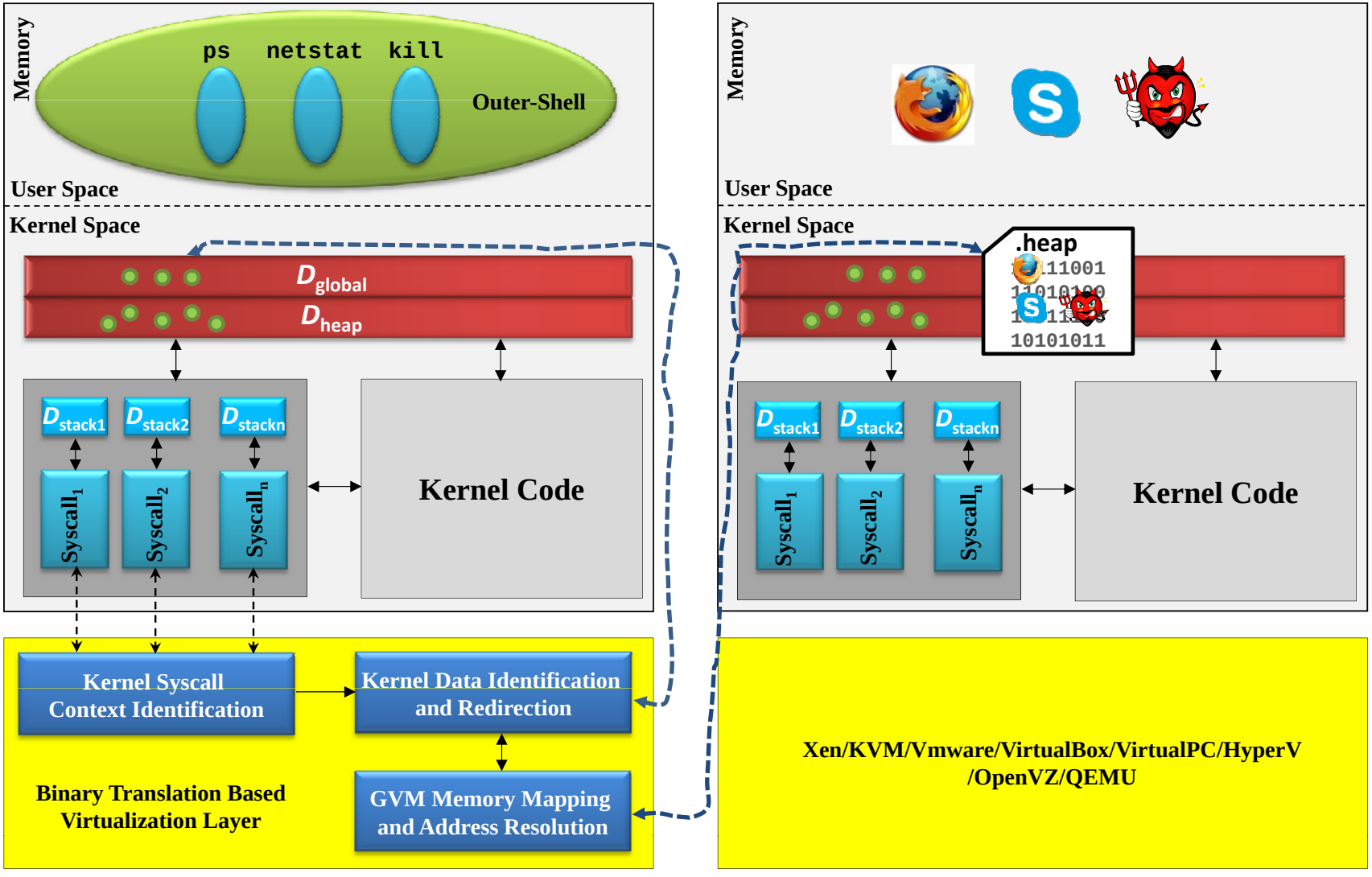


Secure VM (SVM)



Guest VM (GVM)

Approach-I: Design & Implementation [SP'12]



Approach-II: Redirect System call Execution [ATC'14]

In-VM `getpid` Program

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n",getpid());
7     return 0;
8 }
```

```
 1 execve("./getpid",..)           = 0
 2 brk(0)                    = 0x83b8000
 3 access("/etc/ld.so.nohwcap",.) = -1
...
23 getpid()                        = 13849
...
26 write(1, "pid=13849\n", 10)     = 10
27 exit_group(0)                   = ?
```

Approach-II: Redirect System call Execution [ATC'14]

In-VM `getpid` Program

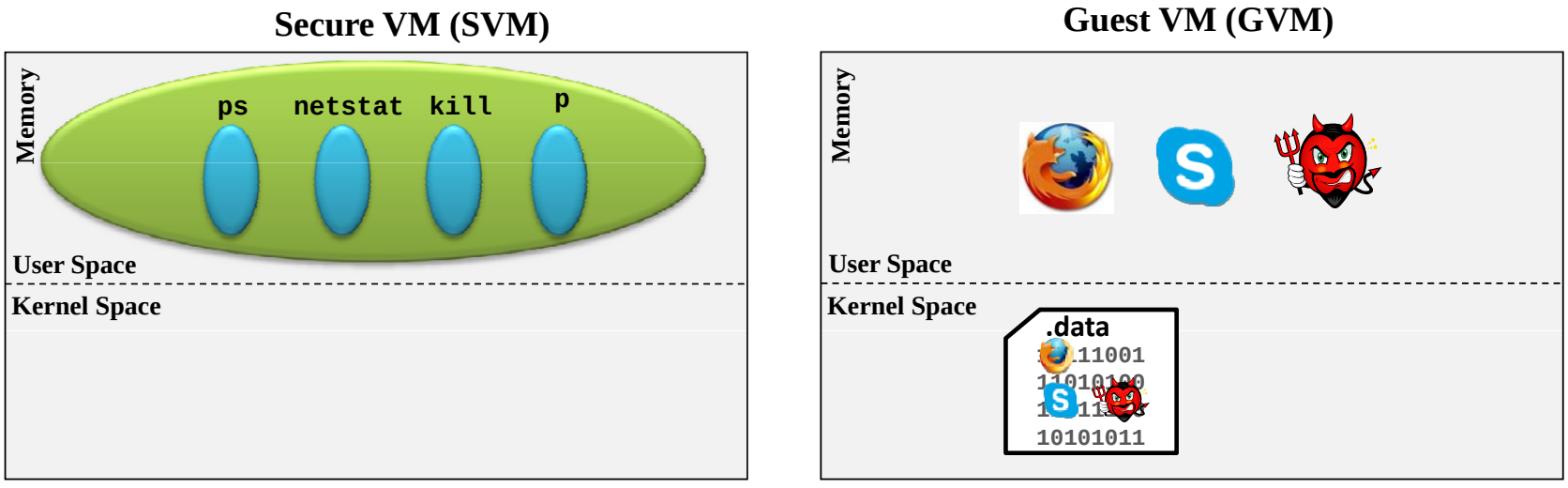
```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n",getpid());
7     return 0;
8 }
```

```
 1 execve("./getpid",..)           = 0
 2 brk(0)                    = 0x83b8000
 3 access("/etc/ld.so.nohwcap",.) = -1
...
23 getpid()                        = 13849
...
26 write(1, "pid=13849\n", 10)     = 10
27 exit_group(0)                   = ?
```

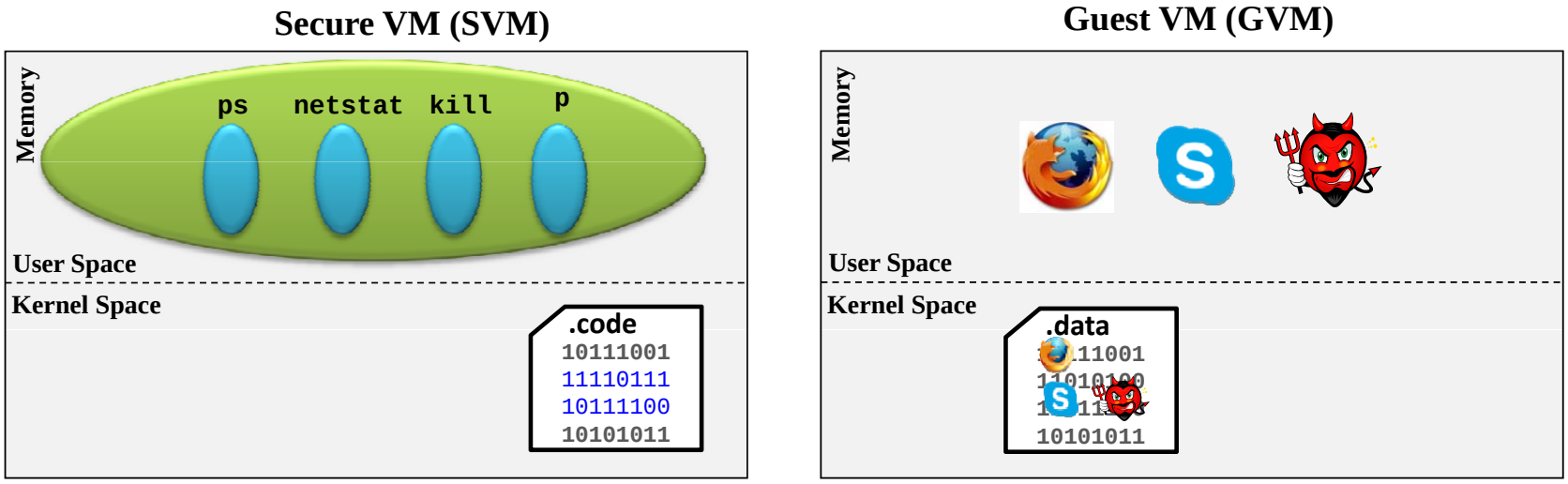
Key Insight

- System call is the only interface to request OS service.
- Pushing the execution of `getpid` system call from SVM to GVM.

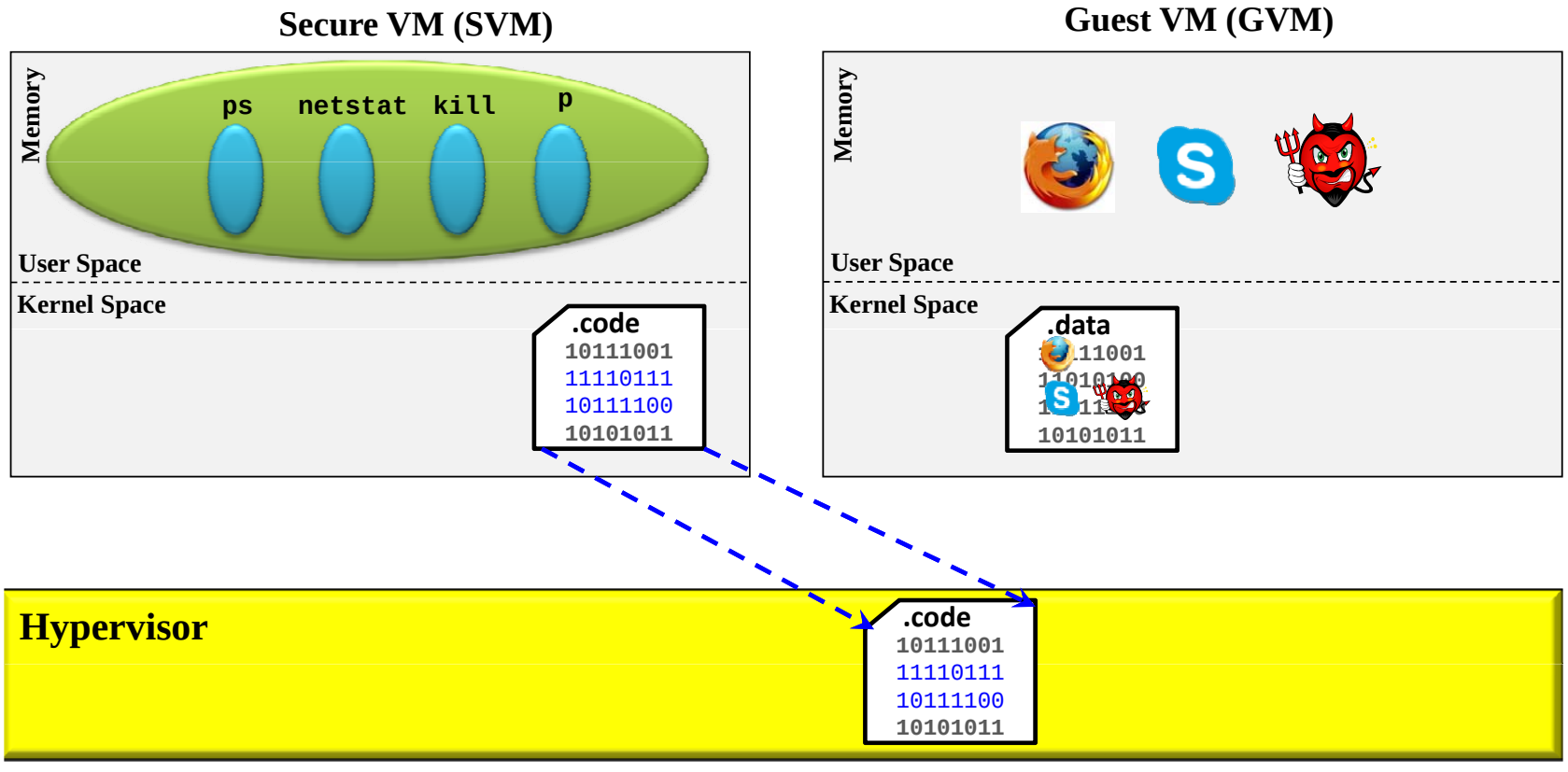
Approach-II: Redirect System call Execution [ATC'14]



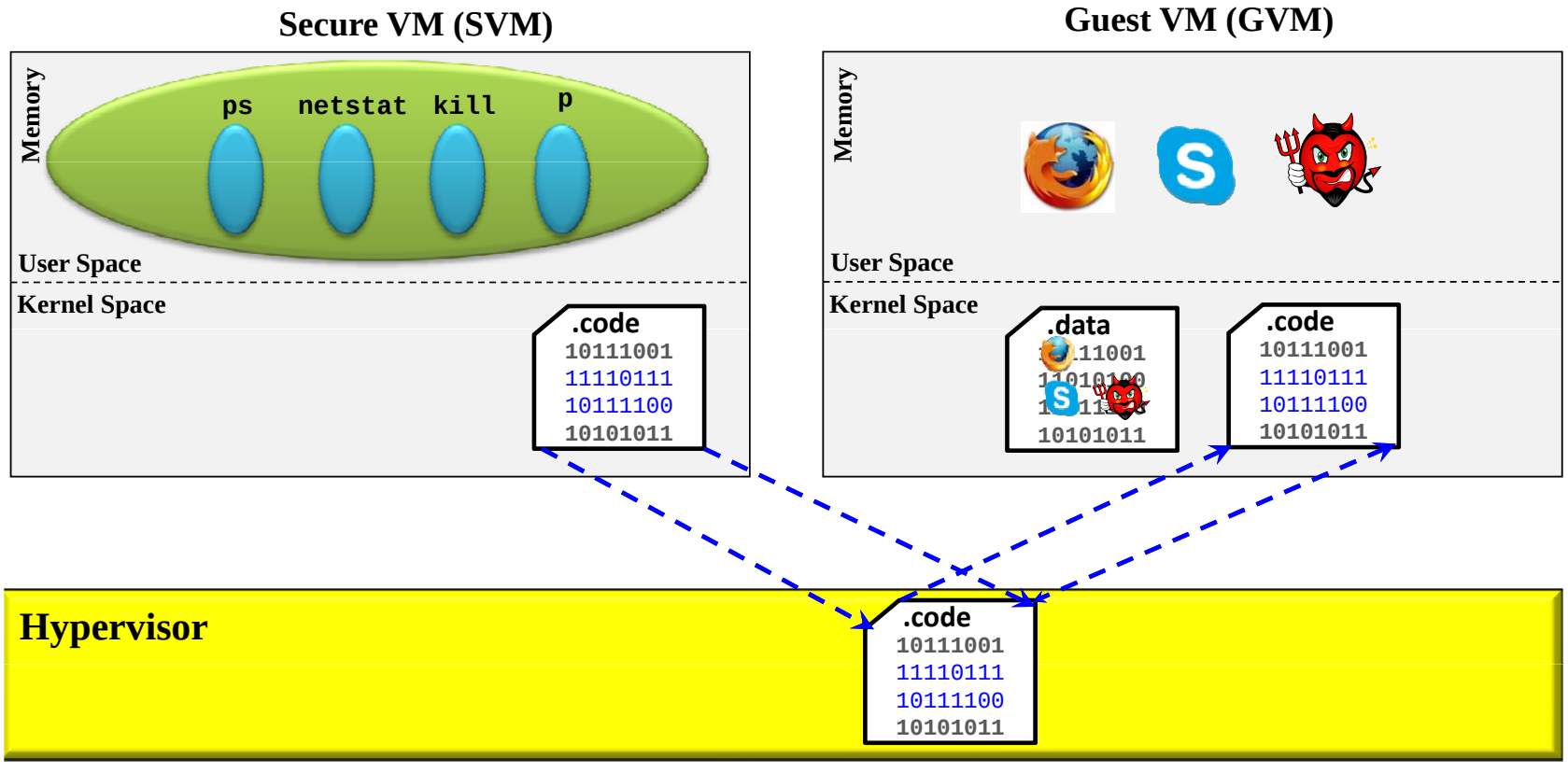
Approach-II: Redirect System call Execution [ATC'14]



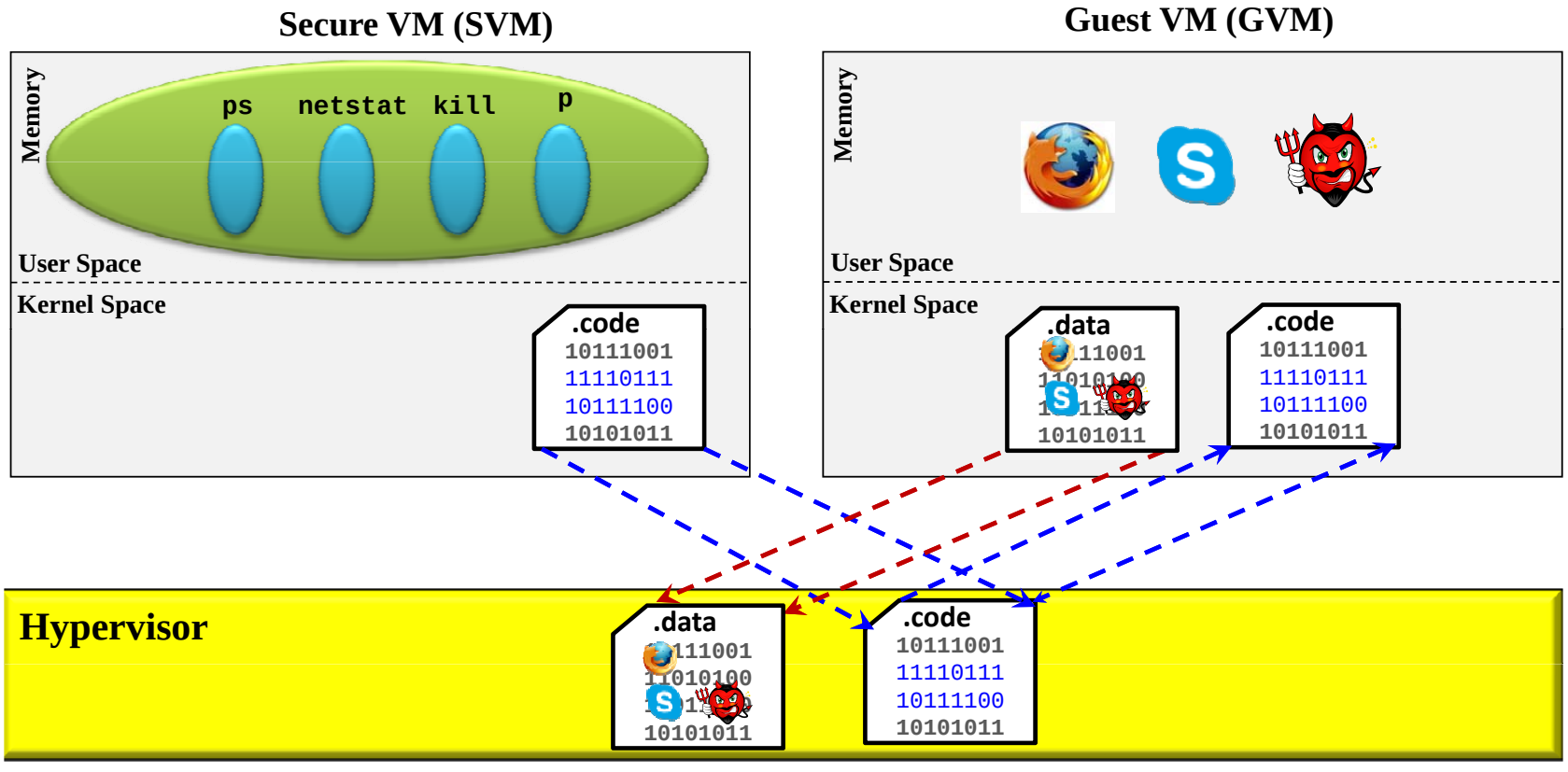
Approach-II: Redirect System call Execution [ATC'14]



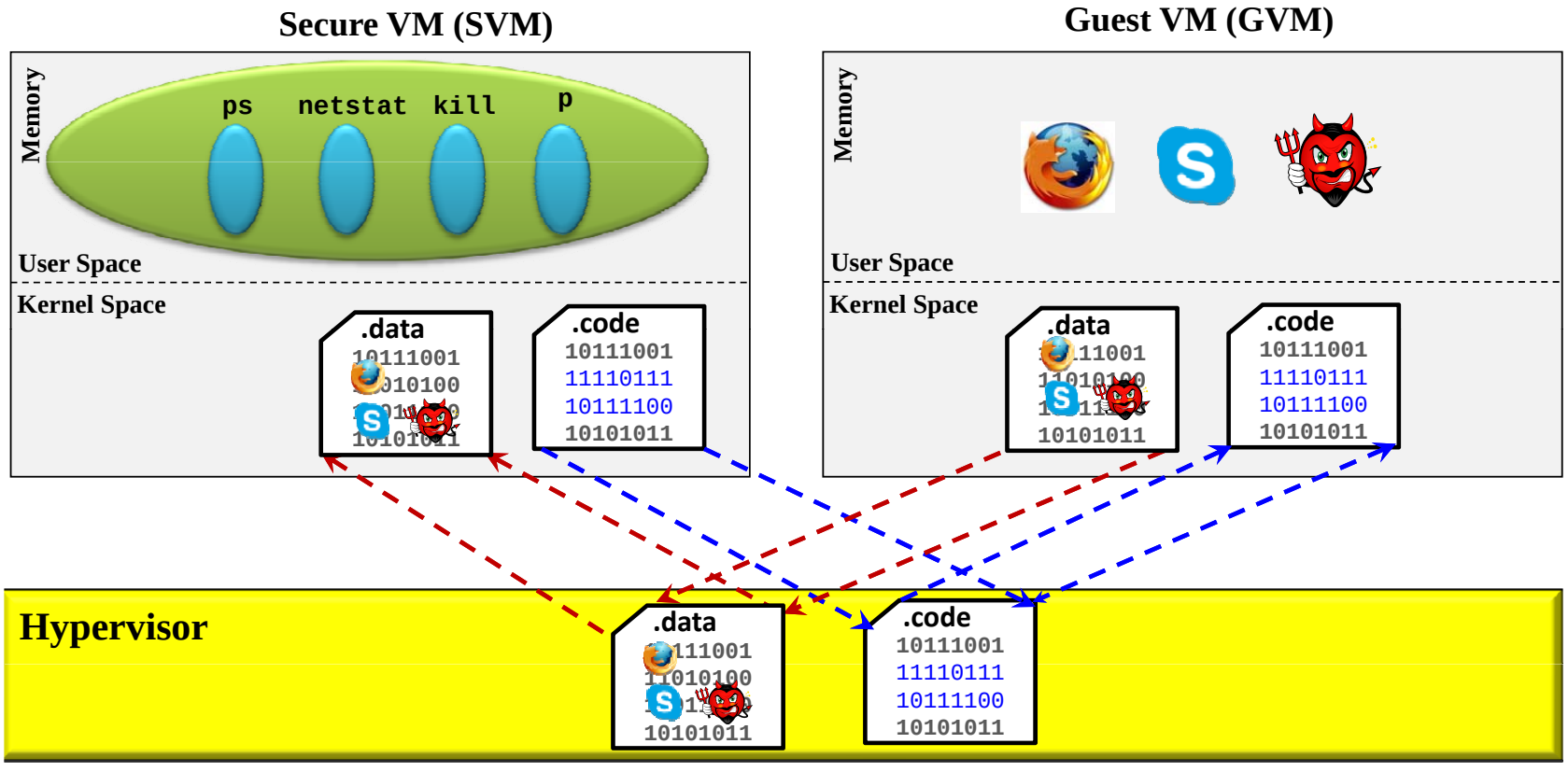
Approach-II: Redirect System call Execution [ATC'14]



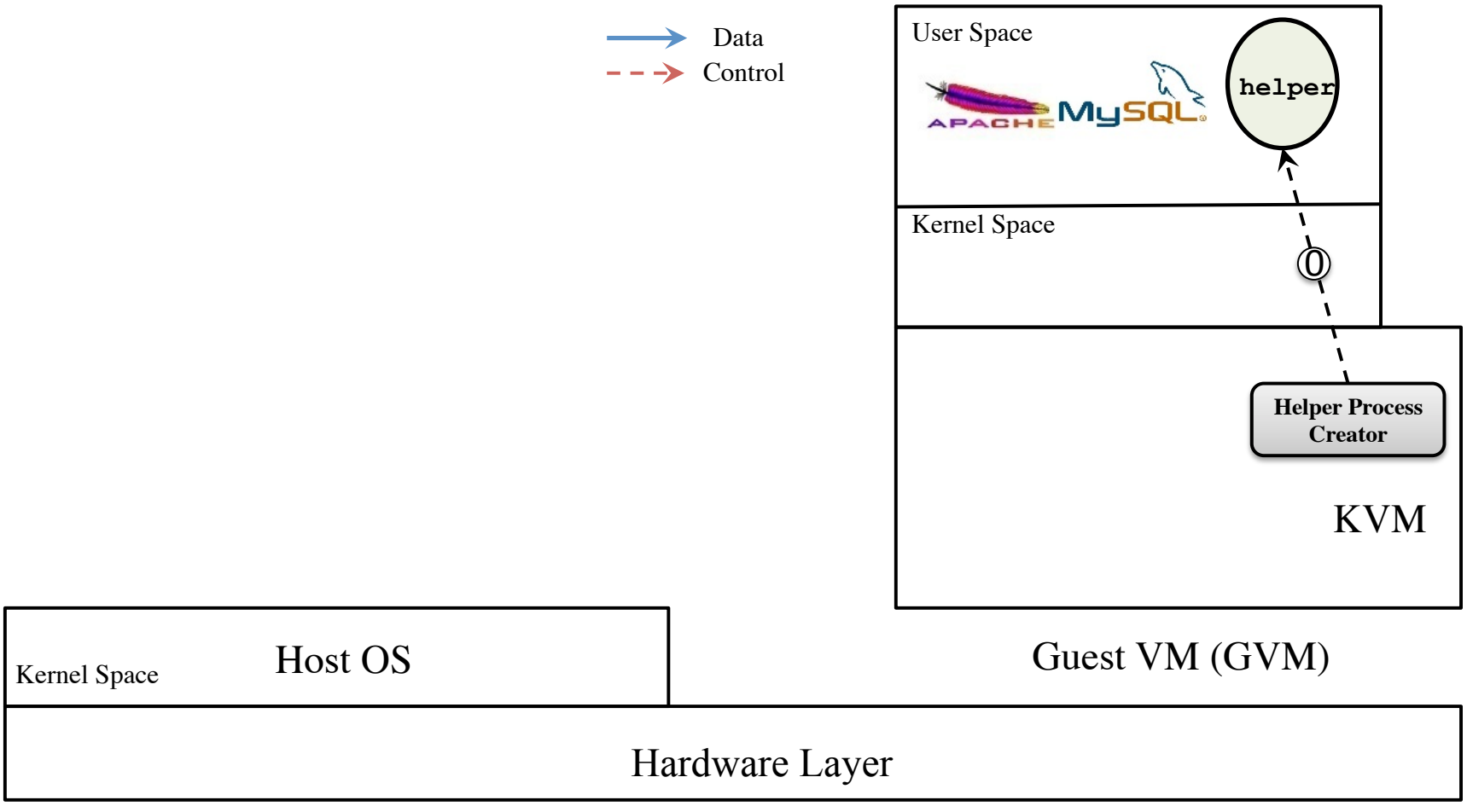
Approach-II: Redirect System call Execution [ATC'14]



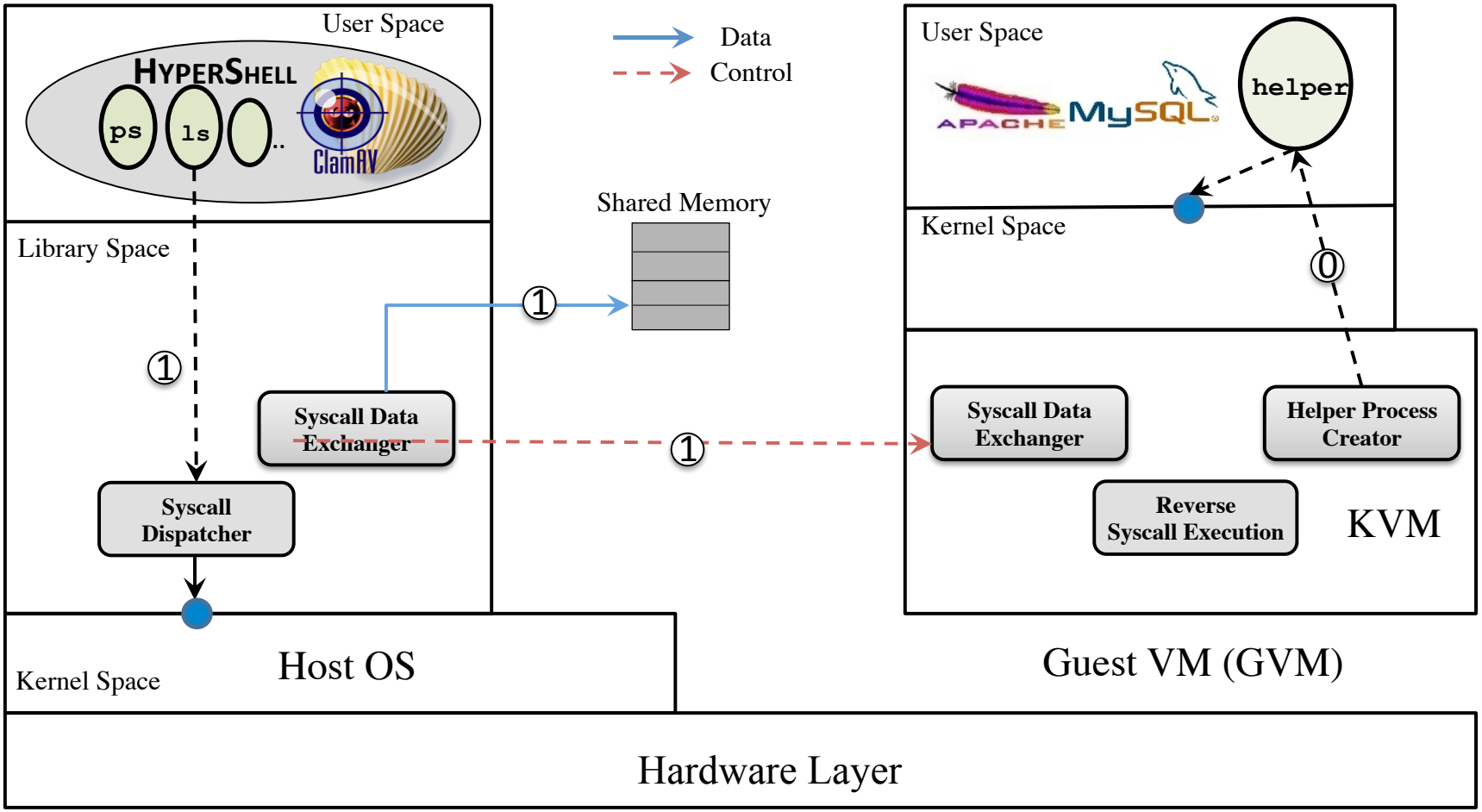
Approach-II: Redirect System call Execution [ATC'14]



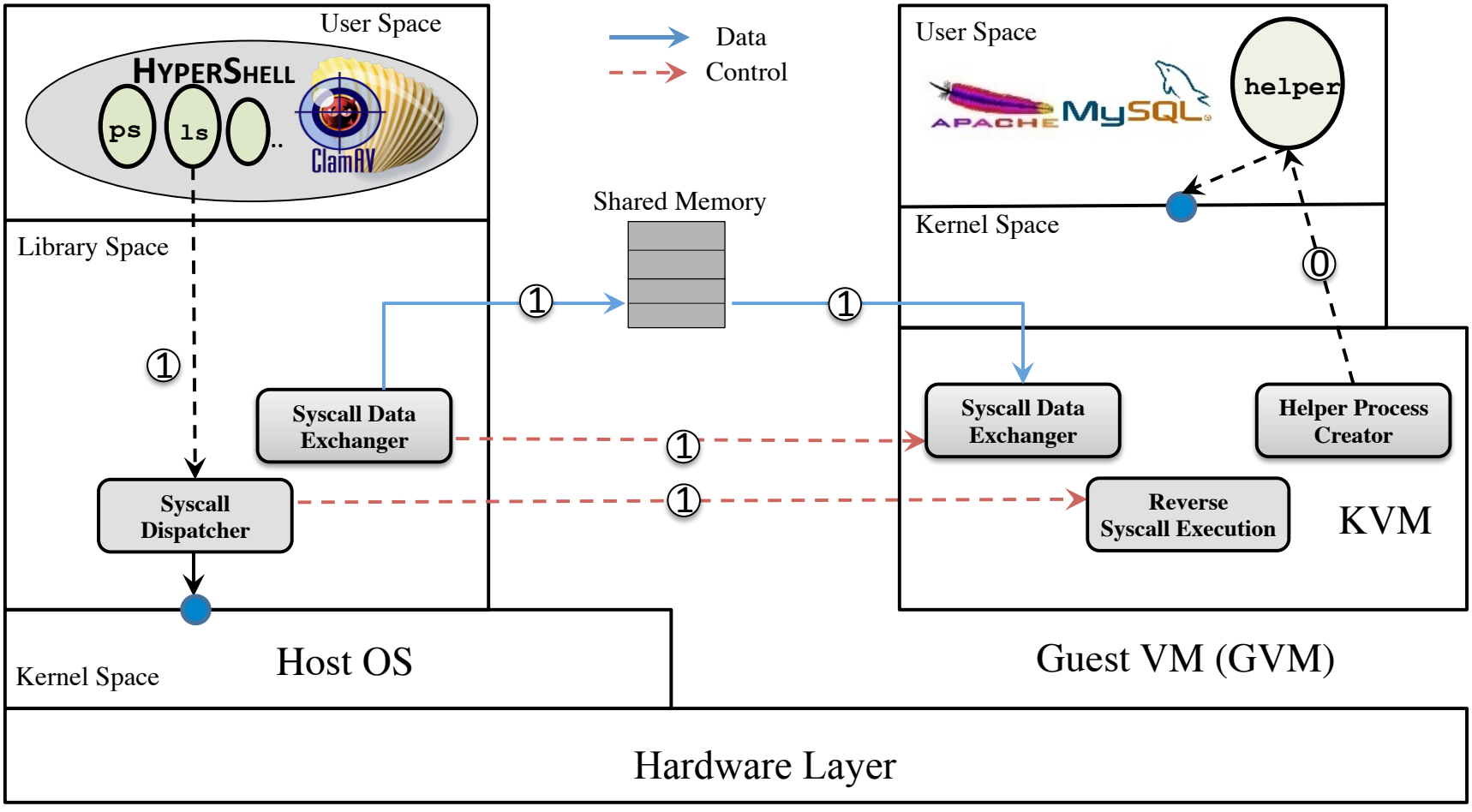
Approach-II: Design & Implementation [ATC'14]



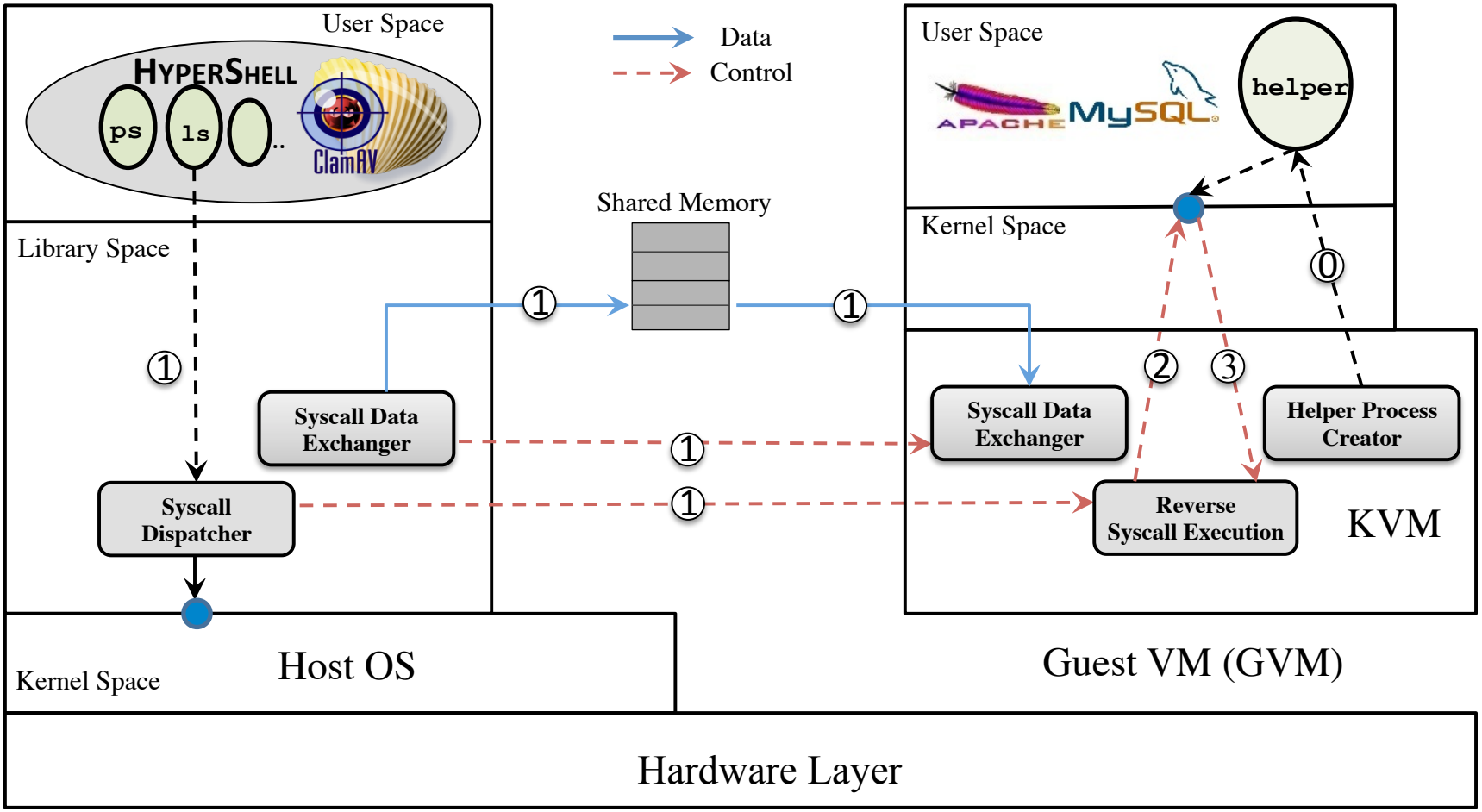
Approach-II: Design & Implementation [ATC'14]



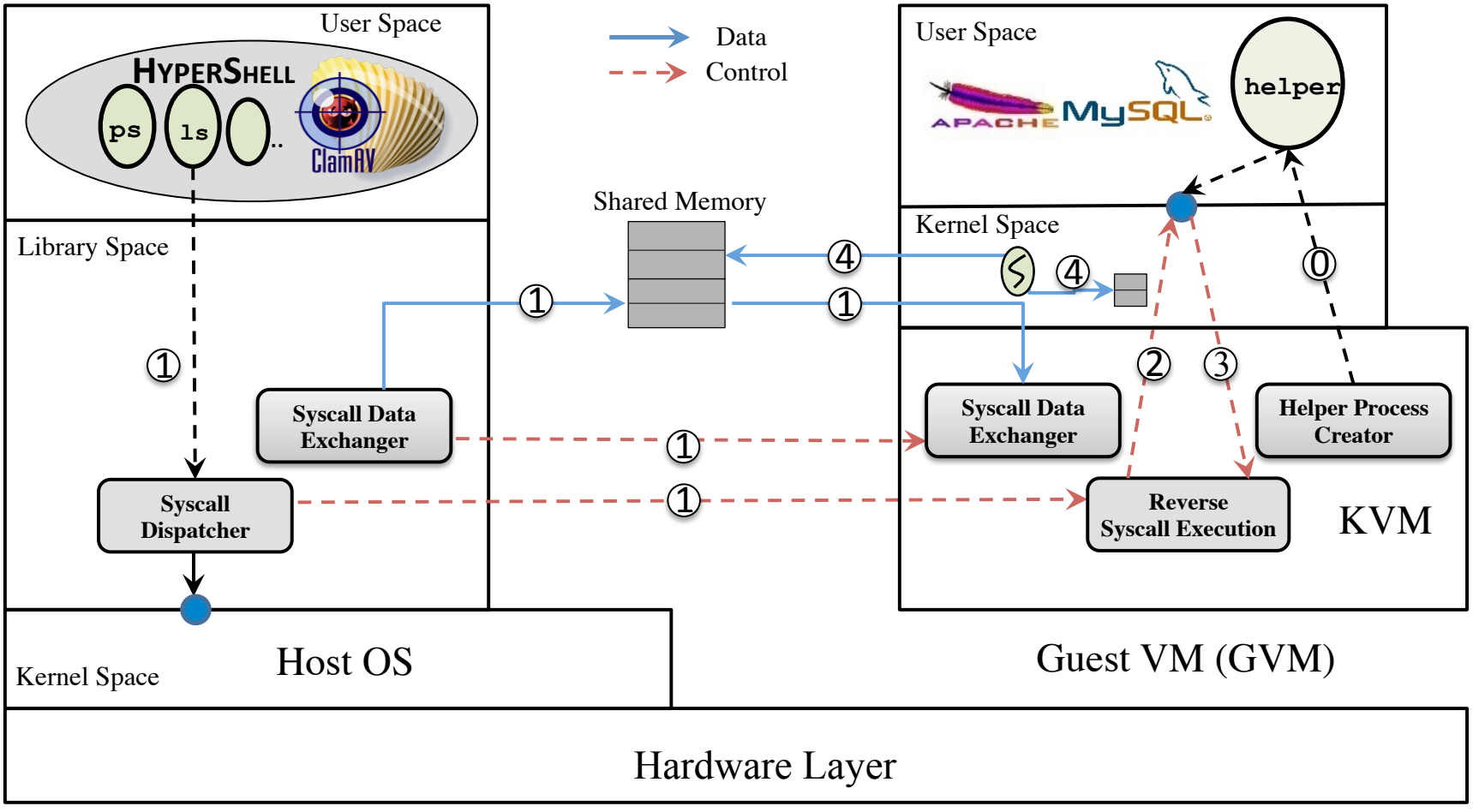
Approach-II: Design & Implementation [ATC'14]



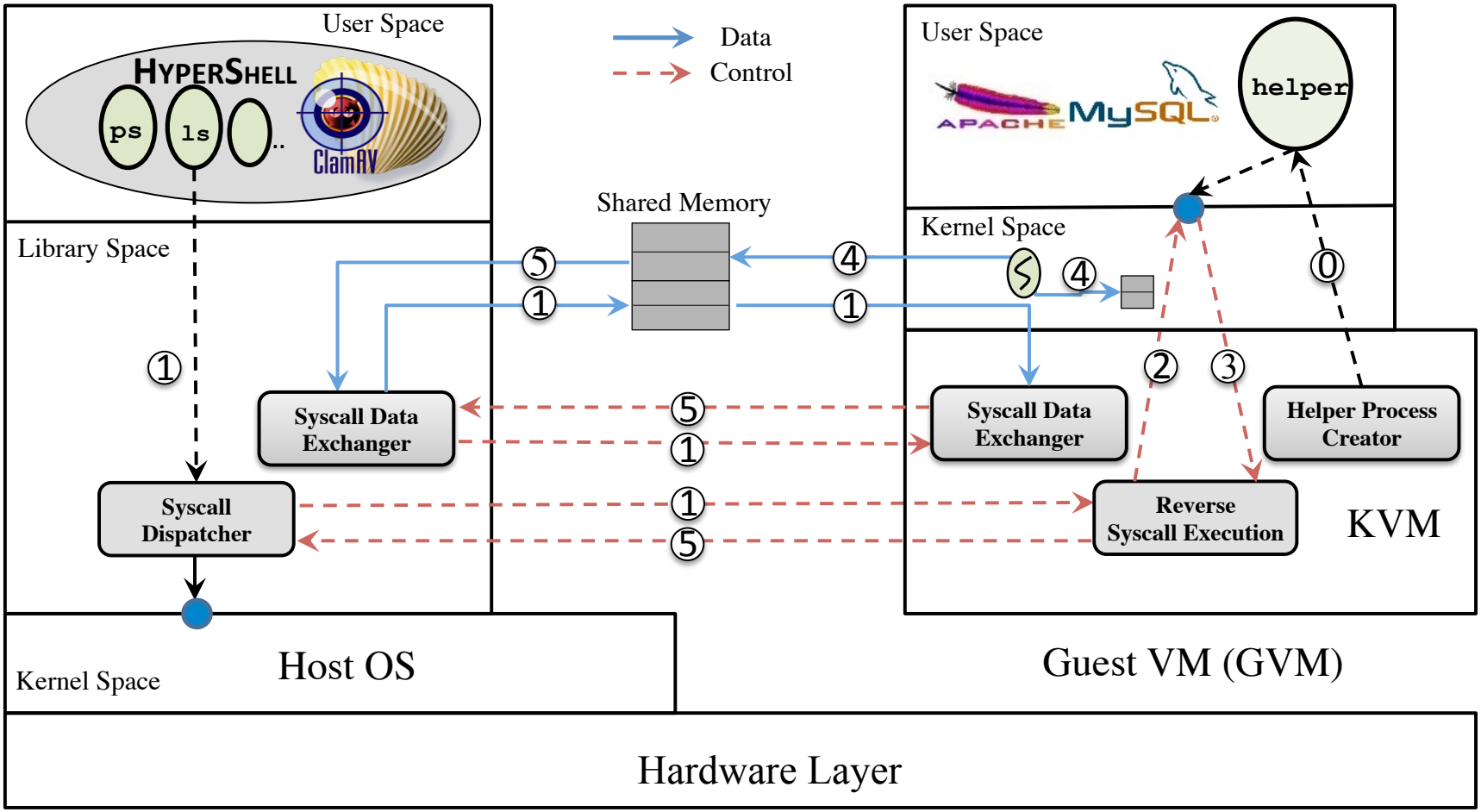
Approach-II: Design & Implementation [ATC'14]



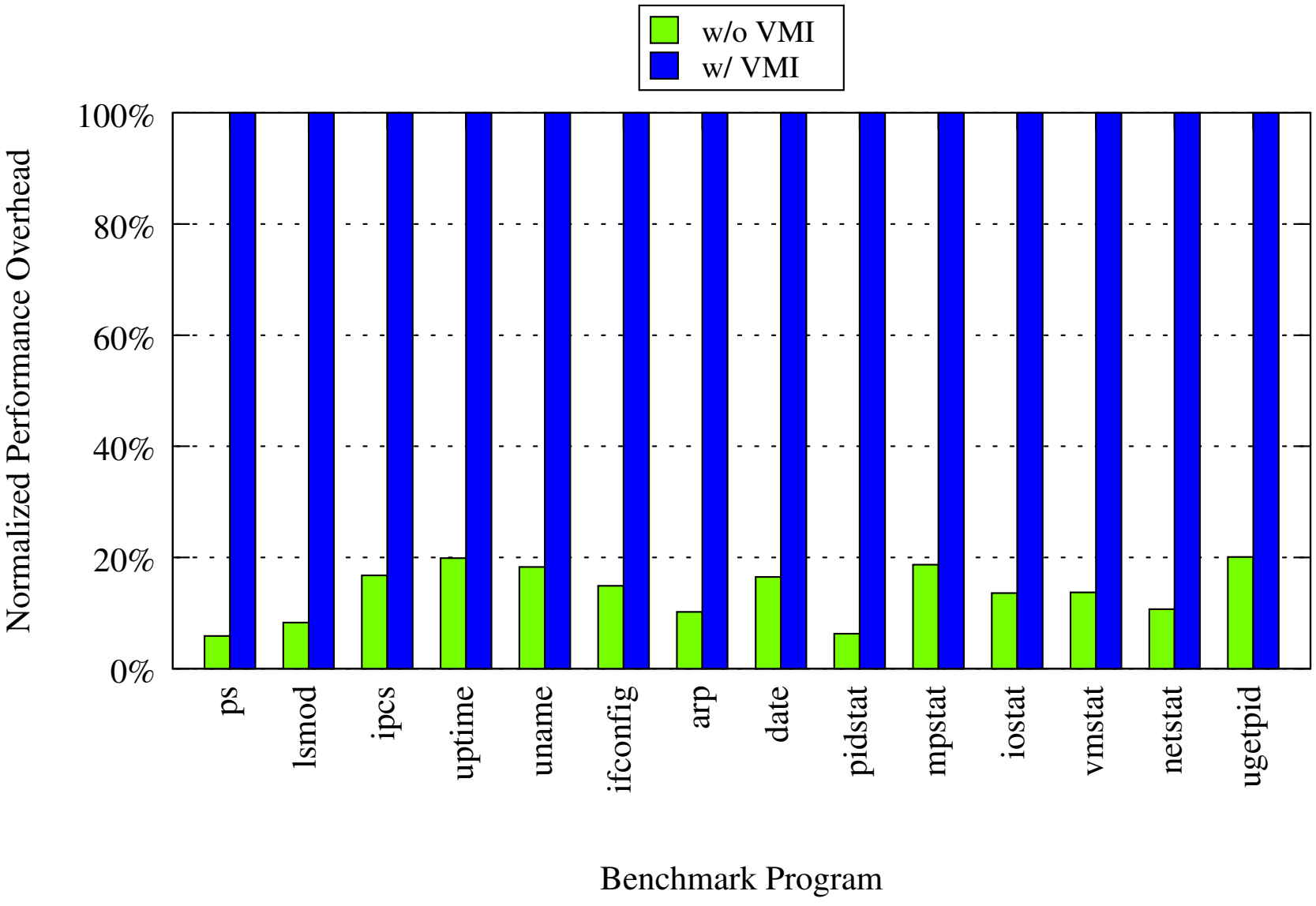
Approach-II: Design & Implementation [ATC'14]



Approach-II: Design & Implementation [ATC'14]



I. Virtual Machine Introspection ([SP'12])

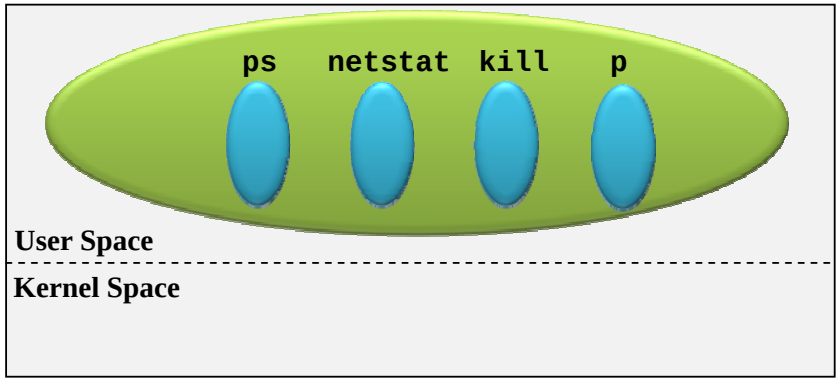


II. Out-of-Box Attack Recovery, Repair

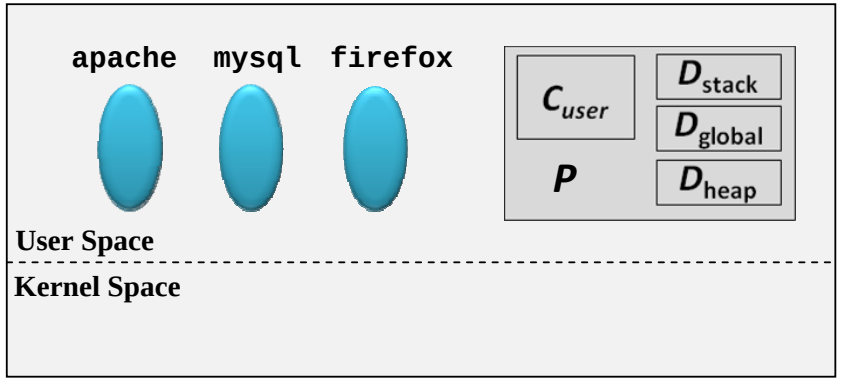
Rootkit	Targeted Function Pointer	Repaired?
adore-2.6	kernel global, heap object	✗
hookswrite	IDT table	✓
int3backdoor	IDT table	✓
kbdv3	syscall table	✓
kbeast-v1	syscall table, tcp4_seq_show	✓
mood-nt-2.3	syscall table	✓
override	syscall table	✓
phalanx-b6	syscall table, tcp4_seq_show	✓
rkit-1.01	syscall table	✓
rial	syscall table	✓
suckit-2	IDT table	✓
synapsys-0.4	syscall table	✓

Table : Rootkit Repairing with An Exterior [VEE'13] Tool.

III. Developing Out-of-VM Programs Natively



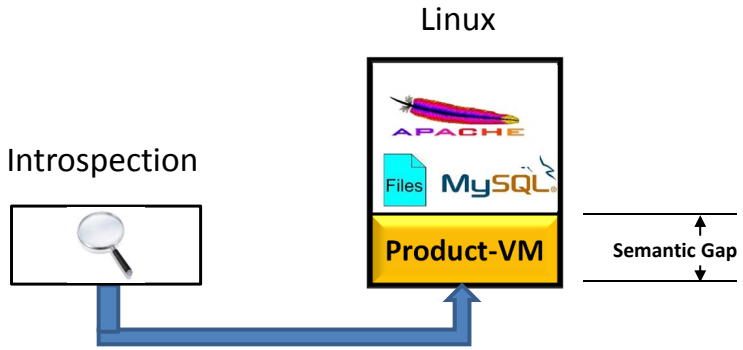
Secure VM (SVM)



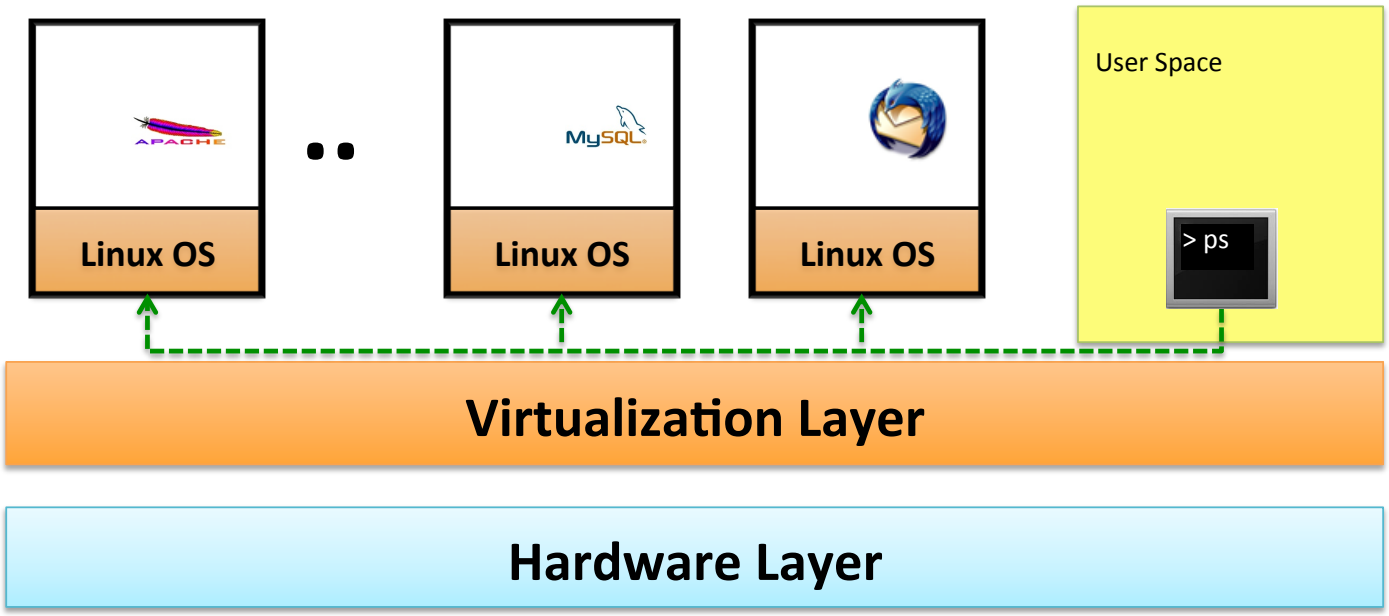
Guest VM (GVM)

In-VM `getpid` Program

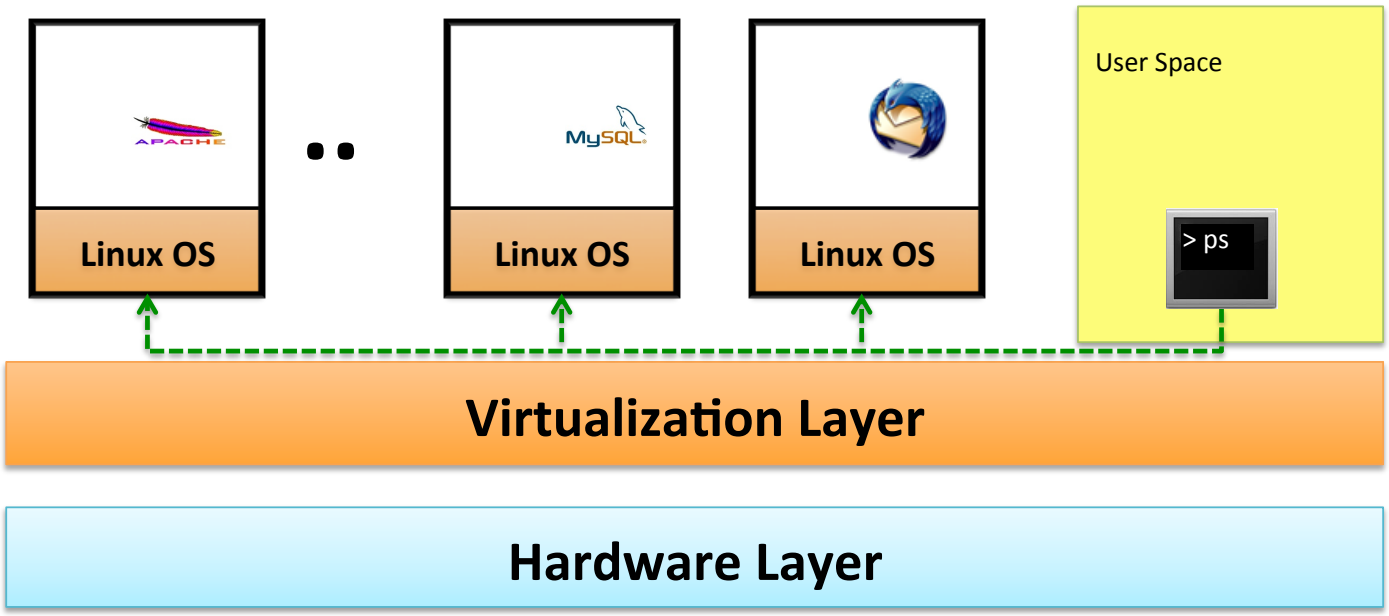
```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n",getpid());
7     return 0;
8 }
```



IV. Writable VMI [VEE'13, ATC'14]



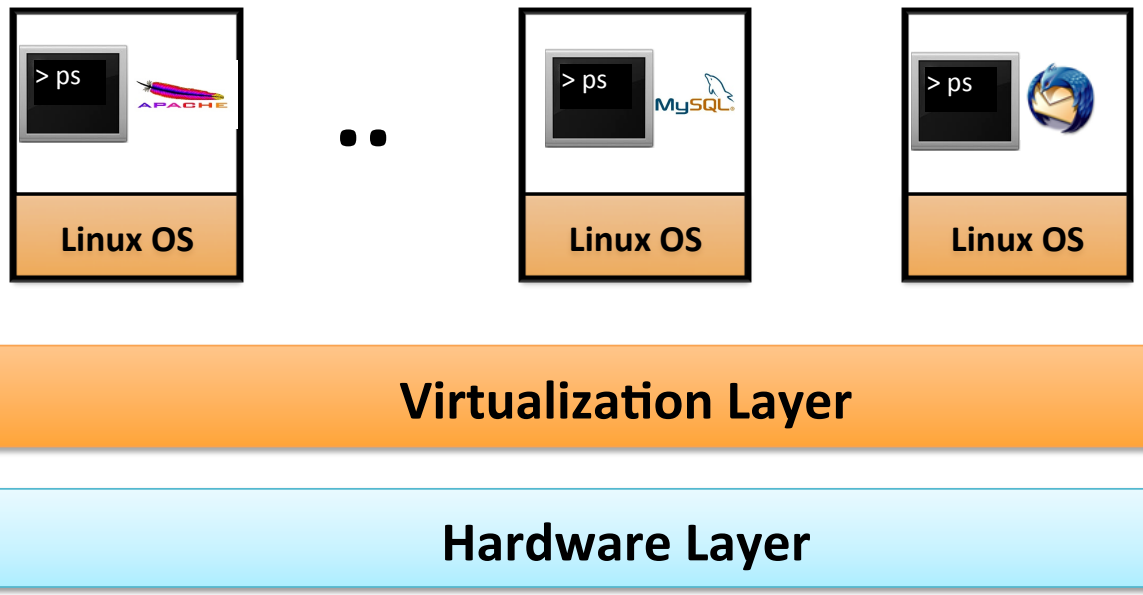
IV. Writable VMI [VEE'13, ATC'14]



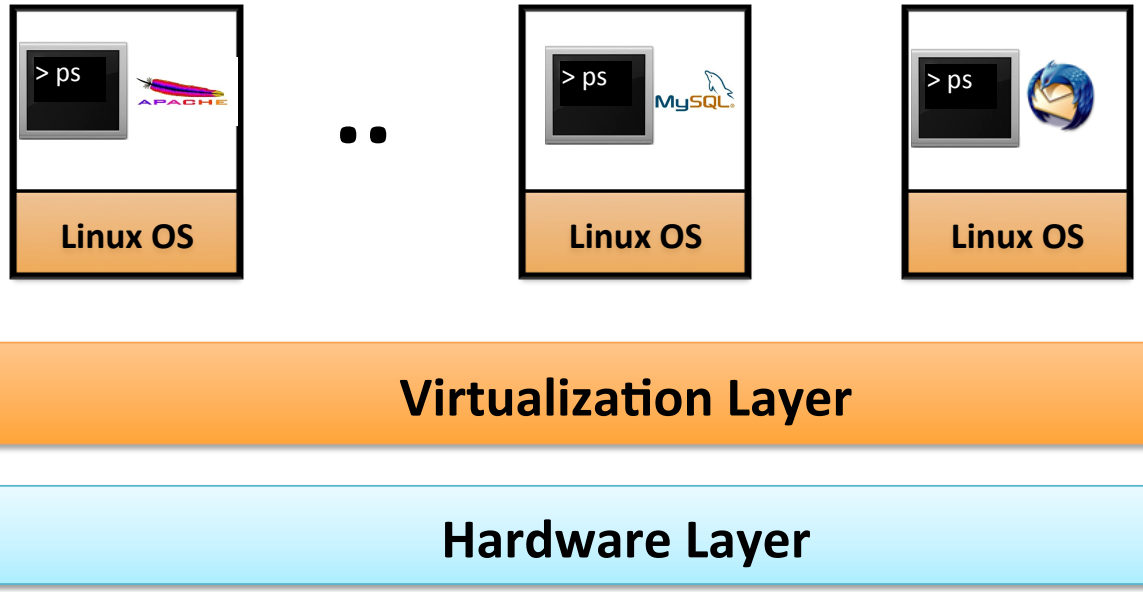
Advantages

- Only install the management utilities at hypervisor layer.
- Automated, uniformed, and centralized management.

In-VM Management: Existing Approaches



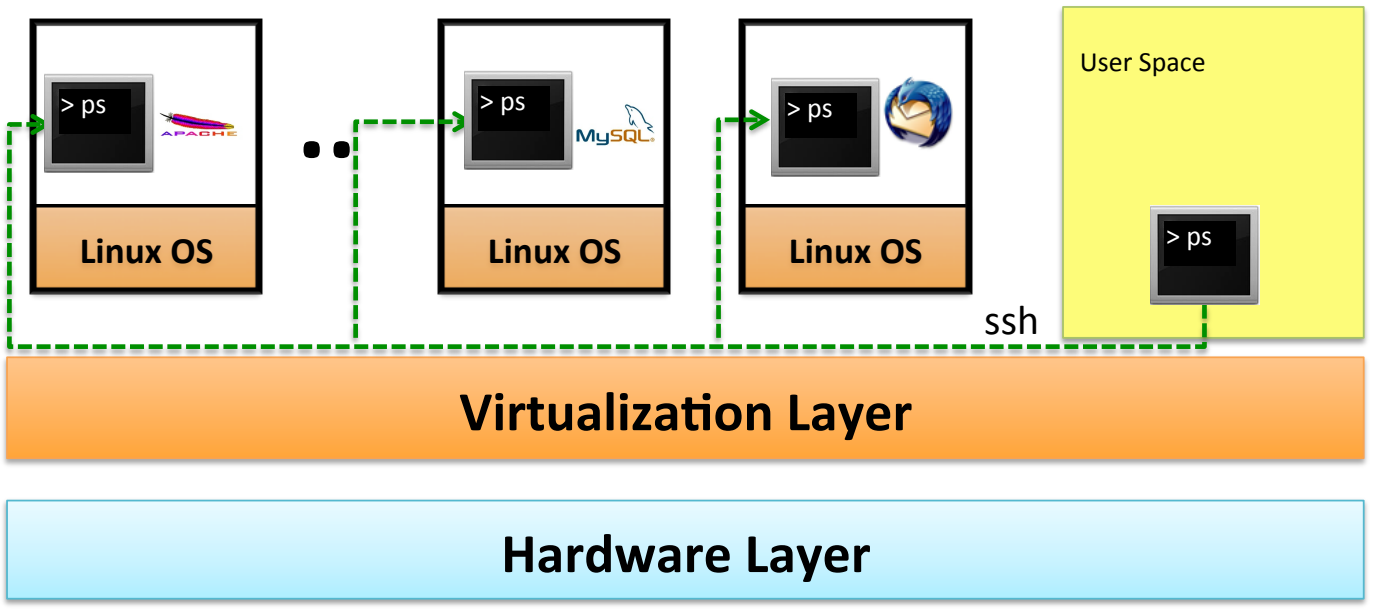
In-VM Management: Existing Approaches



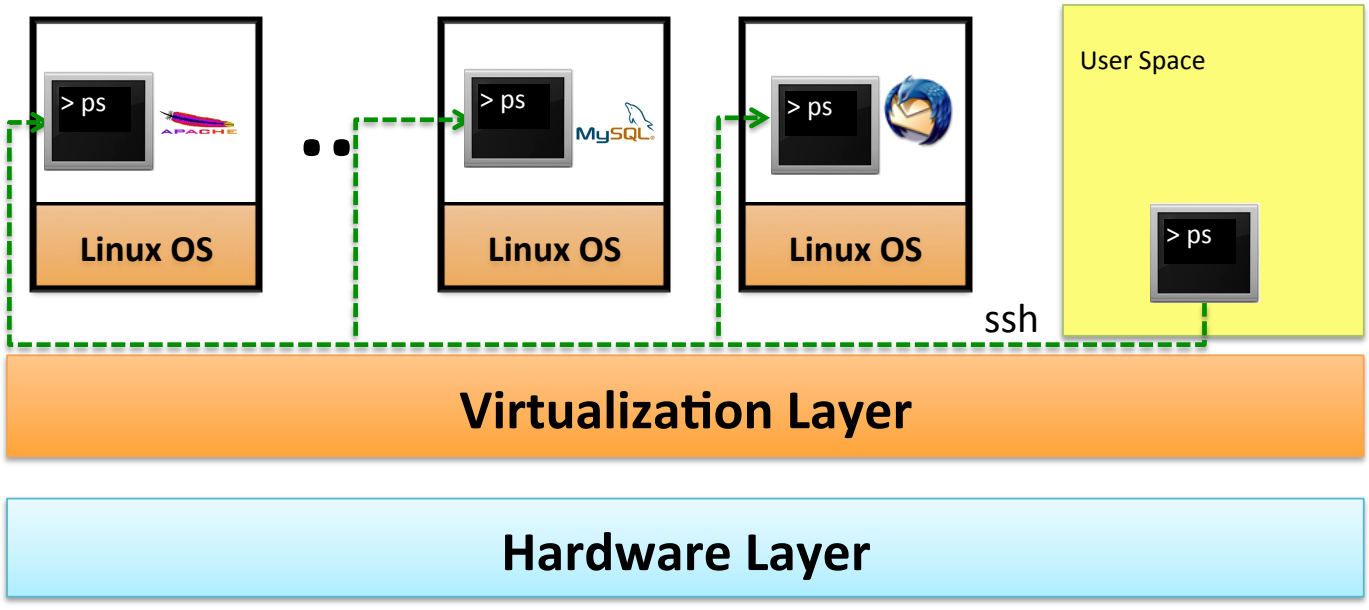
Disadvantages

- Scattered, distributed
- Install, update, and execute in each VM

In-VM Management: Existing Approaches



In-VM Management: Existing Approaches



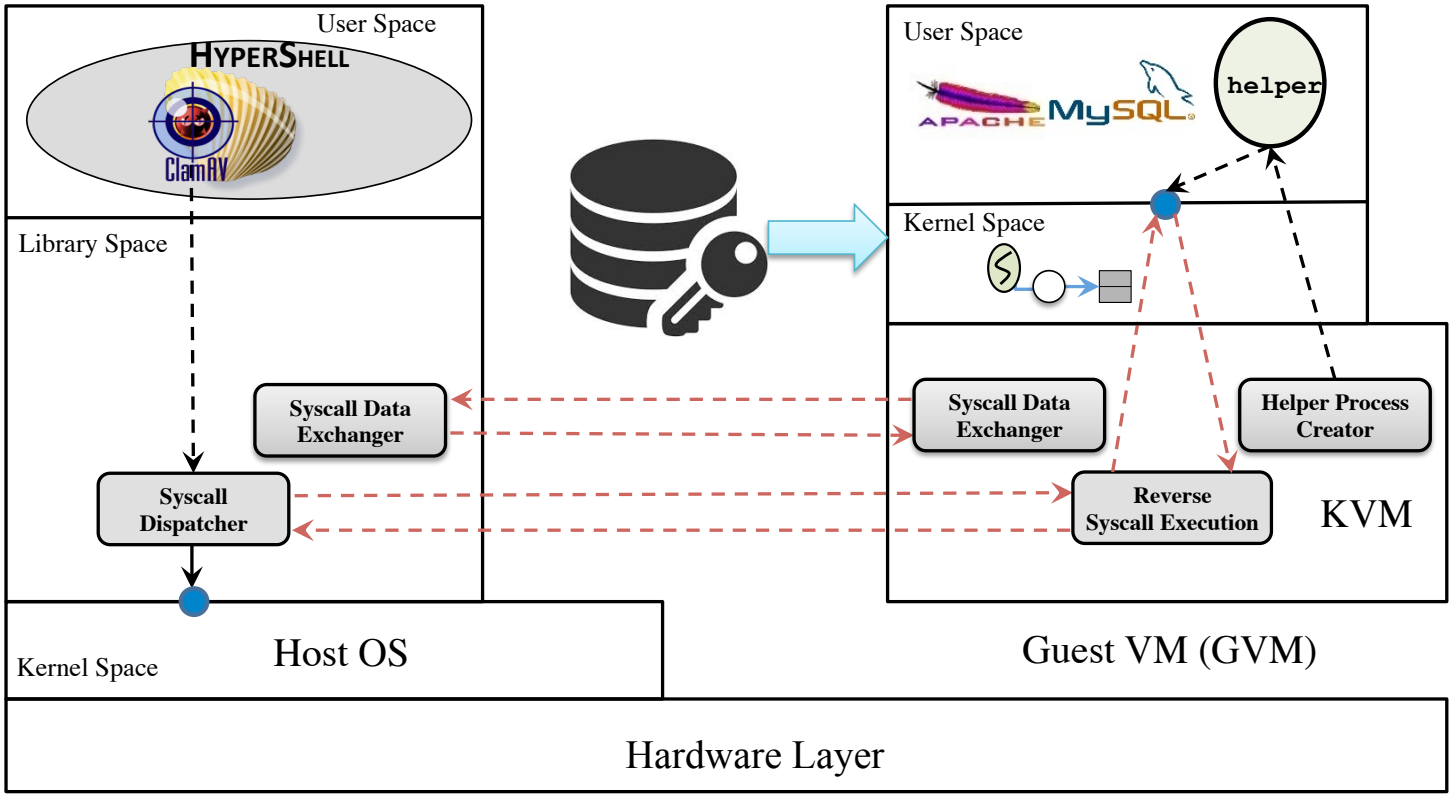
Disadvantages

- Requiring the (admin) login password.
- Requiring install the management utilities in each VM.

Performance Impact: HyperShell [ATC'14]

Process	**S**	$B(ms)$	$D(ms)$	$T(X)$
ps	✗	1.33	5.42	4.08
pidstat	✗	1.95	7.56	3.88
nice	✓	0.07	0.11	1.57
getpid	✓	0.01	0.02	2.00
mpstat	✗	0.29	0.66	2.28
pstree	✗	0.69	6.03	8.74
chrt	✓	0.11	0.16	1.45
renice	✓	0.11	0.18	1.64
top	✗	504.92	510.85	1.01
nproc	✓	0.07	0.26	3.71
sleep	✓	1.27	1.28	1.01
pgrep	✓	0.89	4.72	5.30
pkill	✓	0.87	4.33	4.98
snice	✓	0.17	0.65	3.82
echo	✓	0.07	0.09	1.29
pwdx	✓	0.05	0.07	1.40
pmap	✓	0.16	0.36	2.25
kill	✓	0.01	0.04	4.00
killall	✓	0.62	3.03	4.89
Memory	**S**	$B(ms)$	$D(ms)$	$T(X)$
free	✗	0.04	0.08	2.00
vmstat	✗	0.19	0.33	1.74
slabtop	✗	0.22	0.36	1.64
Modules	**S**	$B(ms)$	$D(ms)$	$T(X)$
rmmod	✓	0.51	3.14	6.16
modinfo	✓	0.48	1.54	3.21
lsmod	✓	0.10	0.17	1.70
Environment	**S**	$B(ms)$	$D(ms)$	$T(X)$
who	✓	0.14	0.72	5.14
env	✓	0.07	0.11	1.57
printenv	✓	0.07	0.1	1.43
whoami	✓	0.19	0.45	2.37
stty	✓	0.11	0.46	4.18
users	✓	0.09	0.53	5.89
uname	✓	0.09	0.11	1.22
id	✓	0.26	0.85	3.27
date	✗	0.11	0.12	1.09
w	✗	0.95	6.62	6.97
hostname	✓	0.04	0.06	1.50
groups	✓	0.21	0.62	2.95
hostid	✓	0.16	0.56	3.50
locale	✓	0.09	0.17	1.89
getconf	✓	0.09	0.34	3.78
System Utils	**S**	$B(ms)$	$D(ms)$	$T(X)$
uptime	✗	0.07	0.47	6.71
sysctl	✓	8.5	42.72	5.03
arch	✓	0.07	0.11	1.57
dmesg	✓	0.38	0.51	1.34
lscpu	✓	0.26	1.21	4.65
mcookie	✗	0.29	0.49	1.69
Disk/Devices	**S**	$B(ms)$	$D(ms)$	$T(X)$
blkid	✓	0.14	0.61	4.36
badblocks	✓	0.35	0.44	1.26
lspci	✓	31.40	36.52	1.16
iostat	✓	0.45	1.04	2.31
du	✓	0.11	0.53	4.82
df	✓	0.16	0.35	2.19
Filesystem	**S**	$B(ms)$	$D(ms)$	$T(X)$
sync	✓	8.07	6.53	0.81
getcap	✓	0.04	0.08	2.00
lsof	✓	3.31	6.12	1.85
pwd	✓	0.07	0.11	1.57
Files	**S**	$B(ms)$	$D(ms)$	$T(X)$
chgrp	✓	0.19	0.47	2.47
chmod	✓	0.07	0.14	2.00
chown	✓	0.19	0.47	2.47
cp	✓	0.11	0.27	2.45
uniq	✓	0.09	0.35	3.89
file	✓	0.87	1.72	1.98
find	✓	0.20	0.58	2.90
grep	✓	0.35	2.14	6.11
ln	✓	0.08	0.14	1.75
ls	✓	0.14	0.27	1.93
mkdir	✓	0.10	0.19	1.90
mkfifo	✓	0.10	0.19	1.90
mknod	✓	0.10	0.19	1.90
mv	✓	0.15	0.31	2.07
rm	✓	0.08	0.15	1.88
od	✓	0.12	0.35	2.92
cat	✓	0.07	0.18	2.57
link	✓	0.07	0.13	1.86
comm	✓	0.08	0.22	2.75
shred	✗	0.72	0.92	1.28
truncate	✓	0.07	0.26	3.71
head	✓	0.07	0.15	2.14
vdir	✓	0.63	3.95	6.27
nl	✓	0.08	0.17	2.13
tail	✓	0.08	0.20	2.50
namei	✓	0.07	0.13	1.86
whereis	✓	2.05	4.86	2.37
stat	✓	0.27	0.78	2.89
readlink	✓	0.07	0.12	1.71
unlink	✓	0.07	0.13	1.86
cut	✓	0.08	0.17	2.13
dir	✓	0.07	0.20	2.86
mktemp	✓	0.09	0.18	2.00
rmdir	✓	0.07	0.13	1.86
ptx	✓	0.12	0.45	3.75
chcon	✓	0.06	0.12	2.00
Network	**S**	$B(ms)$	$D(ms)$	$T(X)$
ifconfig	✗	0.32	1.15	3.59
ip	✓	0.10	0.20	2.00
route	✓	138.65	150.32	1.08
ipmaddr	✓	0.13	0.34	2.62
iptunnel	✓	0.09	0.29	3.22
nameif	✓	0.10	0.21	2.10
netstat	✗	0.25	0.37	1.48
arp	✓	0.14	0.24	1.71
ping	✗	15.02	18.2	1.21
Avg.	-	7.27	8.45	2.73

Disk Introspection: FDE disk virus scanning [ATC'14]



Disk Introspection: FDE disk virus scanning [ATC'14]

1. Encypted by dm-crypt
2. 101,415 files
3. 1336.09 megabytes in size

Disk Introspection: FDE disk virus scanning [ATC'14]

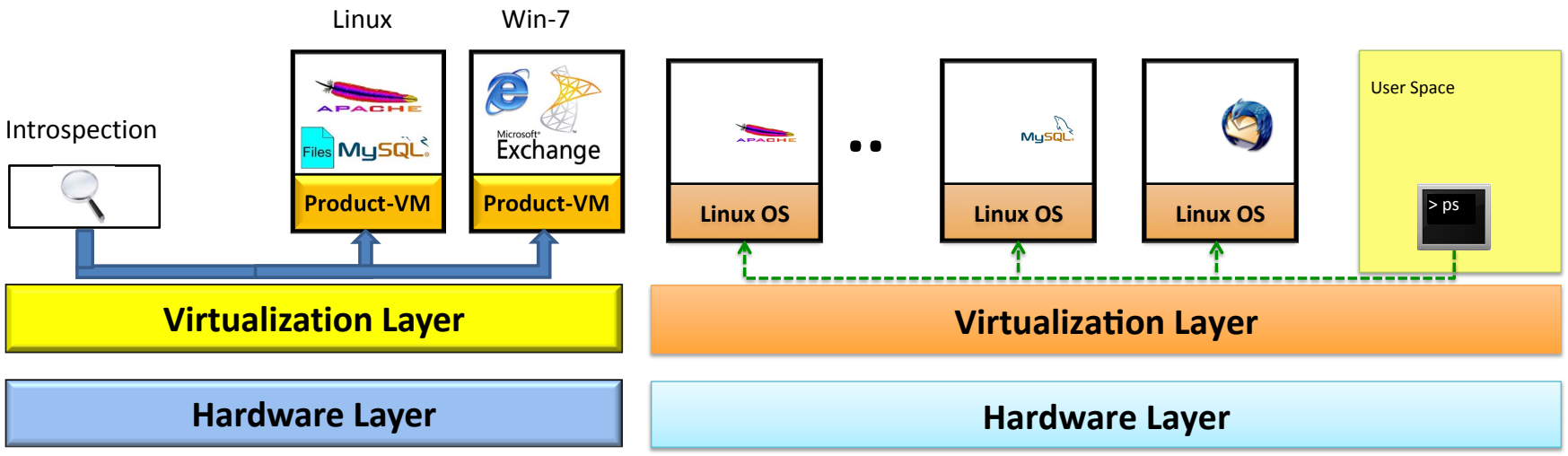
1. Encypted by dm-crypt
2. 101,415 files
3. 1336.09 megabytes in size

Clamav successfully detect two viruses!!

Comparison with the most related work

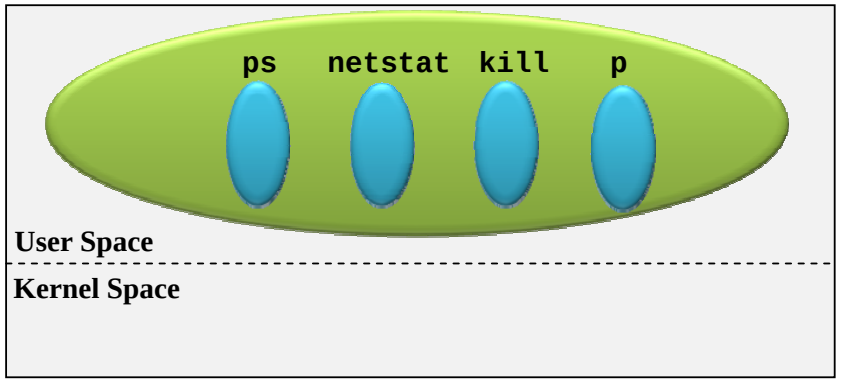
Systems	Execution Context Reuse	wo/ Dual-VM Architecture	wo/ Identical Kernel	wo/ Trust to Guest Kernel	High Code Coverage	Fully Automated	Memory Introspection	Disk Introspection	Guest Management	Process Monitoring
VIRTUOSO	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗
VMST	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗
EXTERIOR	✓	✗	✗	✓	✓	✓	✓	✗	✓	✗
PROCESSIMPLANTING	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗
PROCESSOUTGRAFTING	✓	✗	✓	✓	✓	✓	✗	✗	✗	✓
GEARS	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓
HYPERSHELL	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓

Virtualization (Hypervisor) Layer Applications

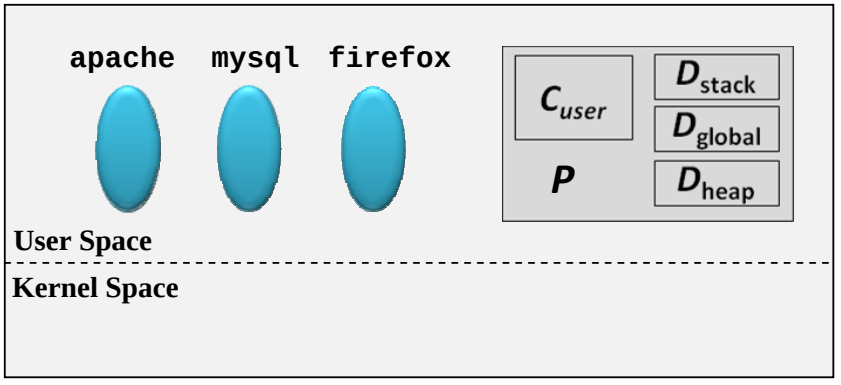


- Virtual Machine Introspection
- Virtual Machine (Re)Configuration, Repair
- Automated Out-of-VM Management

Two Approaches to bridging the semantic gap

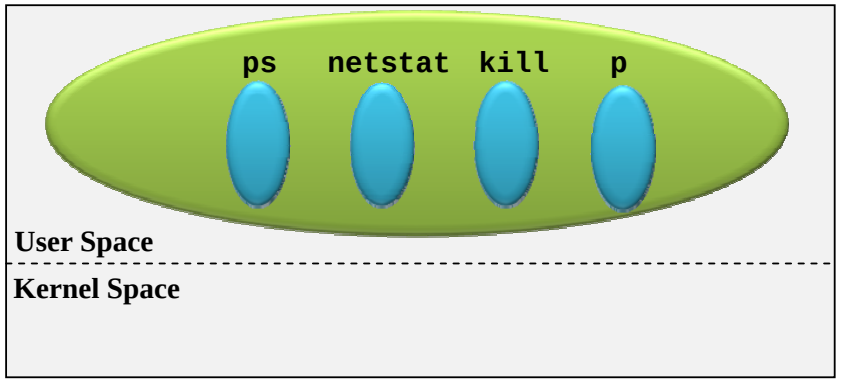


Secure VM (SVM)

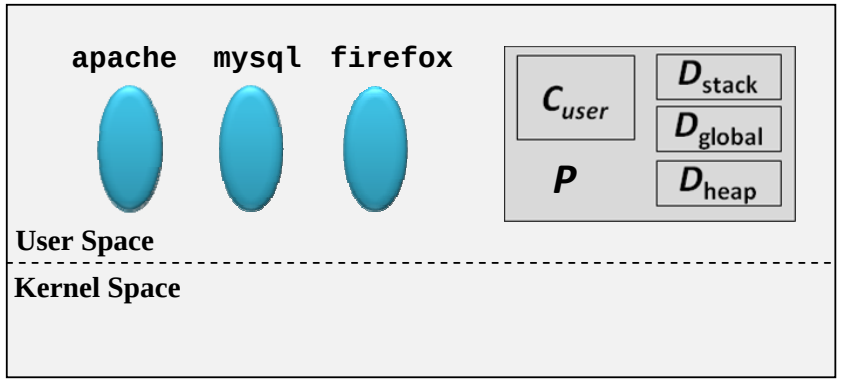


Guest VM (GVM)

Two Approaches to bridging the semantic gap



Secure VM (SVM)

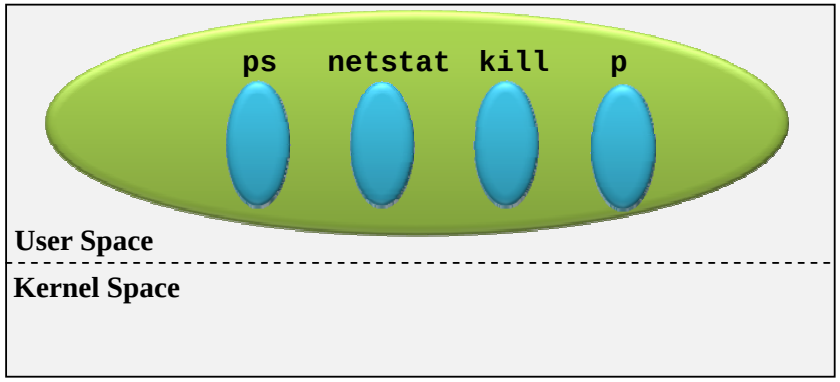


Guest VM (GVM)

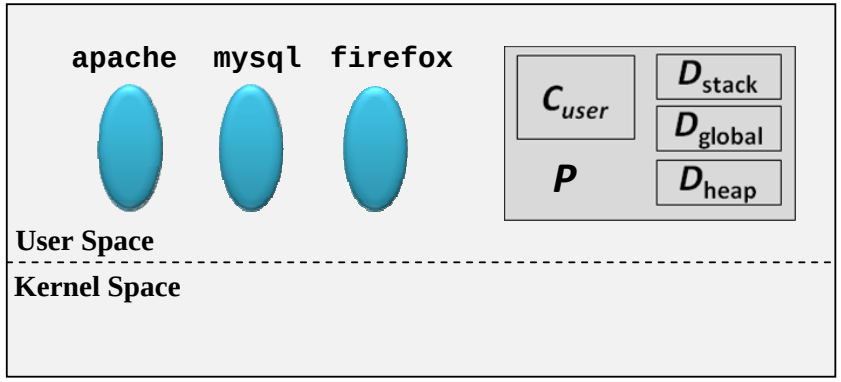
Reusing **(legacy) binary code** with a trusted sibling VM to intropsect the running guest VM.

1. Redirect kernel data [SP'12, VEE'13, NDSS'14] → Fine-grained, slower performance
2. Redirect system call execution [USENIX ATC'14] → More practical, fast performance

For Cloud Developers: No Gap, Everything is Native



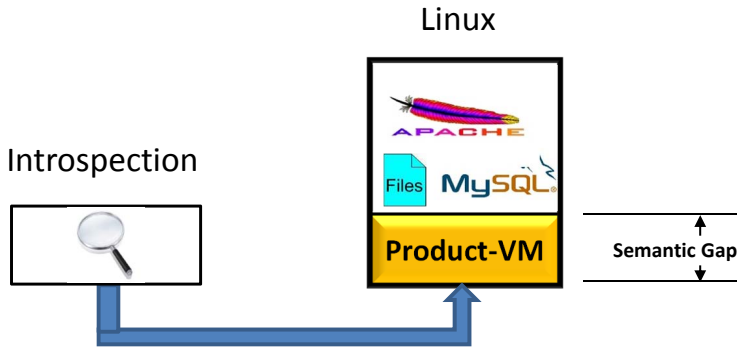
Secure VM (SVM)



Guest VM (GVM)

In-VM `getpid` Program

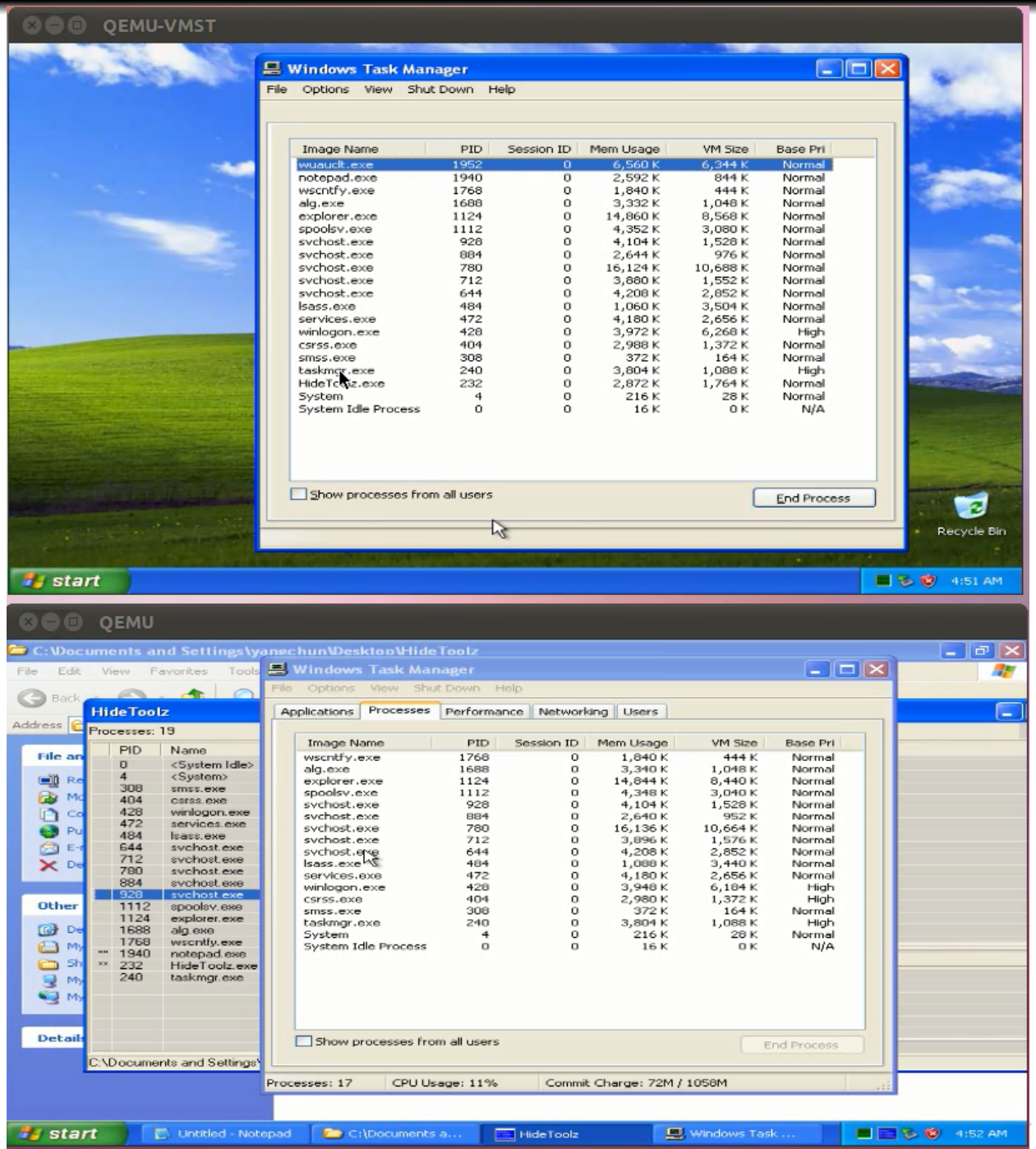
```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n",getpid());
7     return 0;
8 }
```

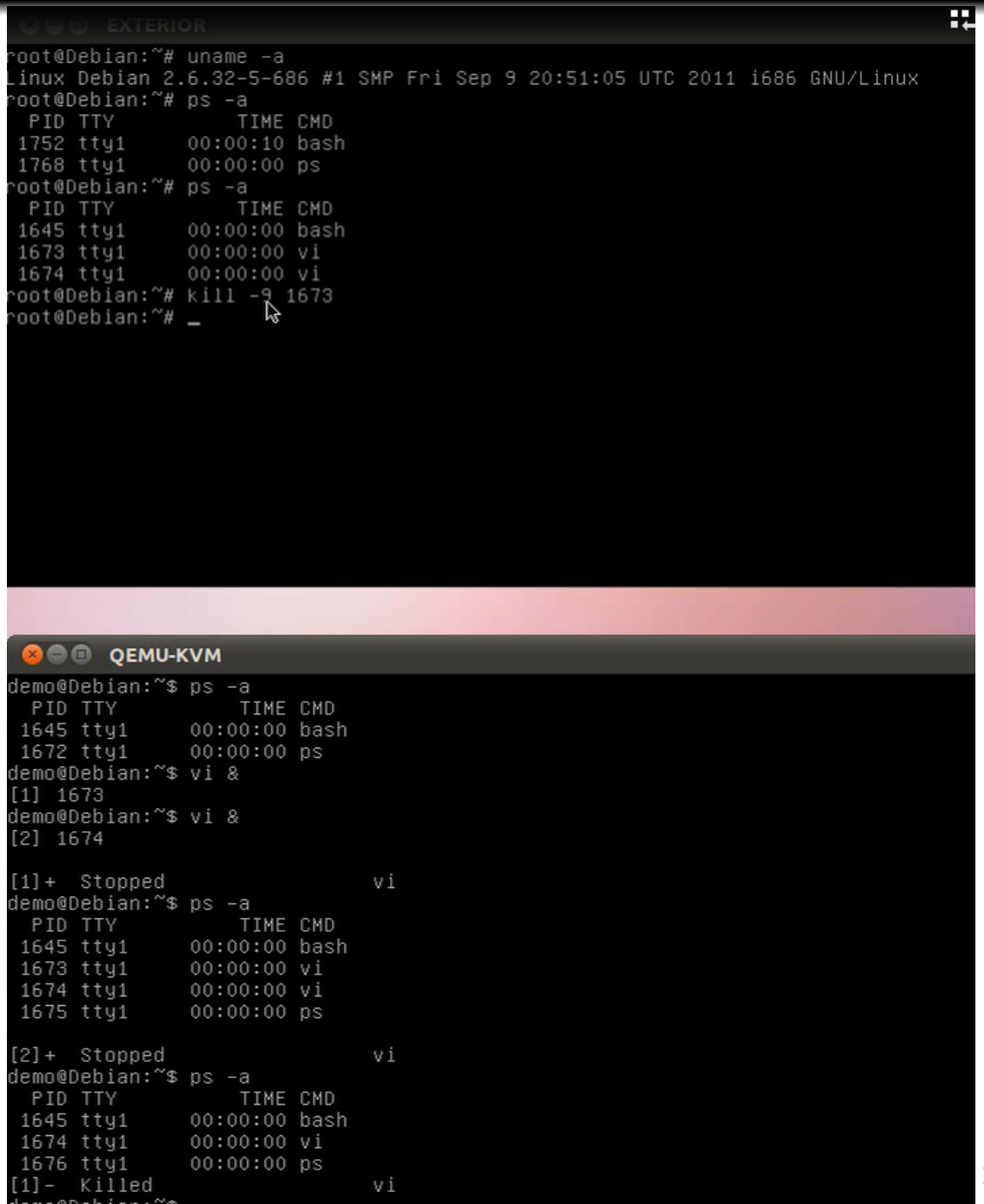


References

1. **VM Space Traveling: Automatically Bridging the Semantic Gap in Virtual Machine Introspection via Online Kernel Data Redirection** (*IEEE Symposium on Security and Privacy* [SP'12])
2. **EXTERIOR: Using A Dual-VM Enabled External Shell for Guest-OS Introspection, Configuration, and Recovery** (*ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* [VEE'13])
3. **Hybrid-Bridge: Efficiently Bridging the Semantic-Gap in Virtual Machine Introspection via Decoupled Execution and Training Memoization** (*Network and Distributed System Symposium* [NDSS'14])
4. **HyperShell: A Practical Hypervisor Layer Guest OS Shell for Automated In-VM Management** (*USENIX Annual Technical Conference* [ATC'14])

`http://www.utdallas.edu/~zhiqiang.lin/s3.html`


QEMU-VMST
Windows Task Manager
File Options View Shut Down Help
Image Name PID Session ID Mem Usage VM Size Base Pri
wuauclt.exe 1952 0 6,560 K 6,344 K Normal
notepad.exe 1940 0 2,592 K 844 K Normal
wscntfy.exe 1768 0 1,840 K 444 K Normal
alg.exe 1688 0 3,332 K 1,048 K Normal
explorer.exe 1124 0 14,860 K 8,568 K Normal
spoolsv.exe 1112 0 4,352 K 3,080 K Normal
svchost.exe 928 0 4,104 K 1,528 K Normal
svchost.exe 884 0 2,644 K 976 K Normal
svchost.exe 780 0 16,124 K 10,688 K Normal
svchost.exe 712 0 3,880 K 1,552 K Normal
svchost.exe 644 0 4,208 K 2,852 K Normal
lsass.exe 484 0 1,060 K 3,504 K Normal
services.exe 472 0 4,180 K 2,656 K Normal
winlogon.exe 428 0 3,972 K 6,268 K High
csrss.exe 404 0 2,988 K 1,372 K Normal
smss.exe 308 0 372 K 164 K Normal
taskmgr.exe 240 0 3,804 K 1,088 K High
HideToolz.exe 232 0 2,872 K 1,764 K Normal
System 4 0 216 K 28 K Normal
System Idle Process 0 0 16 K 0 K N/A
Show processes from all users
End Process
Recycle Bin
start
4:51 AM
QEMU
C:\Documents and Settings\yanechun\Desktop\HideToolz
HideToolz
Processes: 19
PID Name
0 <System Idle>
4 <System>
308 smss.exe
404 csrss.exe
428 winlogon.exe
472 services.exe
484 lsass.exe
644 svchost.exe
712 svchost.exe
780 svchost.exe
884 svchost.exe
928 svchost.exe
1112 spoolsv.exe
1124 explorer.exe
1688 alg.exe
1768 wscntfy.exe
1940 notepad.exe
232 HideToolz.exe
240 taskmgr.exe
Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Image Name PID Session ID Mem Usage VM Size Base Pri
wscntfy.exe 1768 0 1,840 K 444 K Normal
alg.exe 1688 0 3,340 K 1,048 K Normal
explorer.exe 1124 0 14,844 K 8,440 K Normal
spoolsv.exe 1112 0 4,348 K 3,040 K Normal
svchost.exe 928 0 4,104 K 1,528 K Normal
svchost.exe 884 0 2,640 K 952 K Normal
svchost.exe 780 0 16,136 K 10,664 K Normal
svchost.exe 712 0 3,896 K 1,576 K Normal
svchost.exe 644 0 4,208 K 2,852 K Normal
lsass.exe 484 0 1,088 K 3,440 K Normal
services.exe 472 0 4,180 K 2,656 K Normal
winlogon.exe 428 0 3,948 K 6,184 K High
csrss.exe 404 0 2,980 K 1,372 K High
smss.exe 308 0 372 K 164 K Normal
taskmgr.exe 240 0 3,804 K 1,088 K High
System 4 0 216 K 28 K Normal
System Idle Process 0 0 16 K 0 K N/A
Show processes from all users
End Process
Processes: 17 CPU Usage: 11% Commit Charge: 72M / 1058M
start
Untitled - Notepad
C:\Documents a...
HideToolz
Windows Task ...
4:52 AM


EXTERIOR
root@Debian:~# uname -a
Linux Debian 2.6.32-5-686 #1 SMP Fri Sep 9 20:51:05 UTC 2011 i686 GNU/Linux
root@Debian:~# ps -a
PID TTY TIME CMD
1752 tty1 00:00:10 bash
1768 tty1 00:00:00 ps
root@Debian:~# ps -a
PID TTY TIME CMD
1645 tty1 00:00:00 bash
1673 tty1 00:00:00 vi
1674 tty1 00:00:00 vi
root@Debian:~# kill -9 1673
root@Debian:~#
QEMU-KVM
demo@Debian:~$ ps -a
PID TTY TIME CMD
1645 tty1 00:00:00 bash
1672 tty1 00:00:00 ps
demo@Debian:~$ vi &
[1] 1673
demo@Debian:~$ vi &
[2] 1674
[1]+ Stopped vi
demo@Debian:~$ ps -a
PID TTY TIME CMD
1645 tty1 00:00:00 bash
1673 tty1 00:00:00 vi
1674 tty1 00:00:00 vi
1675 tty1 00:00:00 ps
[2]+ Stopped vi
demo@Debian:~$ ps -a
PID TTY TIME CMD
1645 tty1 00:00:00 bash
1674 tty1 00:00:00 vi
1676 tty1 00:00:00 ps
[1]- Killed vi