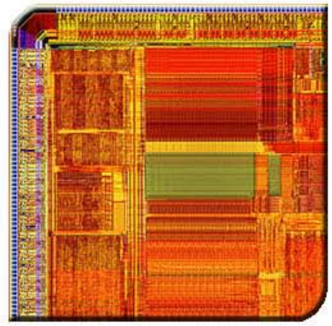
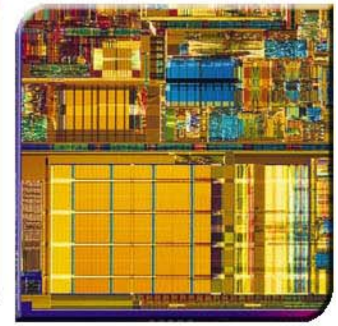




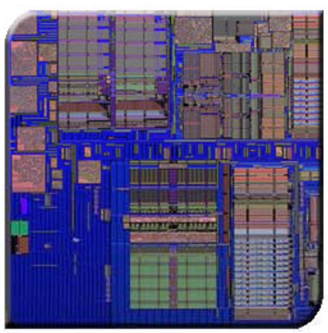
SHARK: Architectural Support for Autonomic Protection Against Stealth by Rootkit Exploits

Vikas R. Vasisht

Hsien-Hsin S. Lee

School of Electrical and Computer Engineering

Georgia Tech



Rootkit Definition

A set of programs that allows a permanent or consistent, undetectable presence on a computer

- Not an exploit to gain elevated access
- Conceal all evidences and malware activities

Rootkit's functions:

Hide processes, files, network connections and conceal malware activities

Example - Hidden Keylogger

Adversary

Task Manager looks clean

www.anybank.com
…login…
…password…

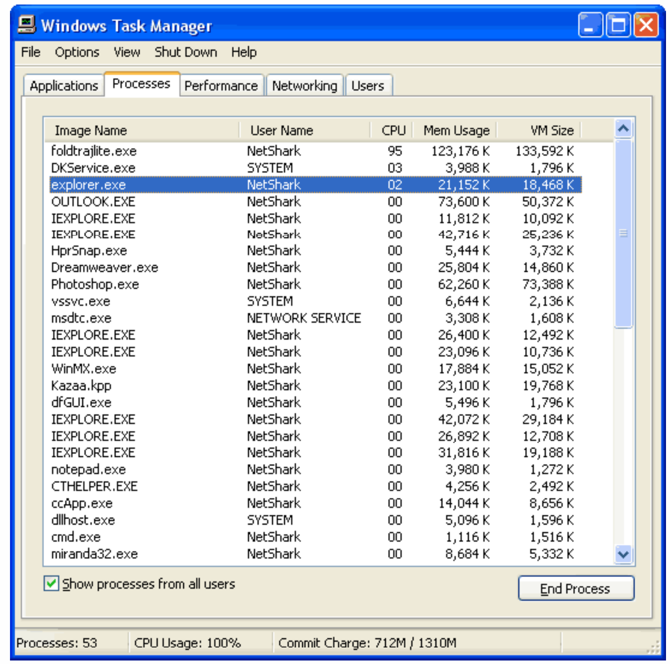
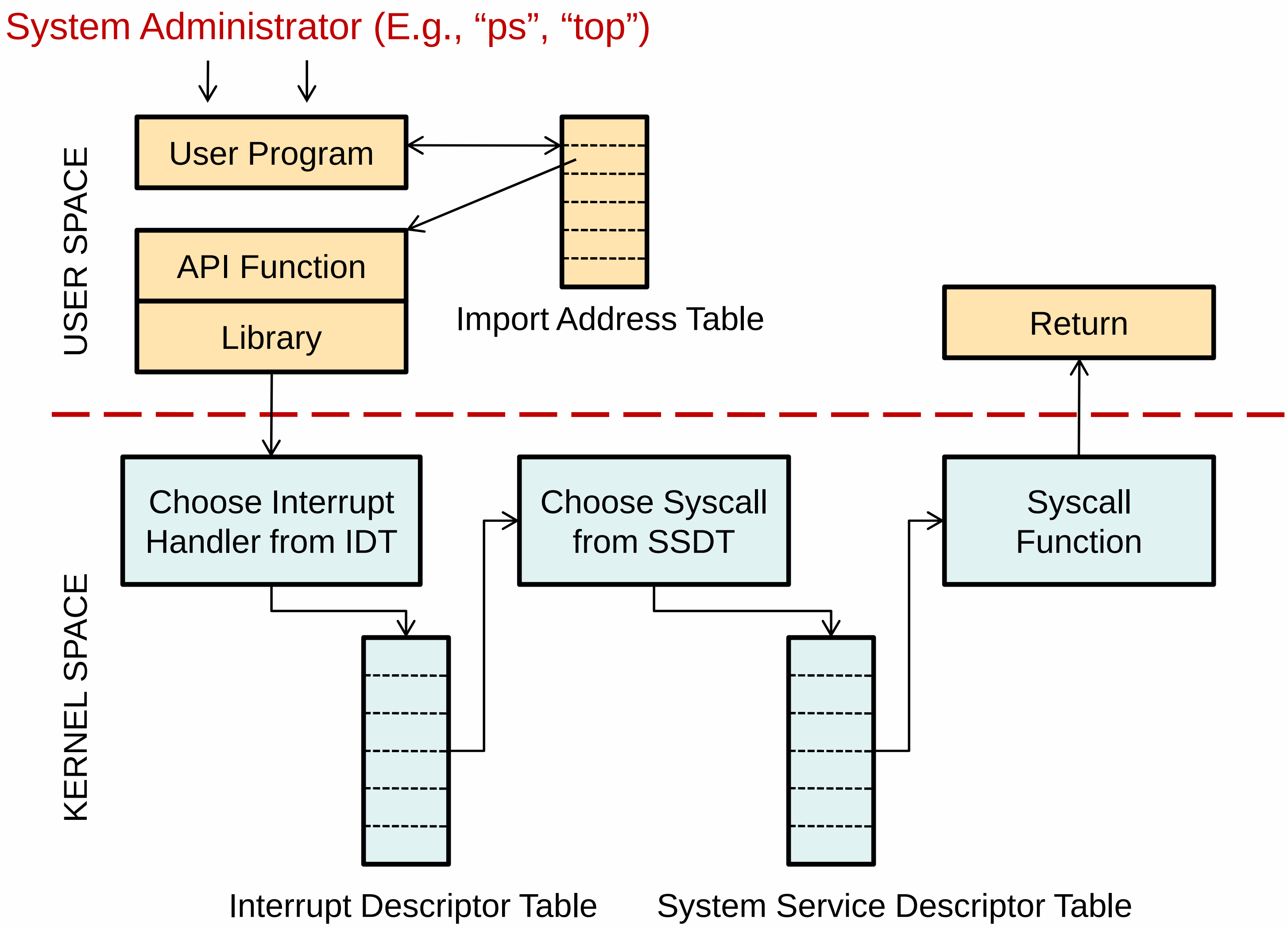

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users

Image Name	User Name	CPU	Mem Usage	VM Size
foldtrajlite.exe	NetShark	95	123,176 K	133,592 K
DKService.exe	SYSTEM	03	3,988 K	1,796 K
explorer.exe	NetShark	02	21,152 K	18,468 K
OUTLOOK.EXE	NetShark	00	73,600 K	50,372 K
IEXPLORE.EXE	NetShark	00	11,812 K	10,092 K
IEXPLORE.EXE	NetShark	00	42,716 K	25,236 K
HprSnap.exe	NetShark	00	5,444 K	3,732 K
Dreamweaver.exe	NetShark	00	25,804 K	14,860 K
Photoshop.exe	NetShark	00	62,260 K	73,388 K
vssvc.exe	SYSTEM	00	6,644 K	2,136 K
msdtc.exe	NETWORK SERVICE	00	3,308 K	1,608 K
IEXPLORE.EXE	NetShark	00	26,400 K	12,492 K
IEXPLORE.EXE	NetShark	00	23,096 K	10,736 K
WinMX.exe	NetShark	00	17,884 K	15,052 K
Kazaa.kpp	NetShark	00	23,100 K	19,768 K
dfGUI.exe	NetShark	00	5,496 K	1,796 K
IEXPLORE.EXE	NetShark	00	42,072 K	29,184 K
IEXPLORE.EXE	NetShark	00	26,892 K	12,708 K
IEXPLORE.EXE	NetShark	00	31,816 K	19,188 K
notepad.exe	NetShark	00	3,980 K	1,272 K
CTHELPER.EXE	NetShark	00	4,256 K	2,492 K
ccApp.exe	NetShark	00	14,044 K	8,656 K
dllhost.exe	SYSTEM	00	5,096 K	1,596 K
cmd.exe	NetShark	00	1,116 K	1,516 K
miranda32.exe	NetShark	00	8,684 K	5,332 K

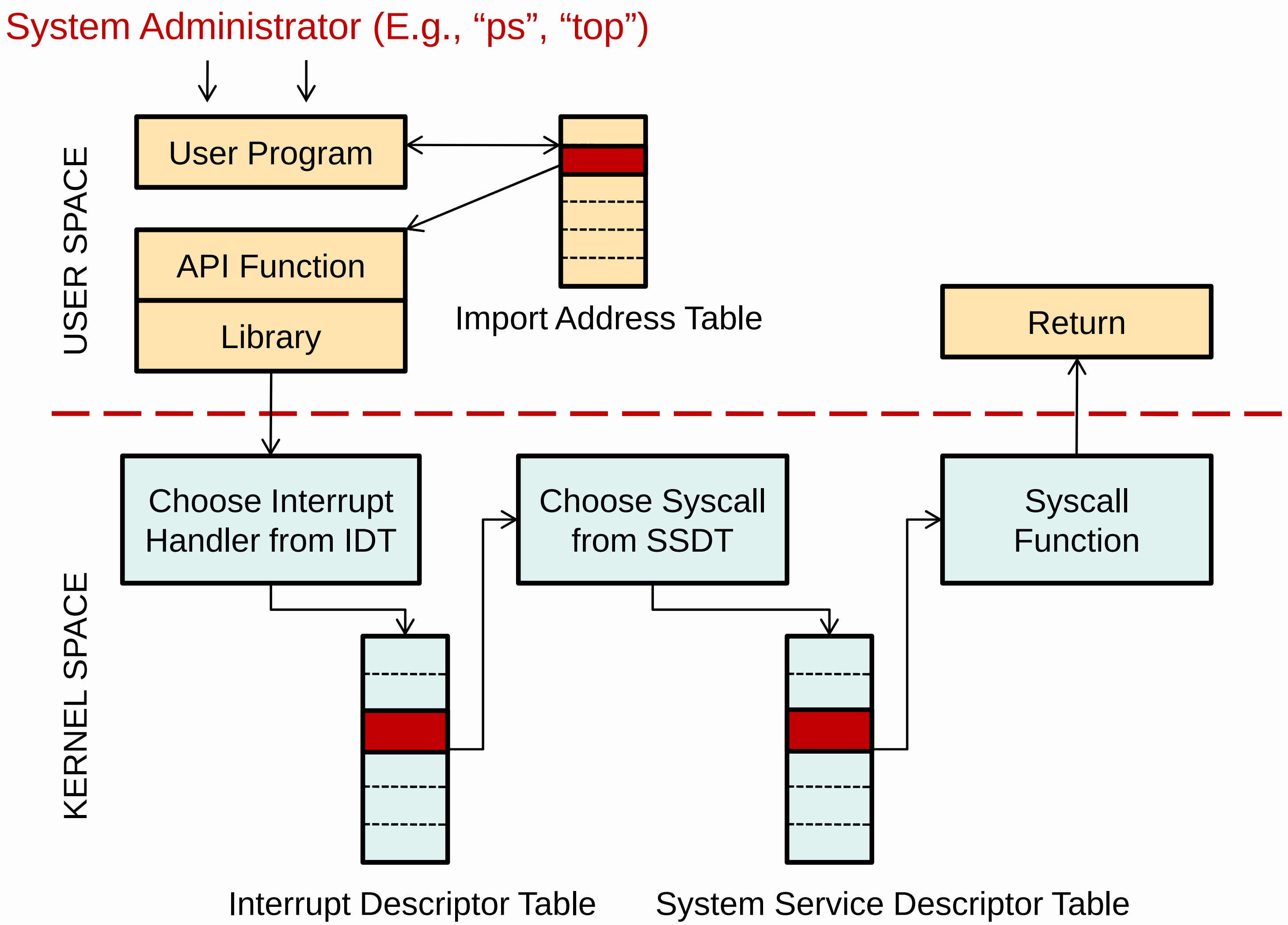
Show processes from all users
End Process
Processes: 53 CPU Usage: 100% Commit Charge: 712M / 1310M

OS compromised & Rootkit installed

Rootkit Technique (I)

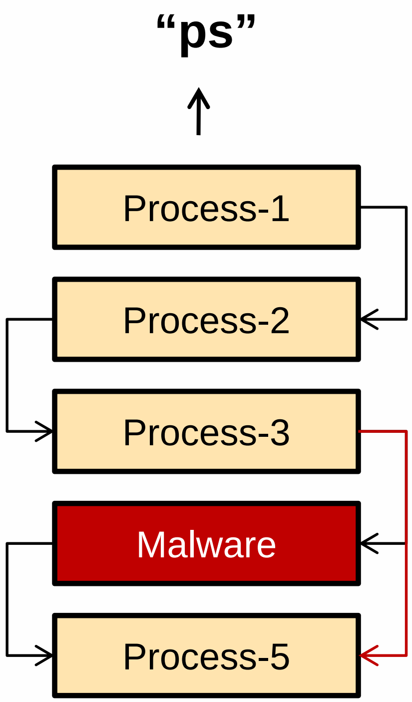


Rootkit Technique (I)



Modify OS execution flow to hide traces of malware

Rootkit Technique (II)



Safe machine	Compromised Machine
>>ps	>>ps
P-1	P-1
P-2	P-2
P-3	P-3
Malware	P-5
P-5	

Direct Kernel Object Modification

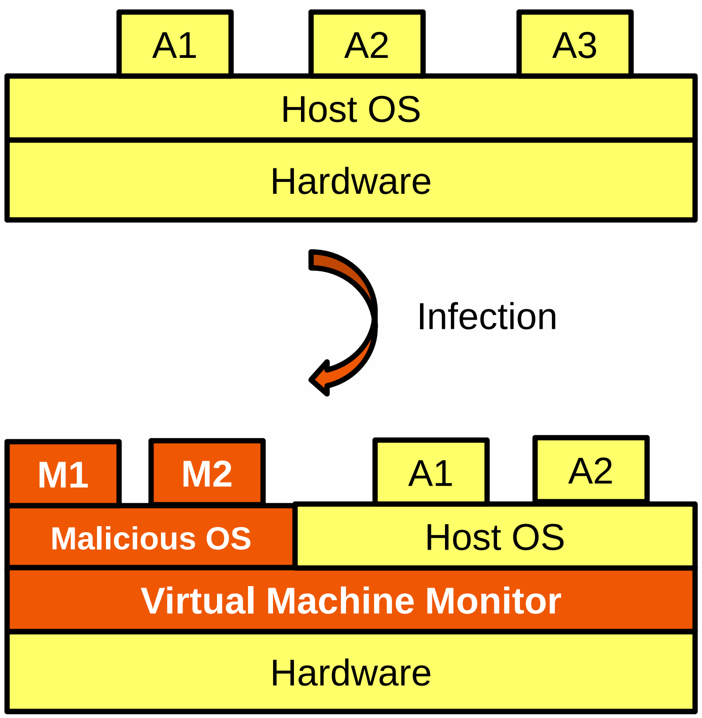
Manipulate Kernel Data to remove malware information

Rootkit Detection Techniques

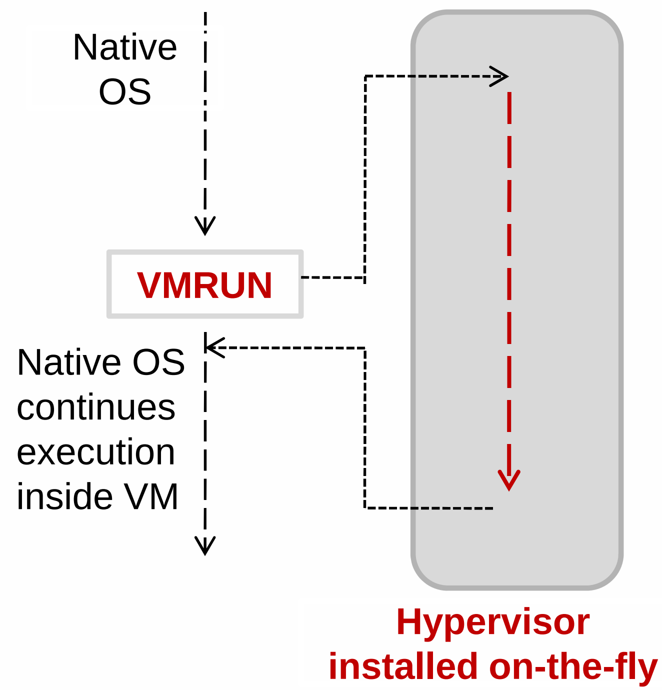
- Software based techniques:
  - Signature/Behavioral detection
    - ☠ Works for only known rootkits
  - Cross-View based detection
    - ☠ Complex rootkits compromise low level OS view
  - Integrity based detection
    - ☠ Rootkits fake memory contents – Shadow Walker rootkit
- Hardware based techniques:
  - CoPilot *(N. Petroni et al. [USENIX'04])*
    - √ Integrity of host memory checked in a remote admin station
    - ☠ Send a faked memory snapshot to the remote machine.

Sophisticated Rootkits

Sub-Virt[1]



Bluepill[2]



Host OS downgraded to VM

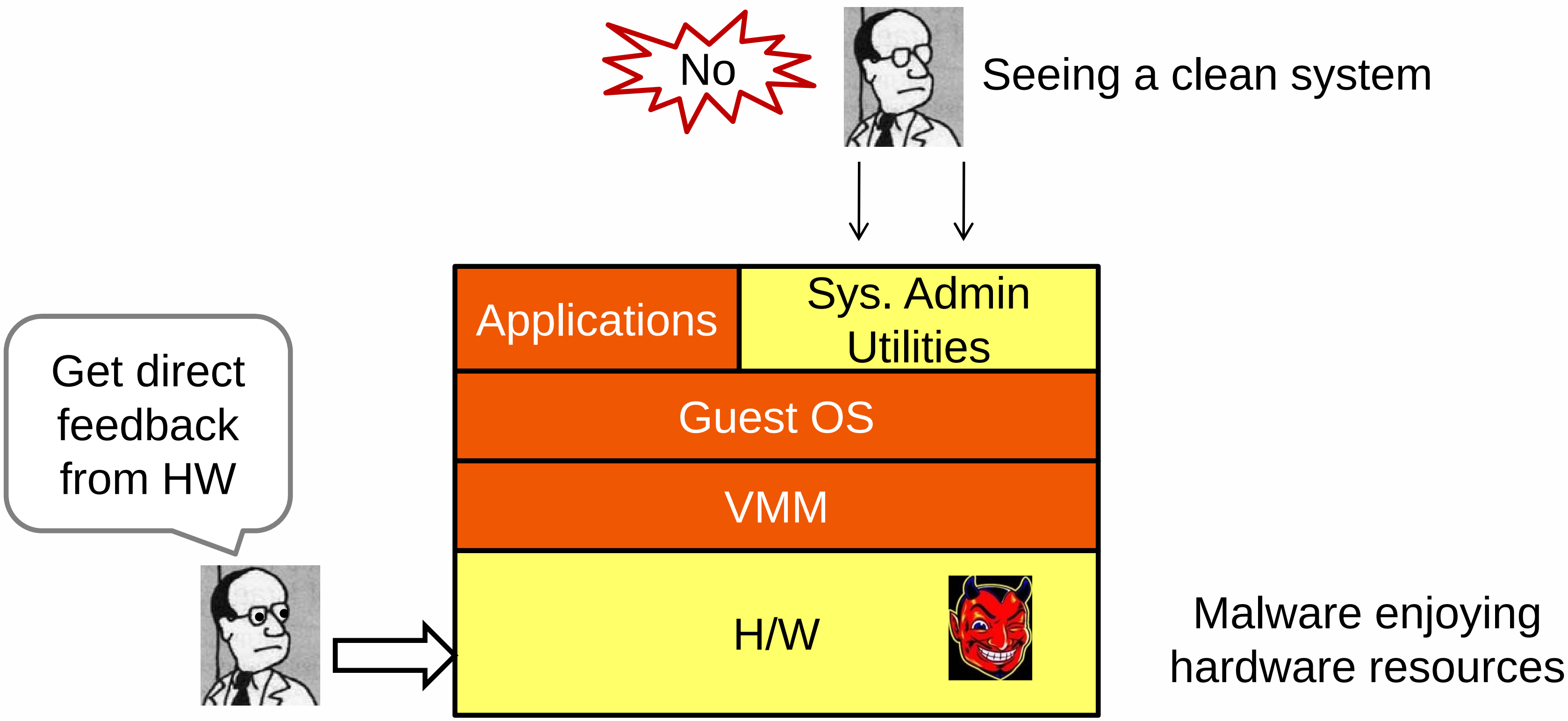
Hypervisor below the host OS

1. King et al. [Symposium on Security and Privacy'06]

2. Joanna Rutkowska [Black Hat'06]

Challenge

We cannot detect hidden processes, VMs and VMMs using software techniques

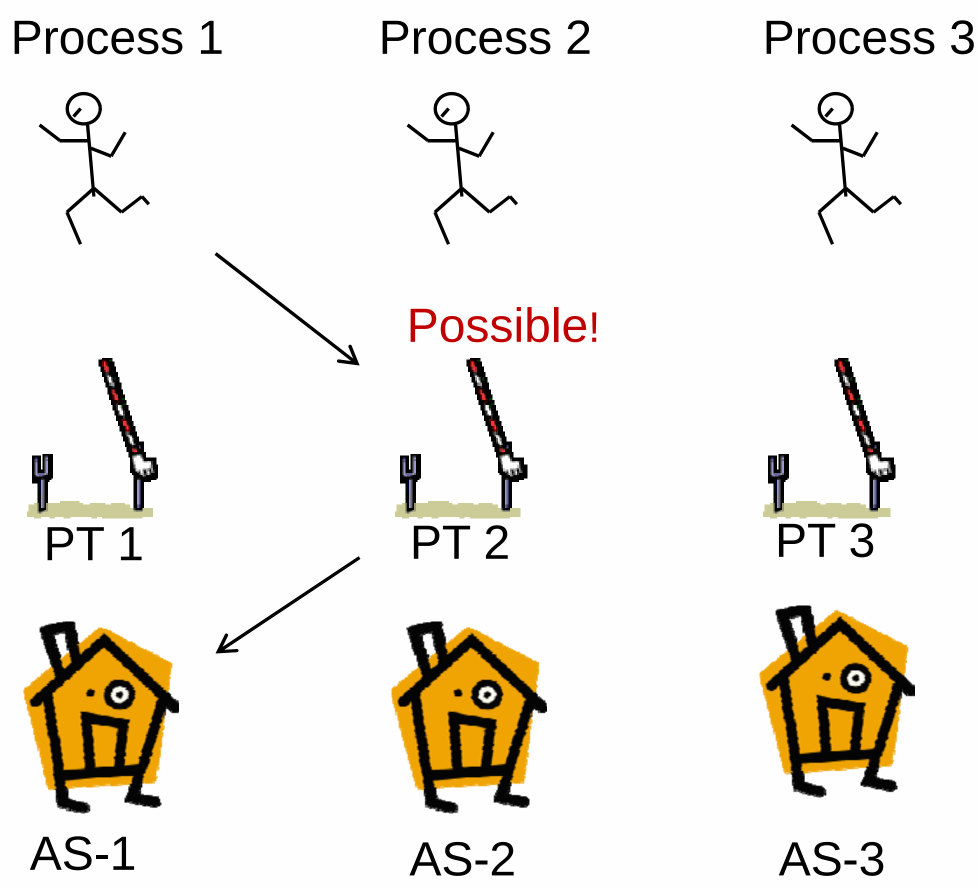


Motivation –
Process Context Aware Architecture

Processes Spawned

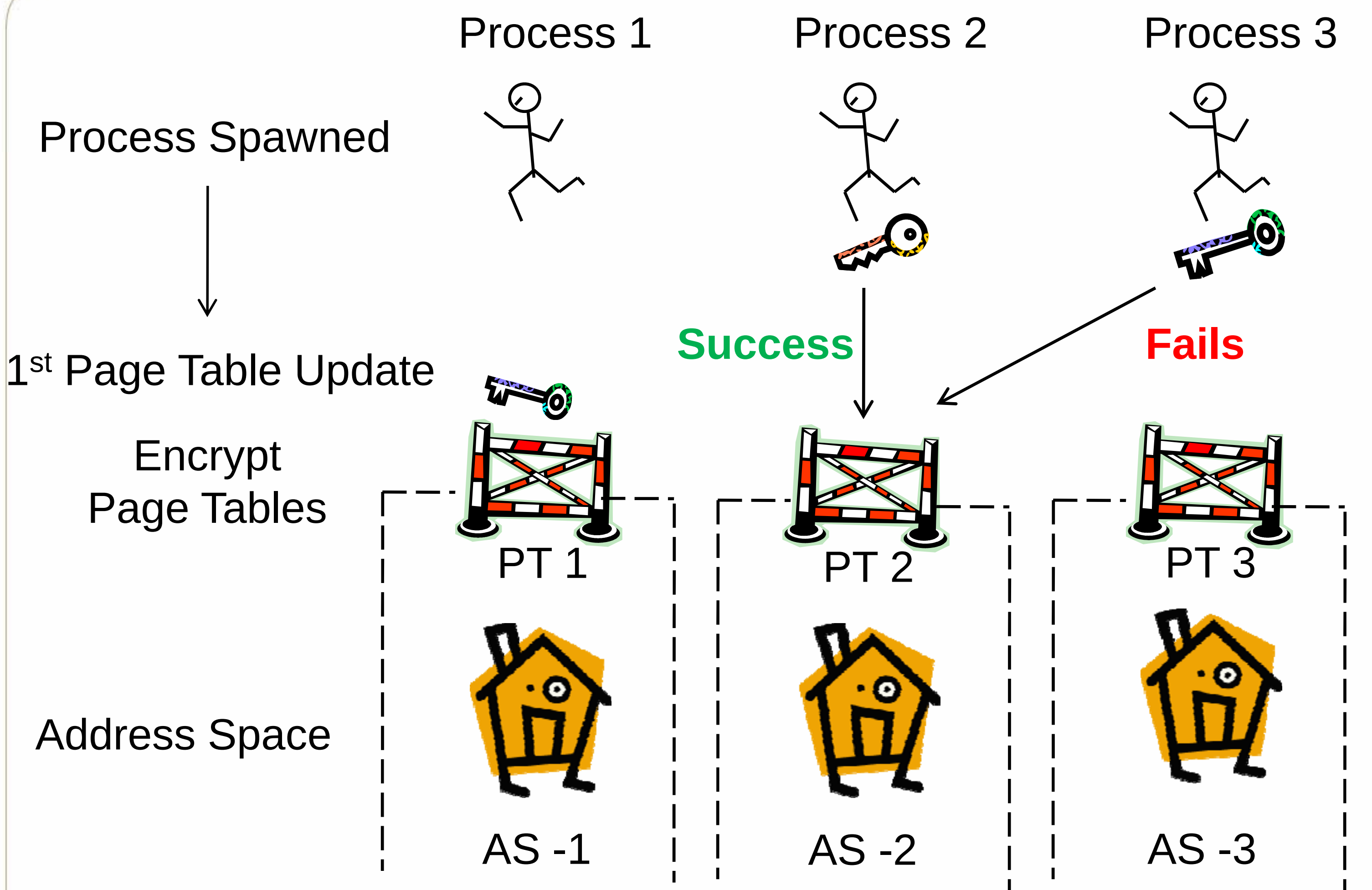
Page Tables

Address Space



OS completely manages processes and HW can be fooled

SHARK Big Picture

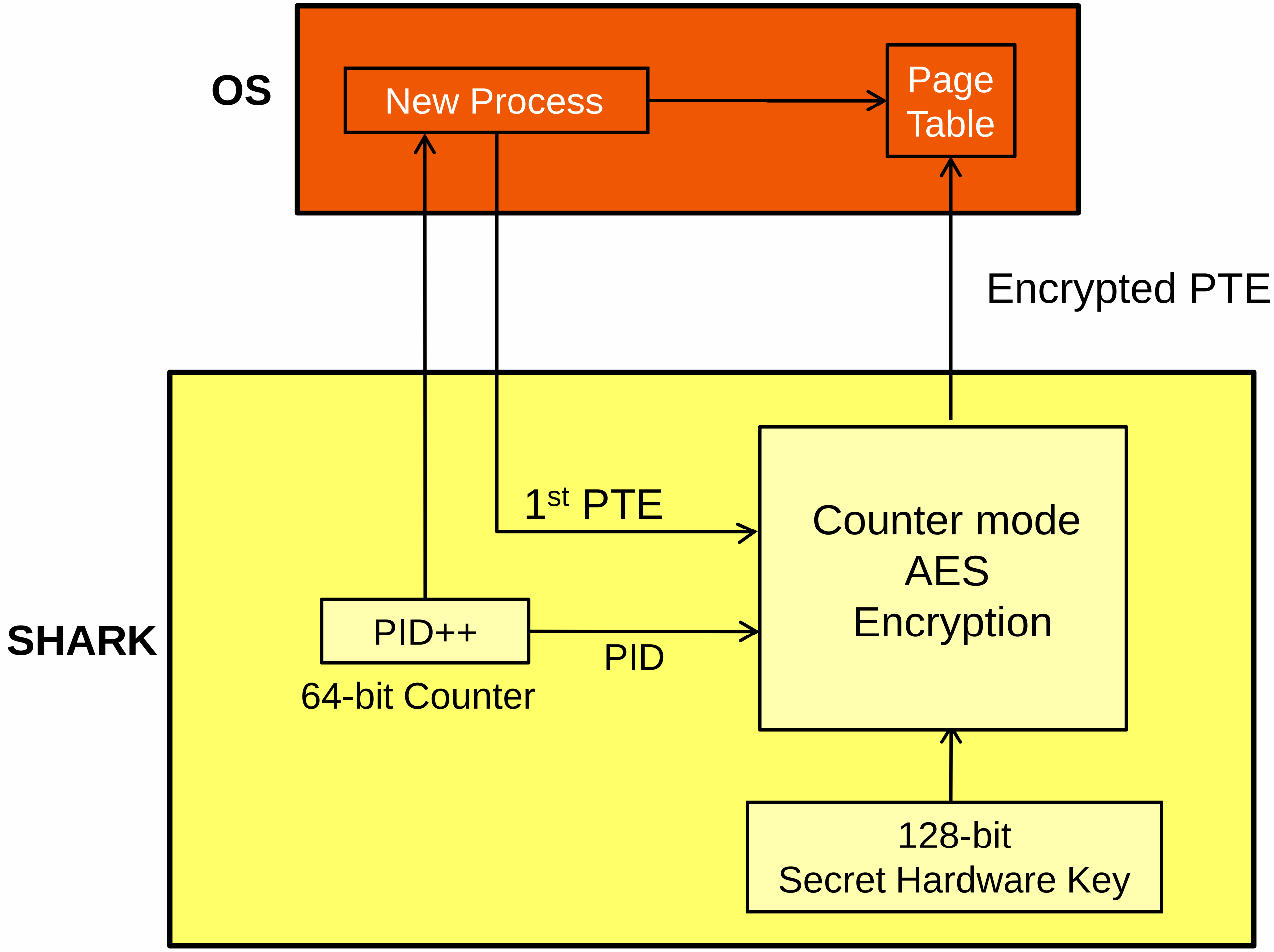


Address space isolation achieved by page table encryption

SHARK – Secure Hardware Against RootKits

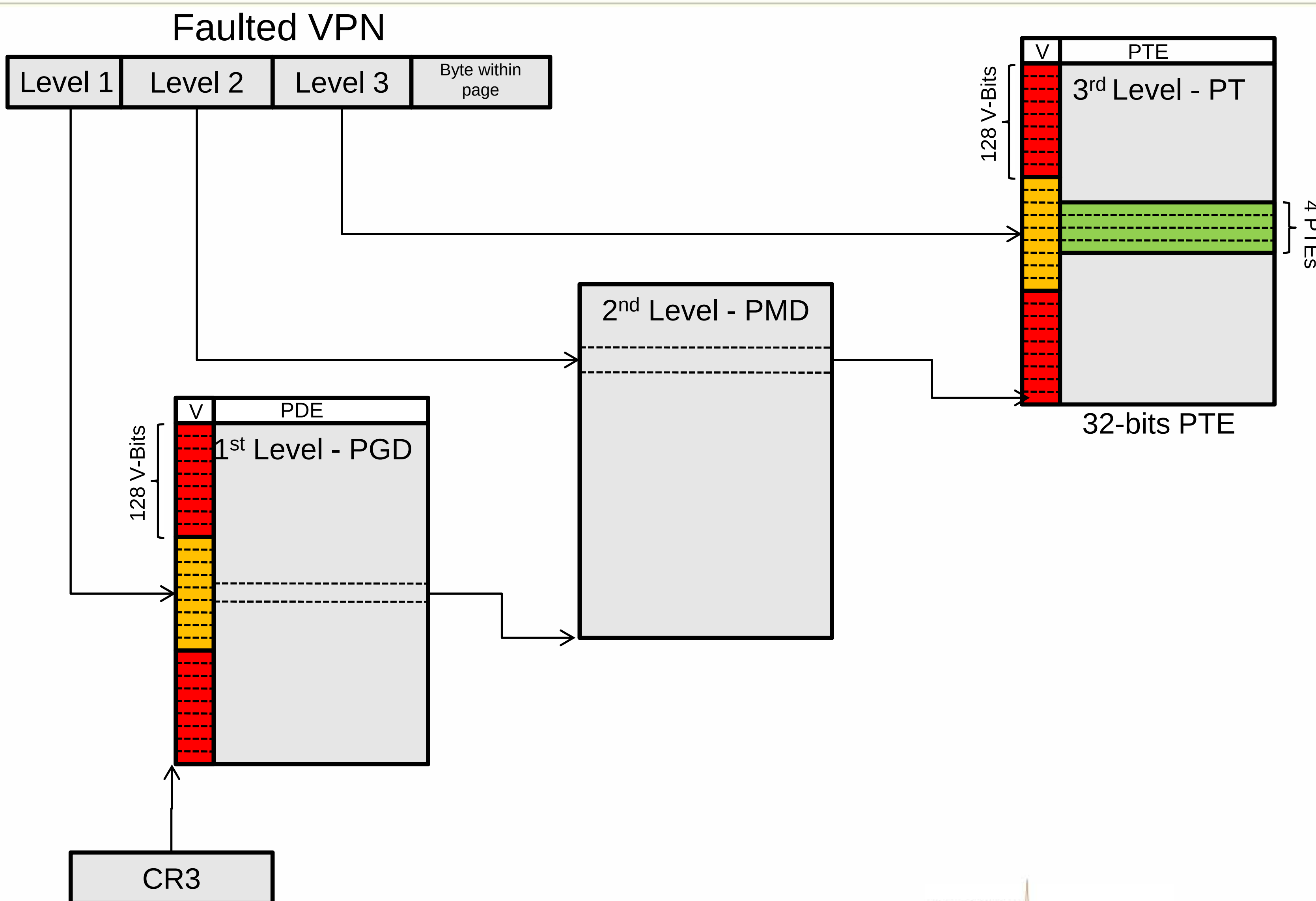
- Hardware assisted PID Generation
  - Software PIDs vulnerable
- Page Table Encryption/Decryption
  - Page table update: Hardware support for every update
  - TLB miss: Page table decryption
- Process Authentication
  - On a context switch, PID $\rightarrow$ HPID Register
  - TLB miss: HPID used for decryption

Hardware Assisted PID Generation

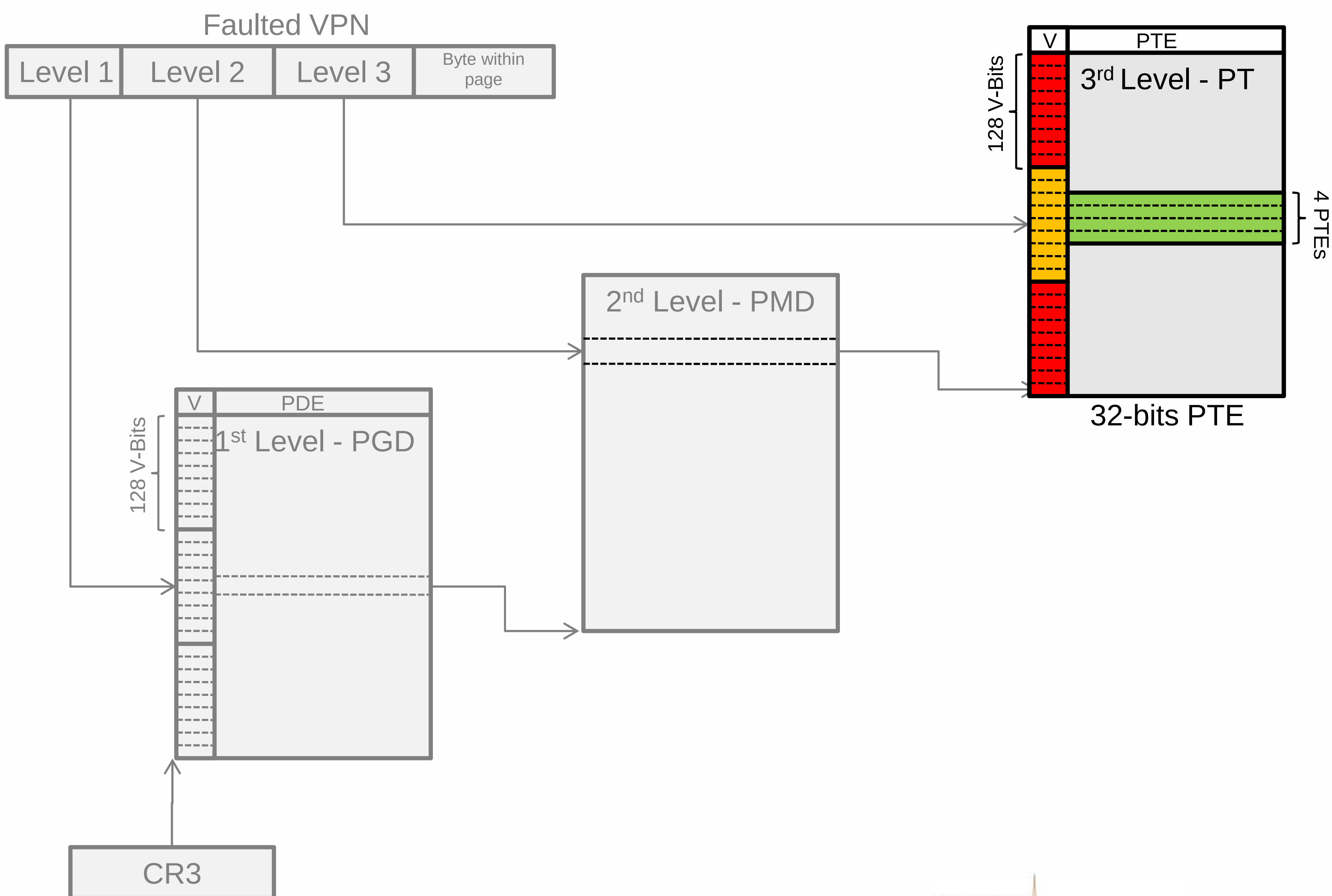


PID returned to the OS only after initial encryption

Page Table Encryption (x86)

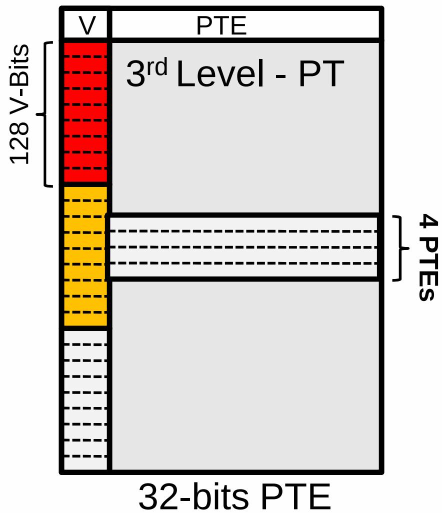


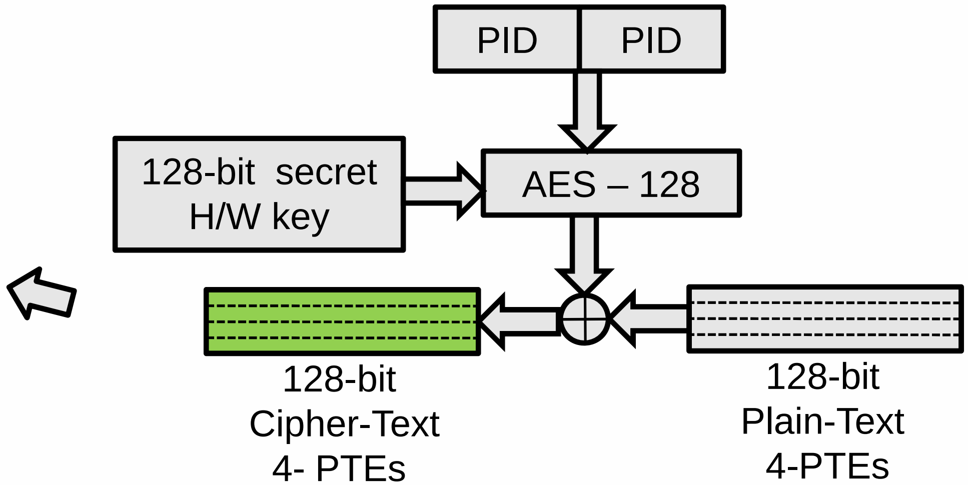
Page Table Encryption (x86)

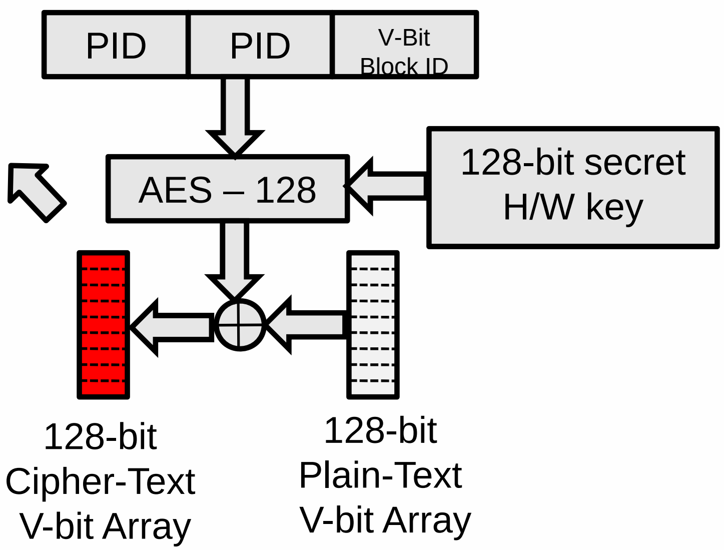


SHARK: Vasisht & Lee

Page Table Encryption (x86)



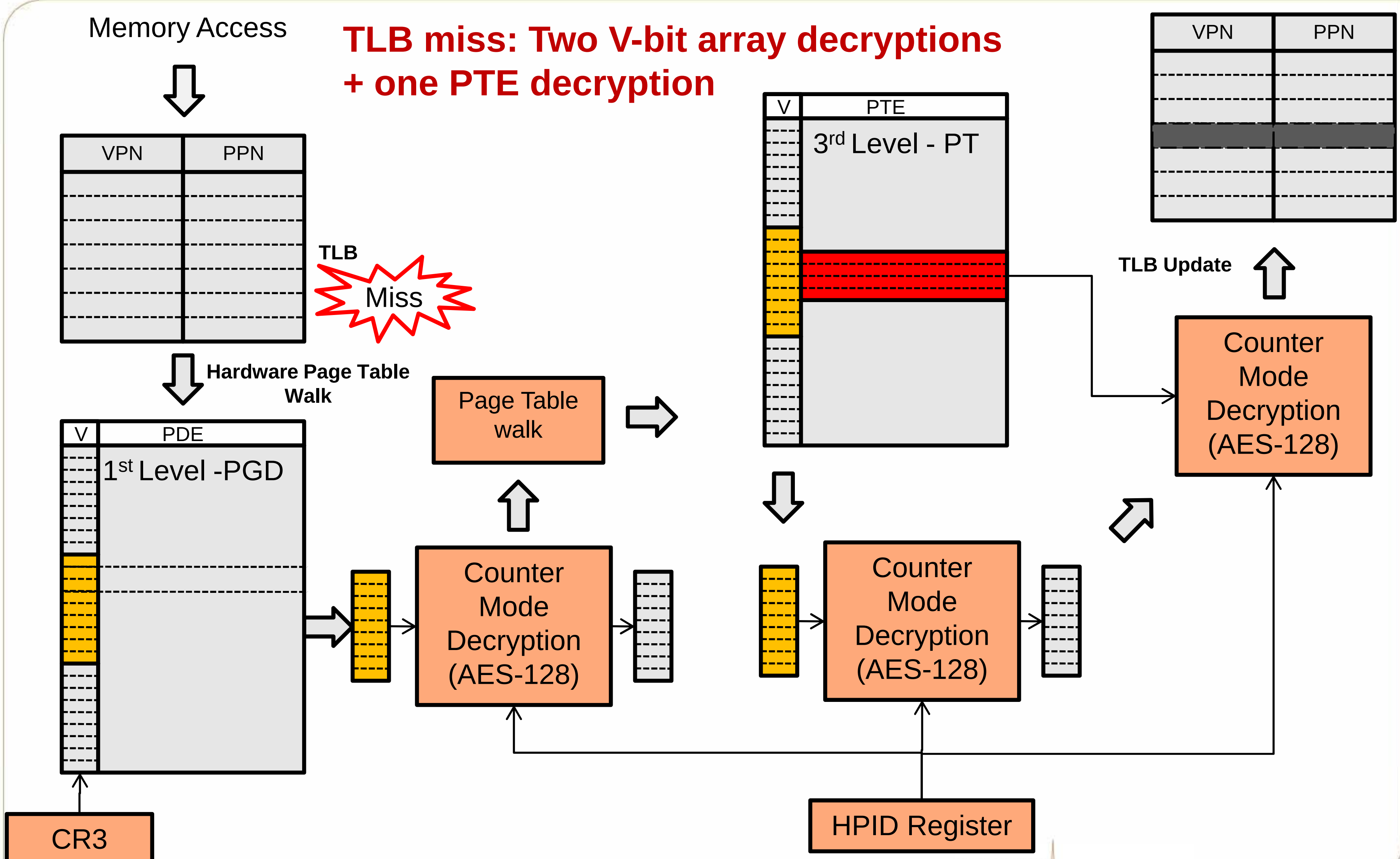




Counter(PID) not a secret;
HW key is secret

TLB Update (x86) – Handled by SSM

TLB miss: Two V-bit array decryptions + one PTE decryption

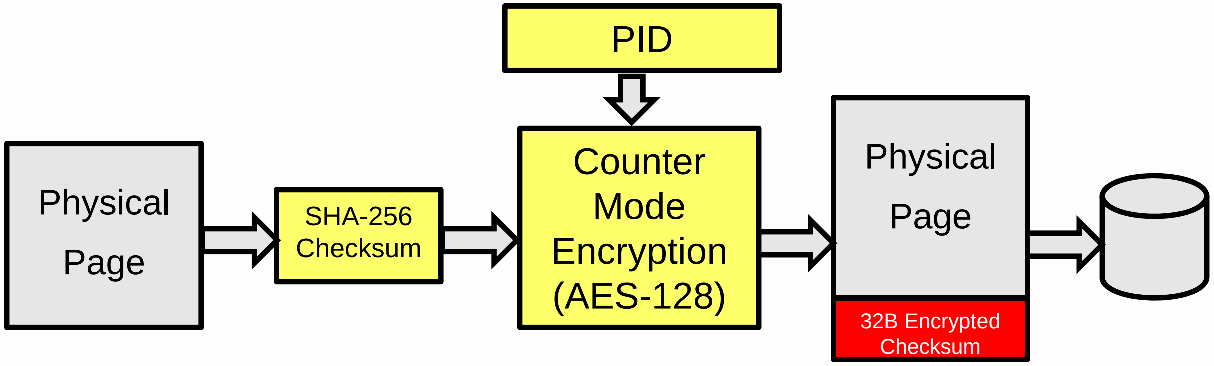


Instructions supported in SHARK

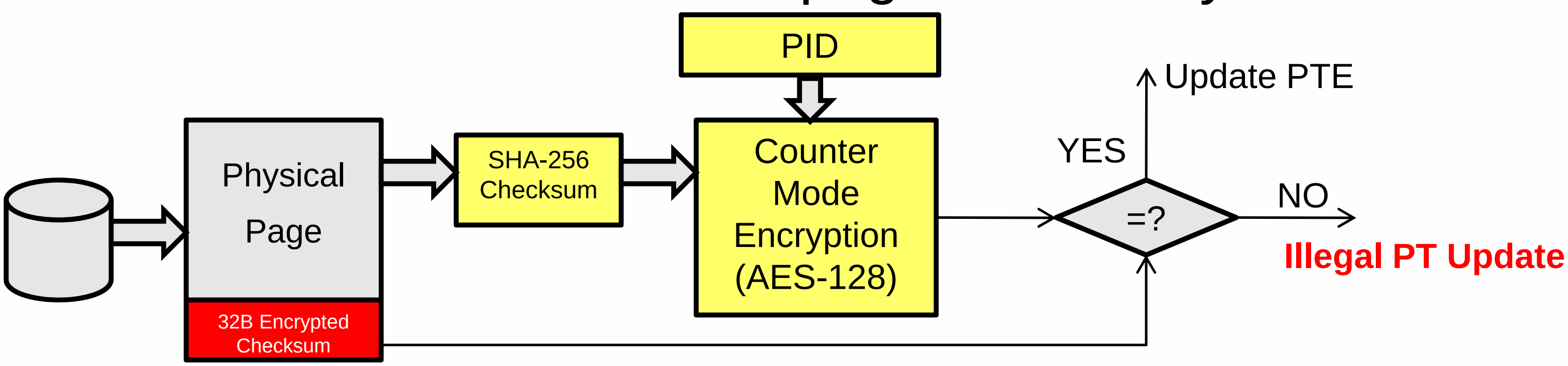
- GENPID- Generate a new PID
  - Used when a new process is created
- MODPT- Update the page table of a process
  - Used when page tables have to be modified
- DECPT- Decrypt a process' page table entry
  - Used to know the physical pages of processes

MODPT: Physical Page Tracking

- MODPT used to Invalidate a page table entry:

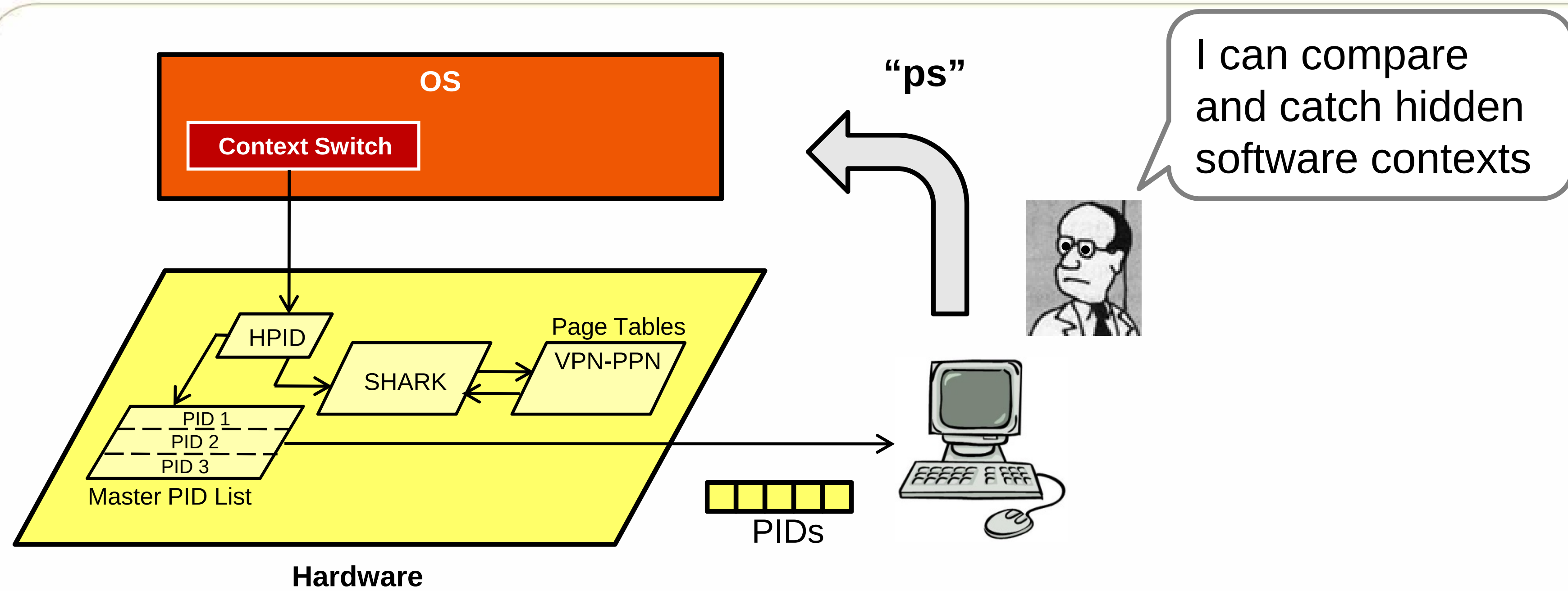


- MODPT used to Validate a page table entry:



Tracks the association of memory page and owning process

Stealth Checker



- Implemented in Firmware
- Encrypts and sends PIDs to a remote system admin machine
- Hardware and software lists compared in the remote machine

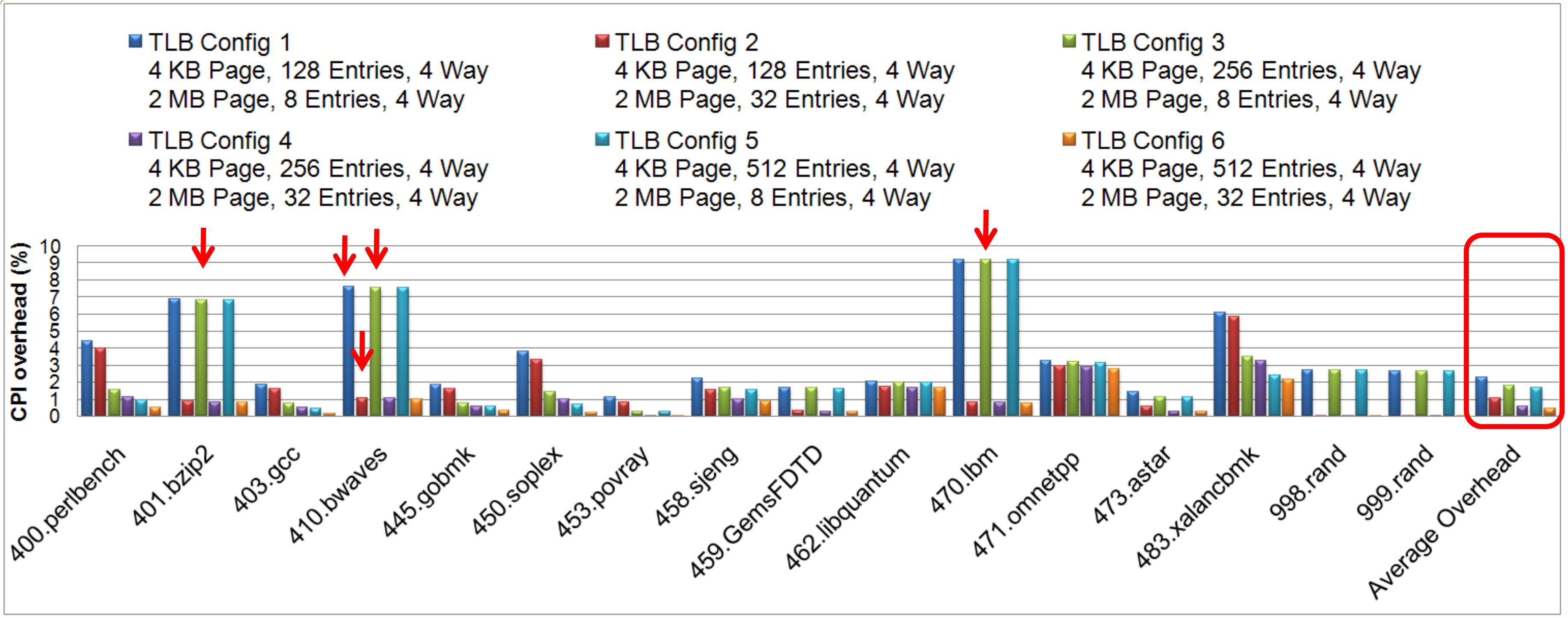
Experimental Analysis

- Functionality Evaluation
  - BOCHS emulator + modified Linux 2.6.16.33
  - Rootkits installed: Adore 0.42, Knark 2.4.3, Phide, Enyelkm.en.v1.1, and Mood-nt-2.3
  - SHARK was able to detect all rootkits
- Performance Evaluation
  - VirtuTech SIMICS
  - Performance overhead due to encryption/decryption

Performance Evaluation

- SPEC 2006 benchmark suite
- Emulated first 2B instructions
  - More page faults and TLB updates
- SHARK Overhead in recompiled Linux kernel 2.6.16.33
  - MODPT instruction: 6 * AES + SHA-256
  - TLB Refill: 3 * AES
  - DECPT instruction: 3 * AES
- Sensitivity study for different TLB configurations
  - 4 KB and 2 MB pages supported (x86)
  - Varied number of TLB entries
- TLB flushed upon every context switch as in x86 machines

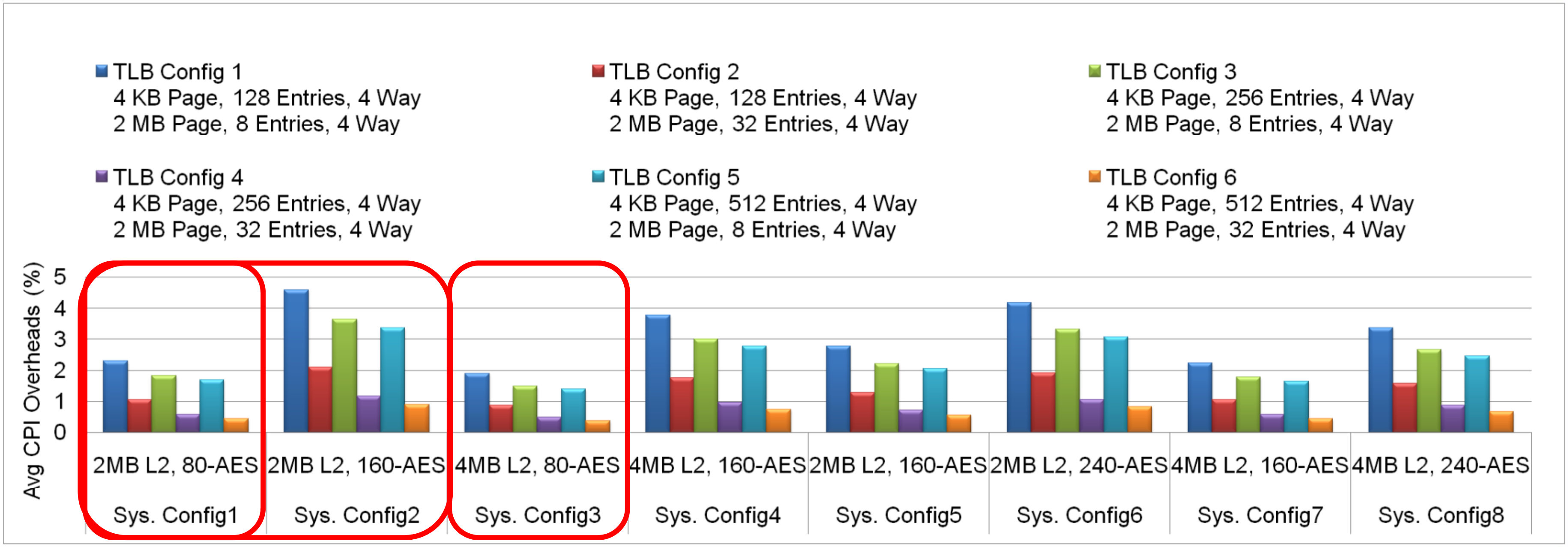
SPEC2006



Performance impact with different TLB organizations

- More context switches and more TLB misses
- Sensitive to the number of entries for 2MB pages in TLB
- Average CPI overhead is 1.3%

SPEC2006 (6 System Configurations)



- Larger AES latency increases the overhead
- Larger L2 cache (longer L2 latency) lowers the overhead
- Average overhead:

  Range : 0.45% - 4.7%

Conclusions

- SHARK is the first synergistic micro-architecture and OS technique to address the Rootkit exploits
- Concealed activity at User, Kernel and VMM levels will be revealed
- Low performance overhead makes it practical

Thank you

Home of the 1996 Olympic Village

Georgia Tech

http://arch.ece.gatech.edu