BTAC
与
机场安全模型

Lenx Wei & Xiaoning Li

Authors

* Xiaoning Li 李晓宁
  * Intel Security Lab资深华人安全研究员
  * 对Intel CPU Security Feature有众多贡献
  * 每月往返于Portland和湾区

Authors

* Lenx Wei 韦韬
  * Baidu X-Lab 百度安全实验室
  * 链接中美，衔接学术界与工业界
  * 每月往返于北京与湾区

BTAC与机场安全模型

两个往返于机场的奶爸对内存安全的思考

Lenx Wei & Xiaoning Li

源起

内存漏洞伴随可寻址内存的产生而产生

* C/ASM/C++... 传统系统编程语言缺乏内存安全机制
  * buffer overflow/UAF/Random RW/...
* 人脑无法掌控Raw pointer引发的复杂性爆炸
  * C语言魔咒
* 产品Benchmark大战加剧了过度优化和指针滥用
  * Benchmark魔咒

洞是挖不完的
防御者的噩梦

* C语言魔咒/Benchmark魔咒
  * 别忘了 编译器、ld、lib、JVM、Driver、Kernel等等
* Exploit Mitigation漏洞缓解技术日益获得重视
  * Stack Cookie
  * DEP
  * ASLR
  * EMET的努力
* 但是，还不够...
  * Pwn2Own每年打脸...
  * iOS Jailbreak...

CFI的提出

* Control-Flow Integrity Principles, Implementations, and Applications, *Martin Abadi, Mihai Budiu, Ulfar Erlingsson, Jay Ligatti*, CCS'05

* 维持程序里原有的控制流完整性，防止控制流注入和任意劫持

CFI的6年沉默

* Performance overheads

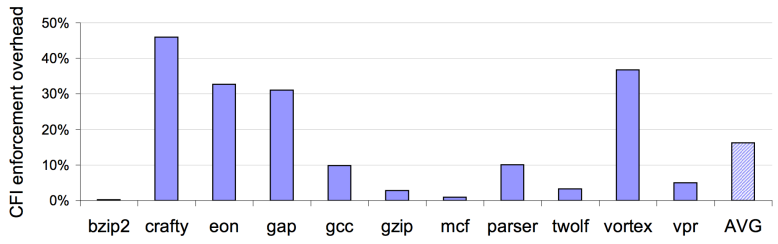
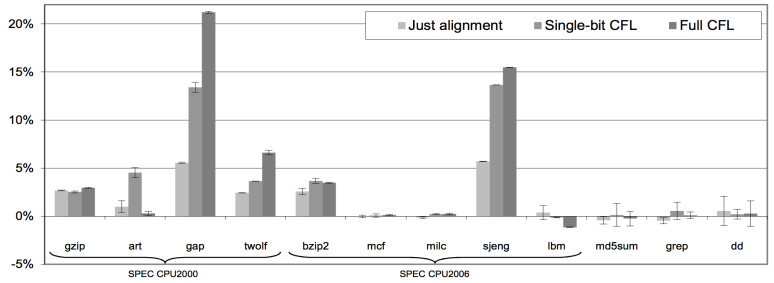


Figure 4: Execution overhead of inlined CFI enforcement on SPEC2000 benchmarks.



* Compatibility nightmare
  * Legacy libraries

CFI的再起

* FPGate by PKU&BitBlaze teams(2012)
  * Protect Function pointers, virtual methods and so on
  * Provide a binary compatibility solution for PE files
  * **Max: 1.41%, Avg: 0.36%**
  * MS Bluehat Prize Contest Special Recognition Award

  * Followed by
    * Control-flow Guard, MS, 2014
    * Enforcing Forward-Edge Control-Flow Integrity in GCC & LLVM, *Google*, 2014

CFI的复兴

* CCFIR, S&P'13
* CFI for COTS, Usenix Security'13

* KCoFI, S&P'14
* MCFI, PLDI'14

* OCFI, NDSS'15
* CCFI, CCS'15

CFI的困境

* OoC, S&P'14
* DVE, by Yuange
* CFB, Control-flow bending, Usenix Security'15
* Control Jujutsu, CCS'15

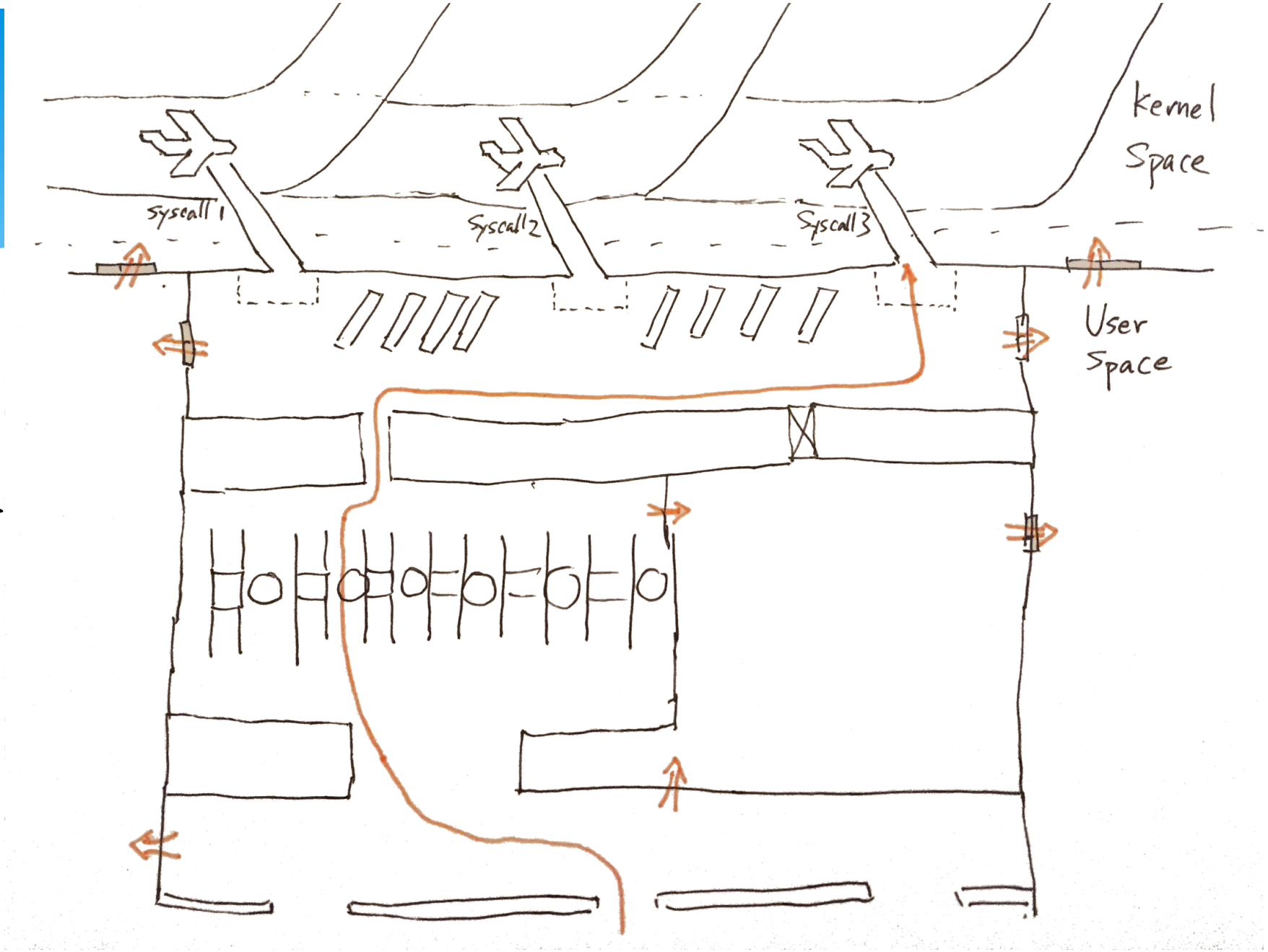
To be CFI, or not to be?

* 细粒度不是CFI的未来
  * Performance overhead
  * Compatibility issues
  * 内存里的猪队友
    * DVE attacks
* 何去何从?

CFI角色的反思

* Con:
  * CFI不可能完成所有的防护
* Pro:
  * CFI增加了一个新的防护层，一个新的基石

机场安全类比



机场安全模型

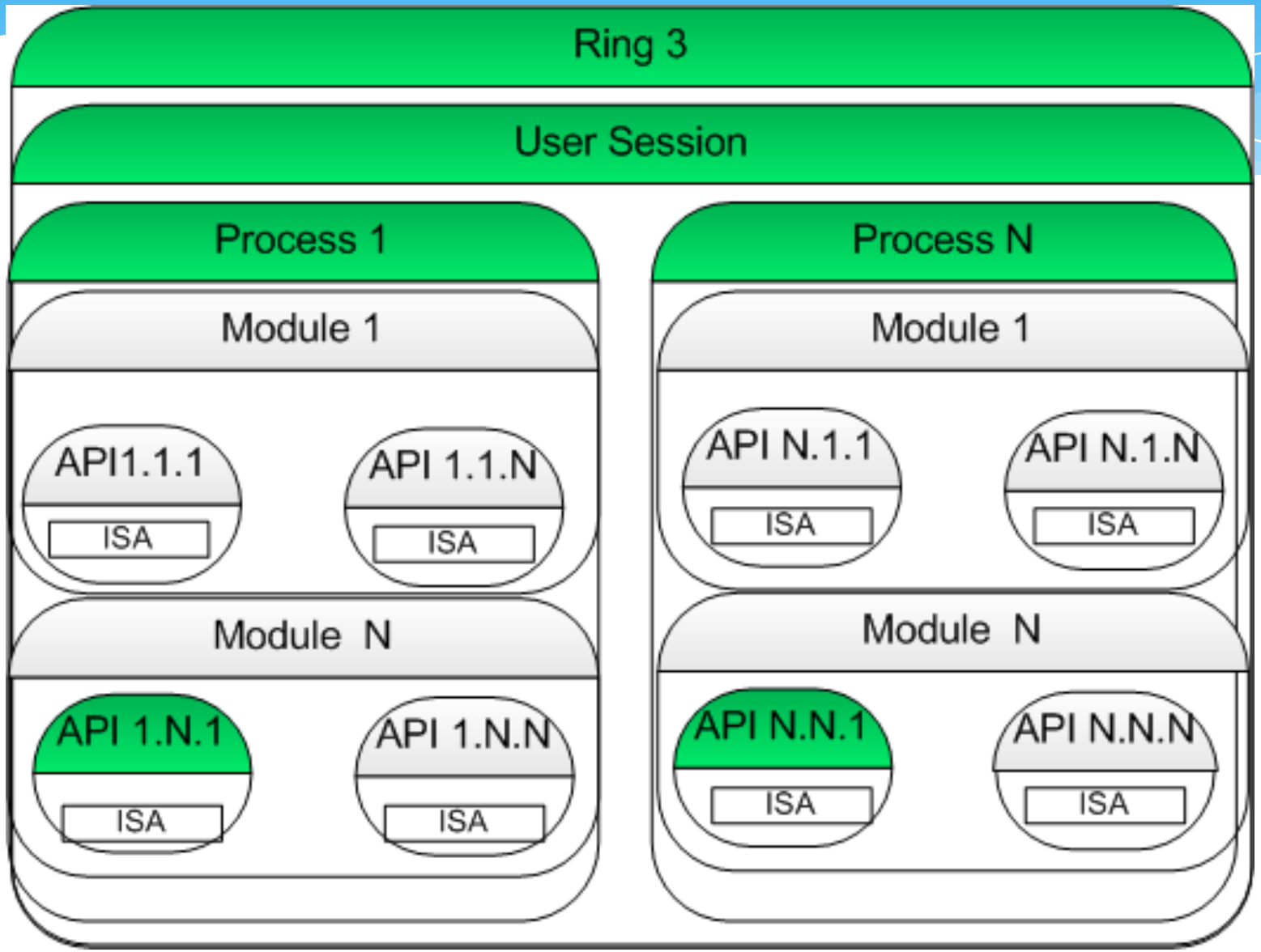
* CFI保证了显性的执行通道
  * 攻击者很难击穿墙壁，绕过安全门
  * 攻击者依然可以选择所有允许的通道

* 关键点(API/Syscall)必须要可靠的Access Control
  * 每个访问的授权凭证(Passport/Boarding card)
  * 关键检查点的不可绕过可以依靠CFI来保证

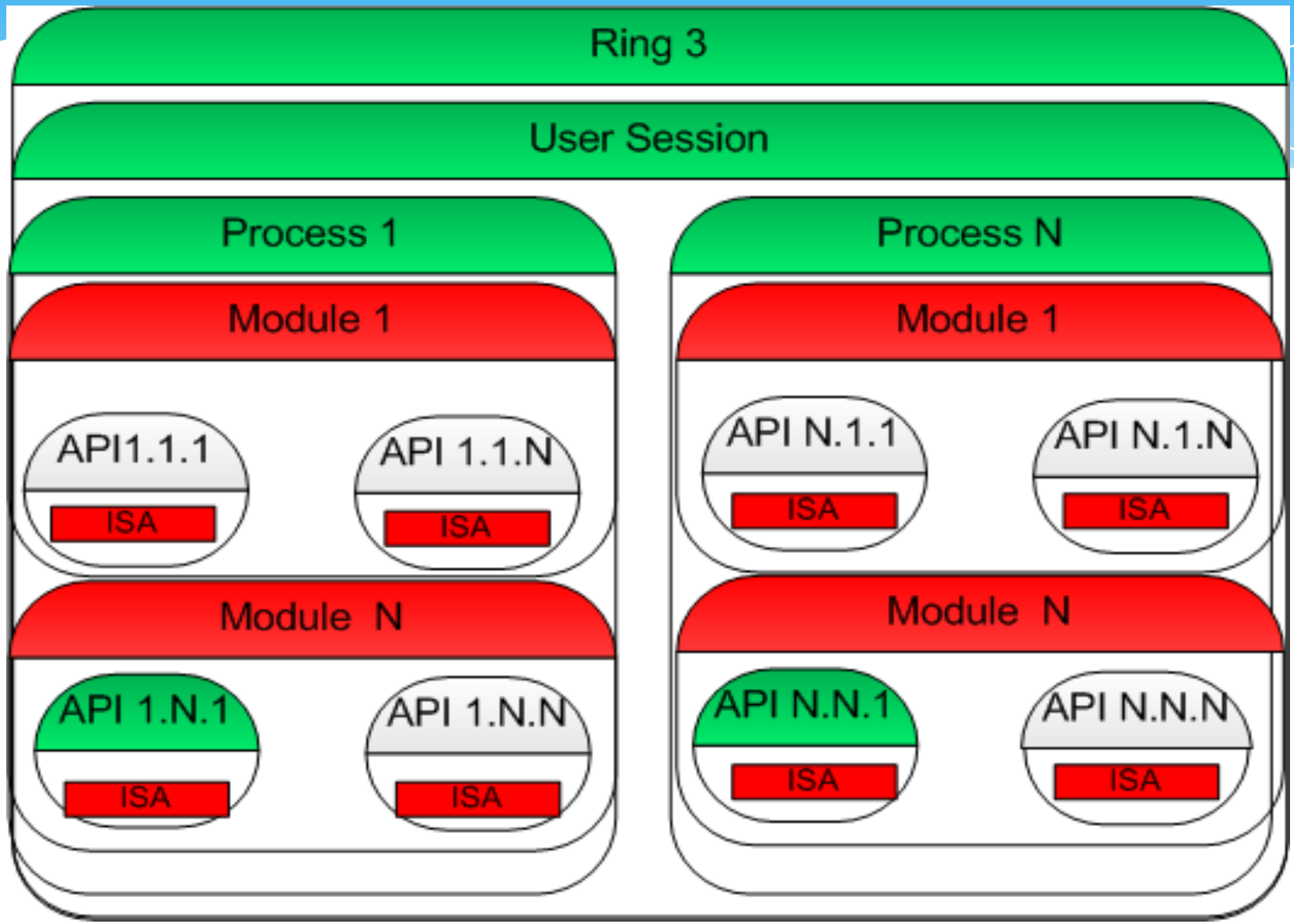
Existing System Access Controls

* User level
  * Resource access per user permission
* Process Level
  * Resource access per process integrity level
* API Level
  * API resource access per HIPS policy

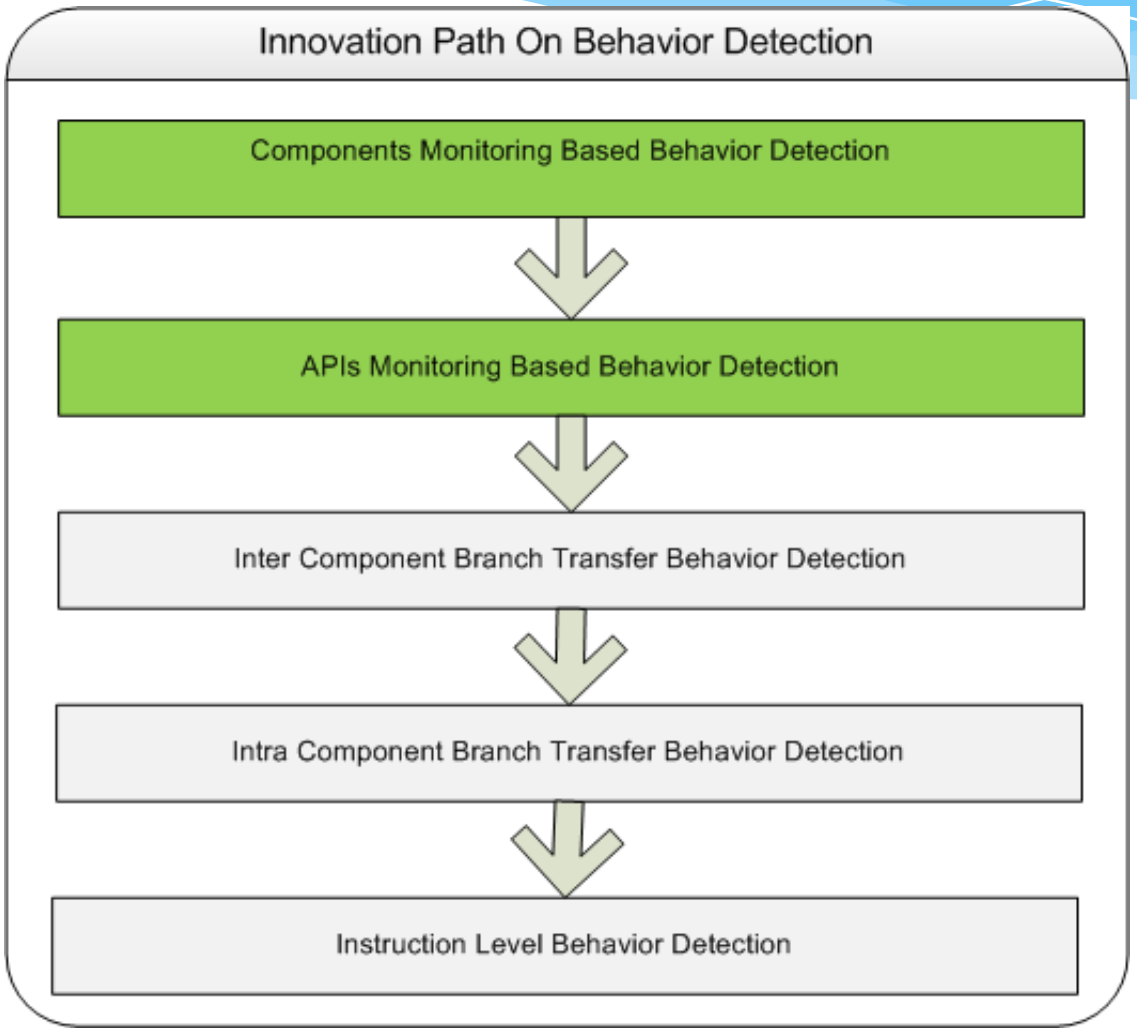
Granularity from OS Ring 3 view



Module/ISA Level granularity gap

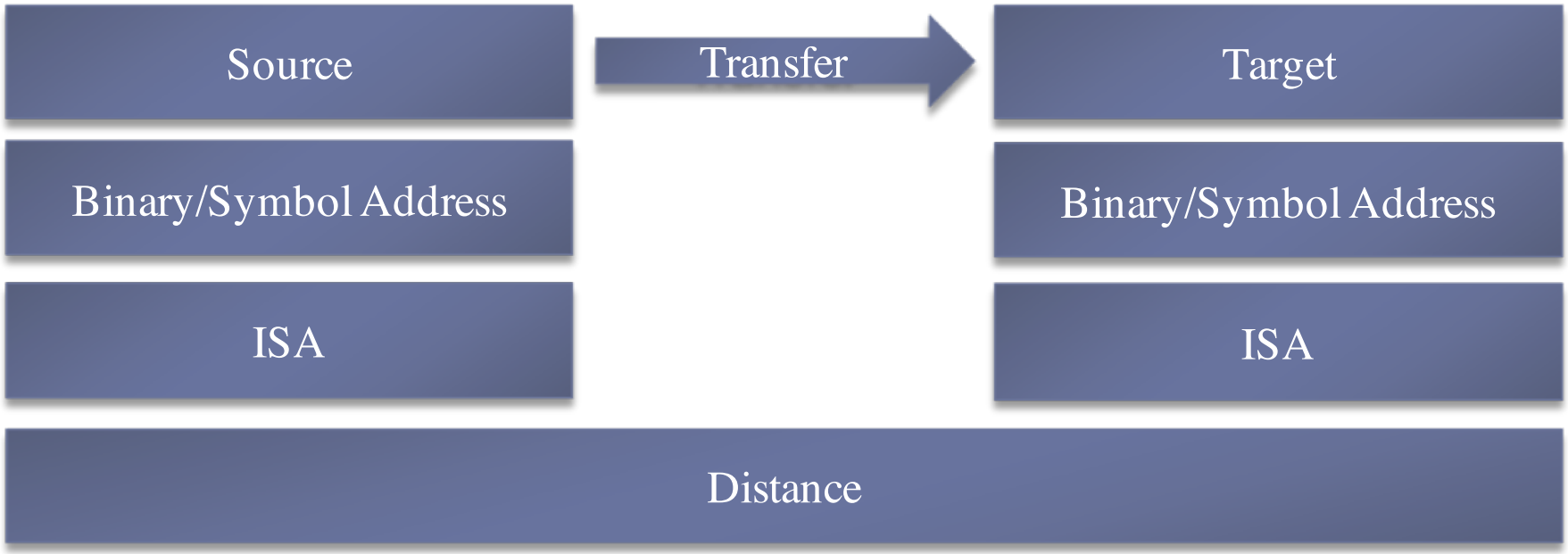


需要更细的AC粒度


Innovation Path On Behavior Detection
Components Monitoring Based Behavior Detection
APIs Monitoring Based Behavior Detection
Inter Component Branch Transfer Behavior Detection
Intra Component Branch Transfer Behavior Detection
Instruction Level Behavior Detection

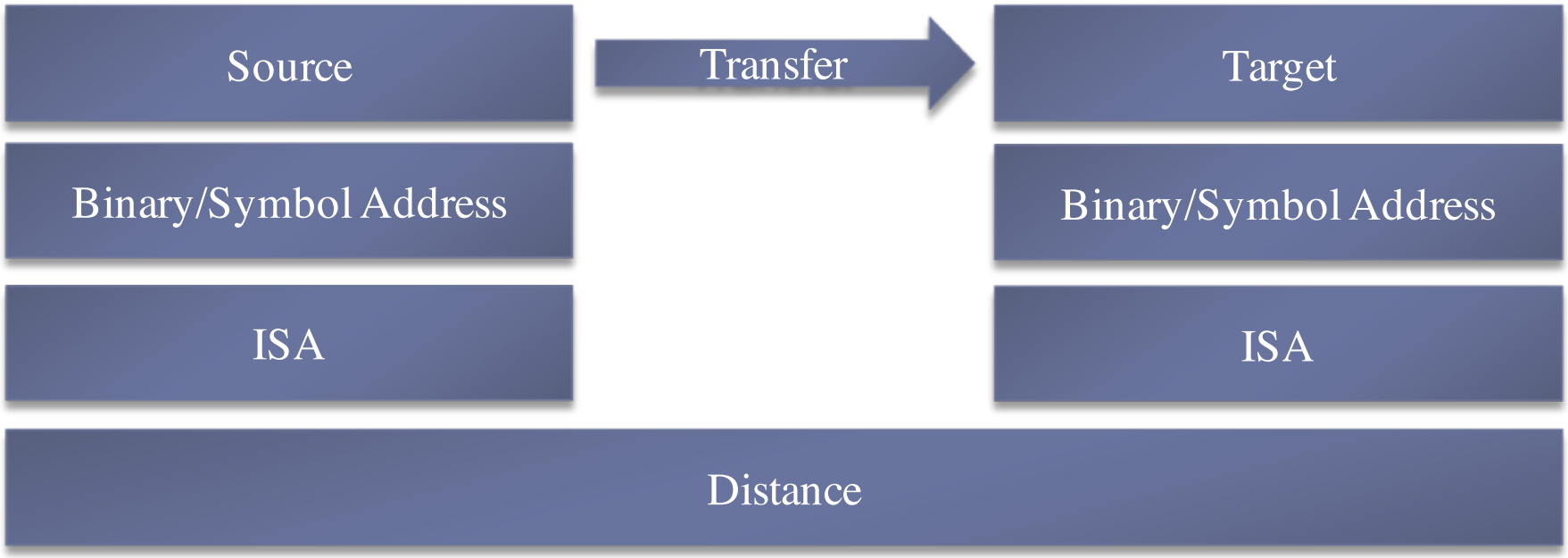
Branch Transfer Access Control (BTAC)

* Inter-components Branch Transfer Access Control
  * Branch access per Component
* Intra-components Branch Transfer Access Control
  * Branch access per branch transfer properties



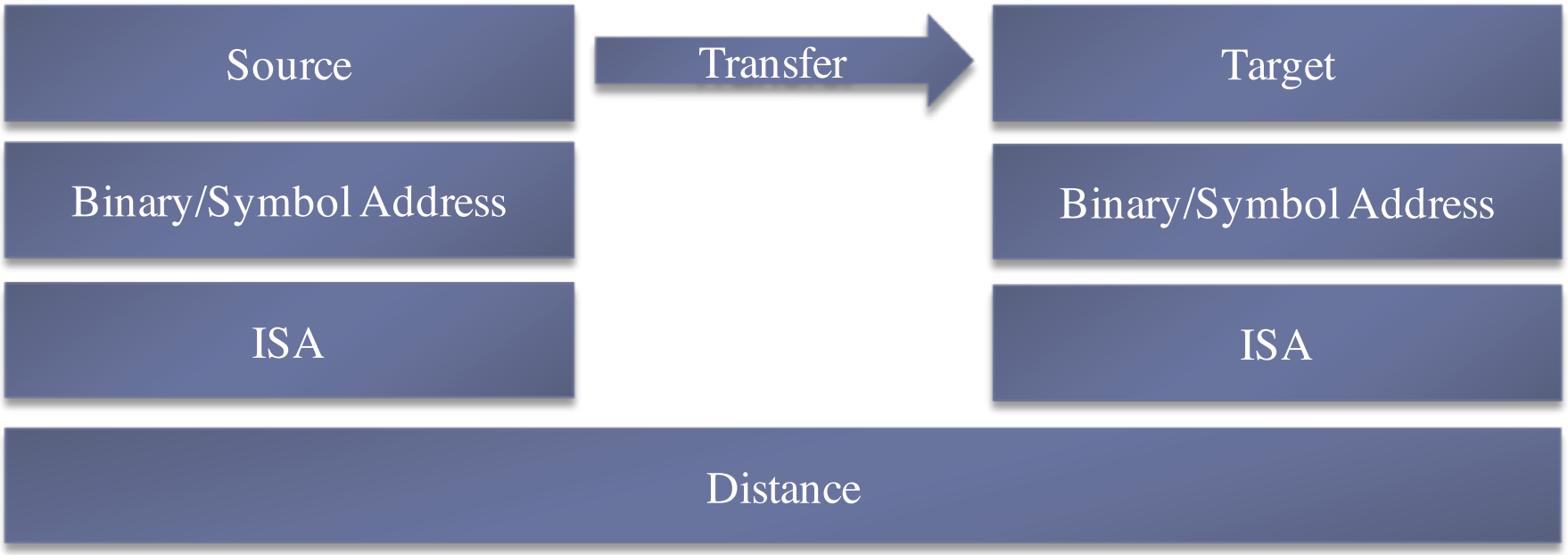
BTAC vs CFI

* BTAC能够使用更多的上下文信息来判定跳转是否合法
* CFI仅考虑control flow integrity



Valid Gadgets

* 细粒度CFI再细，也无法阻止valid gadgets
* Ret to Call-site， call2libc



Syscall Access Control (SCAC)

* BTAC的特例： Syscall Access Control
  * Explicit legal syscall instructions
    * Guaranteed by BTAC or CFI
  * Access control based on In-process context
    * e.g. memory sealing
    * More e.g. User-driven Access Control (S&P'12)
  * Performance
    * Syscall is much less frequent invoked than branch transfer
    * Thus we can check more context

Branch Transfer Access Control with BTF

* BTF is the flag in MSR_DEBUGCTLA MSR
* Used to enable single-step on branches

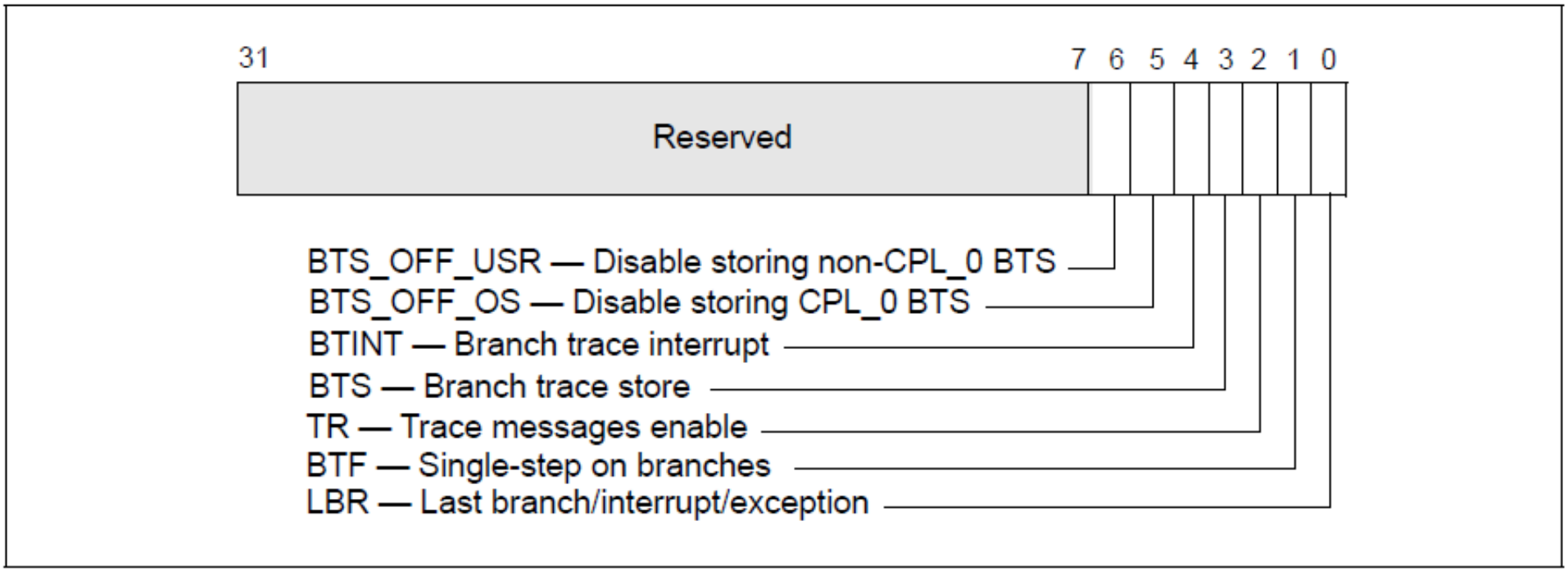
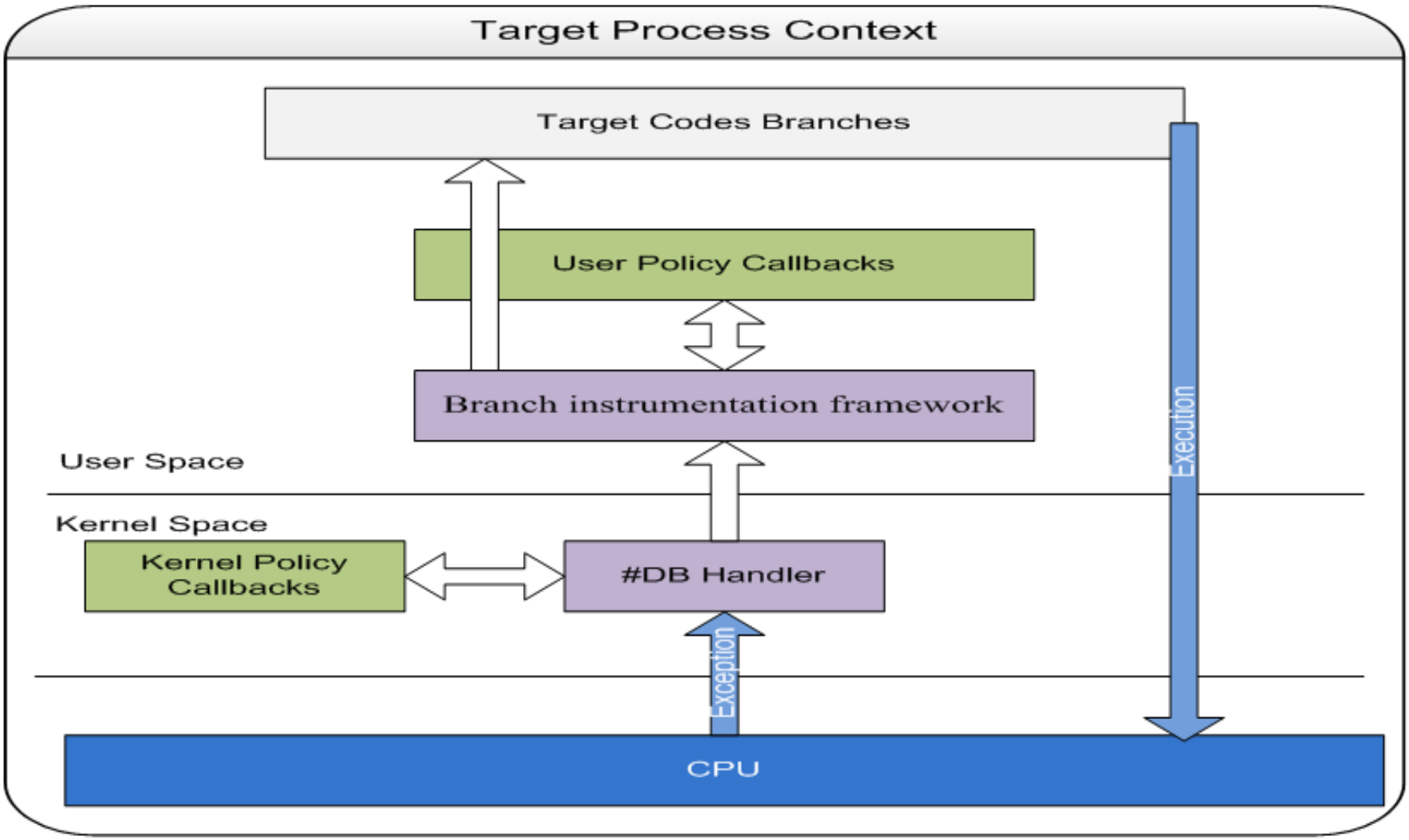


Figure 17-12. MSR_DEBUGCTLA MSR for Pentium 4 and Intel Xeon Processors

BTF based BTAC示例



BTAC vs. GIFT

* GIFT by TK
* Internal function should never be called by arbitrary component, for example MSHTML.DLL -> NTDLL
* The branch transfers happen between many external components' function and NTDLL internal functions are invalid cases

BTAC vs. Vital Point Strike

* 点穴 by TK
* Use Script engine safe mode to call privileged APIs
* The branch transfer between script engine and privileged APIs with sensitive contents should be blocked

BTAC vs. Interdimensional Execution

* DVE by Yuange
* Use script engine to create arbitrary API calls
* JIT code should not directly call into many sensitive APIs

结论

* CFI is another important security layer, not the final solution
* CFI is necessary and effective to support in-process access control, like BTAC, to build a full Airport Security Model
* Branch Transfer Access Control is missed part in today's access control facility
* BTAC provides solid protections against system level intrusions
  * Con: still needs more memory safety support to guarantee in-process abusing
  * Pro: don't require full memory safety

Thanks!

lenx@baidu.com
ldpatchguard@gmail.com