

Pointer Flow Integrity

Формальная спецификация на языке Z

Сессия отладки ядра Windows 11 ARM64 (build 26100)

2026-07-25

1 Назначение и статус

PFI формализует, как *визор* (DBI/секвенсор) наблюдает и изолирует управление и поток данных задачи и как он обнаруживает нелегитимный перехват со стороны эмуляторов антивирусов (AVM) и виртуализационных прослоек (VMP / VM-call). Документ содержит:

1. абстрактную модель секвенции против машинного потока, графа потока указателей (PFG) и инварианта целостности данных (DFI);
2. перенаправление IDP (программный анклав), предикат обнаружения AVM и подмену адреса возврата S-route для атомарных плюзов;
3. конкретную инстанциацию на верифицированной диспетчерской таблице ядра с теоремами.

Замечание о гигиене Z: для адресной арифметики используется функция Shift — *не* символ $\oplus$, поскольку в $Z \oplus$ обозначает overriding функций/отношений.

2 Базовые множества и определения

$[ADDR, INSTR, REF, REG, VAL, SSN, EC, Process]$

$ATOM ::= syscall \mid sysenter \mid indirectCall \mid vmcall \mid gate$

$Mode ::= User \mid Kernel$

$Access ::= Rd \mid Wr \mid Ex$

$Prot ::= ReadOnly \mid RW$

$FlowElem == INSTR \cup ATOM$

$RetGate == \{ syscall, sysenter \}$

$GateAtom == ATOM$

$EA == ADDR$

$Ptr == ADDR$

$Bool ::= true \mid false$

$PACKey == ADDR \times ADDR$

$SignedAddr == ADDR$

$EVENT ::= Enter\langle ATOM \times ADDR \rangle \mid Exit\langle ATOM \times ADDR \rangle$

Глобальные сигнатуры (фиксируются семантикой ISA):

$Accesses : FlowElem \leftrightarrow ADDR$	— отношение «элемент обращается по адресу»
$Select : FlowElem \rightarrow \mathbb{N}$	— размер выборки (байты)
$Shift : ADDR \times \mathbb{Z} \rightarrow ADDR$	— адресный сдвиг
$PageOf : ADDR \rightarrow ADDR$	— база страницы (здесь: large page / PDE)
$Protect : ADDR \rightarrow Prot$	
$WriteFault : \mathbb{P} ADDR$	— адреса, запись в которые вызывает отказ
$Exec : ADDR \rightarrow Bool$	— страница исполняемая

3 Поток управления: секвенция против машинного потока

Визор производит *секвенцированный* поток Cf (исполняется из буфера визора), процессор исполняет *машинный* поток Xf . Они расходятся на атомах, когда исполнение покидает буфер (смена адреса снимает хуки AV-эмулятора).

Sequencing
$Cf : \text{seq } FlowElem$
$Xf : \text{seq } FlowElem$
$Cf \neq \langle \rangle$

Diverges
$a? : ATOM$
$k : \mathbb{N}$
$Cf\ k = a?$
$Xf\ k \neq a? \vee \#Xf \neq \#Cf$

Атом $a?$ в позиции k потока Cf *расходится*, если машинный поток в этой позиции не равен $a?$ или длины потоков различны — формальная запись утверждения «атомы уходят из-под секвенсора».

4 Состояние задачи

TS
$regs : REG \rightarrow VAL$
$frame : \text{seq } VAL$
$prevMode : Mode$
$lr : ADDR$
$retSlot : ADDR$
$prevMode \in \{User, Kernel\}$

Поле $retSlot$ — адрес, по которому возобновляется исполнение после системного атома; на верифицированном ядре это пара lr/fp , восстанавливаемая в `KiSystemServiceCopyEnd+0x38` инструкцией `ldp fp,lr,[sp,#0x50]`.

5 Граф потока указателей (PFG)

PFG
$ts : TS$
$CS : \mathbb{P} Ptr$
$IS : FlowElem \rightarrow \mathbb{P} Ptr$
$enclave : \mathbb{P} ADDR$
$\delta : \mathbb{Z}$
$enclave \neq \emptyset$

CS — текущее множество живых указателей, IS — разрешённое входное множество на элемент потока, $enclave$ — изолированная область (программный анклав), δ — смещение перенаправления IDP.

6 Инвариант целостности данных (DFI)

Центральное свойство: любое обращение в память, выполняемое секвенированным элементом, должно быть по адресу из текущего множества или из входного множества этого элемента.

DFI
PFG
$\forall e : FlowElem; ea : EA @$ $(e \mapsto ea) \in Accesses \Rightarrow ea \in CS \cup IS(e)$

Обращение, чьего EA нет ни в одном из множеств, не может быть порождено санкционированным потоком — это сигнатура внедрённого перехвата (критерий фиксации AVM / OP-инъектов).

7 IDP — перенаправление EA в программный анклав

Redirect
$ea? : EA$
$ea! : EA$
$ea! = Shift(ea?, \delta)$
$ea! \in enclave$

При секвенции визор сдвигает эффективный адрес на δ , направляя выборку данных в изолированный анклав; эмулятор АВ, читающий *исходный* адрес, не видит трафика.

8 Обнаружение AVM / эмулятора

AVMDetected
$e? : FlowElem$
$ea? : EA$
$(e? \mapsto ea?) \in Accesses \wedge ea? \notin CS \cup IS(e?)$

Свидетельство AVMDetected в ходе секвенции $\Rightarrow$ присутствует нелегитимная эмуляция/перехват. Это двойник инварианта DFI: DFI утверждает отрицание для санкционированного потока.

9 Атомы и S-route

$enabled : \mathbb{P} ATOM$ — множество наблюдаемых атомов
--

Наблюдение атомов (ядерный аналог — KiTrackSystemCallEntry/Exit, включается битом 0 по PpmPolicyConfigTable+0xD00):

ObserveAtom
$a? : ATOM$
$target? : ADDR$
$evt! : EVENT$
$a? \in enabled$
$evt! = Enter(a?, target?)$

S-route: для возвратного атома визор сохраняет исходный адрес возврата и подменяет его своей заглушкой, чтобы управление вернулось в секвенсор, а не в API эмулятора АВ.

$STUB : ADDR$	— заглушка возобновления S-route
$OK : EC$	
$NtCode : \mathbb{P} ADDR$	— кодовая область обработчиков nt!Nt*

SRoute
ΔPFG
$a? : ATOM$
$saved! : ADDR$
$a? \in RetGate$
$saved! = ts.retSlot$
$ts.retSlot' = STUB$
$CS' = CS; IS' = IS; enclave' = enclave; \delta' = \delta$

RestoreReturn
ΔPFG
$saved? : ADDR$
$ts.retSlot' = saved?$

10 Конкретная инстанциация ядром (верифицировано)

Значения сняты непосредственно с живого ядра.

$KiServiceTableBase : ADDR$
$KiArgTable : ADDR$
$KiServiceLimitValue : \mathbb{N}$
$DescriptorAddr : ADDR$
$InvalidSvcEC : EC$
$KiServiceLimitValue = 489$

Функция декодирования реализует инструкцию `add x11,x11,x10,asr#4`:

$Decode : ADDR \times \mathbb{N} \rightarrow ADDR$
$\forall b : ADDR; e : \mathbb{N} @$ $Decode(b, e) = Shift(b, e \div 16)$

10.1 Дескриптор и таблица сервисов

ServiceTable
$base : ADDR$
$limit : \mathbb{N}$
$number : ADDR$
$table : SSN \rightarrow \mathbb{N}$
$base = KiServiceTableBase$
$limit = KiServiceLimitValue$
$number = KiArgTable$

10.2 Диспетчеризация (успешный путь)

Реализует `cmp x8,x10; bhs fail` и `ldrsw/add ... asr#4` из `KiSystemService`:

Dispatch
<i>ServiceTable</i> <i>ssn?</i> : <i>SSN</i> <i>handler!</i> : <i>ADDR</i> <i>status!</i> : <i>EC</i>
<i>ssn?</i> < <i>limit</i> <i>ssn?</i> ∈ dom <i>table</i> <i>handler!</i> = <i>Decode(base, (table ssn?) ÷ 16)</i> <i>status!</i> = <i>OK</i>

10.3 Выход за пределы (путь ошибки)

Реализует `mov x0,#0x1C; movk x0,#0xC000,lsl#0x10` в `KiSystemServiceCopyEnd+0x88`:

InvalidService
<i>ServiceTable</i> <i>ssn?</i> : <i>SSN</i> <i>status!</i> : <i>EC</i>
<i>ssn?</i> ≥ <i>limit</i> <i>status!</i> = <i>InvalidSvcEC</i>

10.4 Целостность таблицы (отображение только для чтения)

Проверено командой `!pte fffff8004bc96c28`: атрибуты PDE -R-GA-K-LV (large page, **без W**), тогда как обёртка-дескриптор по адресу `fffff8004cc01900`: -W-GADK-LV (доступна для записи):

TableIntegrity
<i>ServiceTable</i> <i>Protect(PageOf(base))</i> = <i>ReadOnly</i> <i>Protect(PageOf(DescriptorAddr))</i> = <i>RW</i>

Сама таблица смещений обработчиков отображена read-only на уровне PDE (large page); запись в неё из ядра вызывает data abort. Это ядерный аналог IDP-защиты визора.

11 Дополнительная верификация: аппаратная и ядерная реализация целостности

11.1 Trap-фрейм

TrapFrame
<i>pc</i> : <i>ADDR</i> <i>spsr</i> : <i>ADDR</i> <i>regs</i> : <i>REG</i> → <i>VAL</i>

11.2 PAC — аутентификация указателей (ARMv8.3)

Аппаратный аналог целостности указателей из `rfi.pdf`. Функции ядра подписывают адрес возврата в прологе инструкцией `pacibsp` (PAC, В-ключ, модификатор SP); `paciasp/pacibsp` одно-

временно служит landing pad для BTI.

$Sign : ADDR \times PACKey \rightarrow SignedAddr$ $Auth : SignedAddr \times PACKey \rightarrow ADDR$ $PACKeyOf : Process \rightarrow PACKey$ $EnIBOf : Process \rightarrow Bool$

PACSign
$lr? : ADDR$
$k? : PACKey$
$lr! : SignedAddr$
$lr! = Sign(lr?, k?)$

Свидетельство (дизассемблер): `nt!NtClose`, `nt!NtQueryInformationProcess`, `nt!KiTrackSystemCallEntry` начинаются с `d503237f pacibsp`.

KernelCtxt
$apdbKey : PACKey$
$enIB : Bool$

PACContextSwitch
$\Delta KernelCtxt$
$p? : Process$
$apdbKey' = PACKeyOf(p?)$
$enIB' = EnIBOf(p?)$

Свидетельство: в `nt!KiSystemServiceExit` — `mrs SCTL_R_EL1`, переключение бита `EnIB` (bit 30) по флагу процесса, загрузка `APDBKeyLo/Hi` (`S3_0_C2_C1_2/3`), `isb`. Ключи PAC переключаются по процессам.

Аутентификация (полный цикл PAC): `nt!KiTrackSystemCallExit` завершается инструкцией `d50323ff autibsp` непосредственно перед `ret` — пара подпись (`pacibsp`) на входе → аутентификация (`autibsp`) на возврате подтверждена.

Входная конфигурация (зеркало выхода): `nt!KiConfigPointerAuthKernelEntry` на входе в ядро — `mrs SCTL_R_EL1`; `bfi` бита `EnIB` по флагу `[+0x880]` (bit 2); загрузка ключей из `[+0x8F0]` в `APDBKeyLo/Hi` (`S3_0_C2_C1_2/3`); `isb`. Источник ключей и флаг `EnIB` — per-process.

11.3 Возврат в R3: `ELR_EL1` — точка подмены S-route

SystemCallReturn
$tf : TrapFrame$
$elr' : ADDR$
$spsr' : ADDR$
$elr' = tf.pc$
$spsr' = tf.spsr$

Свидетельство: `msr ELR_EL1, x2`, где $x2 \leftarrow [sp, \#0x140]$ (trap-фрейм), затем очистка регистров/SIMD и `eret` по адресу `fffff8004c236068`. Именно `tf.pc` → `ELR_EL1` есть целевой адрес возврата в R3 — объект подмены S-route. Перед возвратом вызывается `nt!KiPreflightReturnToUserMode` — `preflight` проверки целостности.

11.4 Теневая и фильтрующая таблицы

ShadowServiceTable
$ntBase : ADDR; ntLimit : \mathbb{N}$
$guiBase : ADDR; guiLimit : \mathbb{N}$
$ntBase = KiServiceTableBase$
$ntLimit = 489$
$guiLimit = 1493$

Свидетельство: KeServiceDescriptorTableShadow — дескриптор[0] = nt (489), дескриптор[1] = win32k.sys (Limit=0x5D5=1493); KeServiceDescriptorTableFilter — отдельный base для win32k (sandbox).
Выбор per-thread: tst x10,#0x80; tst x10,#0x200000 в KiSystemService.

11.5 Наблюдение атомов: KiTrackSystemCallEntry

Схему ObserveAtom уточняем верифицированными деталями: чтение PreviousMode (ldrsb w0,[x8,#0x252] — поле _KTHREAD+0x252); фильтр nt!KeIsTraceCallbackAllowed; поиск по ключу-обработчику в nt!KiSystemServiceTraceTable; атомарный счётчик (ldaddal).

11.6 Граница user/kernel и Probe (DFI)

$MMU_{userProbe} : ADDR$

Свидетельство: nt!ProbeForRead — проверка выравнивания (tst x8,x0) и диапазона против константы 0x7FFFFFFF0000 (mov x8,#0x7FFFFFFF0000; cmp x9,x8; ccmpls), затем принудительный fault по пробуемому адресу (ldr w8,[x8]); nt!ProbeForWrite — read/write-touch (ldrsb/strb) через границы страниц. Гейт по PreviousMode находится в вызывающей Nt-функции, а не в самой Probe*: в nt!NtQueryInformationProcess — ldr x8,[xpr,#0x988]; ldrsb w21,[x8,#0x252] (чтение PreviousMode); cbnz w21,... (ветвление в user-путь с пробингом через 0x7FFFFFFF0000).

ProbeRead
$ptr? : ADDR$
$len? : \mathbb{N}$
$ptr? + len? \leq MMU_{userProbe}$

11.7 Копирование аргументов: Input Set

$Number : SSN \rightarrow \mathbb{N}$
— число доп. аргументов на user-стеке

ArgCopy
$ssn? : SSN$
$n! : \mathbb{N}$
$n! = Number(ssn?)$

Свидетельство: счётчик из Number-таблицы управляет computed branch adr x8,CopyEnd; sub x10,x8,x10,lsl#3; br x10 в лестницу ldr/str (nt!KiSystemServiceCopyStart) — копируется ровно столько qword с user-стека, сколько декларировано во входном множестве вызова.

11.8 Атом syscall на ARM64 — SVC

На ARM64 атом системного вызова (вместо x86 sysenter/syscall) — инструкция SVC. Цепочка входа разрешена: SVC → nt!Hv1UserSyntheticExceptionHandler (вектор lower-EL/AArch64, trap-фрейм 0x370) → nt!KiConfigPointerAuthKernelEntry (конфигурация PAC на входе) → nt!KiBuildTrap nt!KiSyntheticException → nt!KiSystemService.

11.9 PAN/UAO — не используются (отрицательное заключение)

На данном билде PAN/UAO в тракте syscall не применяются. Проверены конфигураторы PAC на входе/выходе (`nt!KiConfigPointerAuthKernelEntry`, `nt!KiSystemServiceExit`) и сами функции syscall — ни одного `msr pan/msr uao/ldtr`. Копирование user-стека и пробинг выполняются обычными `ldr/str`. Разделение user/kernel здесь обеспечивается *программно*: гейт `PreviousMode + ProbeFor` ница `MM_USER_PROBE_ADDRESS`; отказы при user-доступе ловит SEH-обработчик `nt!KiSystemServiceHandler` `nt!RtlUnwindEx` (для fault-PC в диапазоне `KiSystemServiceExit..CopyEnd`).

11.10 Контроль цели косвенного вызова (CFG/CET)

В `nt!KiTrackSystemCallExit` перед `blr x15` вызывается `nt!KscpCfgCheckUserCallTargetEs` — проверка цели вызова по конфигурации CFG (Control Flow Guard): косвенный вызов санкционируется отдельно, поверх PAC/BTI. Дополнительный механизм целостности потока управления.

11.11 W^X и очистка состояния

Свидетельство: кодовая страница `nt!NtClose` — PDE -RXGA-K-LV (исполняемая, без W $\Rightarrow$ W^X). На выходе из системного вызова ядро обнуляет SIMD v0–v31 и x1–x15 перед `eret` (DFI: отсутствие утечки данных ядра через регистры).

12 Теоремы

Теорема 1 (Корректность диспетчеризации). *Ограниченный, присутствующий в таблице SSN разрешается в nt-обработчик.*

$$\vdash \forall ServiceTable; Dispatch @ handler! \in NtCode$$

Свидетельство: `KiServiceTable[0]  $\rightarrow$  nt!NtAccessCheck`, `[1]  $\rightarrow$  nt!NtWorkerFactoryWorkerReady`.

Теорема 2 (Отклонение вне диапазона). *Любой SSN не меньше лимита даёт верифицированный код ошибки.*

$$\vdash \forall ServiceTable; InvalidService @ status! = InvalidSvcEC$$

Теорема 3 (Принудительная целостность). *Отображение read-only вызывает отказ при любой записи на страницу таблицы.*

$$\vdash \forall ServiceTable; TableIntegrity @ Protect(PageOf(base)) = ReadOnly \Rightarrow PageOf(base) \in WriteFault$$

Теорема 4 (DFI исключает AVM). *При выполнении DFI ни одна пара (элемент, адрес) не удовлетворяет предикату обнаружения.*

$$\vdash DFI \Rightarrow \neg(\exists e : FlowElem; ea : EA @ AVMDetected(e, ea))$$

Доказательство (набросок). *Раскрывая определения: DFI даёт $(e \mapsto ea) \in Accesses \Rightarrow ea \in CS \cup IS(e)$, что прямо противоречит посылке $ea \notin CS \cup IS(e)$ из AVMDetected.* $\square$

Теорема 5 (S-route удерживает управление секвенсора). *После S-route над возвратным атомом слот возврата ядра не равен исходному, поэтому управление после атома не может уйти в API эмулятора.*

$$\vdash \forall SRoute @ ts.retSlot' \neq ts.retSlot$$

Доказательство (набросок). *$ts.retSlot' = STUB$, причём STUB контролируется визором, откуда $STUB \neq saved! = ts.retSlot$.* $\square$

Теорема 6 (Целостность возврата по PAC). Адрес возврата, подписанный ключом процесса, нельзя подменить на произвольный адрес без отказа аутентификации.

$$\vdash \forall PACSign; p? : Process @ PACKeyOf(p?) = k? \Rightarrow \forall a \neq lr? @ Auth(lr!, k?) \neq a$$

Теорема 7 (W^X). Исполняемые страницы ядра недоступны для записи.

$$\vdash \forall a : NtCode @ Protect(PageOf(a)) = ReadOnly \wedge Exec(PageOf(a))$$

Теорема 8 (Граница user/kernel). Любая попытка probes диапазоном, выходящим за `MM_USER_PROBE_ADDRES`, отклоняется (*fault*).

$$\vdash \forall ProbeRead @ (ptr? + len? > MMUserProbe) \Rightarrow (ptr? + len? \in WriteFault)$$

13 Таблица соответствия понятий

Понятие <i>pfi.pdf</i>	Схема / определение Z	Якорь в ядре (верифицировано)
Секвенция <i>Cf</i> против потока <i>Xf</i>	Sequencing, Diverges	буфер визора против <code>blr x11</code>
Атом	ATOM (свободный тип)	шлюз <code>syscall/sysenter</code> в <code>KiSystemService</code>
TS (состояние задачи)	TS	trap-фрейм, <code>prevMode_KTHREAD+0x252</code>
CS / IS / EA / P	компоненты PFG	входы декодирования <code>KiServiceTable</code>
Инвариант DFI	DFI	<code>ProbeForRead/Write</code> (<code>prevMode=User</code>)
IDP + анклав	<code>Redirect, enclave, δ</code>	со стороны визора; ядерный аналог — read-only таблица критерий рассогласования <code>InputSet</code>
Обнаружение AVM	<code>AVMDetected</code>	
S-route	<code>SRoute, RestoreReturn, STUB</code>	<code>[sp, #0x50]</code> <code>retSlot</code> , <code>KiSystemServiceCopyEnd+0x38</code>
Диспетчерская таблица	<code>ServiceTable, Dispatch, Decode</code>	<code>KiServiceTable, Shift(b, $\cdot \div 16$)</code>
Лимит сервисов	<code>ServiceTable.limit = 489</code>	<code>KiServiceLimit = 0 $\times$ 1E9</code>
Неверный сервис	<code>InvalidService</code>	<code>0 $\times$ C000001C</code>
Целостность таблицы	<code>TableIntegrity, \Protect</code>	PDE <code>-R-GA-K-LV</code> (read-only large page)
Наблюдение атомов	<code>ObserveAtom, enabled</code>	<code>KiTrackSystemCallEntry/Exit</code> (бит 0), <code>+0x252</code>
Целостность указателей (PAC)	<code>PACSign, PACContextSwitch</code>	<code>pacibsp</code> , <code>APDBKey</code> , <code>SCTLR_EL1.EnIB</code>
Возврат в R3 / S-route	<code>SystemCallReturn (ELR_EL1)</code>	<code>msr ELR_EL1, [sp, #0x140]; eret</code>
Preflight	<code>KiPreflightReturnToUserMode</code>	проверка перед возвратом в R3
Теневые/фильтр. таблицы	<code>ShadowServiceTable</code>	<code>Shadow(win32k 1493)/Filter; tst per-thread</code>
BTI / W^X	<code>Exec, \Protect</code>	PDE <code>-RXGA-K-LV</code> ; <code>pacibsp</code> как landing pad
Очистка состояния	(гигиена / DFI no-leak)	<code>v0-v31, x1-x15 $\rightarrow$ 0</code> перед <code>eret</code>

Граница user/kernel (DFI)	ProbeRead, MMUserProbe	ProbeForRead/Write, 0x7FFFFFFF0000
Input Set (копирование арг.)	ArgCopy, Number	br x10 в лестницу ldr/str, CopyStart
Атом syscall (ARM64)	SVC HvlUserSyntheticExceptionHandler → KiSystemService	→ вместо x86 sysenter/syscall
PAN/UAO	— (не используется)	программный гейт: PreviousMode + Probe + MM_USER_PROBE_ADDRESS
SEH syscall	KiSystemServiceHandler RtlUnwindEx	→ fault при user-доступе → исклю- чение
Входная конфигурация PAC	KiConfigPointerAuthKernelEntry	ключи [+0x8F0], EnIB [+0x880] bit 2
PAC: аутентификация	Auth	autibsp перед ret (KiTrackSystemCallExit)
DFI: гейт PreviousMode	(Nt-вызывающий)	ldrsb [+0x252]; cbnz в NtQueryInformationProcess
CFG (цель вызова)	—	KscpCfgCheckUserCallTargetEs перед blr

Конец спецификации.