DASICS-安全处理器设计用户手册

DASICS - Secure Processor Design User Manual

版本：v2.1.2

中国科学院计算技术研究所
先进计算机系统实验室系统结构组

2023 年 4 月 10 日

目录

1 DASICS 安全处理器设计方案概述

DASICS(Dynamic in-Address-Space Isolation by Code Segments) 是一种安全处理器设计方案。该方案通过对不同代码片段所访问的内存地址空间进行隔离并设置各自的访存权限，同时对代码片段间的控制流加以限制，从而实现对非预期的越界访存和跳转的防护。这类越界访存主要来自于非可信的第三方库代码。DASICS 的特点是通过硬件检查的方式对每条访存和跳转指令进行检查，在出现权限错误的时候触发用户态中断以处理错误：

DASICS 的权限是以代码片段作为一个主体（subject）单位进行权限划分的。所谓代码片段，是指一段地址连续的代码，其范围可以由权限更高的代码片段动态的在一组 DASICS 特殊寄存器中进行设置。而经过设定权限后，硬件会根据当前指令所处的指令地址（PC）所在的代码片段范围判断当前的访存权限。代码片段的指令如果访问超出设定的范围，则发生用户态（或同级）中断，由权限更高的代码片段进行处理。同时为了防止低权限代码跳转到高权限代码进行非法的权限升级，DASICS 规定低权限代码仅能通过很少的方式进入高权限代码，从而对代码片段间的控制流进行了严格限制。

DASICS 将代码片段分为可信区和不可信区，可信区是在同一个地址空间中具有最高权限的代码片段，主要包含可信管理函数、例外处理函数、可信调用函数等。可信区代码可以修改当前系统特权态（内核态、用户态、管理态）所允许的所有地址空间和寄存器，可以跳转到所有（可执行）代码入口，不会受到 DASICS 访问控制的影响。而非可信区函数主要用来放置不可信的第三方代码（包括第三方库或者第三方内核驱动）。非可信区代码的访问权限要受到由可信区代码所制定的访问权限限制，同时非可信区到可信区的控制流需要进行检查和限制，非可信区只能通过函数返回和可信调用等控制流转移进入可信区。可信区代码通过设置前面说的 DASICS 寄存器来限制非可信区的访问。

DASICS 还实现了可信区对非可信区的系统调用截获。非可信区可能通过系统调用来进行间接访存从而绕过 DASICS 内存保护机制，因此 DASICS 中一旦非可信区进行了系统调用，就触发例外跳转到可信区的例外处理函数，例外处理函数对这些系统调用的参数进行检查，禁止违法的系统调用并放行正常的系统调用，从而减少了系统调用层面的攻击。

2 DASICS 新增寄存器介绍

2.1 配置寄存器 (config)

2.1.1 DasicsMainCfg

该寄存器为 DasicsSMainCfg(0xBC0) 和 DasicsUMainCfg(0x5C0) 两个逻辑寄存器所共用的物理寄存器，DasicsMainCfg 的组织结构如图 1 所示，其中高 XLEN-4 位暂时保留。低 4 位中，M 态可读写全部低 4 位，S 态主函数区仅能读写 UENA 和 UCLS 两位。

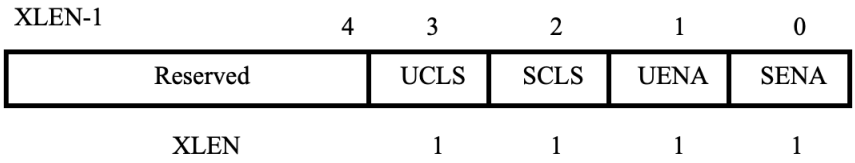

图 1: DasicsMainCfg 寄存器结构示意图

- **DasicsSMainCfg(0xBC0)**：控制 S 态主函数区的逻辑 CSR 寄存器，对应的物理寄存器为 DasicsMainCfg
  - M 态可以读写 DasicsMainCfg 的全部低 4 位
  - DasicsMainCfg 的第 0 位为使能位 (SCFG_ENA)，该位置一时，启动对 S 态的分区保护机制，即位于 DasicsSMainLo 与 DasicsSMainHi 之间的代码段将会被视为 S 态主函数区域；置零则关闭 S 态分区保护机制，即认为所有在 S 态执行的代码都是可信的，但 U 态的分区保护机制仍可以正常运行。
  - DasicsMainCfg 的第 2 位为 S 态清零位 (SCFG_CLS)，该位置一时，将会在下一个时钟周期时对除 DasicsMainCfg 以外的 DASICS 寄存器进行清零，并将 DasicsMainCfg 的 SCLS、UENA、UCLS 位清零。
- **DasicsUMainCfg(0x5C0)**：控制 U 态主函数区的逻辑 CSR 寄存器，对应的物理寄存器为 DasicsMainCfg
  - S 态主函数区仅能读写 DasicsMainCfg 的 UENA 和 UCLS 两位，M 态可以读写 DasicsMainCfg 的全部低 4 位
  - DasicsMainCfg 的第 1 位为使能位 (UENA)，该位用来使能 U 态的 DASICS 保护机制，即位于 DasicsUMainLo 与 DasicsUMainHi 之间的代码段被视为 U 态的主函数区域。此外，操作系统可以在 spawn 的时候进行设置，从而创造可信进程。
  - DasicsMainCfg 的第 3 位为清零位 (UCLS)，该位置一时，将会在下一个时钟周期时对除 DasicsMainCfg、DasicsSMainBoundHi、DasicsSMainBoundLo 以外的 DASICS 寄存器进行清零，并将 DasicsMainCfg 的 UCLS 位清零。

2.1.2 DasicsLibCfg0(0x881)、DasicsLibCfg1(0x882)

- DasicsLibCfg 的设计模仿了 PMP 的 config 寄存器：每个 DasicsLibCfg 包含 8 个 4-bit 的 config，每个 config 对应一组 DasicsLibBound 寄存器。当 CSR 寄存器位宽 XLEN=64 时，每个 DasicsLibCfg 奇数索引的 config 不会使用。
- 对于单个 config：
  - 第 3 位为有效位 (LIBCFG_V)，表示该 config 对应的 LibBound 是否生效；
  - 第 2 位为执行位 (LIBCFG_X)，在目前的设计中，该位置 1 时，LibBound 对应的区域成为活跃区，从而无须设置额外的活跃区寄存器；
  - 第 1 位为可读位 (LIBCFG_R)，表示该 config 对应的 LibBound 内的数据能否被库函数读取；
  - 第 0 位为可写位 (LIBCFG_W)，表示该 config 对应的 LibBound 内的数据能否被库函数写入。

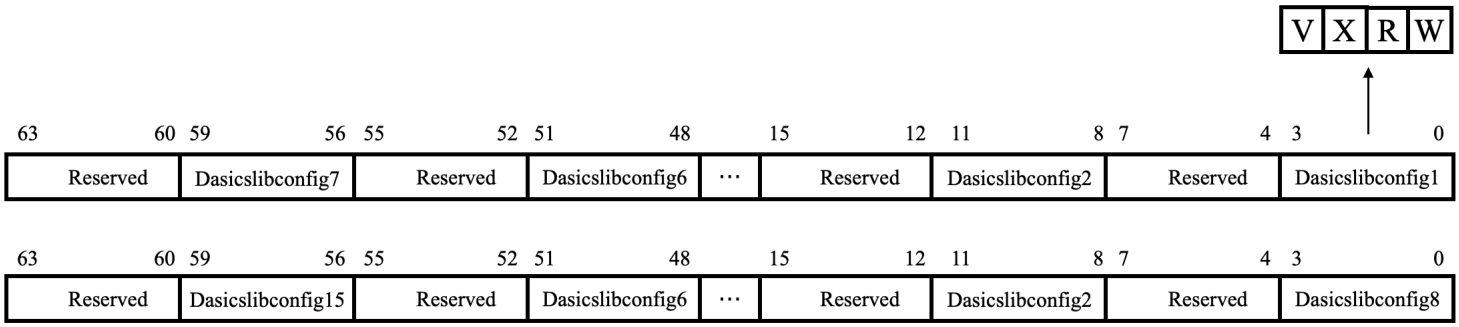

图 2: 64 位架构下 DasicsLibCfg 寄存器结构示意图

2.2 限界寄存器 (bound)

2.2.1 DasicsSMainBoundHi(0xBC1), DasicsSMainBoundLo(0xBC2)

这两个寄存器为 S 态主函数代码区上下界寄存器，M 态、使用 sbi 调用的 S 态主函数可配置

- 目前的实现中，S 态主函数区边界先由 M 态的 BBL 在 minit.c 中进行静态配置（配置为全空间），再在进入 S 态主函数区范围之后，由主函数区代码进行 sbi 调用（sbi_modify_smain_bound）来更新寄存器。此外，在该调用的实现中，会先判断发起调用的代码是否为 S 态主函数区，如果不是的话会报错终止程序。
- S 态的主函数在链接的时候，会被集中放置在.smaintext 段中。因此，S 态主函数通过 sbi 调用来设置 DasicsSMainBoundHi 与 DasicsSMainBoundLo 分别为该段的上下界。

2.2.2 DasicsUMainBoundHi(0x5C1)、DasicsUMainBoundLo(0x5C2)

这两个寄存器为 U 态主函数区上下界寄存器，M 态、S 态主函数可配置

- U 态的可信函数在链接的时候，会被集中放置在.umaintext 段中。因此，S 态主函数可以根据该段的上下界来配置 DasicsUMainBoundHi 与 DasicsUMainBoundLo 两个寄存器。

2.2.3 DasicsLibBound(0x883 0x8A2)

一共 16 组 32 个库函数边界寄存器，M 态、S 态主函数、U 态主函数可配置

- 在 0x883-0x8A2 这 32 个寄存器中，由低地址到高地址按照 (Hi, Lo) 来排列库函数边界寄存器，即第 i 组的库函数边界寄存器，其上界编号为 0x883 + i * 2，下界编号为 0x884 + i * 2，并且与 DasicsLibCfg0（i<8）或者 DasicsLibCfg1（i>=8）中的 config 相对应。

2.3 控制流 & 可信调用相关寄存器

2.3.1 DasicsReturnPC(0x8A4)

可信调用库函数返回地址寄存器。在可信调用库函数时，硬件会自动保存调用结束时的返回地址 (PC+4)。如果程序从库函数跳转回主函数区的目标地址不为 DasicsReturnPC 所记录的返回地址，也不为可信调用入口地址，则引发 DASICS 跳转例外。该寄存器设置为可读写，是因为在可信调用的入口中，需要暂时保存寄存器的值，并在调用结束时恢复。

2.3.2 DasicsFreeZoneReturnPC(0x8A5)

活跃区返回地址寄存器，当控制流由库函数非活跃区跳转进入库函数活跃区时，硬件会自动保存库函数非活跃区的调用返回地址 (Pc+4)。如果程序从库函数活跃区返回库函数非活跃区时的目标地址不为 DasicsFreeZoneReturnPC 所记录的返回地址，则会引发 DASICS 跳转例外。该寄存器设置为可读写，理由同 DasicsReturnPC。

2.3.3 DasicsMaincallEntry(0x8A3)

可信调用入口寄存器。该寄存器可以在程序上下文切换的时候进行复用，即在发生程序上下文切换前的 U 态时，该寄存器为 U 态可信调用的入口地址；在发生程序上下文切换后的 S 态时，该寄存器为 S 态可信调用的入口地址。在 S 态主函数和 U 态主函数中，程序可以自己定义可信调用入口函数，并设置 DasicsMaincallEntry 的值为该函数的起始地址。可信调用模仿了系统调用，为库函数提供了一个可信区代码的调用入口，同时可以在入口函数中对传入的调用参数进行检查。默认的可信调用入口的工作流程为：

1. 由于入口函数之后要跳转到其他主函数区函数，而这些主函数区的函数又会调用其他的库函数，从而会覆盖掉先前保存的返回地址，因此需要先保存 DasicsReturnPC 和 DasicsFreeZonePC 的值。
2. 之后入口函数根据传入的调用号跳转并执行对应的主函数区函数。
3. 执行完成后，恢复先前保存的 DasicsReturnPC 和 DasicsFreeZonePC 的值，并使用 pulpret 指令返回。

2.4 用户态中断相关寄存器、指令、例外号

我们基于 RV-Draft-20210612 中的 N 扩展说明，实现了 RISC-V 架构 N 扩展以实现用户态中断，具体如下：

- 除 CYCLE、TIME、INSTRET 以及浮点相关的用户态 CSR 寄存器以外，我们实现了手册中规定的 N 扩展寄存器及其逻辑，以及 sedeleg、sideleg 两个代理寄存器
- 除 sedeleg 和 sideleg 由 M 态、S 态主函数配置以外，其他诸如 utval 的寄存器由 M 态、S 态主函数、U 态主函数进行配置。
- 实现了 uret 指令，用于用户态中断的例外返回。

3 DASIC 新增指令介绍

目前的 DASICS 项目只增加了一条 *DASICSRET* 指令，其余对 DASICS 寄存器的读写逻辑使用了 RISC-V 架构本身的 CSR 指令。*DASICSRET* 指令码格式如下：

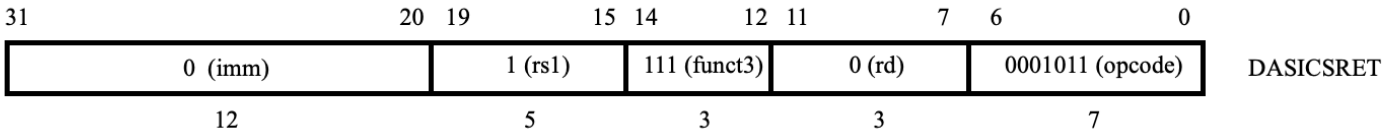

图 3: DASICSRET 指令码格式图

DASICSRET 指令语义和一条正常的 *RET* 指令（即 *JALR x0,0(ra)*）一样，只不过通过特殊的 opcode 进行了标记。该指令主要用于支持可信调用的功能。在可信调用中，控制流由可信区返回非可信区时，使用该条指令说明是主函数嗲用的返回，告知硬件不要修改调用库函数返回地址寄存器 DasicsReturnPC，以区别于正常的库函数调用（控制流也是可信区到非可信区）。

4 DASIC 新增例外介绍

DASICS 新增了 8 种例外，以实现发现 DASICS 访问或者跳转违例时触发例外的功能，以及进行系统调用劫持。

DASICS 例外名	例外号	描述
DASICS_U/S_INST_FAULT	0x18 / 0x19	U 态/S 态跳转例外
DASICS_U/S_LOAD_FAULT	0x1a / 0x1b	U 态/S 态无读权限
DASICS_U/S_STORE_FAULT	0x1c / 0x1d	U 态/S 态无写权限
DASICS_U/S_ECALL_FAULT.	0x1e / 0x1f	U 态/S 态系统调用例外

- 对于 DASICS S 态/U 态执行例外（INST_FAULT）而言，当非可信区函数跳转到可信区的跳转目标地址不等于 dretpc 或者 dmaincall 时，或是非可信活跃区函数想要跳转到返回地址之外的地址时，亦或是非可信区代码跳转到边界寄存器所允许的跳转范围时，引发该例外。
- 对于 DASICS S 态/U 态访存例外（LOAD_FAULT and STORE_FAULT ）而言，当非可信区试图访问允许的访存范围外的地址，即访存地址与读写标记在 DASICS 库函数查找表中没有匹配项时，则根据当前指令是 load 还是 store 分别引发对应特权态下的 DASICS 读/写访存例外。同时，当该例外发生时，需要取消掉 LSU 模块中正要发送给 AXI 总线的访存请求。
- 对于 DASICS S 态/U 态系统调用例外 (ECALL_FAULT) 而言，如果发现当前指令是一条 RISC-V 系统调用指令（即 ecall 指令）就触发该例外，在例外函数中进行系统调用的截获检查。
- DASICS 例外的优先级随着例外号的增加而降低。

5 主要功能介绍

5.1 内核加载进程

含义：进程启动时需要由内核设置好主函数代码区上下界，这样才能使得后面的代码执行具有权限，保证后续的主函数区和库函数区权限设置正常。

流程：当下的实现中使用链接器脚本对.text 和.data 进行了分区。内核根据主函数代码段（.smaintext, .umaintext) 的上下界分别设置 S 态和 U 态的主函数代码区上下界寄存器，把加载函数映射到 S 态主函数区，由加载函数负责加载 elf 文件（合理放置文件中的各个段，代码里库函数区指定为.slibtext .ulibtext，活跃区指定为.sfreezonetext .ufreezonetext）

5.2 可信调用库函数

含义：需要保证跳转到库函数区之后程序权限受到正确的限制。

流程：用户应用程序的主函数中设置库函数数据区上下界寄存器组，跳转时可以使用正常的跳转指令到库函数区，同时硬件设置主函数返回地址寄存器。对于活跃区边界寄存器，可以在 U 态主函数区或者 S 态主函数区进行初始化，目前的实现中将所有需要嵌套调用的子函数视为活跃区，因此把它们放置在同一个代码段之后，在 S 态主函数区进行初始化。

5.3 库函数返回主函数

含义：需要保证控制流的正常，即库函数返回主函数时，返回地址必须和之前跳进库函数区的地址相同。另外，需要特殊区分库函数进行可信调用的情况。

流程：保存跳转地址到库函数返回地址寄存器（DasicsReturnPC 寄存器）。硬件检查跳转地址是否等于主函数返回地址寄存器或可信调用入口寄存器（DasicsMaincallEntry 寄存器），检查失败触发用户态中断。如果等于主函数返回地址寄存器，则继续执行主函数；如果等于可信调用入口，则执行可信调用流程。

5.4 库函数进行可信调用

含义：支持可信调用功能，对库函数区的一些权限进行修改。

流程：主函数根据调用参数进行处理，可能修改库函数权限寄存器和库函数数据区上下界寄存器以及主函数数据区。主函数入口函数先保存 DasicsReturnPC 和 DasicsFreeZoneReturnPC 的值，之后完成调用后恢复两个寄存器的值，再利用 PULPRET 指令返回，保证主函数返回地址寄存器上下文在可信调用前后一致。

5.5 用户态中断

含义：对于权限出现错误，需要进行系统处理的时候，使用用户态中断而非系统中断提高程序性能。

流程：有 6 种中断可能被触发：U 态/S 态跳转错误、U 态/S 态读错误、U 态/S 态写错误，根据代理寄存器配置情况决定是否代理给用户态处理。此外，安全寄存器访问错误按照 RV 规范，视为 illegalInstr 例外处理；系统调用检查错误计划在移植 yzh 师兄的工作之后实现。用户态中断返回使用 URET 指令，返回到 uepc 寄存器所指的位置。

5.6 库函数调用库函数（活跃区）

含义：不同的库函数之间可能存在调用，需要设置库函数跳转到库函数的活跃区来允许嵌套调用。

流程：设置活跃区之后，对于库函数之间的跳转情况，可以分为如下四种类型进行处理：

1. 活跃区-> 活跃区：允许，硬件不会做额外的操作
2. 非活跃区 -> 活跃区：允许，硬件会记录返回非活跃区的返回地址 (DasicsFreeZoneReturnPC)

3. 活跃区 -> 非活跃区：硬件将会检查跳转的目标地址是否等于 DasicsFreeZoneReturnPC，如果不等的话，将会引发 DASICS 跳转例外

4. 非活跃区 -> 非活跃区：不允许，硬件会直接引发 DASICS 跳转例外

5.7 对内核的 DASICS 保护

含义：将内核的某些部分（例如设备驱动，文件系统）通过 DASICS 机制保护，避免第三方驱动或内核模块对内核的攻击。

实现的功能与处理流程：

1. 内核启动时对部分代码功能（例如设备驱动，文件系统，插入的内核模块）进行分区隔离，设置为库函数区。内核单独有一套主函数代码区配置边界寄存器，在切换用户态和内核态时，硬件自动切换使用对应的配置边界寄存器。库函数代码区配置边界寄存器则是内核和用户复用，保存/恢复程序上下文的时候把这些寄存器切给内核/用户使用。主函数代码区配置寄存器暂时与用户态分开，但如果后续两个寄存器都不需要使用太多的 reserved bits 的话，可以将其合并在一起。

2. 修改例外处理流程，在进入系统调用或者中断等例外的时候硬件切到内核的边界寄存器，这一点通过内核/用户各使用一套主函数边界寄存器的方法得到了解决。

3. 内核中触发权限错误需要触发中断，现有的 scause 寄存器需要增加新的对应值。这一点通过引入新的 8 种 DASICS 例外得到了解决。

4. 如果用户态不想处理中断，允许其转交给内核处理。目前的实现是：用户态中断有默认的处理函数，该函数里面调 syscall 交给内核处理。

5.8 系统调用截获检查

含义：将库函数进行的系统调用进行截获检查，防止库函数通过系统调用进行特权提升或者关闭保护机制。

实现的功能与处理流程：

1. 硬件检测到执行 RISC-V 系统调用指令 *ECALL* 时，就触发 DASICS ECALL_FAULT)，跳转到例外处理函数

2. 例外处理函数中复制该次系统调用的参数，包括系统调用号以及对应的参数列表，并根据判断规则决定非可信区是否可以进行此次系统调用，以及进行系统调用的参数是否在可信区允许的范围内。

3. 例外处理函数进行上述检查后，对于不能进行的系统调用，直接报错退出，而对于可以进行的系统调用，则由位于可信区的例外处理函数进行代理系统调用，并在调用之后将结果传递给非可信区。

6 DASICS 软件 API 介绍

7 DASICS 使用实例