RISC-V Zicfilp拡張に対応したバイナリの生成とlpad命令の有効性の評価

髙名 典雅[1]　大山 恵弘[1]

1. 本研究の背景

RISC-V では，バッファオーバーフローや Return Oriented Programming (ROP) に起因する不正な遷移を防ぐ対策として“RISC-V Shadow Stacks and Landing Pads”と呼ばれる仕様の策定が近年進んでいる [1]．この仕様は Control Flow Integrity (CFI) に関するもので，shadow stack をサポートする Zicfiss と landing pad をサポートする Zicfilp の 2 つの拡張から構成される．

2 つのうち Zicfilp 拡張は分岐先に landing pad と呼ばれる目印となる命令を設置し，実際に分岐した先にその命令が置かれているかを確かめることで不正な遷移を検出する拡張である．特定の間接分岐命令が実行されたとき，次の命令が landing pad 命令（lpad 命令）でなければ攻撃者による不正な遷移が発生したとして例外を発生させ，それ以上の命令の実行を防ぐ．lpad 命令を導入することで分岐先が限定されるため攻撃者の使えるコード片の数を大幅に減らせる一方，単なる目印であるためメモリの使用量が増加するなどの問題も存在し，トレードオフの関係にある．

現在 Zicfilp 拡張は仕様が新しく完全に安定していないため，対応したコンパイラやエミュレータが存在しない．また，lpad 命令はラベルの設定を行えるなど他のアーキテクチャには無い独自の機能を備えているが，実行する環境が無いため ROP に利用できそうなコード断片（ガジェット）の削減効果について実環境での具体的な検証が不十分である．

2. 研究の概要と目的

本研究の目的は Zicfilp 拡張における lpad 命令を配置するシステムの実現と，それを利用した Zicfilp 拡張の有効性の評価である．

そこでまず，Zicfilp 拡張の仕様に沿って lpad 命令を配置し Zicfilp 拡張に対応した環境で動作可能なバイナリを生成できるソフトウェアを開発する．gcc などの既存のコンパイラを改変してコード生成の過程で命令を埋め込むことやバイナリを解析して直接命令を対象ファイルに追加する手法によって実現する．

次に，拡張自体の有効性を評価するためにエミュレータへ修正を加える．エミュレータに必要な修正は大きく以下の 3 点である．

- 既存の CSRs への状態を保持するフィールドの追加.
- lpad 命令を要求する/しない，例外を発生させる/させないといった状態の定義と判定の追加.
- 新しい例外番号の定義“landing pad fault (code=2)”の追加.

実行環境を再現でき次第，メモリの使用量の増加やガジェットの削減効果について測定する．また，lpad 命令の効率的な配置方法についても実装の上で考察し，報告する予定である．

3. Zicfilp 拡張に関する調査

Zicfilp 拡張の仕様書を確認し，分岐先に lpad 命令を要求する命令と lpad 命令が置かれる条件を確認した．分岐先に lpad 命令を要求する命令は，間接分岐命令（JALR, C.JR，C.JALR）でありかつアドレスのベースアドレスを指定するレジスタが x1 (ra)，x5 (t0)，x7 (t2) でない命令である [2]．lpad 命令はこの間接分岐命令の対象となるアドレスに置かれる．

既存のアーキテクチャで実装されている類似した命令と異なる点として，lpad 命令では 20 bit のラベルと呼ばれる数値をオペランドに指定できる特徴がある．x86_64 で採用されている endbr64 命令ではオペランドは存在せずただ単に目印として置かれており [3]，Arm の bti 命令の場合は分岐元の命令の種類と対応するオペランドの値を比較して一致していなければ例外を発生させる仕組みになっている [4]．一方，RISC-V では分岐前に特定のレジスタ (x7)

[1] 筑波大学
University of Tsukuba

に任意の値をセットして分岐先の lpad 命令のラベルの値と比較し，一致していなければ例外を発生させる．任意のラベルを設定可能なことにより更にコード片の数を抑制できるが，設定には lui 命令など別でもう 1 命令必要になる．

4. 予備実験

まず，RISC-V のバイナリを Rust から解析可能にする crate[*1] を開発し，命令単位での解析を可能にした．RISC-V では ret 命令などが擬似命令として実装されており，また圧縮命令と通常命令の両方で同じ内容の命令を表現可能なため，このようなオペコードレベルでの解析は不可欠である．ret 命令や，lpad 命令を期待する分岐命令を検知してダンプ時に強調するプログラムを作成し，利用できそうなガジェットの大まかな数や lpad 命令の判定が必要になりそうな頻度などを可視化した．

また命令を fetch するアドレスを意図的に 2 byte ずらして，意図しない命令がどの程度成立するかについても実験した．RISC-V は固定長命令だが，32 bit の通常命令の他に 16 bit の圧縮命令が定義されているため，通常命令の中間の位置（16 bit 目）から fetch することで圧縮命令として解釈可能な場合や続く圧縮命令と合わせて通常命令として解釈可能な場合がある．実際の例を図 1 に示す．

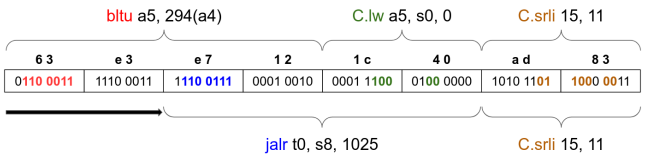

図 1 通常命令の途中と圧縮命令を組み合わせ，別の命令として解釈が可能な例

図 1 では命令を表す byte 列がリトルエンディアンで並んでおり，上が本来解釈されるべき命令列，下が fetch するアドレスを 2 byte ずらした場合に解釈される命令列を表す．また，各命令の色に対応した bit 列がオペコードにあたる．上段の命令列には出てこない JALR 命令が，読み出すアドレスを 2 byte ずらすことで現れることが分かる．

通常命令の下位 16 bit と次の通常命令の上位 16 bit を組み合わせて別の通常命令として解釈できる例を図 2 に，通常命令の下位 16 bit が圧縮命令として解釈できる例を図 3 に示す．

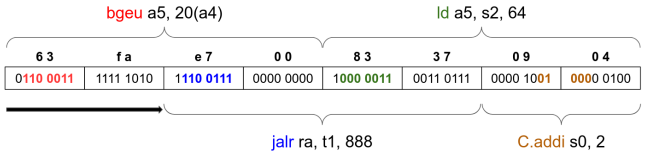

図 2 通常命令の下位 16 bit と次の通常命令の上位 16 bit を組み合わせて別の通常命令として解釈可能な例

[*1] https://github.com/Alignof/raki

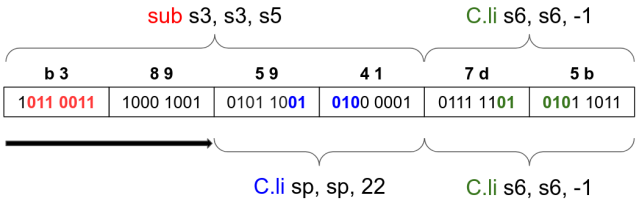

図 3 通常命令の下位 16 bit が圧縮命令として解釈できる例

図 1-3 の例で重要なのは，2 byte 分 fetch するアドレスをずらした後でも，続く命令のアドレスは変わらないことがあるということである．例えば，図 2 では上段の通常命令 2 つの $4 \times 2 = 8$ byte が，下段の 2 byte 分のずれと通常命令 (4 byte) と圧縮命令 (2 byte) の $2 + 4 + 2 = 8$ byte に解釈されており，内容は異なるもののずらす操作の前後で次の命令のアドレスは変わらない．これによりアドレスのずれによって無効な命令が含まれる確率が大幅に下がる．

実際に riscv64-unknown-elf-gcc (g2ee5e430018, v12.2.0) の環境で静的リンクされた実行ファイルに対して，通常命令だった場合に fetch する位置を 2 byte ずらして解釈するプログラムを実行したところ，約 10.82%の命令が無効な命令（全て 0 で構成される illegal instruction を除く）として解釈された[*2]．言い換えれば残りの約 89%の命令は有効な命令として解釈され続け意図しない命令列を構成する．

Zicfilp 拡張では lpad 命令が置かれるアドレスに 4 byte 境界を要求することで，アドレスをずらして不正に lpad 命令として解釈させる手法に対処している．

5. 今後の展望

今後はエミュレータ部分の実装を行い実行環境を整えた上で，lpad 命令を追加したことによる時間的・空間的なロスやガジェットがどの程度減少したかの観点から Zicfilp 拡張の有効性について定量的に評価する予定である．また，lpad 命令にラベルを指定した場合の効果についても評価する予定である．実装・評価を通して lpad 命令の効率的な配置やラベルの効果的な利用法についても定量的に評価することを目標とする．

参考文献

[1] RISC-V International. Specification Status. https://wiki.riscv.org/display/HOME/Specification+Status, 2023 年 10 月 24 日参照.
[2] RISC-V Shadow-stack and Landing-pads Task Group. RISC-V Shadow Stacks and Landing Pads (v0.3.2, 2023-10-03). 2023, p.23.
[3] Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer's Manual. 2023, Vol.2A, p.3-347.
[4] Arm Limited. Arm® Architecture Reference Manual for A-profile architecture. 2023, C6.2.41, p.1318.

[*2] 2 byte ずらさない場合でも 0.17%は無効な命令として検出された．これは実行権限のあるセグメントを機械的に対象にしたため命令でないデータも含んでいるからと考えられる．