6A-01

Linux のデータ完全性保護方式の性能比較

山田　竜也†

三菱電機株式会社　情報技術総合研究所†

1. はじめに

近年、IoT デバイスの増加によってネットワーク接続される組込み機器が増加してきている。これらの機器は工場などのインフラにも使われるようになってきているため、サイバー攻撃によって意図しない動作をすると社会的に大きな被害が出る可能性がある。これに対処するためのセキュリティ機能の一つとして、データ完全性保護がある。機器に搭載されるファームウェアや重要なデータについて使用前に完全性を検査し、改竄が検知された場合はエラーとして通知する機能である。データ完全性保護機能は、組込み機器では特に速度と記憶効率が求められるため、適切な機能の選定・開発が必要である。

2. 本論文の目的

組込み機器においても汎用 OS である Linux が採用されるようになってきており、データ完全性保護機能を実装した多数のソフトウェアが利用可能となっている [1]。これに Linux カーネルにおける標準提供の機能を加えて、「継続的な開発」「フリー（無料）で使用可能」を条件として、データ完全性保護機能を抽出した。結果、dm-verity と dm-integrity が適当であると判明したため、それぞれについて性能を測定し、各方式の差異による特性を示す。

3. 性能比較

dm-verity と dm-integrity について、スループットとその時の CPU 使用率を比較した。

3.1. 測定方法

性能測定環境を表 1、表 2に示す。

表 1　評価環境（ハードウェア）

評価機	Raspberry Pi 3 Model B
・CPU	ARM Cortex-A53 (ARMv8)
・メモリ	949,432 kB
microSD カード	Transcend 16GB (Class10)

dm-verity はリードオンリー(RO)パーティションを対象としており、パーティション全体を指定したベンチマークは実行不可である。そのため、ランダムデータで構成されるファイルをすべてのベンチマークの対象として使用した。

表 2　評価環境（ソフトウェア）

Linux カーネル	4.14.58 (32bit)
ベンチマークアプリ	iozone 3.482
対象ファイル	ランダムデータ 512MiB

測定・比較対象は表 3の 3 種類である。dm-verity の前方誤り訂正は未使用、dm-integrity のジャーナリングは有効化している。

表 3　測定・比較対象

	方式	完全性保証アルゴリズム
①	dm-verity	SHA256
②	dm-integrity	SHA256
③	dm-integrity	HMAC-SHA256(鍵長 256bit)

ベンチマークアプリ iozone による読込み向けテスト項目を表 4に示す。他の読込み向けテスト項目"Fread"と"Re-fread"はユーザライブラリの性能測定であるため、"Random Read"は RO パーティションでは実行不可であるため除外した。

表 4　iozone で用いたテスト項目

項目	説明
Reader	読込みテスト
Re-reader	再読込みテスト
Backward Read	後端からの読込みテスト
Stride Read	幅のある読込みテスト

3.2. 測定結果

3.1で示した 3 種類の測定対象における読込みスループットの比較を図 1 に示す。横軸はテスト項目、縦軸は kB/秒である。

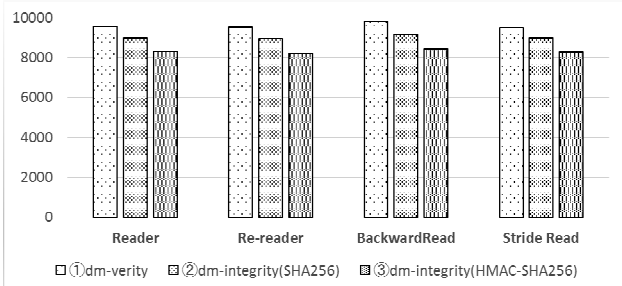

図 1　読込みスループット比較

図 2は、図 1の測定時における CPU 使用率（縦軸）である。さらに、図 1の比較対象に加えて、dm-integrity のスループットを dm-verity

A performance measurement on different style integrity assurance systems for Linux device-mapper subsystem.

† Tatsuya Yamada, Information Technology R&D Center, Mitsubishi Electric Corporation.

と同等とみなして換算した CPU 使用率も追加した。それぞれ④dm-integrity(SHA256)換算、⑤dm-integrity(HMAC-SHA256)換算である。

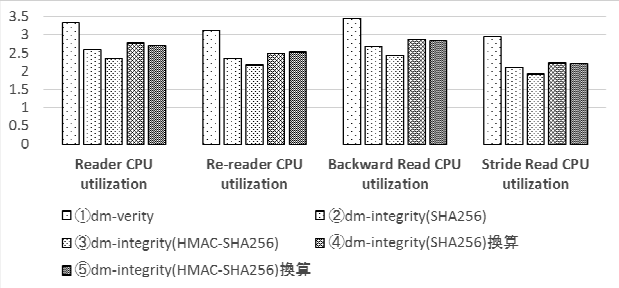

図 2　CPU 使用率比較

4. 性能差異の検討

図 1の結果より、完全性保証アルゴリズムが SHA256 で同一の場合(①対②)、dm-integrity は dm-verity に対して 94.0%のスループットとなった。dm-integrity が HMAC-SHA256 である場合(①対③)、dm-verity に対して 86.5%のスループットとなった。図 2の結果より、CPU 使用率では、それぞれ 75.4%、68.9%となった。また、dm-integrity のスループットを dm-verity と同様としたときの CPU 使用率は、それぞれ 80.2%(①対④)、79.7%(①対⑤)となった。

以降で各方式の特徴を示し、要因を分析する。

4.1. dm-verity の完全性保証方式

dm-verity のデータ構造を図 3に示す。あらかじめデータのブロック単位でハッシュを計算し、ハッシュツリーを構成しておく。ブロックの読込みごとに当該ブロックのハッシュを再計算して root hash を導出し、格納してある root hash と比較することで完全性を保証している。ハッシュツリーは初回読込み時に先読みしてメモリ上にキャッシュされる。

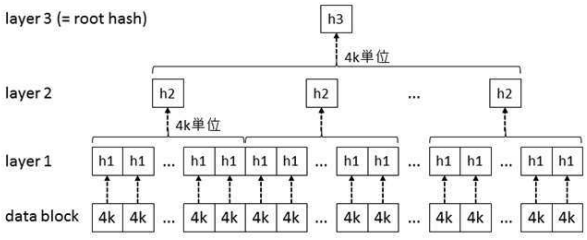

図 3　dm-verity データ構造

ブロックアクセスごとに、4KiB ブロック（デフォルト値）単位でハッシュツリーのレイヤ数回のハッシュ計算が行われる。SHA256 ではハッシュ値が 32 バイトのため、4KiB ブロックのとき、3 層のハッシュツリーでは最大 64MiB、4 層では最大 8GiB のデータブロックが扱える。したがって、今回の性能測定ではブロックアクセスごとに 4 回のハッシュ計算を行っている。

4.2. dm-integrity の完全性保証方式

dm-integrity のデータ構造を図 4に示す。データはセクタ領域に分割されて格納され、セクタ単位の完全性情報などがタグ領域に格納される。完全性情報には、チェックサム、暗号学的ハッシュ関数、メッセージ認証符号(HMAC)が使用可能である。チェックサムやハッシュ関数を用いた場合は、dm-verity の root hash に相当する root-of-trust が存在しないため、セキュアブートなどには HMAC の使用が必要である。

また、dm-verity と異なり書き込みにも対応しているため、ジャーナリング機能も具備する。

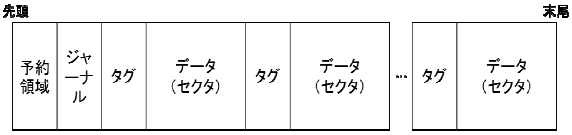

図 4　dm-integrity データ構造

セクタへの読込み単位で完全性情報が再計算され、タグに格納されている値と比較して完全性を保証する。dm-verity とは異なり、セクタの読込み単位の完全性情報の計算は高々1 回である。

4.3. 性能差異の要因

スループットにおいて、SHA256 の使用時に dm-integrity が dm-verity より性能が低い理由は、ジャーナルの正当性の計算が付随する点と、セクタに対応するタグが先読みできない点にあると考えられる。HMAC の使用時は、一般に SHA256 よりもさらに性能低下する。

CPU 使用率において、図 2で示したとおり dm-integrity は CPU 使用率が低くなっている。これは、4.1と4.2で示したとおり、今回の性能測定において dm-verity では 4 回、dm-integrity では 1 回の完全性計算が理由と考えられる。

5. おわりに

dm-integrity は CPU 使用率が低いため、低性能の CPU での使用が適している。

dm-verity で CPU 使用率を下げるには、ハッシュツリーのレイヤ数を減らすことが有効である。一方でデータブロックサイズが大きくなるため、ストレージの利用効率とのトレードオフであるといえる。また、オフライン攻撃への対策に目的を限定し、初回の読込み時のみデータ完全性を検査することによっても CPU 使用率の削減が可能となっている。

参考文献

[1] Wikipedia, “File integrity monitoring,” [オンライン]. Available: https://en.wikipedia.org/wiki/File_integrity_monitoring. [アクセス日: 17 10 2018].