Linux Kernel リリース　モデル

2 月 5 日木曜日　2018 年

初めに

この投稿は、2016 年初めに書いたホワイトペーパーを基にしています。このホワイトペーパーは、多くの企業が Linux Kernel リリース　モデルを理解して LTS の更新をもっと頻繁に採用することを奨励するために書かれたものです。Linux Recipes conference in September 2017（https://kernel-recipes.org/en/2017/）での私の発表も、この内容を基にしました。（こちらで見られます https://www.youtube.com/watch?v=RKadXpQLmPU）

最近起きた Meltdown と Spectre に関する大騒動の最中に、Linux のリリース方法やセキュリティ　パッチの管理について書かれた多くのものを

見ましたが、その内容は全く間違っていました。それで、ホワイトペーパーのほこりを払う時機が来たことを知り、数か所だけ書き換え、皆さんのためになるようにここに公開することにしました。

最初のホワイトペーパーが世に出るのを助けてくれたレビュワーに感謝します。このホワイトペーパーは、自分たちのデバイスに使っている Kernel への無作為なパッチの「つまみ食い」（必要そうなところだけの選択）を止めた方がいいと多くの企業に理解させるのに役立ちました。レビュワーの手助けなしには、この投稿は完全にめちゃくちゃな文章になっていたことでしょう。勿論、すべての問題や誤りは、私に全責任があります。もし、何か気づいたことや質問があれば、私までお知らせ下さい。

全般

この投稿では、Linux Kernel 開発モデルがどのように運用されているか、長期サポート（long term supported）Kernel（訳注：本文書内で LTS は Long Term Stable を指します。長期サポートを意味するときには Long Term Support が使われます。）とはどのようなものか、Kernel 開発者はセキュリティ不具合にどのように挑むか、また、Linux を利用する全シス

テムはパッチのつまみ食い（必要そうなところだけの選択）を止めてすべての Stable パッチを取り込むべき理由、を説明します。

Linux Kernel 開発モデル

Linux kernel は、これまでで最大のコラボレーティブ ソフトウェア開発プロジェクトです。2017 年には、530 を超える企業の 4,300 人以上の開発者がプロジェクトへ貢献しました。2017 年には 5 回のリリースが行われ、毎回 12,000 から 14,500 か所が変更されています。平均すると、1 日の 1 時間、1 時間ごとに 8.5 個の変更が Linux Kernel で受け付けられています。非科学的な調査（つまり、グレッグのメールボックス）によると、各変更は、Kernel ソース ツリーへ受領されるまでに 2、3 回の再提出を必要としています。というのも、変更提案のすべてがパスする必要がある非常に厳しいレビューとテストプロセスのためで、エンジニアリングの労力として実際にかかっているのは 1 時間に 8 変更に対応するよりもずっと大きなものになっています。

2017 年末の時点で、Linux Kernel の規模は、コード、ビルド スクリプトやドキュメントを含めて 6,100 ファイル以上、2,500 万行以上に達しています（Kernel release 4.14）。Linux Kernel は、サポートしているすべての

アーキテクチャとハードウェア ドライバに対するコードを含んでいます。このため、個別システムは、全コードベース中の一部分だけを使用します。平均的なラップトップ PC は、動作させるのに 5,000 ファイル、200 万行程度の Kernel コードを使っていますが、一方で、スマートフォンは 6,000 ファイル、320 万行程度の Kernel コードを使います。これは、使用している SoC の複雑さが大きくなっていることによるものです。

Kernel リリースモデル

2003 年 12 月の Kernel 2.6 のリリースから、Kernel コミュニティは、個別の開発ブランチと Stable ブランチの 2 つを持つモデルから、Stable ブランチだけのモデルへ切り替えました。新リリースは、2、3 か月ごとに実行されます。該当するリリースは「Stable」と宣言され、全ユーザーはそれを使うことを推奨されます。この開発モデルの変更は、Kernel 2.6 より以前は非常に長いリリース サイクル（3 年程度）であったことと、コードベース中の 2 つの異なるブランチを同時に維持することに苦闘したことが理由で行われました。

Kernel のバージョン番号付けは 2.6.x から始めました。x は、リリースごとに増加する数字を意味します。数字は、前 Kernel リリースよりも新しいこ

とを示す以外に特別な意味はありません。2011 年 7 月に Kernel 2.6.39 がリリースされた後に、リーナス トーバルズ（Linus Torvalds）は、Kernel バージョンを 3.x に変更しました。これが起きたのは、大きなバージョン番号がユーザーに混乱をもたらし始めていたこと、Stable Kernel メンテナーのグレッグ クロー ハートマン（Greg Kroah-Hartman）が大きな数字に飽き飽きしてリーナスを日本産の高級ウィスキーで買収したことに起因します。

3.x 番号付けへの変更は、メジャー リリース番号の変更以外の意味はありません。2015 年 4 月に、再び 3.19 から 4.0 へのリリース番号の変更がありました。この時に、ウィスキーが手渡されたかどうか定かではありません。現在の Kernel リリースの頻度からいくと、2018 年に 5.x への変更があるでしょう。

Stable Kernel リリース

Linux Kernel Stable リリース モデルは、2005 年に始まりました。既存の Kernel 開発モデル（2、3 か月ごとに新リリース）はほとんどのユーザーのニーズを満たしていないと判断されたのです。ユーザーは、この 2，3 か月の期間に不具合修正が実行されることを望んでいましたが、Linux

ディストリビューションは、コミュニティからのフィードバックがない中で彼らの Kernel を最新に保つ作業に疲弊していました。最新の不具合修正を取り込みながら個々の Kernel をセキュアに保つことは、混乱を伴いながら多数の個人によって行われた膨大な努力でした。

こうしたことから、Stable Kernel リリースが開始されたのです。これらのリリースは、リーナスのリリースに基づいて、さまざまな外部要因（時節、適用可能なパッチ、メンテナーの負荷、等）にも依存しますが、毎週リリースされます。

Stable リリースの番号付けは、Kernel リリースの番号から始まり、その後ろに追加番号が付きます。

たとえば、Kernel 4.9 がリーナスからリリースされると、この Kernel リリースに対応する Stable Kernel リリースは、4.9.1、4.9.2、4.9.3、 というように番号付けされていきます。Stable Kernel ツリーを参照する場合、この系列は、通常「4.9.y」と短く表現されます。各 Stable Kernel リリース ツリーは、一人の Kernel 開発者によって保守されます。その開発者が、リリースの必要があるパッチを選択し、レビュー／リリース プロセスを実行します。これらの変更がどこに加えられるかを以下で説明します。

Stable Kernel は、現在の開発サイクルが実行されている間だけ保守されます。リーナスが新しい Kernel をリリースした後は、前 Stable Kernel ツリーの保守は終了しますので、ユーザーは新しくリリースされた Kernel へ移行するべきです。

Long-Term Stable Kernel

この新しい Stable リリース プロセスを 1 年実施してみたところ、多くの Linux ユーザーはほんの数か月ではなくもっと長く Kernel を保守してほしいと願っていることが分かりました。これを背景として、長期サポート（LTS）Kernel リリースが登場しました。最初の LTS Kernel は、2006 年にリリースされたバージョン 2.6.16 でした。それ以降、新しい LTS Kernel は、毎年 1 つずつ選定されています。この Kernel は、Kernel コミュニティによって最低でも 2 年間保守されます。LTS Kernel の選定方法については、次節をご覧下さい。

本稿執筆時の LTS Kernel は、リリース 4.4.y、4.9.y、4.14.y です。これらに対する新しい Kernel は、平均すると週に 1 回程度リリースされます。これら 3 つの Kernel リリースとは別に、少数の古い Kernel は Kernel 開発者によって保守され続けていますが、あるユーザーやディストリビュー

ションの要望に応える形でもっとゆっくりとしたリリース サイクルになっています。

すべての LTS Kernel の情報（誰が責任者で、いつまで保守されるか）については kernel.org release page（https://www.kernel.org/）で見ることができます。

普通の Stable Kernel リリースが 1 日に 10 から 15 のパッチを受領するのに対して、LTS Kernel リリースは、平均して 9 から 10 のパッチを受領します。パッチ数は、開発 Kernel リリースの影響やその他外部要因によって、リリースごとに変動します。また、多くの新しい不具合修正は古い Kernel に無関係ですので、LTS Kernel が古いほど、適用可能なパッチの数は少なくなります。しかしながら、コードベースが変化していきますので、Kernel が古いほど必要な不具合修正を適用させるのはより難しくなります。適用されるパッチの数は少ないといっても、LTS Kernel 保守に要する労力は、通常の Stable Kernel の保守よりも大きなものとなります。

LTS Kernel の選定

LTS Kernel の選定とメンテナーの選定は、数年の間に、半分無作為的な方法から、より信頼性がある方法へと変わりました。

当初は、Stable メンテナーの所属する企業の製品に利用されている Kernel が、保守を楽にする目的で選ばれました。（2.6.16.y と 2.6.27.y）この保守モデルの利点に気づいた他のディストリビューション メンテナーたちは、皆で相談して、企業に気づかれずに、同じ Kernel バージョンを製品に載せるように働きかけました。(2.6.32.y) このモデルは大変成功し、開発者は企業の枠を超えて作業を分担することができましたが、その後、企業は企業間で分担することをやめさせました。その後の LTS Kernel はある特定のディストリビューションの要望に基づいて選定され、異なる開発者によって保守されることになりましたが、これは関係者に大きな作業と混乱をもたらすことになりました。（3.0.y、3.2.y、3.12.y、3.16.y、3.18.y）

特定の Linux ディストリビューションの要望にだけ応える特別な方法は、Linux を利用しているものの伝統的な Linux ディストリビューションをベースにしていない組み込みシステム デバイスにはメリットがありませんでした。こうした理由で、グレッグ クロー ハートマンは、LTS Kernel 選定手法を、企業が LTS Kernel を使った製品を計画しやすいように改めるべきであると判断しました。ルールは、「毎年 1 つの Kernel が選定され、2 年

間保守される」となりました。このルールに従い、Kernel 3.4.y、3.10.y、3.14.y が選定されました。

同じ年に異なるすべての LTS Kernel から多数のリリースが行われて、ベンダーもユーザーも混乱したことから、個々のディストリビューションの要望に応じた LTS Kernel は今後作成されないことが決まりました。このことは、年次の Linux Kernel Summit で賛同され、LTS 4.1.y の選定から開始されました。

これまでは、LTS Kernel はリリースされた後にしか宣言されませんでしたので、このことは、企業が製品へどの Kernel を利用するか事前に計画するのを難しくし、多くの推測や誤った情報が広まる原因ともなりました。以前、企業や開発者が次の LTS Kernel になるものを事前に知っていたときに、通常は厳しいはずのレビュープロセスを緩和し、テストしていない多数のコードをマージした（2.6.32.y）ことが理由で、こういう宣言の方法が行われました。 この混乱を収拾して Kernel を適切なレベルに戻すのに数か月を要しました。

Kernel コミュニティは、年次会議でこの問題を議論した上で、Kernel 4.4.y を LTS Kernel とすることに決めました。関係者すべてが大きく驚い

たことに、次のホリデーシーズン（2017 年）向け製品の投入を計画する十分な時間を企業に与えるように、次の LTS Kernel は 2016 年の最後の Kernel リリースと決められたのです。同じように、Kernel 4.9.y と 4.14.y も選定されました。

この選定プロセスは、うまく機能しているように見えます。4.9.y ツリーは、過去にリリースされた中で最大の Kernel ですが、16,000 か所の変更を施したにも関わらず、問題は多くは報告されていません。

この先の LTS Kernel は、このリリース サイクルを基に計画するべきでしょう。（つまり、その年の最後の Kernel）このプロセスによれば、SoC ベンダーも、新しいチップセットに古くて保守されなくなる LTS Kernel バージョンを採用しないように事前に開発サイクルを計画できます。

Stable Kernel パッチ ルール

Stable Kernel リリースに何を加えることができるかについてのルールは、過去 12 年間ほとんどそのまま維持されてきました。Stable Kernel リリースで受領できるパッチについてのルールは、

Documentation/process/stable_kernel_rules.rst ファイルに書いてあります。ここでは要約を書きます。Stable Kernel への変更は:

- 明らかに正しく、かつテストされていなければならない。
- 100 行より大きくてはいけない。
- 1 つのことだけ修正しなくてはならない。
- 問題として報告されたものを修正しなくてはならない。
- 新しい Device ID やハードウェアの癖への対処であってもよいが、メジャーな新規機能の追加であってはならない。
- リーナス ツリーに既にマージされていなくてはならない。

最後のルール「リーナス ツリーに既にマージされていなくてはならない」は、Kernel コミュニティが修正をし忘れないようにあります。コミュニティは、リーナス ツリーに入ってない修正が Stable Kernel に入ることを望みません。こうすることで、Kernel をアップグレードしたときにリグレッションを経験せずに済みます。これは、Stable と開発のブランチを保守している他のプロジェクトで生じうる多くの問題を予防します。

Kernel アップデート

Linux Kernel コミュニティは、ユーザーに対して、前リリースで動作している機能はアップグレードしても損なわれないことを約束しました。これは、2007 年にイギリスのケンブリッジで開催された年次 Kernel Developer Summit で約束され、今日に至るまで守られています。リグレッションが発生すると、最優先の不具合となります。これらは、すぐに解決されるか、そうでなければリグレッションを起こしている変更を Linux Kernel ツリーからすぐに外します。

この約束は、3 か月ごとに実行される「メジャー」アップデートと同様に、インクリメンタルに出される Stable Kernel アップデートでも順守されます。

Kernel コミュニティは、Linux Kernel ツリーにマージされるコードに対してのみこの約束をしています。kernel.org のリリースに記載がないデバイスの Kernel にマージされているコードは、未知のものであり、それとの相互作用は計画も検討もされていません。Linux を採用するデバイスは、大きなパッチセットを持っており、新しい Kernel へアップデートする際に大きな問題を抱える可能性があります。というのもリリース間で莫大な数の変更をする必要があるからです。特に SoC のパッチセットは、新しい Kernel へのアップデートに伴う問題を持っていることが知られています。

これは、アーキテクチャに由来する大規模で重い変更と、ときには核となる部分、Kernel コードにも変更をしているためです。

ほとんどの SoC ベンダーは、自分たちのチップが出荷される前に彼らのコードをアップストリームへマージしたいと思っています。しかし、SoC プロジェクト計画サイクルの現実と、究極的には企業のビジネス優先度が、アップストリーム作業に十分な時間を割くことを阻んでいます。組み込みデバイスへアップデートを導入することへの歴史的な困難も加わって、上記が原因となり、ほとんどすべてのデバイスはライフスパンで特定の Kernel リリースにとどまり続けています。

大きなツリー外パッチセットが理由で、ほとんどの SoC ベンダーは、自社のデバイスに LTS リリースを使うことを標準化しようとしています。この取り組みによって、伝統的に問題への対応が遅い SoC ベンダーのバックポーティングに依存せず、不具合やセキュリティに関するアップデートを直接 Linux Kernel コミュニティからデバイスが受け取ることができます。

Android プロジェクトが「最低限の Kernel バージョン要求（minimum kernel version requirement）」として LTS Kernel を標準化しているのを是非参考にして下さい。こうしたやり方を採用することで、SoC ベンダーが

ユーザーに対してセキュアなデバイスを提供するために、デバイスの Kernel を継続してアップデートすることが可能となるでしょう。

セキュリティ

Kernel リリースを行う際に、Linux Kernel コミュニティは、特定の変更を「セキュリティ修正（security fixes）」と宣言することはほとんどありません。その不具合修正がセキュリティ修正かどうかをその時点で判断することが難しいからです。多くの不具合修正はかなり時間が経過した後でしか決められないものです。セキュリティへの言及がないからといって、ユーザーがパッチを適用しないことがないように、Kernel コミュニティはリリースされたすべての不具合修正を適用するように強く推奨します。

リーナスは、2008 年に Linux Kernel ML に出したメールの中で、この行動の背景にある理由についてまとめています。

```
On Wed, 16 Jul 2008, pageexec@freemail.hu
wrote:
>
> ここ数回の Stable リリースをチェックしてみるべきです。
> セキュリティ不具合を修正しているにもかかわらず
```

> セキュリティに関するアナウンスがないことがわかるでしょう。

うーん。どの部分が「単なる通常の不具合」なのでしょう？

あなたにはっきりと言いましたが、セキュリティ不具合はそういうようにマークするべきではないです。
不具合は不具合でしかないです。

> 別の言い方をすれば、そのコミットがセキュリティ不具合を修正していると
> いうためにマークするのでしょう。

違います。

> > 私は、ただ、マークの意味がないのに、何のためにマークを付けるのか、と言っているだけです。
> > それを信じている人は本当に間違っています。
>
> 特に、何が間違っているのでしょうか。

あなたに従えば、2 つのケースが考えられます：

- 人々がマークは信用できると考えている

    人々は間違っていて、部分マーキングを誤解して、
    マークされていないものは「あまり重要でない」と考える。そうではないです。

- 人々は、それを重要と考えない。

    人々は正しく、マーキングは無意味である。

いずれのケースも、マークするのはバカげています。そうしたくないのです
というのも、「セキュリティ修正」は「単なる普通の不具合修正」とは別物であるという神話を
不朽のものにしたくないからです。

すべては修正です。すべては重要です。新しい機能として、それは
問題なのです。

> セキュリティ不具合を修正するパッチをコミットしようとするとき、
> コミットの中でそれに言及するのがなぜ悪いのですか？

それは意味がないし、間違ってもいます。というのも、そのやり方は、人々に、他の不具合で
セキュリティが修正される可能性を考えさせないからです。

そのことは明白でしょう。

リーナス

この Email はここ（https://marc.info/?l=linux-kernel&m=121616463003140）にあります。このトピックに関心のある方は、スレッド全体（https://marc.info/?t=121507404600023）を是非読んでみて下さい。

セキュリティ問題が Kernel コミュニティに報告されると、それらは可能な限り早く修正されて、開発ツリーで公開された上で、Stable リリースされます。上に述べたように、こうした変更は、Kernel の不具合修正と同じように扱われ、「セキュリティ修正」と記載されることはまずありません。この

やり方ですと、問題の報告者がそれをアナウンスする前に、影響を受ける組織が自分たちのシステムをアップデートすることができます。

リーナスは、同じ Email のスレッドでこの開発手法を説明しています。

```
On Wed, 16 Jul 2008, pageexec@freemail.hu
wrote:
>
> 我々はこれをやり通します。あなたが言ったようにセキュリ
ティ不具合は
> 普通の不具合のようには扱われません。というのも、コミッ
トから関連する情報が
> 取り除かれているからです。

実に根本的なところで同意できません。同意できないのは、
「関連する」という単語のところです。

その不具合を再現する方法を明白に説明することに、それが役
立ち、関連するとは
思えません。不具合を追いかけているときには、それは役立
ち、関連もするでしょう。
しかし、一度修正してしまえば、それは関連しなくなります。
```

セキュリティ問題を明白に指し示すことこそが本当に重要だと
考えているから、
それがいつも「関連する」と考えるわけです。私はほとんど
逆の見方をしています。それを指し示すことは反生産的なもの
だと
考えています。

例えば、私の好きな仕事のやり方は、ある人が私と Kernel
List 宛に修正パッチを送ってくれて、
その直後のメールで問題の悪用例をセキュリティ メーリング
リストに（プライベートに）
送ってくれるというものです。
（何故プライベートで送るかというと、世界中の人がその問題
をテストしようとするのを
好まないからです。）それが、
物事が動くべき方法というものでしょう。

> 悪用方法をコミットメッセージに書くべきでしょうか？とんでもないです。そんな理由から
> プライベートであるわけです。たとえ、それが重要な情報であっても、です。それは、開発者が
> 何故パッチが必要なのかを説明するためには重要な情報です。しかし、
> 一度説明してしまえば、それは不必要に広めるべきではありません。
>
> リーナス

セキュリティ不具合を可能な限り早期に修正するための、Kernel コミュニティへの報告方法は、Kernel ファイル Documentation/admin-guide/security-bugs.rst に詳細な説明があります。

セキュリティ不具合は Kernel チームから一般にアナウンスされません。

ですから、Linux Kernel 関連問題の CVE 番号は、もしあったとしても、通常、修正が Stable と開発ブランチにマージされてから数週間後、数か月後、ときには数年後にしかリリースされません。

システムをセキュアに保つ

Linux を利用するデバイスを市場投入するのであれば、製造者に対して、以下の点を強く推奨します。すなわち、すべての LTS Kernel アップデートに対応した上で、アップデートが動作することを適切なテストで確認してそれをユーザーに示すことです。上にも書きましたように、LTS リリースからパッチを選択して取り込むのは賢い方法ではありません。それは以下の理由です:

- リリースは、Kernel 開発者によってコード全体としてレビューされていて、個別の一部分だけがレビューされているわけではありません。
- どのパッチが「セキュリティ」問題を解決し、どれが解決しないかを決めるのは、たとえ不可能でないにしても難しいです。ほとんどすべての LTS リリースは、少なくとも一つの既知のセキュリティ修正と、多くの知られていないセキュリティ修正を含んでいます。
- テストで問題が見つかれば、Kernel 開発者コミュニティはすぐにその問題を解決するように動きます。アップデートを数か月、数年待つと、Kernel 開発者コミュニティは、アップデートが長い期間されないものが何なのかさえ覚えていられないでしょう。
- あなたがビルドしない／使わない Kernel 部分への変更は、確かに、あなたのシステムに問題を起こさないかもしれません。でも、あなたが

> 必要な変更だけをフィルターすることこそが、Kernel ツリーを、将来のアップストリーム リリースと正しくマージできなくしてしまうでしょう。

注記：本著者は、アップストリーム LTS リリースから無作為にパッチをつまみ食い（必要そうなところだけの選択）している多くの SoC Kernel ツリーを監査しました。すべてのケースで、深刻なセキュリティ修正が見過ごされ、適用されていませんでした。

これを証明するために、私は、上で紹介した Kernel Recipes のトークで、市場で売られている Android スマートフォンの最新のフラッグシップ モデルすべてをクラッシュさせるのはいかにささいなことでできるかを、小さなユーザースペース プログラムを使ってデモしました。 この問題の修正は、デバイスが使っている LTS Kernel には 6 か月前にリリースされていましたが、一つのデバイスとしてこの問題を解決するために、Kernel のアップグレードや修正をしていませんでした。本稿の執筆時点（5 か月後）、たった 2 つのデバイスだけが Kernel を修正し、この特定の不具合に関してですが、脆弱性を取り除いています。

グレッグ クロー ハートマン投稿 2 月 5 日木曜日 2018 年 kernel, linux, stable