***注：***

***この翻訳文書のライセンスは、原ブログと同じCC BY-NC-SA ３．０です。***

***原ブログ：***[*http://www.kroah.com/log/blog/2018/02/05/linux-kernel-release-model/*](http://www.kroah.com/log/blog/2018/02/05/linux-kernel-release-model/)

***CC BY-NC-SA 3.0***

***https://creativecommons.org/licenses/by-nc-sa/3.0/***

***Original English: Copyright © 2018 Greg Kroah-Hartman***

***Japanese translation: Hiroyuki.Fukuchi@sony.com***

**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/）](https://kernel-recipes.org/en/2017/)での私の発表も、この内容を基にしました。（[ここで見られますhttps://www.youtube.com/watch?v=RKadXpQLmPU](https://www.youtube.com/watch?v=RKadXpQLmPU)）

[最近起きたMeltdownとSpectreに関する大騒ぎ](http://www.kroah.com/log/blog/2018/01/06/meltdown-status/)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、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たのです。

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である理由、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か所の変更が入っています。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。と言うのも、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、 と言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/）](https://www.kernel.org/category/releases.html)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、こう言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](https://www.kernel.org/doc/html/latest/process/stable-kernel-rules.html)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。と言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言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

> 言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

と言うのも、「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。と言うのも、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）](http://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）](http://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](http://marc.info/?l=linux-kernel&m=121616807207387)。

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。と言うのも、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」と言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）

送ってくれると言うものです。

（何故プライベートで送るかと言うと、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言うものでしょう。

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。](https://www.kernel.org/doc/html/latest/admin-guide/security-bugs.html)

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](http://kroah.com/log/blog/categories/kernel/), [linux](http://kroah.com/log/blog/categories/linux/), [stable](http://kroah.com/log/blog/categories/stable/)