連 載

情報倫理研究の最前線（3）
負担なきところにクオリティの恵みは訪れる

アンドリュー A. アダムス （Andrew A. Adams）
明治大学大学院経営学研究科・ビジネス情報倫理研究所
（翻訳・加筆：村田　潔）

1. 露呈するセキュリティの脆弱性

最近，インターネットで一般的に利用されている「セキュアな」通信プロトコルに，セキュリティ上の重大な脆弱性があることが明らかにされてきている．このことはフリーソフトウェアあるいはFLOSS（Free Libre Open Source Software）のクオリティに対する信頼性を大きく揺るがすこととなっている．こうした状況を踏まえ，本稿では，ソフトウェアの開発・生産における「神話と現実」について考察する．本稿の議論は，とりわけFLOSSに焦点を当てて進められ，これを通じてあらゆる種類のソフトウェアの生産環境にとって有用な次の教訓を導き出すことを目的とする．すなわち，「開発要員への過度の負担がソフトウェアの貧弱なクオリティを生み出す」ことを指摘したい．

2013年6月に，米国NSA（National Security Agency：国家安全保障局）の局員であったエドワード・スノーデンは，英紙ガーディアンなど複数のメディアの協力を得て，NSAと英国GCHQ（Government Communications Headquarters：政府通信本部）が，ありとあらゆる種類の通信を監視する能力を有しており，実際にその能力を広範囲にわたって行使しているという事実を白日の下にさらした．このことは，コンピュータ通信ソフトウェアのセキュリティをいかに確保するのかについて，活発な議論を巻き起こすこととなり，とりわけ，通信トラフィックにおけるエンド・ツー・エンドでの暗号化のあり方が注目を集めることとなった．

こうした中，本年になって，世界中で広く使われている3つのシステムに深刻なセキュリティ上の欠陥があることが立て続けに報じられてきている．まず2月に，iOSとMacOS Xのプログラムコードにいわゆる「goto fail」バグが含まれていることが明らかにされた．これは同社が提供する2つのオペレーティングシステムにおけるクライアントとサーバ間のSSL（Secure Sockets Layer）通信のためのプログラムに「goto fail」というコードが2行続けて書かれているために，暗号化された通信チャネルを形成する際に行われる電子証明書の検証プロセスにおいて，不適切な電子証明書であっても，そのまま検証プロセスをすり抜けてしまうという事態が発生するというものであった［1］．

3月になると，同様の事態を引き起こす類似のバグがFLOSSプロジェクトとして開発されたSSL/TSLライブラリのGnuTLS（GNU Transport Layer Security）にも発見され［2］，4月にはさらに深刻な致命的ともいえる欠陥が，やはりFLOSSプロジェクトとして開発されているOpenSSLに存在していることが公表された．これはHeartbleedバグと呼ばれるもので，悪意を持ったユーザがこれを利用すれば，クライアント側とサーバ側双方のメモリの大部分にアクセスすることができ，暗号化に用いられている公開鍵と秘密鍵のペアや暗号化されたセッションにアクセスするためのパスワードなどを盗み取ることができる．OpenSSLはウェブサーバの2/3で利用されているとも言われており，しかもこのバグは2年間もその存在がオフィシャルには認識されず，それゆえ何らの対応もとられていなかったため，広範囲にわたる被害が気づかれないままに発生している可能性がある．また，米国政府は否定しているものの［3］，NSAがこのバグを利用して情報収集をしていたという報道もなされている［4］．

それでは，プロプライエタリソフトウェアとオープンソースソフトウェアのどちらが本質的にはよりセキュアなのであろうか．Anderson［5］とFord［6］

の研究が示しているように，また上記のようにプロプライエタリソフトウェアとFLOSSの双方に脆弱性が相次いで見つかっているという事実が物語るように，この問いに対する一般的な結論を導くことは不可能である．

Heartbleedバグがそもそもなぜ存在していたのかに関する詳細が浮かび上がってくるにつれて，OpenSSLの開発に関わったプログラマ，プログラムコードの点検担当者，プロジェクトマネージャのいずれもが，この広く流通し，多くのオンラインシステムの通信セキュリティ基盤となっているセキュリティプロトコルパッケージに根本的な欠陥が存在する状況を作り出してしまったことについて落ち度があり，責任があることが明らかになってきた．しかし，そうした落ち度を指摘し，それに関わった者を非難することよりもむしろ，こうしたソフトウェアがどのようにして開発され，そのプロセスにおいてなぜこのようなバグが忍び込み，長期にわたって見つからないままでいたのかを，信頼できるシステム設計のための教訓を得るために，検証していくことがより重要である．

2. リーナスの法則の帰結としてのHeatbleedバグ

『伽藍とバザール』の中でレイモンドは，GNUオペレーティングシステムのLinuxカーネルの開発者であるリーナス・トーバルズにちなんで，次のような内容をリーナスの法則と呼んでいる．

ベータテスタと共同開発者の基盤さえ十分大きければ，ほとんどすべての問題はすぐに見つけだされて，その直し方もだれかにはすぐわかるはず．あるいはもっとくだけた表現だと，「目玉の数さえ十分あれば，どんなバグも深刻ではない」[7]（訳は山形による）．

ソフトウェアのソースコードの可視性が，より良いソフトウェアクオリティを生み出すというこの主張は，FLOSSをプロプライエタリソフトウェアよりも優先すべきである，あるいはFLOSSのほうが好ましいということの根拠として取り上げられることが多い．しかしながら，この法則が有効性を維持するためには，明示的には書かれていない次の前提が成り立たつ必要がある．すなわち，「十分な数の目玉とは，十分な数の注視し続ける目玉でなければならない」のである．

Hill［8］が指摘しているように，大多数のFLOSSプロジェクトにおいては，ただ一人しかソフトウェア開発者，プロジェクトマネージャ，プログラムコードの点検担当者が存在していない，あるいは存在してこなかったのであり，「十分な数の注視し続ける目玉」を確保できているわけではない．実際，OpenSSLの開発プロジェクトが，非常に多くのシステムの信頼性の基礎をなす，共通のコアセキュリティインフラストラクチャを作り上げることを目指すものであることを考えれば，このプロジェクトに携わる要員の少なさ［9］は異常なほどであり，また多くのシステム開発プロジェクトとユーザが，OpenSSLのクオリティについて何らのチェックも行うことなく，このセキュリティプロトコルを全面的に導入し，それへの依存状態を作り上げてきたことは，まったくもって驚くべきことなのである．Hearbleedバグの存在は，リーナスの法則の当然の帰結なのかもしれない．

3.「人月の神話」が教えてくれること

1970年代にブルックスはその高名な著書『人月の神話』において，建物や製品のような物理的なモノを作るプロジェクトとは異なり，ソフトウェアの開発プロジェクトにおいては，開発要員の追加がソフトウェア開発促進の手助けにならないばかりか，多くの場合，余計なコミュニケーションのオーバーヘッドをもたらすことによって，開発スケジュールにネガティブな影響を与えると指摘した［10］．ここでいうコミュニケーションのオーバーヘッドとは，たとえば，既存のスタッフが新入りに対して，取り組んでいるソフトウェア開発プロジェクトの状況や事情をよく説明するために使われる時間や，さらにすべてのプロジェクトチームメンバーが当該ソフトウェアの開発に関するあらゆる変更点を確実に認識するようにするために費やされる時間を指す．

しかし，Michlmayr and Hill［11］が指摘するところによれば，多くのFLOSSプロジェクトおいては，システム開発が誰の手助けも受けていない一人の個人に依存する形で進められており，このことは論理的にはブルックスのジレンマを回避できる典型

的な状況が生みだされていると考えることができる一方で，実際にはそこにソフトウェアプロジェクト全体の単一障害点（single point of failure）が形成され，そこでの失敗がソフトウェア全体の致命的な欠陥を招くという状況がもたらされている．個人と組織の経済活動や社会活動が情報通信技術に全面的に依存している今日の社会において，プロフェッショナルとしての責任を全うすることが求められるソフトウェアエンジニアにとって，こうしたソフトウェア開発環境は決して望ましいものではない．

ソフトウェア開発プロジェクトがどのような意図あるいは野望を持つものなのか，どのようなユーザのために，何を達成しようとしているのか，そしてそのためにはどのようなプロジェクト管理が必要とされるのか，ということを考えて質量ともに適切な要員がプロジェクトに投入されなければならないことは理の当然である．こうした要員はブルックスの指摘に十分な注意を払いつつ的確に割り当てられなければならない．しかしその一方で，優れたソフトウェアエンジニアリングアプローチを採用し，きちんとしたプロジェクト管理をしていくことによって，プロジェクトの遂行に大いに寄与する「要員の追加」を認識することもまた可能なのである．たとえば，既存のプログラムコードとの間のインターフェースの変更を必要としないソフトウェアへの新規機能の追加は，既存のソフトウェアシステムのクオリティ向上に大いに資するものとなる．このとき，追加されるべき機能と現実のプログラムコードとを突き合わせて行われるクオリティチェックや，コンパイルされたプログラムのランダムテストと系統的テストの実施は，ブルックスの指摘とは関係なしに，追加された要員によって実施可能である．もちろんこうしたチェックやテストによって問題が発見された場合，ソフトウェアシステムの開発は遅延することになる．しかし，プログラムコードのクオリティを高く維持することは，長期的に見れば，システムの保守管理やアップデートを容易にするのである．

4. FLOSSのクオリティ管理のためにすべきこと

インターネットやウェブのユーザのほとんどは，それが商用ユーザであっても個人ユーザであっても，ネット環境のさまざまなインフラストラクチャを構築するために行われてきたプロジェクトの有難みを理解せず，正当な評価も与えていない．ユーザは初期のインターネット環境を整備するために政府機関や大学，ボランティアたちによって行われた投資に，いわばタダ乗りしてきているのである．しかし少なくとも，組織のシステムの信頼性が外部者の仕事に依存しているのであれば，そして実際インターネット関連のシステムのほとんどがそうであるのだから，そうした外部者の仕事のクオリティに注意が払われて当然なのである．

Adams［12］において論じたように，企業は自社が使っているあらゆるソフトウェアについて，それがFLOSSライセンスの下にあるものか，あるいは別の形態の契約に基づいて使用しているのかにかかわらず，そのソースコードの可視性が確保されることと，ソースコードを自由に修正する権利を要求すべきである．そして同時に，インターネットやウェブの共通のインフラストラクチャシステムとして広く使われているFLOSSに関しては，それを利用しているあらゆる企業が，そのクオリティの確保のために積極的に貢献すべきなのである．具体的な貢献策として最も有効なものは，FLOSS開発にとって有益・有能な人材が自社にいるのであれば，そうした人材をFLOSS開発プロジェクトに派遣することや，あるいはFLOSS開発プロジェクトが適切な人材を雇用できるよう，資金提供することである．FLOSSプロジェクトの現状を見るまでもなく，要員に過度の負担がかかるような生産システムからは，高いクオリティのアウトプットが生まれるはずもないからである．

また，人員に対する支援を行う一方で，FLOSSのユーザである企業は，FLOSS開発プロジェクトのクオリティ管理プロセスを監視すべきであり，FLOSSプロジェクト側が企業に対してこのプロセスを可視化する手段を提供するよう要求すべきである．クオリティ管理プロセスの評価を的確に行うことのできる専門知識を有する人材が企業内にいない場合や，複数のユーザ企業がそれぞれに監視をするという無駄を省くためには，独立の第三者組織にクオリティ管理プロセスのチェックを外注してもよいであろう．

実際のところ，Google，Amazon，Microsoftをは

じめとする12社がそれぞれ年間10万ドルを，今後3年間にわたり非営利の技術コンソーシアムであるリナックスファウンデーションに貸付し，重要なインフラストラクチャFLOSSの開発プロジェクトに潤沢な資金を提供し始めたところである［13］．OpenSSLプロジェクトがこの資金提供の第一候補となっており，他のプロジェクトにも資金が提供される予定である．今後は，リナックスファウンデーションがさまざまなプロジェクトのニーズと有効性を評価して資金の配分を定めることとなっている．

ソフトウェアの高いクオリティを確保することは，「プロフェッショナル」としてのソフトウェアエンジニアの責任であると言われることが多い．しかし，このことは決してエンジニアの犠牲の上に成り立つべきものではなく，プロフェッショナルにふさわしい労働環境の下で達成されるべきことなのである．

参考文献

［1］Goodin, D. (2014). Extremely Critical Crypto Flaw in iOS May Also Affect Fully Patched Macs. *Ars Technica.* Available online at http://arstechnica.com/security/2014/02/extremely-critical-crypto-flaw-in-ios-may-also-affect-fully-patched-macs/.

［2］Goodin, D. (2014). Critical Crypto Bug Leaves Linux, Hundreds of Apps Open to Eavesdropping. *Ars Technica.* Available online at http://arstechnica.com/security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping/.

［3］Office of the Director of National Intelligence (2014). Statement on Bloomberg News Story That NSA Knew about the “Heartbleed Bug” Flaw and Regularly Used It to Gather Critical Intelligence. Available online at http://icontherecord.tumblr.com/post/82416436703/statement-on-bloomberg-news-story-that-nsa-knew.

［4］Riley, M. (2014). NSA Said to Exploit Heartbleed Bug for Intelligence for Years, *Bloomberg.* Available online at http://www.bloomberg.com/news/2014-04-11/nsa-said-to-have-used-heartbleed-bug-exposing-consumers.html

［5］Anderson, R. (2002). Security in open versus closed systems: the dance of Boltzmann, Coase and Moore. Presented at Open Source Software Economics. Available online at http://www.cl.cam.ac.uk/~rja14/Papers/toulouse.pdf.

［6］Ford, R. (2007). Open vs. Closed: Which Source is More Secure? *ACM Queue*, 5(1), pp. 32–38.

［7］Raymond, E. S. (2001). *The Cathedral and the Bazaar.* Available online at www.catb.org/~esr/writings/cathedral-bazaar（山形浩生訳（2010）『伽藍とバザール』USP研究所）.

［8］Hill, B. M. (2011). When Free Software Isn’t Better. Presentation at OKCon, Berlin. Available online at http://2011.okcon.org/2011/programme/when-free-software-isnt-better.

［9］Perlroth, N. (2014). Heartbleed Highlights a Contradiction in the Web. *New York Times.* Available online at http: //www.nytimes.com/2014/04/19/technology/heartbleed-highlights-a-contradiction-in-the-web.html?_r=0.

［10］Brooks Jr, F. P. (1995). *The Mythical Man-Month, Anniversary Edition: Essays on Software Engineering.* Upper Saddle River, NJ: Pearson Education.

［11］Michlmayr, M. and B. M. Hill (2003). Quality and the Reliance on Individuals in Free Software Projects. *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pp. 105–109.

［12］Adams, A. A. (2010). The Open vs Closed Debate. 日本情報経営学会誌，30(3)，pp. 30–47.

［13］Rayman, N. (2014). Google, Amazon, Microsoft Plan $3.6M for Open Source Projects After Heartbleed Bug. *TIME.* Available online at http://time.com/75440/google-amazon-microsoft-openssl-heartbleed/.

略歴

アンドリュー A. アダムス（Andrew A. Adams）

1969年生．1991年Leeds大学卒業．1997年St Andrews大学にてPhD取得．同年，同大学リサーチフェロー．2000年Reading大学講師．2010年より明治大学大学院特任教授．ビジネス情報倫理研究所副所長として情報倫理，情報社会に関する研究に従事．British Computer Society，ACM，IEEE，British Association for the Advancement of Science，Society for Computers and Law各会員．