Manuscript Information

Title

オープンソース開発におけるネブラスカ問題

The Nebraska Problem in Open Source Software Development

Authors

Name: 八田真行　Masayuki Hatta

Affiliation: 駿河台大学経済経営学部　Faculty of Economics and Management, Surugadai University

ORCID ID: https://orcid.org/0000-0002-6700-7341

Corresponding author

Name: 八田真行　Masayuki Hatta

Full postal address: 698 Azu, Hanno-Shi, Saitama, 357-8555.

Email address: hatta.masayuki@surugadai.ac.jp

Keywords

オープンソース、ソフトウェア開発、Unix 哲学、Heartbleed、SBOM、リーナスの法則

Open source, software development, Unix philosophy, Heartbleed, SBOM, Linus's Law

Authorship Contribution Statement

MH is the sole author.

Competing Interests

The author declares there are no competing interests.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP18K11993.

CC BY

オープンソース開発におけるネブラスカ問題

The Nebraska Problem in Open Source Software Development

Abstract: Unix 哲学に基づいて構築されたオープンソースの世界では、日の当たらない場所にある縁の下の力持ち的なプログラムが、知名度がない一人かごく少数の人間によって、主に個人的な理由で細々と維持されていることがある。しかし、ひとたび、底に近いこの細いプログラムが折れてしまうと、現代社会のインフラ全体がバランスを失って崩壊してしまうかもしれない。このことを本稿では「ネブラスカ問題」と呼ぶ。実際に発生した深刻な Heartbleed バグの事例から、これまではリーナスの法則で当たり前だと思われていた「目玉の数」を意図的に確保することが必要であり、かつ SBOM のような補完策でリスクを事前に検討する必要があることがわかった。

Keywords: open source, software development, Unix philosophy, Heartbleed, SBOM, Linus's Law

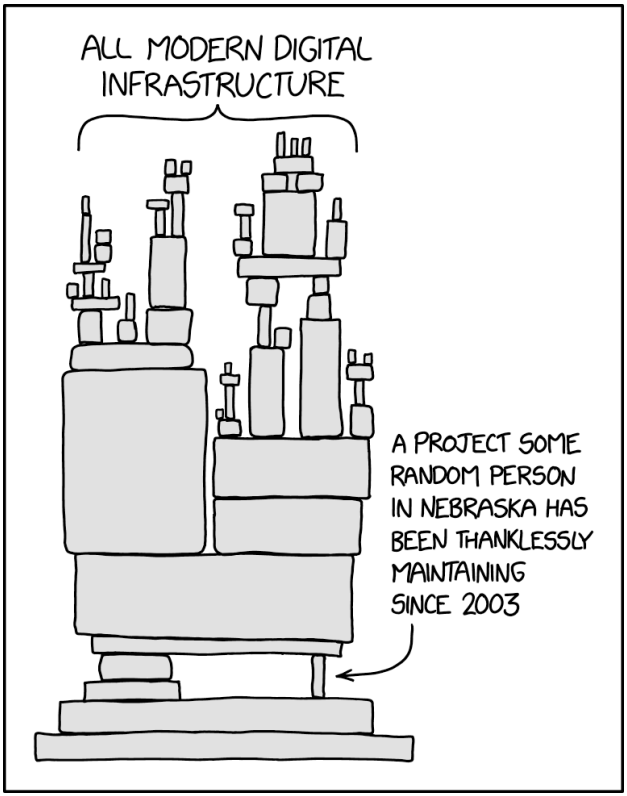
ネブラスカ問題とは何か What's The Nebraska Problem?

「ネブラスカ問題」(Nebraska Problem)とは筆者の造語である。ハッカーに人気のある Randall Munroe 作のウェブ漫画、xkcd の第 2347 回[1]か

[1] この漫画の公開日時は明記されていないが、掲示板サイト Reddit の書き込み(https://www.reddit.com/r/xkcd/comments/ibnqr3/xkcd_2347_dependency/)からは 2020 年の作ではないかと思われる。

ら着想を得た。

Figure 1. Dependency.



Sources: https://xkcd.com/2347/ (CC BY-NC 2.5)

Munroe 作の “dependency” と題された漫画は、Figure 1 に示されているようなもので、長方形のブロックは、システムを構成するプログラムの開発プロジェクトを表している。下位のブロックの上に上位のブロックが載っていることで、上位の構成プログラムが下位の構成プログラムに依存していることを表現している。そして、下から 3 段目にある細いブロックが、このシステム全体 “all modern digital infrastructure” を支えている。その細いブロックが “a project some random person in Nebraska has

been thanklessly maintaining since 2003” [2]というわけだ。つまり、日の当たらない場所にあるこのプログラムは、知名度がない一人かごく少数の人間が、主に個人的な理由で細々と維持しているという状況にあるのだが、ひとたび、底に近いこの細いプログラムが折れてしまうと、“all modern digital infrastructure” 全体がバランスを失って崩壊してしまう可能性があることを示している。

キャプションによれば、この漫画は ImageMagick で起こった事件を念頭に描かれている。ImageMagick とは、1990 年にリリースされた画像変換ユーティリティで、様々なグラフィックス・ファイル形式間の変換や、その他様々な変換を実行するために使用されている。他にもこれらのタスクを実行するための多数のライブラリや API は存在するが、ImageMagick は昔からあるため、多くのプログラム、特にウェブサービスが、その API を使用するか、必要な変換を実行するために ImageMagick を利用してきた。したがって、これらは皆 ImageMagick に依存しており、ImageMagick がなくなると壊れてしまうような、昔からある縁の下の力持ち的なユーティリティなのである。にもかかわらず、ImageMagick は脆弱性が多いことが知られていて(CVE Details, 2022)、2016 年には、ImageTragick と愛称がつくほどの大きな脆弱性(CVE-2016-3714)が発見された[3](ImageTragick, 2016)。しかも、ImageMagick そのものが多くのライブラリに依存しており、それらに脆弱性がある可能性も問題視されてきた[4]。Figure 1 は、こうした広く使われていてもバグが多いオープンソース・プロジェクトの存在を描いていたのである。

[2] なぜネブラスカなのかは定かではないが、米国のポピュラー文化においてはネブラスカは「何もない田舎」の代名詞として扱われることが多く、不遇なイメージがある。例えば人気テレビドラマ Better Call Saul の主人公が警察やギャングに追われて最後に逃げ延びたのはネブラスカ州オマハだった。

[3] https://imagetragick.com/

[4] 例えば https://news.ycombinator.com/item?id=24193871

Unix 哲学 Unix philosophy

“All modern digital infrastructure” がこの漫画に描かれるような構造をしているのは、コンピューティングの歴史が大きくかかわっている。コンピューティングの世界において、再利用可能なライブラリやコードは技術開発の大きな原動力となった。なぜなら、よく使われる基本的な機能を他のライブラリにアウトソースすることで、開発者が「車輪の再発明」に費やす時間を減らすことが可能となったからである。結果として、多くの小さなパッケージ、その多くは単一機能のパッケージが、より大規模なプログラムに利用され、依存関係が生じることとなった。この特徴は、特にオープンソースにおいて顕著であり、オープンソースを代表する Unix や Linux においては、プログラム 1 つが 1 つの小さなタスクのために使われるのが普通であり、シェルスクリプトによって小さなプログラムを結びつけることが「Unix の哲学」の本質とされてきた(Salus, 1994; Raymond, 2004)。

さらに近年では、JavaScript のプラットフォームである Node.js と Python が広く使われるようになったが、これらは世界中の開発者が集まって、お互いに作った小さな便利なライブラリを利用し合い、より強力で新しいライブラリを構成するよう設計されている。しかし結果として、依存関係の連鎖のどこかで突然新しい予期せぬバグが発生しやすいというリスクも抱えることになった。理論的には、このようなシステムは、より少ない行数のコードを書いて維持する必要があり、そのことは開発者にとっては良いことのように聞こえるが、高度に最適化されたシステムは、急速な変化の影響を非常に受けやすいものでもある。例えば、2016 年に起きた left-pad 事件では、不満のある開発者が、たった 11 行のコードを混入させただけで、全員のビルドを破壊することができ、left-pad に依存していた多くのウェブサービスがビルドできなくなったのである(Avram, 2016; Collins, 2016)。つまり、オープンソースで、ソースコードが公開されているが故に、第三者が手を入れてバグが入ったことに気づかない(再利用性が高く改変しやすい)ことも起こりうるのである[5]。

[5] いわゆる open source software supply chain problem の一部として考えることも出来る。

Heartbleed 問題 Heartbleed problem

Nebraska problem の深刻な事例としては、Heartbleed problem が挙げられる。Heartbleed とは、インターネットにおける暗号通信を実現するライブラリである OpenSSL で発見されたバグの通称である。OpenSSL は 1998 年から開発が続けられているフリーのオープンソース・ソフトウェアで、Apache License の下で公開されている。Heartbleed は 2014 年 4 月に発覚したが、Heartbleed バグにより、安全な通信を保証するオープンソースのライブラリ OpenSSL に脆弱性が生じ、インターネットのかなりの部分が攻撃されやすいことが明らかになった。実は、電子商取引を含むインターネットのセキュアな通信は OpenSSL に依存していたので、大きな問題となった。

Heartbleed バグの発見により、サービス・プロバイダーは OpenSSL のパッチ適用に奔走し、顧客は漏洩の可能性があるパスワードの変更に追われることになった。影響は非常に広範囲に及び、Heartbleed はインターネット上で発生した最悪のセキュリティ・バグとして広く認識されている。セキュリティ専門家の Bruce Schneier は「『壊滅的』という言葉がぴったりだ。1 から 10 までのスケールで言えば、これは 11 だ」(“Catastrophic” is the right word. On the scale of 1 to 10, this is an 11.)と述べた(Schneier, 2014)。

実は、OpenBSD と OpenSSH の創始者である Theo de Raadt によれば、メモリ割り当てライブラリである malloc は、Heartbleed で見られたような悪用を防ぐために、かなり前にパッチが適用されていた。しかし、それと同時に OpenSSL は「malloc & free の周りに wrapper を追加し、ライブラリが独自にメモリをキャッシュし、保護する malloc に解放しないようにした」([Because of an inability to test,] a bug shows up which leaks the content of memory mishandled by malloc() and free(). If the memoory[sic] had been properly returned via free, it would likely have been handed to munmap, and triggered a daemon crash instead of leaking your keys.) (de Raadt, 2014)という。すなわち、malloc 単体では対策されている問題が、OpenSSL では、一部のシステムでパフォーマンスを向上させるという名目のもとで残っていたのである。つまり、潜在的な

バグは検出され、パッチが適用されたものの、セキュリティよりも効率を優先した技術的判断により、回避されたのであった。おそらく、この wrapper は些細なものであり、新しいものを追加するものではないと思われていたため、綿密に調査されなかったのだろう。wrapper は、誰も修正する可能性のないコードの一部として確立されていた。このような指摘から、de Raadt (2014)は、「OpenSSL は責任あるチームによって開発されていない」(OpenSSL is not developed by a responsible team.)と痛烈に批判している。

実際、Heartbleed に関しては、バグそのものもさることながら、それを担っていたチーム体制も関心を集めた。発見直後の CNN 報道(Pagliery, 2014)から、OpenSSL の重要性、特に経済的な重要性とはあまりにも釣り合わない開発支援体制の貧弱さが明らかになったからである。2014 年の発覚当時のニュースでは、「インターネットはスティーブという名の 2 人の男が守っている」(The Internet Is Being Protected By Two Guys Named Steve)という見出しが躍った。実際、このスティーブたちは、ほとんど無名のボランティアだったが、過労で資金不足に陥りながらも、その努力によって世界中の主要なウェブサイトのセキュリティが支えられていたのである。

歴史的に、オープンソース開発モデルは、ホビイストの無償かつ継続的な献身に大きく依存している。Linux のようないくつかのプロジェクトは組織を作るのに足る注目を集めることができるかもしれないが、多くの小さなプロジェクトは、より大きなプロジェクトに再利用され、一人かごく少数の人によってのみメンテナンスされる可能性がある。ライブラリのメンテナンスには、ライブラリそのものだけでなく、そのライブラリの使用例や周辺の広範な知識が必要だが、これは通常、何年もの経験があるメンテナのみが可能で、したがって簡単に交代することはできない。こうしたプロジェクトが停滞した場合、多く起こるのは、誰かが引き継ぐというよりは、そのまま放置され、いわゆる bit-rot が発生するということである(The jargon file 4.4.7, 2022)。こうしたプロジェクトが依存関係の連鎖の川上にある場合は、川下のユーザにとって危機が訪れる可能性が生じる。

Heartbleed に関しては、OpenSSL が、2014 年の時点で約 50 万行と比較的大規模なソフトウェアであったにもかかわらず、資金不足で、開発に集中できなかったり、開発の担い手を集められなかったことも災いした。

OpenSSL 開発者の一人 Steve Marquess によると、OpenSSL の開発を主導していた Stephen Henson は、全くのボランティアであり、かつほぼフルタイムで活動していた。2009 年に、資金を集めて開発を支援するために非営利団体として Marquess が OpenSSL Software Foundation (OSF)を設立したが、状況はあまり変わらなかった。OSF が得る年間寄付は、Heartbleed 発見前は通常約 2000 ドル程度であった。Marquess は、これに加えて、商用ソフトウェアにおける OpenSSL のサポート契約販売(米国防省との年 2 万ドル)やその他コンサルティング請負(時給 250 ドル)など行っているが、Henson の年収は、Marquess が OpenSSL とは無関係のコンサルタントとして得ていた収入の 5 分の 1 にも及ばなかったという(Marquess, 2014)。結果として、Marquess と Henson の「2 人のスティーヴ」を除きフルタイムで OpenSSL の開発に取り組む開発者はおらず、それが Hertbleed バグの混入を見過ごした原因の一つと考えられる。Hertbleed バグが明らかになった後は若干個人寄付に増加が見られたが、それでも年間 9000 ドル程度に留まった。

目玉の数 Given enough eyeballs…

Heartbleed problem は、オープンソース・ソフトウェアの信頼のイメージに打撃を与えた。なぜなら、Eric Raymond は、ソフトウェア開発におけるプロプライエタリな手法とオープンソースの手法を比較して、当時も今も多くの FOSS (free and open source software)開発者が信じている有名な言葉「十分な目玉があれば、すべてのバグは浅いものだ」(given enough eyeballs, all bugs are shallow.) (Raymond, 2001, p.30)という「リーナスの法則」(“Linus's Law”)を提唱し、オープンソースの作業習慣がなぜより高品質でバグの少ないソフトウェアを生み出すと考えられるのかを要約してみせたからである。それ故、オープンソースでは、ソースコードが自由に利用できるので、Heartbleed のようなバグは起きないことになっていた。しかし実際には、OpenSSL のソースコードは完全に公開されていたにもかかわらず、Heartbleed は発見されるまで 2 年以上かかったのだった。

Raymond の主張は、ソフトウェアテストの代わりになるという考えだけでなく、バグを検出するために特別な努力は必要ないという考えも含まれ

ていた。オープンソース・プロジェクトのメンバーは、単に開発者として業務を遂行することによって、自ずとバグに気づき、それを修復することができるはずだったのである。ところが、Heartbleed problem で明らかになったことは、ソースコードが公開されていても、注目度が低いので目玉が少ないことがあるということである。

結論 Concluding remarks

Unix 哲学は、ソフトウェア開発の方法論として長年重視されてきており、極力単機能の小さなプログラムをつなげて利用するというモジュラー化と、それによる再利用性の向上が、ソフトウェア開発においてはグッド・プラクティスとして重視されてきた。その結果、巨大で複雑なシステムを構成するあまり人目につかないピース、しかもボランタリー・ベースに近い形で開発が行われるごく小さなピースに脆弱性があった場合でも、全体が危険にさらされるという「ネブラスカ問題」が生じた。

Figure 1 の Xkcd の漫画の作者は、ブロックの縦横の長さに特に意味を持たせていないようだが、実は、本稿の議論を踏まえれば、ブロックの縦幅がコードの複雑さや量、横幅が注目の量つまり「目玉の数」を表していると考えるとネブラスカ問題の本質が見えてくる。実際、縦が長くとも横が長ければ安定するのであり、横が短いから折れやすくなるのである。これは、ネブラスカ問題を引き起こすメカニズムの本質とも言える。つまり、これまでは放っておいても自然と存在すると考えられていた「目玉の数」を、意図的に増やして確保することが、根本的な解決策なのである。

ただし、仮に目玉が十分にあり、一部の人が気づいて、修正すらした問題であっても、世界中のあらゆるコードを完全に対策できるとは限らないことには注意がいる。たとえば、GNU GRUB のようなケースである。GNU GRUB (GRand Unified Bootloader) は、GNU プロジェクトにおいて開発されている高機能なブートローダである。オペレーティング・システムを起動するのが主たる機能であり、GNU/Linux がインストールされた多くのコンピュータにインストールされている。歴史的経緯から、バージョン 0.9x 系の GRUB Legacy と、1.9x 系の GRUB 2 の 2 種類があり、両者は別々に管理されている。Apple の Silicon Mac への Linux カーネルの移植者

である Martin は、Aug. 21, 2022 に、Twitter で以下のような書き込みを行った。

“So today I filed a trivial GRUB bug (latest version runs out of memory, hardcoded heap size needs a bump). I just realized they haven't fixed *any* bug tracker bugs that weren't typos in the last 5 years. Seriously. I knew they were slow about releases, but wow.” (Martin, 2022)

すなわち、GRUB2 のバグトラッカーを確認すると、些細なものを除きこの 5 年間大規模なコードの追加や修正はなされていないということである。Git のコミットログを見ると、修正パッチの多くは外部の企業やディストロ(Distribution の略)から送られてきたものであり、送られてきたまま放置されていて、メンテナンスは名目上に過ぎないようであるというのである。

より正確に言えば、オープンソースにおける開発は、(1)アップストリームのレベル、(2)ディストロのレベル、に分かれて行われており、場合によっては両者が同期しないことで、(3)実質的なフォークになってしまうこともありうる。それに加えて、GRUB2 の場合は、Linux ディストロ・レベルで活発にパッチが当てられていたが、ディストロによってパッチの内容は異なるので、名目上は同じバージョンであっても、あるディストロでは修正されているバグが、他のディストロではそのまま残っていた可能性が高く、メンテナンスが名目上に過ぎないと指摘されたわけである。言い換えれば、仮に目玉が十分にあり、一部の人が気づいて、修正すらした問題であっても、世界中のあらゆるコードを完全に対策できるとは限らないということである。

そのための補完策としては、米国バイデン政権が 2022 年 8 月 25 日に発表したサイバーセキュリティ大統領令[6]に含まれる連邦政府に製品を販売す

[6] The White House wants new transparency into software components. The security benefits won't arrive quickly. https://www.protocol.com/enterprise/biden-sbom-open-source-software

る組織に義務付ける SBOM (Software Bill Of Materials: ソフトウェア部品表)は効果的かもしれない。SBOM は特定の製品に含まれるすべて(プロプライエタリおよびオープンソースなど)のソフトウェアコンポーネント、ライセンス、依存関係を一覧化したもの(NTIA, 2021)で、これにより、リスクを事前に検討することが可能になるかもしれない。

References

Avram, A. (2016, Mar 24). NPM was broken for 2.5 Hours, InfoQ. Retrieved from https://www.infoq.com/news/2016/03/npm/

CVE Details. (2022). Imagemagick: Security vulnerabilities [Security DataSource]. Retrieved from https://www.cvedetails.com/vulnerability-list/vendor_id-1749/Imagemagick.html

De Raadt, T. (2014), Re: new OpenSSL, Openbsd-tech maling-list, [Message: Message-ID: 201406051927.s55JRmMb018899 () cvs ! openbsd ! org]. Retrieved from https://marc.info/?l=openbsd-tech&m=140199655122732&w=2

Collins, K. (2016, Mar, 27). How one programmer broke the internet by deleting a tiny piece of code. QUARTZ. Retrieved from https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/

Marquess, S. (2014, Apr, 12). Of Money, Responsibility, and Pride, Speeds and Feeds [Blog]. Retrieved from http://veridicalsystems.com/blog/of-money-responsibility-and-pride/index.html.

Martin,H @marcan42. (2022, Aug, 21, 0:23). So today I filed a trivial GRUB bug (latest version runs out of memory, hardcoded heap size needs a bump). I just realized they haven't fixed *any* bug tracker bugs that weren't typos in the last 5 years. Seriously. I knew they were slow about releases, but wow [Twitter moment]. Retrieved from https://twitter.com/marcan42/status/1561011127869673478

Munroe, R. (2020?). Dependency, What if? 2 [Webcomic]. https://xkcd.com/2347/

NTIA (National Telecommunications and Information Administration). (2021). Software bill of materials [Guideline]. Retrieved from https://www.ntia.gov/SBOM

Pagliery, J. (2014, Apr 11). Hartbleed bug: What you need to know. CNN Business. https://money.cnn.com/2014/04/09/technology/security/heartbleed-bug/index.html

Raymond, E. S. (2001). *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary* (Revised ed.). Sebastopol, CA: O'Reilly.

Raymond, E. S. (2004). *The art of Unix programming*. Boston, MA: Addison-Wesley

Salus, P. H. (1994). *A quarter century of UNIX*. Reading, MA: Addison-Wesley.

Schneier, B (2014, Apr 9). Hearbleed [Blog]. Schneier on security. Retrieved from https://www.schneier.com/blog/archives/2014/04/heartbleed.html

The jargon file 4.4.7. (2022). bit rot. Retrieved from http://www.catb.org/jargon/html/B/bit-rot.html