The Nebraska Problem in Open Source Software Development

Masayuki HATTA

Abstract: In the world of open source constructed on the basis of the Unix philosophy, there are cases of unsung heroic programs in obscure locations being maintained in a detailed way by a single unknown person or a small number of unknown people, mainly for personal reasons. However, if once these small programs close to the bottom rung of the ladder break, it may cause a loss of balance and collapse of our entire modern infrastructure. This is referred to as the Nebraska Problem in this paper. We can see from the actual and serious case of the Heartbleed bug that “the number of eyeballs” taken for granted in Linus’s Law up to this point needs to be proactively secured, and we need to consider complementary measures, such as SBOM, against risk in advance.

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

---

a) Surugadai University, 698 Azu, Hanno-Shi, Saitama, 357-8555, Japan, hatta.masayuki@surugadai.ac.jp
The author declares there are no competing interests.

© 2022 Masayuki Hatta. This is an Open Access article distributed under the terms of the Creative Commons Attribution License CC BY 4.0 (Attribution 4.0 International) license. The CC BY 4.0 license permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.
What is the Nebraska Problem?

The Nebraska problem is a term coined by this author. The idea for this was inspired by episode #2347\(^1\) of xkcd, a web comic by Randall Munroe that is popular with hackers.

![Figure 1. Dependency (Munroe, n.d.)](https://xkcd.com/2347/)  
*Source: https://xkcd.com/2347/ (CC BY-NC 2.5).*

---

\(^1\) Although the date and time of this comic’s release is not specified, judging from posts on the bulletin board site Reddit, ([https://www.reddit.com/r/xkcd/comments/ibnqr3/xkcd_2347_dependency/](https://www.reddit.com/r/xkcd/comments/ibnqr3/xkcd_2347_dependency/)) we can estimate that it was from 2020.
The Nebraska problem in open source software development

The Munroe comic entitled “dependency” is shown in Figure 1, and the rectangular block represents the program development project for configuring the system. Upper-level blocks are placed on lower-level blocks and the programs constituting the upper level depend on the programs constituting the lower level. On the third level from the bottom, there is a thin block that supports the entire system of “all modern digital infrastructure.” This thin block is “a project some random person in Nebraska has been thanklessly maintaining since 2003.” In other words, these programs, in obscure locations, are being maintained in detail by a single unknown person or a small number of unknown people, mainly for personal reasons, and if once these detailed programs, which are close to the bottom break, then the total balance of “all modern digital infrastructure” may be lost, causing it to collapse.

According to the caption, the illustrator drew the comic with an incident that occurred with ImageMagick in mind. ImageMagick is an image conversion utility released in 1990, and it is used for conversion between various graphics file formats, as well as executing various other types of conversions. There are numerous other libraries and APIs that are capable of performing these tasks, but as ImageMagick has been around for a long time, many programs, especially those used by web services, have relied on ImageMagick for its APIs or for performing the necessary conversions. As they are reliant on ImageMagick, a behind-the-scenes utility that has existed for a long time, their programs would break if ImageMagick were to disappear. Despite this being the case, ImageMagick is known to have numerous vulnerabilities (e.g., CVE Details, 2022), and in 2016, a

---

2 It is not clear why Nebraska was chosen, but it remains true that in U.S. popular culture, Nebraska is often treated as a synonym for “a country area with nothing to see or do” and has an unfavorable image. For example, the leading character in the popular TV drama “Better Call Saul” was last seen escaping to Omaha, Nebraska, pursued by both police and gangsters.
major vulnerability (CVE-2016-3714) nicknamed ImageTragick was discovered. Figure 1 depicts the existence of open source projects that have many bugs, even though they are widely used.

### Unix Philosophy

The reason that “All modern digital infrastructure” is depicted with the structure shown in this comic is a result of the history of computing. In the world of computing, reusable libraries and code have been a major driving force for technological development. The reason for this is that by outsourcing commonly-used basic functions to other libraries, developers are spared the cost and time required to “reinvent the wheel.” As a result, many small packages, many of which are packages containing only a single function, can be used in larger-scale programs, and so a dependency occurs. This characteristic is particularly prominent in open source development, and in Unix and Linux, which are representative of open source, it is normal for one program to be used for one small task, with small programs being linked together by a shell script. This is the essence of the “Unix philosophy” (Raymond, 2004; Salus, 1994).

Additionally, in recent years, the JavaScript platform Node.js and Python have been widely used. These languages are designed in a way that developers from around the world can gather to use many small convenient libraries that each has written to make new, more powerful, libraries. The result of this, however, is that they have the risk of new unexpected bugs suddenly occurring somewhere in the chain of dependencies. In theory, such systems can be written and maintained with fewer lines of code. This may sound good for the

---

3 https://imagetragick.com/ Moreover, ImageMagick itself depends on many other libraries, and the fact that these may also contain their own vulnerabilities has been pointed out as a problem. For example, https://news.ycombinator.com/item?id=24193871
The Nebraska problem in open source software development

developer, but a highly optimized system is also extremely susceptible to rapid change. For example, in the left-pad incident that occurred in 2016, a discontented developer, by introducing just 11 lines of code, was able to destroy everyone’s build, and the builds of many web services dependent on left-pad failed (Avram, 2016; Collins, 2016). That is to say, because the code is open source and the source code is available publicly, a third party may modify the code and not notice the bug (due to high reusability and ease of modification).4

Heartbleed Problem

The Heartbleed problem has been raised as a serious case of the Nebraska problem. Heartbleed is the name of a bug discovered in OpenSSL, which is a library that enables encrypted communications over the Internet. OpenSSL is free open source software for which development has continued since 1998, and is published under the Apache License. Heartbleed was discovered in April 2014 and, as a result of the Heartbleed bug, a vulnerability occurred in the OpenSSL open source library that guarantees secure communications, and it was revealed that a significant portion of the Internet had become easy to attack. As secure communications over the Internet, including those used by e-commerce, were in fact reliant on OpenSSL, this was a major problem.

When the Heartbleed bug came to light, service providers rushed to apply OpenSSL patches, and customers were forced into changing passwords that may have been compromised. The impact was extensive and Heartbleed was widely recognized as a horrendous security bug that had occurred over the Internet. Security specialist Bruce Schneier put it as follows: “‘Catastrophic’ is the right word. On

4 This can be thought of as a kind of open source software supply chain problem.
the scale of 1 to 10, this is an 11.” (Schneier, 2014).

According to Theo de Raadt, the founder of OpenBSD and OpenSSH, malloc, which is a memory allocation library, had applied a patch long before this to prevent the kind of abuse seen with Heartbleed. However, at the same time, in the case of OpenSSL, “[Because of an inability to test,] a bug shows up which leaks the content of memory mishandled by malloc() and free(). If the memory [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). Thus, a problem for which measures had been taken solely on malloc had remained on OpenSSL under the guise of improving performance on certain systems. In other words, even though a latent bug had been detected and a patch had been applied, it had been avoided due to a technical decision to prioritize efficiency over security. It is perhaps the case that the wrapper had escaped close examination because it was considered trivial and was not something new being added. The wrapper was established as a piece of code that nobody could modify. Based on this perspective, de Raadt (2014) was highly critical, saying that “OpenSSL is not developed by a responsible team.”

In fact, Heartbleed attracted as much attention for the team structure that was responsible for it as for the bug itself. The CNN coverage broadcast immediately after the discovery (Pagliery, 2014) highlighted a poor development support system that was considered insufficient given the importance of OpenSSL, especially its economic importance. The news at its time of discovery in 2014 led with the headline “The Internet Is Being Protected By Two Guys Named Steve.” In fact, these guys named Steve were virtually unknown volunteers, whose efforts supported the security of major websites around the world, although they were overworked and underfunded.

Historically, the open source development model has been largely dependent on the free and continual contributions of
hobbyists. Some projects, such as Linux, may be large enough to attract sufficient attention to warrant the creation of an organization; however, smaller projects may be reused by larger projects and effectively maintained by a single person or very few people. To maintain a library, it is necessary to have extensive knowledge not only of the library itself but also of its use cases and scope. This is normally only possible with several years of maintenance experience; therefore, the maintaining party cannot easily be replaced. If this kind of project stagnates, it is often the case that rather than somebody taking it over, it is abandoned and the phenomenon known as bit-rot occurs (The jargon file 4.4.7, 2022). When such a project is positioned upstream in the chain of dependencies, there is a high likelihood of risks occurring for downstream users.

In the case of Heartbleed, OpenSSL was a relatively large software project with about 500,000 lines of code as of 2014; however, a lack of funding prevented it from focusing on development and it was unable to attract developers. According to one of the OpenSSL developers, Steve Marquess, Stephen Henson, who was the main lead on OpenSSL development, worked on it virtually full-time, despite being a complete volunteer. In 2009, the OpenSSL Software Foundation (OSF) was established by Marquess as a non-profit organization to raise funds and support development, but the situation remains largely unchanged. Annual donations obtained by the OSF before Heartbleed was discovered were normally in the range of $2,000. In addition to this, Marquess was able to sell support contracts for OpenSSL in commercial software ($20,000/year with the US Department of Defense) and engaged in other contracting work ($250/hour). However, Henson’s annual income was less than a fifth of what Marquess was earning as a consultant unrelated to OpenSSL (Marquess, 2014). As a result, there were no other developers engaged in the development of OpenSSL on a full-time basis other than the “two guys named Steve” (Marquess and Henson),
and this may be one reason that the infiltration of the Heartbleed bug was overlooked. After the Heartbleed bug came to light, there was an increase in personal donations from young people, but this still amounted to no more than $9,000 a year.

**Given Enough Eyeballs…**

The Heartbleed problem was very damaging to the image of trust in open source software. The reason is that Eric Raymond compared proprietary and open source methods of software development. Many FOSS (free and open source software) developers both then and now concur with Linus’s Law, the famous saying that “given enough eyeballs, all bugs are shallow” (Raymond, 2001, p. 30), and he summarized why open source work practices should result in software that was of a higher quality with fewer bugs. The reasoning is such that, in open source, since the source code is freely available, bugs like Heartbleed were not supposed to occur. However, despite the fact that OpenSSL source code was completely accessible, it still took two years or more for Heartbleed to be discovered.

Raymond’s assertion was not only that this would take the place of software tests, but includes the idea that special effort is not required to detect bugs. It had been thought that open source project members would be able to notice and fix bugs during performing their duties as developers. However, what the Heartbleed problem brought to light was that even if the source code is published, the level of attention on it is low, which accounts for the lack of eyeballs.

**Conclusion**

The Unix philosophy has long been focused on as a methodology for software development. Indeed, the modularization of software development by linking small programs with minimal functionality
and improving reusability through such modularization has been lauded as a good practice for software development. As a result, the Nebraska problem arose, in which a vulnerability in even a small piece of a large and complex system, which is developed on a voluntary basis, could compromise the entire system.

Although the author of the xkcd cartoon in Figure 1 does not appear to give any special meaning to the length and width of the block, based on the discussion in this paper we can say that the essence of the Nebraska problem becomes clear if we consider the complexity and quantity of the code to be expressed by the height of the block, and the width to represent the amount of attention, or “number of eyeballs.” In fact, even if it is high, it will be stable if it is wide, but easy to break if narrow. This can be said to be the essence of the mechanism by which the Nebraska problem occurs. In other words, the fundamental solution to the problem lies in intentionally increasing and securing the “number of eyeballs,” which until now was thought to exist naturally even if nothing was done about it.

However, it should be noted that even if there are a sufficient number of eyeballs, and even if some people have noticed and even fixed the problem, it still may not be possible to provide measures for all the code in the world. For example, there is the case of GNU GRUB. GNU GRUB (GRand Unified Bootloader) is a high-functioning boot loader developed as part of the GNU project. Its main function is booting the operating system, so it is installed in a large number of computers in which GNU/Linux is installed. For historical reasons, there are two types, which are managed separately: the version 0.9x series GRUB Legacy and the 1.9x series GRUB 2. Martin, who transplanted the Linux kernel in the Apple Silicon Mac, posted the following on Twitter on Aug 21, 2022.

“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)

That is to say, if we check the GRUB2 bug tracker, we can see that excepting trivial items, no major code additions or modifications have been made in the past five years. If we look at the Git commit log, the majority of the fix patches were sent from external companies or distro (abbreviation for distributors), and were just neglected after being sent, with maintenance only being carried out on a nominal basis.

To be more precise, development in open source is carried out separately at (1) the upstream level and (2) the distro level, and in some cases, due to the fact that they are not synchronized, (3) an effective fork is possible. In addition to this, it has been pointed out that whereas in the case of GRUB2, patches were actively applied at the Linux distro level, as the content of the patches differ depending on the distro, even if they are nominally the same version, there is a high likelihood that bugs fixed on a certain distro will remain on another distro, and that maintenance is only being carried out on a nominal basis. Put another way, even if there are a sufficient number of eyeballs, and even if some people have noticed and even fixed the problem, this does not mean it is possible to provide measures for all the code in the world.

As a complementary measure for this, the SBOM (Software Bill of Materials), which is a mandate for organizations that sell products to the federal government, and was contained in the Cybersecurity Presidential Directive5 issued by the Biden Administration in the US on August 25, 2022, may be effective. The SBOM lists (NTIA, 2021) all (proprietary and open source, etc.) software components, licenses,

5 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
and dependencies included in a specific product. This means that any risk can be considered in advance.

**Acknowledgments**

This work was supported by JSPS KAKENHI Grant Number JP18K11993.

**References**

Avram, A. (2016, March 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, March 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, April 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, August 21). *So today I filed a trivial GRUB bug (latest version runs out of memory, hardcoded heap size needs a bump).* [Tweet]. Twitter. Retrieved from https://twitter.com/marcan42/status/1561011127869673478

Munroe, R. (n.d.). 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, April 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, April 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