THE SECOND MAINTAINER

Inside the XZ Backdoor

Ethan Hawksley

The Second Maintainer

Inside the XZ Backdoor

Ethan Hawksley

Imprint: Lulu Press, Inc.

ISBN: 978-1-291-54737-5 (Ebook)

https://hawksley.dev

Contents

1: The Tukaani Project

2005, Finland: open-source software was experiencing an explosive growth in popularity. Whilst the world migrated to Windows XP, a global enclave of nerds was using Linux. It was a free operating system, born out of the early 90s, maintained by a community of volunteers.

Lasse Collin, a Finnish student in his early twenties, belonged to this group of nerds. He was embedded deep among programmers, and many of his friendships were formed in online chat rooms.

Linux is the overarching term that encompasses many distributions, or distros for short. There are many options to choose from: Ubuntu, Debian, Arch, Fedora, and more. Each had their different software and communities, and Lasse's personal favourite was Slackware Linux: a distro that had been going strong since its inception in 1993. Consequently, his appreciation for the distro prompted him to make a derivative of it.

Logging on to the internet, he informed his friends of the plan. They jumped on board and began brainstorming immediately. Naturally, any good project needed a good project name and, after much deliberation, they settled on titling it Tukaani Project, hence calling the distro Tukaani Linux. Tukaani is Finnish for a toucan, so Lasse's friend Ville Koskinen designed a fitting mascot for the project - Bob the Toucan.

However, they soon stumbled into a problem: Slackware Linux included plenty of preinstalled software. It may have been convenient whilst using it, but it made the installer much larger. It was so large that it couldn't fit on a CD-ROM and instead required a more expensive DVD-ROM.

Lasse suggested that instead of removing software, they should try compressing the installer. Slackware already used "gzip", a popular compression algorithm designed in 1992, which could compress the installer to one gigabyte. Better, but not small enough. Lasse then tried using "LZMA", a newer and slightly less well-known compression algorithm.

Perfect. The installer compressed to just under 700 megabytes, narrowly fitting onto the CD-ROM.

LZMA compressed files better than gzip, yet it was majorly underutilised, which shocked Lasse. He promptly made a decision to change this. The world needed to start using LZMA, and he would be the one to introduce it to them.

He informed the members of the Tukaani Project about the capabilities of LZMA, and they too were surprised by its

efficiency. Each of them started creating small but useful programs that used LZMA.

* * *

Over the course of a year, the Tukaani Project created a suite of software named "LZMA Utils". It was a jack-of-all-trades and could manipulate LZMA in every way imaginable. The tools ran on the smallest of chips and on the biggest of servers.

It didn't take long for the word to spread. By 2007, LZMA Utils had accumulated a large community of both users and contributors. Although gzip still stood as the dominant algorithm, LZMA was finally respected.

Lasse wasn't satisfied, though. In his regular use of LZMA he had learned all of its flaws. The algorithm was good, yes, but the files it produced were fragile: it was prone to corrupting during file transfers, leaving it entirely useless. Lasse's coding skills had also improved, and starting afresh would let him build back stronger.

He needed to make a new file format.

* * *

His plans for the format were fuzzy at best. He knew he wanted to use LZMA, but there were so many other unknowns. He couldn't make it alone, so he contacted the person who knew compression best. He reached out to Igor Pavlov, the original creator of LZMA.

Igor was very receptive to Lasse's inquiries, so together they worked on a new format, with assistance from Ville. They chose to use Igor's latest algorithm - LZMA2. It split data into smaller

chunks and compressed each in parallel. If a chunk couldn't be compressed, it would be left as-is.

Development of the file format took a year - due to Lasse's personal priorities drawing him away - but at last, in 2009, the format was ready. He titled it "XZ", and officially rebranded LZMA Utils to XZ Utils.

* * *

XZ was an instant success with developers and became used almost everywhere. Linux distros started including XZ Utils pre-installed. Some distros even used XZ Utils to compress the installer itself. Competitor file formats came and went, but XZ was one of the few that stayed. Lasse's project was a success.

After the creation of XZ Utils, the Tukaani Project began to slowly drift apart. Some of Lasse's friends found jobs, others had families. He, too, became increasingly busy. Yet, as its creator, XZ Utils remained his responsibility. People would occasionally submit code to add, but Lasse was the sole maintainer. As the software spread and became used by millions, he continued to maintain the project.

2: A New Face

In 2021, twelve years after its creation, XZ remained a dominant file format. With its popularity, however, came bug reports. Plenty of bug reports. Even though many bugs were minor, they constantly stole time out of his day for Lasse to deal with them all.

It was November 10th when he checked his inbox and found an email which had arrived two weeks prior. Lasse was not known for regularly checking his emails - he frequently took internet breaks. The email was addressed from another developer named Jia Tan, and it contained a patch for XZ Utils. He proposed adding the file *.editorconfig*, that provided configuration for how text editors would format code. Lasse hoped it could settle the debates he'd had over formatting. However, when he added the patch to the code, problems ensued. He began to draft a reply to Jia.

> "Thanks! I hadn't heard about this before, but it sounds nice."

There hadn't been much activity on XZ Utils recently, so Lasse was glad to see any new faces. Who knew, if he was lucky, maybe Jia would stay?

The main issue with the proposed *.editorconfig* was that it needed to be more thorough. Jia had added formatting rules for some files but not all.

> "There are multiple indentation styles, even under src."

Lasse proceeded to explain the different styles that had accumulated over the years. It had grown quite complicated, and only he understood it all. He ended his email warmly.

> "Is it good enough or did I add bad bugs? :-)"

It was hard work to maintain software, but when you got genuine help from others, it was all worthwhile.

* * *

It took two days for Jia to reply. When he did, he found and fixed a couple more problems with the patch. He ended his email with a nitpick:

> "Why do you use tab width 8 for your source files? Either way, it's your preference, and I will follow your lead."

Lasse smiled as he read it. When he indented code, he would indent it eight characters to the side, instead of the more commonly used four. It made no technical difference: he just

thought it looked nicer, and he appreciated that Jia didn't mind. XZ Utils was Lasse's project, after all.

He applied Jia's patch to XZ Utils, and then uploaded the code to GitHub - a website for hosting code publicly. This exchange had rejuvenated Lasse's hope for the project and gave way to the idea that he wouldn't have to keep developing XZ Utils alone.

* * *

Christmas rolled around the corner, and a new email appeared on the mailing list. It was from Sebastian Siewior - a developer of the Debian distro. He asked if parallel decompression would get implemented, since decompressing different chunks simultaneously would greatly speed up the process.

Lasse took the time to consider it. He'd been meaning to get round to this for a while. Over the years, XZ had earned a reputation for compressing well, but decompressing slowly. If he implemented parallel decompression, it would help close the gap between XZ and other compression formats.

However, implementing parallel decompression would be no simple task. It could take quite some time before a finished implementation was ready, though the arrival of Jia gave him some hope. If he stayed, the extra help would be invaluable.

Lasse created an early draft of the feature. As he had hoped, Jia stuck around to comment. He highlighted where Lasse should restructure his code, and spotted a simple typo in the documentation. They both continued to iterate on improving parallel decompression.

Unfortunately for Sebastian, parallel decompression was not ready in time for Christmas. Major releases of XZ Utils were infrequent, and the feature required more testing than usual due to its complexity. Meanwhile, more development was in the works.

* * *

Two weeks after Sebastian's request, Jia had prepared a new patch. He had ported over the existing automated tests to a new framework named *seatest*. These automated tests were responsible for ensuring the software always functioned properly - if new code broke XZ Utils, he would be alerted.

What shocked Lasse most was the size: the patch was hundreds of lines of code, with clear effort put into it. He smiled. It looked like Jia was here to stay. It was a shame to lose the testing infrastructure he had built himself over the years, but it was a step in the right direction for the project. Jia's patch needed some improvements here and there, but was remarkably high quality - he clearly knew his stuff.

3: Public Pressure

Fast-forward five months to April 2022, and XZ Utils was the busiest it had been in years. Lasse and Jia were working through the backlog of bugs and feature requests. Jia prepared another patch for Lasse that allowed users to easily control the steps taken in compression. This idea had been long requested and served as another incremental improvement.

When Lasse went to check the mailing list, an email had arrived from a new, opinionated developer, Jigar Kumar. He criticised Jia's latest patch, yet also complimented Jia's high-quality code. The message signed off with a quiet lament over the pace of development. Lasse had to silently agree: despite the increased pace of development, the backlog was still sizeable. He didn't have the time to commit fully to XZ Utils.

Jia fought back against Jigar. He quickly replied and explained how development was moving at a steady pace, given that nobody was paid to work on the project. After all, it was just a hobby project, even if the hobby project was used by millions, he believed there was no need for more urgency. He

took Jigar's feedback into consideration, acted upon one, and discarded the rest.

Jigar strongly objected to Jia's optimism and wrote a cutting reply.

> "Patches spend years on this mailing list. 5.2.0 release was 7 years ago. There is no reason to think anything is coming soon."

Although Lasse intended to release a new version this year, he found it hard to disagree with the sentiment. He too was unhappy with the pace of XZ Utils.

* * *

Another month passed where Lasse and Jia uneventfully worked on XZ Utils. Bugs were fixed and documentation was improved - business as usual.

When Lasse next checked the mailing list, Dennis Ens - a programmer with expertise in the Java programming language - was there to greet him. He inquired about the status of XZ Utils' port to Java.

> "Is XZ for Java still maintained? When I view the git log, I can see it has not updated in over a year."

Lasse sighed. Amongst all the work he and Jia had been putting into XZ Utils, he had neglected the Java port. Dennis was right, he should pay it more attention than he had been. He began to write a reply.

> "After the latest XZ for Java release, I've tried to focus on XZ Utils (and ignored XZ for Java). Obviously that hasn't worked so well either, even if some progress has happened with XZ Utils."

He paused. How was he supposed to properly support XZ for Java? XZ Utils was neglected enough. As is, he couldn't afford to divert attention from it. He thought of how long it took for him to reply to Jia's emails. He continued the email.

> "Jia Tan has helped me off-list with XZ Utils, and he might have a bigger role in the future. It's clear that my resources are too limited (thus the many emails waiting for replies), so something has to change in the long term."

Jia was the project's hope - Lasse's time was too infrequent for the current arrangement to continue much longer.

* * *

As spring turned to summer, Jigar remained a thorn in the side of the project. His public complaints were incessant.

> "There is no reason to think anything is coming soon. Over 1 month and no closer to being merged. Not a surprise."

Progress on XZ Utils had been going well between Lasse and Jia. Jia's feature to add new options was almost ready to add to the main software. They developed in private, so it was no surprise Jigar complained about the seeming lack of progress.

> "Progress will not happen until there is a new maintainer. Submitting patches here has no purpose these days."

Lasse continued to ignore the messages piling up. Jigar was wrong, and while he may have been a good programmer, he was insistently impatient.

> "The current maintainer lost interest or doesn't care to maintain anymore."

This was the straw that broke the camel's back. Lasse cared so much about XZ Utils: it was his child, having been with him for over a decade. If the project was anybody's, it was his.

He started to draft a response, but paused. There was a grain of truth to Jigar's complaints. Lasse hadn't lost interest, no, but he had lost the time to commit enough to the project. On the other hand, Jia had plenty of time. He was always punctual and had been making even bigger patches. He began to write.

> The thought of finding new maintainers has existed for a long time too. Perhaps the moment after the 5.4.0 release would be a convenient moment to make changes in the list of project maintainer(s).

Version 5.4.0 was the next planned version for XZ Utils. Lasse hoped it would be ready by the year's end. Jigar's reply did not embody the same hope.

> "With your current rate, I very much doubt we will see a 5.4.0 release this year. Why wait until 5.4.0 to change maintainer? Why delay what your repo needs?"

This was enough for Lasse. Jigar clearly wasn't listening to what he was saying. In an attempt to appease him, Lasse sent one final email to Jigar.

> "Jia Tan has been helping a lot off-list and is practically a co-maintainer already. :-) In any case, some change in maintainership is already in progress, at least for XZ Utils."

Jigar was all out of snarky replies. Ironically, Jigar instead of Lasse seemed to lose interest in the project. He never sent another email to the mailing list.

4: Handing Over the Keys

For the next few months, Lasse and Jia quietly continued work on XZ Utils. The crown jewel of the next release - parallel decompression - was polished and tested to perfection. Dennis, the Java developer, had drifted away, leaving the project free of drama again.

Lasse admired the state of the project. It was finally back on track and making steady progress. He admired Jia too. What better help could a maintainer want?

* * *

The question of Jia's promotion to maintainer stayed at the forefront of Lasse's mind. He had been intending to get round to it for some time, and in late October he finally decided to pull the trigger. After almost a year of collaboration with Jia, he officially invited Jia to become a maintainer of the Tukaani Project. He gladly accepted the offer and thanked Lasse for the past year of work together. It was just the two of them in Tukaani now - everyone else from the project's early days left long ago.

Alongside the promotion, Jia suggested they modify the project's bug report email. At the time, it had been sending all emails just to Lasse. Jia asked if the emails could be forwarded to them both, and Lasse agreed. He was at his computer infrequently, so it made perfect sense for somebody a bit more punctual to receive the emails too. The bug report email was changed to *xz@tukaani.org*, an address they both had access to.

Becoming a maintainer gave Jia the ability to alter XZ Utils' code without having to ask Lasse first. As a result, development quickened drastically. Although they each worked across the whole project, they specialised in specific areas: Lasse focused on compression and decompression, whereas Jia revamped the automated testing.

* * *

2023 rolled around and there were no significant features planned, so Lasse and Jia completed project chores. They improved documentation, fixed a smattering of warnings, and streamlined the compilation.

Lasse was very pleased with the software's state. Through his and Jia's hard work, there were no major issues left unresolved. He watched Jia grow accustomed to the code, and he could tell XZ Utils was now in safe hands. The next version - 5.4.1 - was ready for release. He figured this version should be his last, and future releases would be Jia's responsibility. Lasse created and published the new version online. Jia subsequently announced the new release and explained all the new features. Lasse was impressed by his style - the version announcement mirrored his past announcements well.

As winter continued, development was uneventful. Lasse and Jia fixed a handful of bugs in XZ Utils and published version 5.4.2. Jia's promotion to the second maintainer had been a major success.

5: Introducing Ifuncs

Google operates a free service named "OSS-Fuzz", where developers of popular software can submit their code to be automatically tested on Google's servers. OSS-Fuzz constantly checks for bugs and exploits, reporting any that it finds to the project's maintainers. Since XZ Utils was so widely used, it was part of the OSS-Fuzz program.

Shortly after the release of 5.4.2, Jia requested Google to let him access the OSS-Fuzz reports. When Google contacted Lasse for confirmation, he approved. Since Jia was a maintainer now, he was allowed greater control over the project.

A week later, Lasse was contacted by Google again. Gabriela Gutierrez - a Google employee on their security team - had noticed XZ Utils didn't have a proper reporting procedure. If users found bugs in the software, it wasn't obvious how they should let maintainers know. Gabriela suggested an example procedure and Lasse gave it the green light. However, Jia had a few objections. He made a revised version that kept the intent similar, but reworded several paragraphs.

Version 5.4.3 was ready not too long after. It only contained a couple of fixes to the translations and some edge cases when compiling the software. The main highlight was Jia's new role. He was in charge of the release from start to finish. Whilst Lasse watched, Jia compiled the software according to the instructions. He finished by adding his digital signature to XZ Utils.

The transition wasn't entirely seamless: automated systems flagged the new digital signature, so some administrators had to manually approve the update. This update wasn't urgent, though. No one was going to lose any sleep over a couple translation fixes being delayed, after all.

* * *

Development picked up as June began. Three months after the project improved its security posture, a new developer called Hans Jansen appeared. He had prepared a pair of patches that would improve the performance of XZ Utils. They leveraged "indirect functions", commonly shortened to ifuncs. These ifuncs allow a program to rewire itself whenever it runs. Entire functions of code can be swapped with each other, according to the pre-programmed conditions. His patch optimised the *CRC64* function, which was used for checking if a file was corrupt. If the ifunc detected modern hardware, it would switch the *CRC64* function to a new, highly specialised version.

Lasse took a look and noticed Jia had already reviewed the patch. He had picked out the usual reasons a patch needed reworking: bad variable names, too many letters on a line of code, and unnecessary spaces. However, his opinion of the CRC patch was largely positive:

> "Overall, this seems like a nice improvement to our function picking strategy for CRC64. It will likely be useful when we implement CRC32 too :)"

CRC32 is just a smaller version of *CRC64*, which is slightly less accurate but faster to compute. Hans' patch only changed the *CRC64* function, but it was simple enough that Jia could follow in his footsteps to implement it for *CRC32* too.

Lasse wasn't quite as convinced as Jia. When he ran the automated tests on the patch, they returned errors. Looking closer, he noticed all of these errors were related to the optional *AddressSanitizer* memory checker. It was responsible for ensuring there were no unsafe modifications to memory. After researching, it turned out this was a well-known incompatibility. Ifuncs rewire the program before the *AddressSanitizer* is ready, resulting in a crash. The typical fix was to disable ifuncs when using the memory checker. He shrugged. It seemed an acceptable trade-off.

With compatibility solved and out of the way, his other concern was performance. He asked Hans for details about the patch.

> "How big a difference in speed does your patch make with your code? I would like to understand the real-world improvement that ifunc can make."

Hans replied to him quickly.

> "I was noticing a 4-5% improvement. I'm also running all of this on older hardware, which may be contributing to the speedup."

While a 4-5% speed improvement isn't anything spectacular, it seemed like a nice bonus. He fired off a quick private message to Jia to gauge his opinion on the matter. When Jia reaffirmed that he was supportive of the addition, they thanked Hans for his contribution and added his code to XZ Utils.

* * *

After Hans' ifuncs were added to XZ Utils, Lasse and Jia received another email from Google's OSS-Fuzz service. It highlighted the new patch was causing constant crashes whenever the software was tested. They both read through the log trying to spot the issue, and it turned out to be as expected: OSS-Fuzz was utilising *AddressSanitizer*, not unlike their own tests had. The fix was simple: exempt ifuncs from Google's automated scans. Jia created a request for this, and within ten minutes, the change was in action.

This made Lasse uncomfortable. By disabling ifuncs in scans, any problems within the ifunc code wouldn't be easily identified ahead of time. Fortunately XZ Utils didn't use ifuncs in many places. He had read the documentation and knew how to spot the common pitfalls with them. The performance impact was

also too good to pass up. It wasn't ideal, but he could sleep easily.

* * *

Work carried on as per usual. August marked Jia releasing version 5.4.4. It introduced web browser support to XZ Utils but did not support all features - only a subset of them. Regardless, it was a notable feat.

Hans returned to XZ Utils in September. His new patches once again focused on the *CRC* algorithms from earlier. The earlier contributions were for *CRC64*, and he had finally returned to improve *CRC32*. This time it even came with benchmarks, promising up to 70% faster performance. Again, the speed improvement came from using ifuncs. Lasse could hardly believe his eyes - you don't see improvements this good every day.

Hans kept iterating over his patch making small incremental improvements. After a few days of silence, Lasse finally responded.

> "I'm sorry for the delay. Neither Jia nor I have been able to look at this in the past few days :-(We are both happy to see an improved version of CRC32."

The code was genuinely solid. They all bounced ideas off each other for the next few days, until the final patch was as good as possible. He then approved and applied the patch to XZ Utils. Lasse saw this a few hours later and left a message.

> "We (or likely it's mostly Jia) will do a few tests later. Thanks again!"

* * *

Jia worked hard over the next week. He clearly wasn't satisfied with Hans' code - he made new files, rearranged old files, and rewrote comments explaining how it all worked.

Lasse was hard at work too. He implemented "sandboxing", where XZ Utils is cut off from the rest of the computer. Through this, he could mitigate the damage that a vulnerability in XZ Utils would have. Any damage caused by a bug in XZ Utils would be restricted to inside the sandbox.

Jia was inspired by Lasse's progress and also worked on the sandboxing. He fixed the automated tests by disabling sandboxing when using *AddressSanitizer*. He also continued adding sandboxing to more parts of XZ Utils, such as *xzdec* - the part dedicated to decompressing .xz files back to their original form.

* * *

A month later, Jia brought up the topic of the project's homepage. At the time, the XZ Utils was hosted on tukaani.org. Since its inception, it had just been another part of the Tukaani project. Jia thought it was more than that though: he wanted XZ Utils to have its own website and identity. Lasse was reluctant, but folded when Jia offered to handle the site creation himself. He designed the new website and even made a new logo to replace Bob the Toucan: simply the letters *XZ* in bright orange and yellow.

Minor releases continued to be prepared by Jia. Versions 5.4.5 and 5.4.6 both had very minimal changes. The major changes, such as the new work on sandboxing, were set to release in the next major version.

6: Obfuscating the Code

Jia Tan had spent two and a half years working on XZ Utils with Lasse. Two and a half years building trust to become a maintainer. Two and a half years spent waiting. Two and a half years of volunteering as a maintainer from the goodness of his heart? No, his true intentions were to create a backdoor in the software - where he could fully control any computer he wished.

His first obstacle was users: despite there being so many users of XZ Utils, many were still using outdated versions. Jia needed them to update in order to compromise them. The best way to achieve this would be to create some much-wanted features, and he prepared a patch to do exactly that.

XZ Utils included specialist filters to compress code for certain CPU architectures better, such as *x64* and *ARM*. Jia created a filter for the architecture called *RISC-V*. This was a nicher architecture, but steadily growing in popularity. He knew people would update for it.

This feature was combined with sandboxing and released as version 5.5.1alpha. Users quickly responded - they loved the all

the new features. Many stated they were excited for when these features would be stable instead of alpha. Jia had now laid the bait. It was time to add the backdoor.

* * *

Jia's goal wasn't merely to compromise XZ Utils. In his current position, he could do that easily. After this long of a wait, however, he wanted to hijack an even more significant piece of software: OpenSSH.

OpenSSH is what lets administrators log in and manage servers from anywhere in the world. It lets companies rent out servers in vast data centres without needing to send somebody over with a keyboard. It was so useful, in fact, that it has near universal usage among the Fortune 500. Since it was installed on millions of company servers, Jia thought it was a prime target.

However, software as important as OpenSSH is always held under extreme scrutiny. Security researchers pore over every line of code, checking and double-checking for vulnerabilities and backdoors. People have tried and failed to break into OpenSSH - but those people were not Jia Tan.

To save themselves from reinventing the wheel, the developers of OpenSSH borrow plenty of code from other programmers. This is known as a "dependency", where OpenSSH needs another program's code in order to run. OpenSSH had a handful of dependencies, but Jia cared about one: Systemd.

On most Linux distros, Systemd is the glue that connects the operating system to the programs running on it. OpenSSH used it to manage notifications and alerts. Systemd is on par with the

importance of OpenSSH, and is also heavily scrutinised. Unfortunately for them, it had a chink in its armour.

Systemd depended on XZ Utils. Accordingly, every time OpenSSH ran with Systemd, it also ran with XZ Utils. Nobody had noticed this weakness - nobody apart from Jia. Even better, he was now a maintainer and had full control over the project.

* * *

When Hans added ifuncs to the program, he took note. Jia created a backdoor hinging upon using ifuncs to swap functions. Despite the intention for ifuncs to rewire a program's own functions, Jia manipulated the ifuncs to replace code within OpenSSH.

Whilst OpenSSH contained many functions, he cared about one in particular. *RSA_public_decrypt* was responsible for verifying passwords when logging into computers. Jia added a special check so that, when he connected, OpenSSH would run any commands he provided. It gave full reign over any company's computer.

Using ifuncs gave Jia another advantage: most automated tests ran with ifuncs disabled. This meant his backdoor wouldn't be flagged in the tests, and would go undetected. Despite this safety, he still had to be cautious. One discovery could blow his entire cover, and reset years of work. He decided to hide the backdoor behind several layers of obfuscation.

If XZ Utils recognised a researcher was inspecting it with a debugger, it wouldn't inject the backdoor. It would also be disabled if the program wasn't OpenSSH.

Lastly, when XZ Utils was compiled to an executable, Jia made it check what distro it was on. He was targeting both Debian and Red Hat Linux - the two post popular distros for companies and governments.

He hid the backdoor and all these checks in plain sight. The XZ Utils automated tests included lots of .xz files that were decompressed to ensure that the software worked. He split the backdoor into two .xz files. The first seemed corrupted, but Jia knew the cipher to recover the compressed code. Likewise, the second decompressed to random letters and numbers, though could be decrypted to the other half of the backdoor. Splitting the backdoor in two prevented any single file seeming suspicious.

Back in 2008, Lasse had added a note to the folder containing the test files.

> Many of the files have been created by hand. There is no better "source code" than the files themselves.

Thanks to this, Jia now had an excuse to add his archives without anybody inspecting closely. The very next day, he released version 5.6.0 infected with the backdoor. All he had to do now was wait for people to update.

* * *

Just an hour after the update, a Linux distro named Gentoo made a bug report. He panicked. How had they found the backdoor so soon? When he checked the report, he let out a sigh of relief. This version was the first to include Hans'

optimisations, and Gentoo wasn't fully compatible with how they were used. Jia quickly wrote a reply explaining the bug to Gentoo and fixed the problem.

Two days passed, and the Debian distro added version 5.6.0 to their unstable packages. Anybody running unstable Debian would update and be backdoored. Jia sent a few emails to Richard Jones, a packager for the Fedora distro. Fedora's main packager for XZ Utils was often busy, so he had stepped up. Jia asked him to update the version used in Fedora, and Richard happily obliged. Fedora's testing version was now also backdoored.

Jia decided to take precautions to prevent another Gentoo incident. In what seemed like a routine update, he added a subtle typo to the sandboxing code so it would no longer run. The backdoor already bypassed Lasse's sandboxing, but it didn't hurt to be careful.

* * *

A few days later, Red Hat also created a bug report. They were testing the program with the debugging tool *Valgrind*, and noticed a variety of memory errors - all caused by the backdoor's ifuncs. It seemed the Gentoo patch hadn't fixed all of Jia's problems. Since Red Hat Linux was one of his main targets, this was potentially disastrous. He modified the backdoored test files to fix the bug, and invented an excuse to explain his why the test files changed.

> The original files were generated with random local to my machine. To better reproduce these

> files in the future, a constant seed was used to recreate these files.

It was a lie, but a plausible one. As a distraction, he also modified Hans' ifunc code. Red Hat fell for it and didn't realise the problem was fixed because of the new test files. They stopped investigating the software too closely. He then released version 5.6.1, containing all the previously mentioned fixes.

Meanwhile, a patch was prepared that threatened to ruin his entire plan. A developer tweaked Systemd so it no longer depended on XZ Utils. This broke the dependency chain from OpenSSH to Systemd to XZ Utils, foiling Jia's entire plan. It was a race against time to update XZ Utils before Systemd could.

* * *

The next two weeks dragged on for Jia as he waited for the new version to spread. It was already in prereleases for both the Debian and Red Hat distros. He had to wait just a little longer though, as he needed the versions in the regular releases of these distros. As soon as that happened, he would be able to activate the backdoor.

On the 25th March, whilst he waited for it to spread, he simplified the instructions for security researchers interested in XZ Utils. To prevent anyone discovering the backdoor, he asked researchers to report issues privately and without the need for elaborate description. He hoped that it would guide people to examine the software less closely.

Lasse was none the wiser about Jia's true intentions. He kept creating new patches for XZ Utils, entirely unaware that a backdoor lay dormant within his project.

Two days later, Debian unstable updated to 5.6.1, and the next day Jia requested the Ubuntu distro to update to 5.6.1. The XZ Utils backdoor was just about to hit Debian's and Red Hat's stable releases, and Jia could hardly wait.

7: 500 Milliseconds

Rewind back a month to late February. XZ Utils 5.6.0 had just been released, and a man called Andres Freund was performing tests. Andres was a software engineer at Microsoft, where he was paid to improve PostgreSQL - a free, community-designed database. Even the biggest companies like Spotify and Netflix used it. He had been developing PostgreSQL since 2009, with various companies paying him to do so. He had a fine eye for performance, with his primary goal being to optimise PostgreSQL to be as fast as possible.

This time, whilst Andres was testing PostgreSQL's performance, he noticed errors in the Valgrind debugger. They pointed towards the latest XZ Utils update, but Valgrind was well known for flagging even inconsequential issues. Andres shelved the errors to the back of his mind and moved on.

* * *

A month of working on PostgreSQL came and went. It was March 27th, and Andres was testing PostgreSQL's performance

on another server. In order for his benchmarks to be as accurate as possible, he wanted to free up more processing power.

On the internet, there are hundreds of automated scanners that try connecting to servers through OpenSSH. Andres' server was no different: the server's logs showed plenty of failed log-in attempts. However, something in the logs caught his eye...

Connection attempts were taking five hundred milliseconds too long. As a rule of thumb, a normal attempt takes about three hundred milliseconds, but these took eight hundred instead. Most people would have ignored it, or perhaps filed a bug report. Andres Freund was not most people - he had to understand what was causing it.

He used his performance expertise to investigate why exactly OpenSSH was slow. The culprit was clear: XZ Utils had something running very slowly. Andres remembered the PostgreSQL errors he saw back in February, and how XZ Utils was the culprit there too. He concluded there must have been an underlying reason.

When he investigated deeper, he immediately faced an issue. Every time he tested OpenSSH with a debugger, it behaved perfectly. He grew suspicious, it sounded exactly like a program trying to hide something. He tried several other debuggers, but none worked. The program would run perfectly every time without issue.

When all hope seemed lost, he remembered another debugger named Intel Processor Trace. Unlike other debuggers, this ran on the hardware. Programs couldn't even tell it was running, and that included OpenSSH. Andres watched as the entire backdoor unfolded before him.

It was growing late, so he decided to call it for the night. Before he went to sleep, he sent an email to Debian's security team. He warned there was something seriously wrong with XZ Utils, and he would have more information tomorrow.

* * *

The next day at work, Andres kept thinking about XZ Utils. He couldn't focus on his meetings: he knew he was sitting on a backdoor that could be triggered at any moment. As soon as he was finished with work for the day, he went straight to investigating again.

He knew there was a backdoor in the program, but he didn't know how it got there. He read the source code online, though nothing seemed suspicious. Andres decided to sanity-check himself and compare the source code with what was actually in each update.

He spotted it. In the script that compiled the update's code, there were a few extra lines. In isolation, it looked entirely innocuous. With Andres' understanding, however, it explained every question he had. The script was heavily obfuscated, but he dissected the code and saw it load the malicious test files, decipher the backdoor, and compromise XZ Utils. When he checked who added this code, there was only one name: Jia Tan.

8: The Aftermath

Maintainers of major distros all have access to a private mailing list. Dangerous vulnerabilities get reported here so maintainers can fix them before the discoverer publicly announces it. The typical grace period is 16 days, in which maintainers have to fix the issue and have people update to it. When Andres Freund reported the backdoor in XZ Utils, they only had one day.

Andres Freund detailed everything he had discovered so far, and the maintainers scrambled to get it under control. For many distros, a compromised XZ Utils was already being used by various beta testers, and was mere days away from ordinary users. The latest versions of XZ Utils were replaced within hours by older, trusted versions.

It had been an incredibly close call.

* * *

The very next day, Andres publicly disclosed the backdoor for all the world to see. He explained where it came from, its effects, and how to check if you were safe. He also added a disclaimer.

> "I am not a security researcher, nor a reverse engineer."

After all, he wasn't some cybersecurity expert. Andres just enjoyed optimising the performance of databases. Unlike everyone else, he couldn't just ignore 500 milliseconds.

The US federal Cybersecurity and Infrastructure Security Agency was alerted to the exploit, and issued an urgent advisory. They recommended all developers downgrade XZ Utils to version 5.4.6 or older. Many minor Linux distros caught wind and delisted modern versions to protect their users.

Red Hat assigned a severity score of 10.0/10.0 to vulnerability, and published an explanation to their blog detailing everything they knew so far.

GitHub disabled the XZ Utils repository and both Lasse's and Jia's GitHub accounts. This helped prevent the spread of the backdoor, but also prevented researchers from analysing it. Thankfully, Lasse had been hosting a personal copy of the source code on his website, so researchers could continue unimpeded.

* * *

In an attempt to spread the word, Andres created a series of social media posts where he warned his followers about the backdoor. He also shared some more details behind its discovery, and a warning that he almost missed it.

> "We got unreasonably lucky here, and we can't just bank on that going forward."

His posts spread much further than he could have imagined. Mainstream news outlets picked up on the story: The Economist published an article explaining how the backdoor had been discovered, Wired described how Jia earned trust and became a maintainer of XZ Utils, and The New York Times dug into Andres' past and the aftermath of the exploit. He was invited to numerous interviews and podcasts, and had become an overnight household name in cybersecurity.

When interviewed, he explained how disorienting the whole event was.

> "I'm a fairly private person who just sits in front of the computer and hacks on code."

After all, he was a database developer, not a security expert.

* * *

As soon as the news of the backdoor broke, Jia fell silent. He stopped responding to emails or creating XZ Utils patches. For all intents and purposes, he had vanished.

When investigators tried to dig into his past, they came up empty handed. He appeared in 2021 and started to contribute code. Before then, there were no traces whatsoever. It wasn't just Jia Tan: Jigar Kumar, Dennis Ens, and Hans Jansen all had no real counterparts. They contributed to XZ Utils and vanished without a trace.

Many government agencies have tried to track down the perpetrators, but none succeeded. The most popular theory is rival governments used these names as pseudonyms. This does have precedent: events such as Russia's 2020 compromise of

SolarWinds and the consequential supply chain attack. Even so, this pales in comparison to the stakes of the XZ Utils backdoor. If Andres Freund had missed it, this would have been one of the most catastrophic hacks in history.

* * *

Lasse Collin was away on holiday when the news broke. He had suspicions about Jia's latest behaviour, but never would have realised the extent to what he was hiding.

He initially tried logging into his GitHub account, but discovered his account had been disabled. Instead, he turned his attention to the Linux mailing list and began to write an email.

> "I'm on holiday and only happened to look at my emails, and it seems to be a major mess.
>
> My proper investigation efforts likely start in the first days of April. That is, I currently know only a few facts which alone are bad enough."

Lasse stopped forwarding XZ Utils mail to Jia, and removed Jia from the Tukaani Project. Nonetheless, he didn't let it spoil the rest of his holiday.

When he returned home, Lasse immediately got back to work. He persuaded GitHub to reinstate his account, and after a week they enabled the XZ Utils repository again. Then started the tedious part - reviewing every patch that Jia ever made, and he noted down anything suspicious or out of the ordinary. What shocked him was how, even with hindsight, it was remarkably benign. Jia's contributions fixed plenty of bugs and added some invaluable features.

It took Lasse two months, but he finally reviewed all of Jia's code and purged the backdoor from XZ Utils. He even revived Bob the Toucan as the mascot. The project was finally safe with 5.6.2.

Now that Jia was gone, he decided to onboard a second maintainer to assist with development. Sam James - a developer of the Gentoo Linux distro - volunteered himself and was accepted after some vetting. Lets just hope this time, history will not repeat itself...

Afterword

Placeholder Text.

Acknowledgements

Placeholder Text.