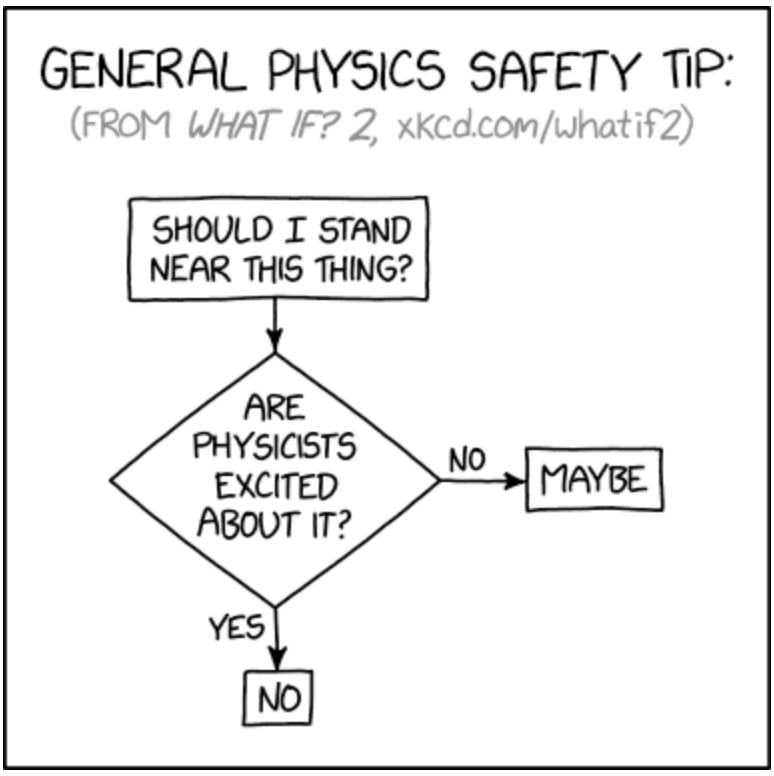
Security Now! #887 - 09-06-22

Embedding AWS Credentials

This week on Security Now!

This week we look at Google's just-announced and launched open source software vulnerability rewards program. We ask the question whether TikTok leaked more than 2 Billion of their user's records. We look at Chrome's urgent update to close its 6th 0-day of 2022 and at a worrisome "feature" — I think it a bug! — in Chrome. A somewhat hidden autorun facility in PyPI's pip tool used for downloading and installing Python packages is being used to run malware. And we examine a recent anti-Quantum computing opinion from an Oxford university quantum physicist. Then I have two bits of miscellany, three pieces of listener feedback, a fun SpinRite video discovery, and my discovery of a wonderful and blessedly prolific science fiction author. And after all that, we look at the result of Symantec's recent research into their discovery of more than 1800 mobile apps which they found to be leaking critical AWS cloud credentials, primarily due to carelessness in the use of today's software supply chain.



https://xkcd.com/2662/

Security News

Google's (newest) Open Source Software Vulnerability Rewards Program

Last week Google announced and launched a new Google-targeted Open Source Software Vulnerability Rewards Program. This is what they said. Yes, it's a little bit of a commercial, but it's still interesting and they are giving away money...

Today, we are launching Google's Open Source Software Vulnerability Rewards Program (OSS VRP) to reward discoveries of vulnerabilities in Google's open source projects. As the maintainer of major projects such as Golang, Angular, and Fuchsia, Google is among the largest contributors and users of open source in the world. With the addition of Google's OSS VRP to our family of Vulnerability Reward Programs (VRPs), researchers can now be rewarded for finding bugs that could potentially impact the entire open source ecosystem.

Google has been committed to supporting security researchers and bug hunters for over a decade. The original VRP program, established to compensate and thank those who help make Google's code more secure, was one of the first in the world and is now approaching its 12th anniversary. Over time, our VRP lineup has expanded to include programs focused on Chrome, Android, and other areas. Collectively, these programs have rewarded more than 13,000 submissions, totaling over $38M paid.

The addition of this new program addresses the ever more prevalent reality of rising supply chain compromises. Last year saw a 650% year-over-year increase in attacks targeting the open source supply chain, including headliner incidents like Codecov and the Log4j vulnerability that showed the destructive potential of a single open source vulnerability. Google's OSS VRP is part of our $10B commitment to improving cybersecurity, including securing the supply chain against these types of attacks for both Google's users and open source consumers worldwide.

Google's OSS VRP encourages researchers to report vulnerabilities with the greatest real, and potential, impact on open source software under the Google portfolio. The program focuses on:

- *All up-to-date versions of open source software (including repository settings) stored in the public repositories of Google-owned GitHub organizations (eg. Google, GoogleAPIs, GoogleCloudPlatform, …).*

- *Those projects' third-party dependencies (with prior notification to the affected dependency required before submission to Google's OSS VRP).*

[So that's nice. Not JUST Google project code, but any errors in anything Google's own project code depends upon.]

The top awards will go to vulnerabilities found in the most sensitive projects: Bazel, Angular, Golang, Protocol buffers, and Fuchsia. After the initial rollout we plan to expand this list. Be sure to check back to see what's been added.

To focus efforts on discoveries that have the greatest impact on the supply chain, we welcome submissions of:

- *Vulnerabilities that lead to supply chain compromise*
- *Design issues that cause product vulnerabilities*
- *Other security issues such as sensitive or leaked credentials, weak passwords, or insecure installations*

Depending on the severity of the vulnerability and the project's importance, rewards will range from $100 to $31,337. The larger amounts will also go to unusual or particularly interesting vulnerabilities, so creativity is encouraged.

Before you start, please see the program rules for more information about out-of-scope projects and vulnerabilities, then get hacking and let us know what you find. If your submission is particularly unusual, we'll reach out and work with you directly for triaging and response. In addition to a reward, you can receive public recognition for your contribution. You can also opt to donate your reward to charity at double the original amount.

Not sure whether a bug you've found is right for Google's OSS VRP? Don't worry, if needed, we'll route your submission to a different VRP that will give you the highest possible payout. We also encourage you to check out our Patch Rewards program, which rewards security improvements to Google's open source projects (for example, up to $20K for fuzzing integrations in OSS-Fuzz).

Google is proud to both support and be a part of the open source software community. Through our existing bug bounty programs, we've rewarded bug hunters from over 84 countries and look forward to increasing that number through this new VRP. The community has continuously surprised us with its creativity and determination, and we cannot wait to see what new bugs and discoveries you have in store. Together, we can help improve the security of the open source ecosystem.

Give it a try, and happy bug hunting!

https://bughunters.google.com/about/rules/6521337925468160/google-open-source-software-vulnerability-reward-program-rules

https://bughunters.google.com/about/rules/4928084514701312/patch-rewards-program-rules

What occurred to me as I was reading through this and including it in today's podcast is the degree to which bug bounties have become a part of today's modern software ecosystem. It's no longer a surprise for a company to be offering a bounty for the discovery and responsible reporting of problems. But it wasn't so long ago that this was unheard of, or that it was an enlightened exception. Today, it's the way large companies do business. If you can't find all of the important bugs yourself — and apparently no one can — you reward the white hats who do. And more significantly, as we've seen, this sort of bug hunting, while not guaranteeing a steady paycheck, could increasingly be considered a valid and workable career.

Did TikTok leak 2.05 BILLION User Records?

Did TikTok leak 2.05 BILLION User Records? TikTok says no, but other independent researchers are not so sure. Every week, while I'm looking through the past week's news, half of what I see are breach reports. This or that company reports that it was breached and bad guys may have obtained the data for typically a couple hundred thousand of their users. Okay, that's not good. But there's generally not much more to say about it. It's a bit like this or that company got hit by ransomware. Not good. We're sorry. Hope you recover.

But the potential exposure of **2.05 BILLION** user records by TikTok, well... We're talking about this one because of who it is and the size, scale and impact of the possible breach.

When news of this appeared last week, TikTok pressed their "Respond to the press" button and out popped a statement reading: *"TikTok prioritizes the privacy and security of our users' data. Our security team investigated these claims and found no evidence of a security breach."* (Then the button popped back out.)

But TikTok's canned-sounding denial follows reports of a hack that surfaced on the Breach Forums message board Saturday, with the threat actor noting that the server holds 2.05 billion records in a humongous 790GB database.

The hacking group known as BlueHornet (aka AgainstTheWest) tweeted: "Who would have thought that TikTok would decide to store all their internal backend source code on one Alibaba Cloud instance using a trashy password?"

Bob Diachenko, threat intelligence researcher at Security Discovery, said the breach is "real" and that the data is likely to have originated from "Hangzhou Julun Network Technology Co., Ltd rather than TikTok." But Troy Hunt isn't yet convinced. Troy Tweeted: "This is so far pretty inconclusive; some data matches production info, albeit publicly accessible info. Some data is junk, but it could be non-production or test data. It's a bit of a mixed bag so far."

I just checked both of their Twitter feeds for any update. Bob Diachenko's feed had two updates: The first was *"UPDATE: while there is definitely a breach, it is still work in progress to confirm the origin of data, could be a third party."* And that was followed up with *"UPDATE 2: OK, #TikTokBreach is real. Our team analyzed publicly exposed repos to confirm partial users data leak."*

But then Troy retweeted a tweet and added: *"The thread on the hacking forum with the samples of alleged TikTok data has been deleted and the user banned for "lying about data breaches"."* The group that was banned, as explained in the Tweet that Troy retweeted, was that same BlueHornet / AgainstTheWest who made the original allegation. So it appears that the whole thing was, as TikTok originally claimed, not a breach at all.

Though this story sort of cancels itself out, I wanted to share this because it's a great example of what is continually going on behind the scenes in the security industry. Not all players are on the up and up. Not everything that's Tweeted is factual (imagine that!!). And Troy, for whom this is certainly not his first rodeo, knows to remain skeptical until all the facts are in and proven.

An urgent Chrome update patches new 0-day flaw
Last Friday, Google bumped Chrome up to v105.0.5195.102 to urgently close a vulnerability that was being actively exploited in the wild.

As usual, little more is being said beyond the CVE of 2022-3075 which involves insufficient data validating in Mojo, which is a library of routine to provide platform-agnostic inter-process communication (IPC). Google credited an anonymous researcher with their report of the high-severity flaw on August 30, 2022. And I'm again impressed by the Chromium team's 3-day incident response.

So this is 0-day number six for the year and while it's not likely to be an emergency for everyone, everyone is advised to be sure they are running the version ending in .102. The update applies to Windows, macOS, and Linux desktop editions of Chrome. And as always, the users of the Chromium-based browsers Edge, Brave, Opera, and Vivaldi are advised to apply the fixes as and when they become available.

Permission-less Browser Clipboard Write
If you, right now, go to https://webplatform.news/ in Chrome or any of the Chromium-based web browsers, without your consent or permission, the following message will placed into your system's global clipboard:

> *Hello, this message is in your clipboard because you visited the website Web Platform News in a browser that allows websites to write to the clipboard without the user's permission. Sorry for the inconvenience. For more information about this issue, see* https://github.com/w3c/clipboard-apis/issues/182.

Hopefully, to our listening audience, it's needless to say, this is not safe!

And more than being unsafe, some consider it to be a major security issue. The problem is that the browser's interaction with the clipboard is somewhat tricky. And web designers have been tinkering around with it, considering the deliberate addition of some non-interactive access. But it seems pretty clear to me that a user should have to clearly highlight and mark something on a web page, then explicitly issue some form of "clipboard copy" command for their web browser to be given permission to modify their system's clipboard. It should always remain under the user's explicit control. The benefits of having a web page announce that something has already been placed on the user's clipboard, slick as that might seem to a web developer, offers bad guys far too much opportunity for carnage. And there appears to be some lack of clarity on this front.

Web developer Jeff Johnson said that what he's calling the "clipboard poisoning attack" was accidentally introduced in Chrome version 104. But looking over the pertinent Chromium discussion thread makes that less clear:

https://bugs.chromium.org/p/chromium/issues/detail?id=1334203

From Microsoft on Monday, Aug 29:

> *It's pri-3 because this behavior has been there since we shipped async clipboard APIs. It is not a regression. However, I agree that this should be fixed, and we should send a breaking change email to blink-dev to figure out the right process to add the transient user activation restriction to the APIs. I guess pri-1 makes sense since Firefox and Safari are also considering adding the transient user activation (instead of a user gesture requirement). See* https://github.com/w3c/clipboard-apis/issues/75#issuecomment-1088932790
> *Currently, I don't have cycles to work on this bug, so marking it as available.*

Unbelievable. That was Monday before last. And then last Tuesday:

From the Chromium group on Tue, Aug 30, 2022:

> *To be clear: READING from the clipboard always requires a permission (like geolocation, microphone, etc). To see what this looks like, check out this demo site:* https://async-clipboard-api.glitch.me/
>
> *WRITING plain text or images to the clipboard* ***can currently be accomplished*** *without a permission or user gesture, although the site and tab in question must be foregrounded. We are looking to tighten up the security model here.* [Gee, ya think?!]
>
> *Neither one of these behaviors has changed recently, nor does the new tab page test rely on permission-less, gesture-less clipboard access. (The New Tab Page is automatically granted permissions, as it is part of the browser rather than the web. The fact that it uses html/js is an implementation detail.)*

Jeff Johnson says:

> *While the problem exists in Apple Safari and Mozilla Firefox as well, what makes the issue more severe in Chrome is that the requirement for a user gesture to copy content to the clipboard is currently broken.* [Though we just saw the Chromium guy saying that it's working the way it should and it's **not** a regression.]
>
> *User gestures include selecting a piece of text and pressing Control+C (or ⌘-C for macOS) or selecting "Copy" from the context menu.*
>
> *Therefore, a gesture as innocent as clicking on a link or pressing the arrow key to scroll down the page gives the website permission to overwrite your system clipboard.*

So, Jeff appears to be saying that Safari and Firefox require the user to have some interaction with the page, though not necessarily a clipboard copy, whereas Chrome currently requires exactly no interaction of any kind. And the Chromium guys agree on that point, saying it's the way it's supposed to be.

The idea that the danger of this is not glaringly apparent to the web developers is a bit surprising. I would have the you-know-what scared out of me if I were to paste my machine's clipboard and find some random message in it talking to me that I did not explicitly copy there. I

would be **SURE** that my machine had been infected with malware of some kind that had messed with my clipboard. But right now, as it is, any webpage I visit in any Chromium-based browser (I tried it in Edge and it worked) can place anything they wish onto my system's clipboard at any time.

And aside from being annoying and wrong, the ability for any web page to replace my system's clipboard data has some obvious security implications. In a hypothetical attack scenario, an adversary could lure a victim to visit a rogue landing page and rewrite the address of a cryptocurrency wallet previously copied by the target with one under their control, resulting in a redirected funds transfer. Or a threat actor could overwrite the clipboard with a link to specially crafted websites, leading victims to download dangerous software. It's so clearly a very bad idea.

Johnson said: *"While you're navigating a web page, the page can without your knowledge erase the current contents of your system clipboard, which may have been valuable to you, and replace them with anything the page wants, which could be dangerous to you the next time you paste."* It's clearly outside of the scope of what any web page should be able to do. But what's frightening is that the Chrome guys appear to be saying: *"It's not a bug, it's a feature!"* Great, give me a way to turn it off!

Speaking of unwanted features...
PyPI is the Python language's popular Package Index repository. Python Package Index, thus PyPI. And in another discovery that could expose developers to increased risk of a supply chain attack, it was found that nearly one-third of the packages in PyPI trigger an automatic code execution upon downloading them.

As with auto-copying to a user's clipboard — if that's what the user wants — having a Python script auto-run after downloading a well designed and benign package can be a convenience. But a researcher at Checkmarx noted in a technical report they published last week that "A worrying feature in pip/PyPI allows code to automatically run when developers are merely downloading a package." He added that the feature is alarming because "a great deal of the malicious packages we are finding in the wild use this feature of code execution upon installation to achieve higher infection rates."

One of the ways by which packages can be installed for Python is by executing the "pip install" command, which, in turn, invokes a file called "setup.py" which comes bundled along with the module. "setup.py," as the name implies, is a setup script that's used to specify metadata associated with the package, including its dependencies. It provides some of the welcome automation that makes package management convenient.

And in what amounts to a documentation flaw, a cautious user might opt to use the safer appearing "pip download" command since its documentation states: "pip download does the same resolution and downloading as pip install, but instead of installing the dependencies, it collects the downloaded distributions into the directory provided (defaulting to the current directory)." In other words, the command can be used to download a Python package without having it installed on the system. But as it turns out, executing the download command also runs the embedded "setup.py" script, resulting in the execution of whatever malicious code it might contain.

It turns out that "setup.py" autorunning only occurs when the package contains a tar.gz file instead of a wheel (.whl) file.

Although pip defaults to using wheels instead of tar.gz files, attackers take advantage of this behavior to intentionally publish python packages without a .whl file, leading to the execution of the malicious code present in the setup script. The Checkmarx report noted that *"When a user downloads a python package from PyPi, pip will preferentially use the .whl file, but will fall back to the tar.gz file if the .whl file is not present."*

At the moment there's not much that Python users can do. If pip is used to either install or download a PyPI package, bad guys can arrange to run their "script.py" on the user's machine. Given the recent troubles with supply-chain attacks, that's a bit nerve wracking.

A Quantum Hype Bubble?

We've been talking about quantum computing recently. Even though the capability to break our current public key cryptographic security protocols remains purely theoretical, the existence of technology that could do so could render the asymmetric cryptography, which we depend upon to manage our symmetric keys, useless for that purpose. In practice, the security of anything and everything we currently protect with certificates, and the public handshakes we make to negotiate secret keys, would be open to circumvention and abuse. Astonishing to me as it is, although bombs are not flying through the air between hostile super-powers, there is clearly very active, continuous and slowly escalating cyberwarfare being conducted among and between the world's hostile super-powers. The only thing keeping this under control and at parity is that none of these super powers has meaningful superiority in cyberspace... or, as Leo would never say: "In Cyber."

If quantum computing's promise is realized, someone will have it first. And that someone will have a massive destabilizing power over the rest of the world. The degree to which we depend upon the stabilizing force of today's status quo cannot be overstated.

And it is for this reason that researchers in cryptography, armed with an understanding of what a future working quantum computer should be able to do, have already been at work for several years on the design and implementation of next-generation "post-quantum" replacement cryptography.

The website "Futurism" runs a column called "The Byte" and last Saturday's title was: *"Oxford physicist unloads on quantum computing industry, says it's basically a hype bubble."*

Now, it would be a full time job to know everything about what's going on in quantum computing. And I suppose that's this guy's job. And that's good because I'm already overbooked. At the same time, I don't know anything about what biases he might have. But I found the synopsis of what we wrote to be interesting and worth sharing with everyone here:

> *Oxford quantum physicist Nikita Gourianov tore into the quantum computing industry this week, comparing the "fanfare" around the tech to a financial bubble in a searing commentary*

piece for the Financial Times. [FT is behind a high paywall or I'd have gone to the source.] *In other words, he wrote, it's far more hype than substance. It's a scathing, but also perhaps insightful, analysis of a burgeoning field that, at the very least, still has a lot to prove.*

Despite billions of dollars being poured into quantum computing, Gourianov argues, the industry has yet to develop a single product that's actually capable of solving practical problems. That means these firms are collecting orders of magnitude more in financing than they're able to earn in actual revenue — a growing bubble that could eventually burst.

Gourianov wrote for the FT: "The little revenue they generate mostly comes from consulting missions aimed at teaching other companies about 'how quantum computers will help their business,' as opposed to genuinely harnessing any advantages that quantum computers have over classical computers."

While I think this is an interesting opinion from someone whom others apparently believe knows something about quantum physics, I'll just note that something is impossible, right up until the time that it isn't. This doesn't guarantee that that something will ever **not** be impossible. But when the stakes are as high as they are, this sort of tax-deductible research is easy to justify to a company's board.

Nikita Gourianov went on so say:

"Contemporary quantum computers are "so error-prone that any information one tries to process with them will almost instantly degenerate into noise," which scientists have been trying to overcome for years. He also took aim at other assumptions about the field, arguing that fears over quantum computers being able to crack even the most secure cryptographic schemes are overblown. And, notably, Gourianov's rant in the Financial Times comes just weeks after a group of researchers found that a conventional computer was able to rival Google's Sycamore quantum computer, undermining Google's 2019 claims of having achieved "quantum supremacy."

Recalling the sentiment "There's gold in them thar hills", despite the industry's lackluster results to date, investors are still funneling untold sums into quantum computing ventures. Yeah!! Because... what if?

*Gourianov wrote: "In essence, the quantum computing industry has yet to demonstrate **any** practical utility, despite the fanfare. Why then is so much money flowing in? Well, it is mainly due to the fanfare." The money, he argues, is coming from investors who typically don't have "any understanding of quantum physics," while "taking senior positions in companies and focusing solely on generating fanfare."*

In short, Gourianov believes it's only a matter of time until the "bubble will pop" and the "funding will dry up."

I wanted to share this presumably-informed perspective, since many tech media outlets covered it, because the Financial Times which carried it is highly regarded, because this Oxford University quantum physicist may know what he's talking about, because on some level it does appear to

fit the evidence, and because it serves as an interesting counterpoint to what does, indeed, despite huge expenditures, still seem to be quite a long ways off, if it is ever even practical. Will this quantum crypto panic ultimately turn out to have been misplaced? Maybe. But the stakes are clearly so high that a great deal of wealth is being transferred. If factoring the number 33 is considered a huge achievement, and if, as Gourianov appears to believe, this particular branch of quantum physics is **not** about to be visited by a breakthrough, then the crypto industry probably has time to get its post-quantum crypto right the first time. Let's do **that!**

Even if "quantum" never happens, the replacement of our aging quantum-**unsafe** crypto only makes sense. And since the replacement of everything we have now **will** take significant time, I'm very glad that we're already working to determine what that replacement will be. And it'll be interesting to see whether it is the replacement of pre-quantum crypto with post-quantum crypto that finally bursts the quantum hype bubble.

Miscellany

All of the BlackHat 2022 Presentation Slides PDFs:
BlackHat is somewhat notorious for being quite slow to release the materials after their conference. So an enterprising researcher has put them all up on Google Drive with open public sharing access. He Tweeted a link to the collection and I've captured it in this week's show notes. To make it easy for those who don't other want this week's show notes, I've also given it a GRC shortcut of "BH2022" so: https://grc.sc/bh2022
https://drive.google.com/drive/u/0/folders/1KHx2rKUEdb53flGUN0mFHRdSdB5PUT4B?fbclid=IwAR1C2Fk3XDPU-ky-4B57ZmKtEKgjB6Yg-9m2c6MTxyJd779yPV7MCCHCvWo&fs=e&s=c

Csurf NPM library mistake:
The NPM-hosted JavaScript library named "Csurf" is an JavaScript library designed to protect applications from Cross-Site Request Forgery attacks, known as CSRFs. Unfortunately, researchers at the security company Fortbridge discovered a CSRF vulnerability in Csurf. And unfortunately, the package apparently cannot be used to protect itself. Since the project's authors have apparently decided it's not worth repairing, they chose, instead, to mark Csurf as "vulnerable & deprecated." And I suppose, at the same time, as evidence of the need for it.

SpinRite

I didn't think I was going to have anything new to share about SpinRite today. Work is proceeding well. It's running and I almost have all of the obvious problems resolved. I may rearrange its real time activities screen to make better use of its real estate as a consequence of other things that have changed. But in any event, this morning I encountered a Twitter direct message from "Matt Foot / @glossywhite", one of our listeners with whom I exchange notes. He explained that he was searching for "How SpinRite works" and encountered a surprisingly recent 2021 YouTube video showing a full demo / walkthrough of SpinRite II running on a 20 megabyte Seagate ST-225 drive. I hadn't seen SpinRite II run in a long time, so I wound up watching the entire 13-minute, well-narrated video. The drive was initially in very bad shape, but SpinRite fixed it. For anyone who's interested, it's this week's shortcut of the week: https://grc.sc/887
And thanks, Matt!

Closing The Loop

Tyson Moore / @tysmo

Hi Steve, just listened to SN886. I have a different take on NIC LEDs you might find interesting:

I worked for a large Canadian telco that had disabled activity lights on almost all of their switches and routers, from small 1U 16-port access switches to 30U multi-terabit routers. On many devices, this was done through undocumented commands or special firmware.

Though it made troubleshooting difficult, the idea is that an adversary wouldn't be able to distinguish dormant or lightly-used links from busy ones, especially when faced with hundreds or thousands of options in a large telephone exchange. I was never fully sold on the usefulness of this measure, but as a defense-in-depth strategy I figure it couldn't hurt.

Via Twitter DM:

Hi Steve - I enjoyed your discussion of the LastPass breach. Whilst users' vaults are strongly encrypted and therefore presumably fairly safe, do you think there could be another risk, in that conceivably the hackers could have introduced a backdoor into the LP code? How can we know the code itself is still safe, given that hackers have had access to it for an unspecified amount of time?

I think that this is where the meaning of words matters. Since everyone is on the outside looking in, we don't know precisely what happened and I doubt we ever will. So we can choose to take their CEO's statement as fact or not. If we do choose to accept it as fact, then exactly what he said matters: *"We have determined that an unauthorized party gained access to portions of the LastPass development environment through a single compromised developer account and took portions of source code and some proprietary LastPass technical information."* Certainly "taking source code" is a world different from "may have modified source code." We must assume that they have well established source code controls in place, and that verifying the extent of the intrusion was their top priority. So again, if we choose to trust the CEO's statement, then there's no danger to us and we can hope that they will respond to this by making anything like this far less likely to occur again. And if we choose not to trust what the CEO said then nothing he said matters anyway.

Bill Crahen / @wcrahen (Saturday)

Hey Steve, just setup my Sandsara last night. Smaller than I imagined, but still fascinating to watch.

One of our listeners Tweeted:

Just listened to SN886. I've been in the foo@duck.com beta for about a week now. It has worked as advertised and as you described. I'm not seeing 85% of my email with trackers, more like 50-60%. It's equally surprising to me who is and who isn't using trackers.

Sci-Fi Discovery

"The Silver Ships" - http://scottjucha.com/silverships.html
Thanks to one of our listeners (whose Twitter handle is "Laforge129" @Laforge129 — Thank you! Thank you! Thank you! "Laforge") I am VERY excited to announce the discovery of a completely new and blessedly prolific science fiction author. This author's name is pronounced Scott "Ū•HĂ" which is spelled Jucha. His website domain is his name, so: scottjucha.com.

First of all, Scott can write, and he has a pleasantly expansive working vocabulary, which is a joy to encounter. He writes stories containing well-formed characters who, at every turn, do what you hope they're going to do, then surprise you as they exceed your expectations. Many of his Amazon reviewers, in their review after they have finished the final book in "The Silver Ships" series, state that this is the best "space opera" series they've ever read. It would take a lot to compete with Ryk Brown's "Frontiers Saga"... but there's certainly room for a top two!

I received "Laforge's" tweet a week ago, today. And since then I've read the first book. Then I needed to see what was about to happen next, so I read just enough of the second book to relieve the pressure. Then I went back and reread/skimmed the first third of the first book in order to relive its key parts, now knowing what the author had in mind. At one point my wife Lorrie asked where I was in my reading and I explained that I was re-reading book one. She just shook her head and said: "We are so different that way." So I explained/asked: "Haven't you ever seen a movie that was so good that you immediately watched it again, able to enjoy it even more, since you knew how everything fit together?" (That generated some more head shaking.)

Anyway, ten years ago, in 2012, Scott began writing. In April of last year he finished his 24-novel series, which he calls "The Silver Ships." All of the first five novels in the series were three times awarded Amazon's #1 Best Selling Sci-Fi book, and they also won the #2 spot twice across multiple science fiction categories of first contact, space flight, and alien invasion.

The story is set in the far future. It follows a number of Earth descended colonies which encounter a very non-human, quite powerful and quite hostile alien race. Ever since I caught up with Ryk Brown's Frontiers Saga series, I've been casting about, looking for something next, while Ryk continues working on his next books. I'm absolutely certain that I found what I've been looking for. I have no idea what's going to happen, but I **love** what has happened so far.

The books are all on Amazon, available through their Kindle Unlimited program, and also on Audible, narrated by Grover Gardner. Look for "The Silver Ships" series.

Embedding AWS Credentials

Last Thursday, Kevin Watkins, a security researcher with Symantec revealed the results of Symantec's sobering research into a previously unappreciated, or at least grossly under-appreciated, serious weakness and vulnerability created by the way today's increasingly powerful mobile applications are being developed. The problem surrounds the collision of the increasingly ubiquitous use of "The Cloud" by mobile apps with the merging of out-sourced code libraries and SDK's which use "The Cloud" to contain their often massive databases. Another problem appears to arise from a failure to follow the old adage which carries the well-known abbreviation: RTFM.

The data belonging to both users and the enterprises hosting these dangerous ill-designed apps are thus put at risk, as is the data of all of the customers of these enterprises. The problem is big. So I want to get specific and put a sharp point on this, fleshing it out by sharing what Symantec's research revealed:

They open with a punchline: *"Over three-quarters of the apps Symantec analyzed contained valid AWS access tokens that allowed access to private AWS cloud services."* That was 1,859 iOS and Android apps which were found to be leaking actionable AWS cloud credentials that MUST be kept private.

Here's how Symantec framed the problem and explained what they found:

Most of us, by now, have been impacted in some way by supply chain issues. An increase in the price of fuel and other items, delivery delays, and a lack of product availability are just some of the consequences of supply chain issues stemming from recent events around the world. However, in the context of software and technology infrastructure, the consequences resulting from supply chain issues are very different. Mobile apps, for example, can contain vulnerabilities introduced in the supply chain that can potentially lead to the exposure of sensitive information, which in turn could be used by threat actors for other attacks.

Mobile app supply chain vulnerabilities are often added by app developers, both knowingly and unknowingly, who are likely unaware of the downstream security impacts putting not only the app users' privacy at risk, but sometimes putting their company and employer's privacy and data at risk too.

In other words, this is what I would call the "modern modular software component assembly dilemma" where it's too easy to plug this library into that library and have this API calling into that API — and where everything just appears to work — but without the developers ever obtaining a full in-depth working understanding of exactly what's going on. And, if course, the whole point of using plug-in modular libraries and APIs is to not NEED to learn everything about what the serving library is doing. The trouble is, this is also the way implementation mistakes occur.

Similar to the supply chain for material goods, mobile application software development undergoes a process that includes: the collection of materials, such as software libraries and software development kit (SDKs); manufacturing or developing the mobile application; and

shipping the end result to the customer, often using mobile app stores. This research examined the type of upstream supply chain issues that can make their way into mobile apps, making them vulnerable. These issues include:

- *Mobile app developers unknowingly using vulnerable external software libraries and SDKs*
- *Companies outsourcing the development of their mobile apps, which then end up with vulnerabilities that put them at risk*
- *Companies, often larger ones, developing multiple apps across teams, using cross-team vulnerable libraries in their apps*

In order to better understand the prevalence and scope of these supply chain vulnerabilities, we took a look at publicly available apps in our global app collection that contained hard-coded Amazon Web Services (AWS) credentials. Hard-coded cloud credentials is a type of vulnerability we've been looking at for years and have extensively covered in the past.

This time, in order to get to the bottom of the supply chain impacts caused by this issue, we've looked into ***why*** *app developers hard-code cloud credentials inside apps; where the hard-coded credentials are located in the apps - tracking the sequence, or chain of events leading to the vulnerability; and finally, the size of the problem and its impact.*

We identified 1,859 publicly available apps, both Android and iOS, containing hard-coded AWS credentials. Almost all were iOS apps (98%) - a trend and difference between the platforms we've been tracking for years, possibly linked to different app store vetting practices and policies. In any case, we examined the scope and extent of the risks involved when AWS credentials were found embedded inside apps. We found the following:

- *Over three-quarters (77%) of the apps contained valid AWS access tokens allowing access to private AWS cloud services*
- *Close to half (47%) of those apps contained valid AWS tokens that also gave full access to numerous, often millions, of private files via the Amazon Simple Storage Service (Amazon S3)* [They get much more specific about this in a minute.]

We will explore the type of private data exposed in the examples discussed later in this blog, but the message is clear: ***apps with hard-coded AWS access tokens are vulnerable, active, and present a serious risk.***

I should explain that it would be entirely possible for apps NOT to embed static AWS access tokens into their code. How many times have we talked about the insanity of a Cisco router, for example, embedding some backdoor access username and password into its firmware where it's ripe for discovery? It's just supreme malpractice and laziness. In the case of well-connected mobile apps, it would be trivial to have apps reach out to obtain the AWS token on-the-fly over a secure encrypted and authenticated connection. That would have the added flexibility of allowing the app's developers to change AWS credentials on-the-fly if some access right problems, such as we'll be discussing shortly, were to be found.

In any event, Symantec continues...

We then looked into why and where exactly the AWS access tokens were inside the apps, and if they are found in other apps. We discovered that over half (53%) of the apps were using the same AWS access tokens found in other apps. Interestingly, these apps were often from different app developers and companies. This pointed to an upstream supply chain vulnerability, and that's exactly what we found. The AWS access tokens could be traced to a shared library, third-party SDK, or other shared component used in developing the apps.

*As for the remaining question of **why** app developers are using hard-coded access keys, we found the reasons to include:*

- *Downloading or uploading assets and resources required for the app, usually large media files, recordings, or images*
- *Accessing configuration files for the app and/or registering the device and collecting device information and storing it in the cloud*
- *Accessing cloud services that require authentication, such as translation services, for example*
- *No specific reason, dead code, and/or used for testing and never removed*

If an access key only has permission to access a specific cloud service or asset, for example accessing public image files from the corporate Amazon S3 service, the impact may be minimal. Some app developers may be assuming this is the case when they embed and use hard-coded AWS access tokens to access a single bucket or file in Amazon S3. The problem is often that the same AWS access token exposes all files and buckets in the Amazon S3 cloud, often corporate files, infrastructure files and components, database backups, etc. Not to mention cloud services beyond Amazon S3 that are accessible using the same AWS access token.

Imagine a business-to-business (B2B) company providing access to its service using a third-party SDK and embedding an AWS hard-coded access key, exposing not only the private data of the app using the third-party SDK, but also the private data of all apps using the third-party component. Unfortunately, this is not an uncommon occurrence, as you can see in the following case study examples.

Intranet platform SDK

We found a B2B company providing an intranet and communication platform had also provided a mobile SDK that its customers could use to access the platform. Unfortunately, the SDK also contained the B2B company's cloud infrastructure keys, exposing all of its customers' private data on the B2B company's platform. Their customers' corporate data, financial records, and employees' private data was exposed. All the files the company used on its intranet for over 15,000 medium-to-large-sized companies were also exposed.

Why did the company hard-code the AWS access token? In order to access the AWS translation service. Instead of limiting the hard-coded access token for use with the translation cloud service, anyone with the token had full unfettered access to all the B2B company's AWS cloud services.

Digital identity and authentication

We discovered several popular banking apps on iOS that rely on the same vulnerable third-party AI Digital Identity SDK. Outsourcing the digital identity and authentication component of an app is a common development pattern as the complexities of providing different forms of authentication, maintaining the secure infrastructure, and accessing and managing the identities can incur a high cost and requires expertise in order to do it right. Unfortunately, in this case, things were not done right.

Embedded in the SDK were cloud credentials that could place entire infrastructures at risk. The credentials could expose private authentication data and keys belonging to every banking and financial app using the SDK. Furthermore, users' biometric digital fingerprints used for authentication, along with users' personal data (names, dates of birth, etc.), were exposed in the cloud.

In addition, the access key exposed the infrastructure server and blueprints, including the API source code and AI models, used for the whole operation. In total, over 300,000 biometric digital fingerprints were leaked across five mobile banking apps using the SDK.

Online gaming technology platform

Often already established companies rely on outsourcing, or partnering, with other B2B companies for their digital and online services. This allows them to quickly move their brand online without having to build and support the underlying technology platform. At the same time, by relying on the outsourced company to run the technology platform, they often have to give exclusive access to their business data. Furthermore, they have to trust that the outsourced company will protect the online private data, not to mention the reputation of the brand overall.

We found a large hospitality and entertainment company depending on another company for their technology platform, even forming a sports betting joint venture with the company.

With a highly regulated sports betting market the complexities of building and supporting infrastructure for online gambling cannot be understated. Unfortunately, by giving the joint venture company exclusive access to that part of its business, the company also exposed its gaming operations, business data, and customer data to the world.

In total, 16 different online gambling apps using the vulnerable library exposed full infrastructure and cloud services across all AWS cloud services with full read/write root account credentials.

All of the organizations whose vulnerable apps were discussed in these case studies have been notified about the issues we uncovered.

Avoiding supply chain issues

Protecting yourself from these types of supply chain issues is possible. Adding security scanning solutions to the app development lifecycle and, if using an outsourced provider, requiring and reviewing Mobile App Report Cards, which can identify any unwanted app behaviors or vulnerabilities for every release of a mobile app, can all be helpful in highlighting potential issues. As an app developer, look for a report card that both scans SDKs and frameworks in your application and identifies the source of any vulnerabilities or unwanted behaviors.