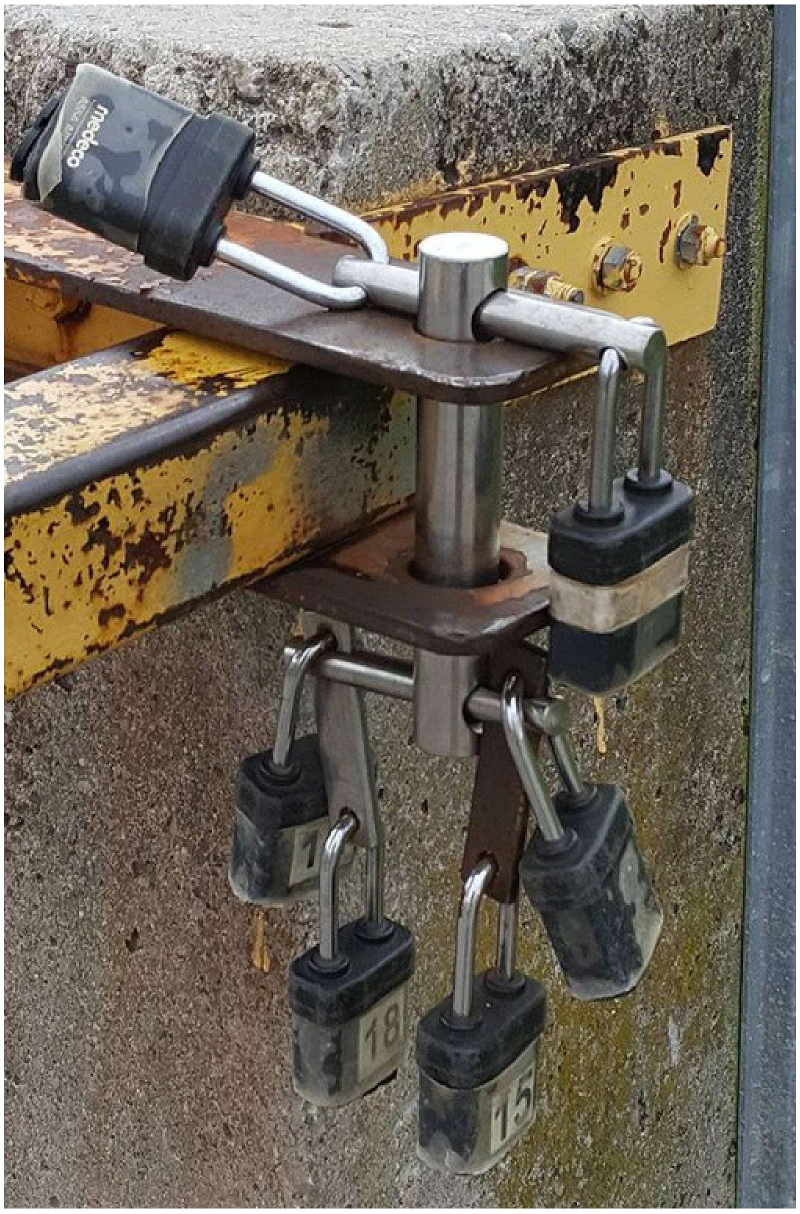
Security Now! #879 - 07-12-22
The Rolling Pwn

This week on Security Now!

This week we look at a recently made and corrected mistake in the super-important OpenSSL crypto library. The NIST has settled upon the first four of eight post-quantum crypto algorithms. Yubico stepped-up to help Ukraine. Apple has added an extreme "Lockdown Mode" to their devices. Microsoft unbelievably re-enables Office VBA macors received from the Internet. The FBI creates a successful encrypted message app for a major sting operation. We close the loop with some of our listeners. Then we examine an even more egregious case of remote automotive wireless unlocking and engine starting.

A Wonderful Mechanical "OR" Gate....



Security News

OpenSSL version 3.0.4 introduced a serious vulnerability which version 3.0.5 just repaired.
A potential remote code execution flaw was recently discovered in an update to the v3 branch of OpenSSL.

The issue was found in OpenSSL version 3.0.4, which was released late last month on June 21, 2022. It impacts x64 systems with the AVX-512 instruction set extension. Not affected are OpenSSL v1.1.1 as well as its BoringSSL and LibreSSL forks. The trouble stems from a remote memory-corruption vulnerability. The Advanced Vector Extensions (AVX) are extensions to the x86 instruction set architecture for microprocessors from Intel and AMD.

There was some back and forth about this in the GitHub issue thread about this. The OpenSSL Foundation's Tomáš Mráz said: "I do not think this is a security vulnerability, it is just a serious bug making the 3.0.4 release unusable on AVX-512 capable machines."

A security researcher, Guido Vranken, said it "can be triggered trivially by an attacker."

Alex Gaynor wrote: "I'm not sure I understand how it's not a security vulnerability. It's a heap buffer overflow that's triggerable by things like RSA signatures, which can easily happen in remote contexts (e.g. a TLS handshake)."

And the postgrad PhD student who originally discovered and reported the bug chimed in, stating that although "I think we shouldn't mark a bug as 'security vulnerability' unless we have some evidence showing it can (or at least, may) be exploited," it's necessary to release version 3.0.5 as soon as possible given the severity of the issue.

... Which is what soon after happened. The issue has been assigned CVE-2022-2274, described as a case of heap memory corruption within RSA private key operations. The advisory notes that: "SSL/TLS servers or other servers using 2048 bit RSA private keys running on machines supporting AVX512IFMA instructions of the X86_64 architecture are affected by this issue."

Calling it a "serious bug in the RSA implementation," (but still apparently not wishing to call it a vulnerability) the maintainers said the flaw could lead to memory corruption during computation that could be weaponized by an attacker to trigger remote code execution on the machine performing the computation. That sure does smack of a security vulnerability. If it quacks like a duck...

Anyway, the flaw has been patched and all users of OpenSSL v3 are urged to move to 3.0.5 especially is you had diligently moved to 3.0.4 which is the buggy release.

NIST Announces First Four Quantum-Resistant Cryptographic Algorithms

Last Tuesday, the US NIST — National Institute of Standards and Technology — announced that the results of the 6-year competition among a set of post-quantum algorithms had resulted in the selection of four initial algorithms. After editing out the various self-congratulatory statements from various bureaucrats who have no clue what this is all about and who certainly didn't write what they are quoted as saying, here's what NIST said:

The U.S. Department of Commerce's National Institute of Standards and Technology (NIST) has chosen the first group of encryption tools that are designed to withstand the assault of a future quantum computer, which could potentially crack the security used to protect privacy in the digital systems we rely on every day — such as online banking and email software. The four selected encryption algorithms will become part of NIST's post-quantum cryptographic standard, expected to be finalized in about two years.

The announcement follows a six-year effort managed by NIST, which in 2016 called upon the world's cryptographers to devise and then vet encryption methods that could resist an attack from a future quantum computer that is more powerful than the comparatively limited machines available today. Today's selection constitutes the beginning of the finale of the agency's post-quantum cryptography standardization project.

Four additional algorithms are under consideration for inclusion in the standard, and NIST plans to announce the finalists from that round at a future date. NIST is announcing its choices in two stages because of the need for a robust variety of defense tools. As cryptographers have recognized from the beginning of NIST's effort, there are different systems and tasks that use encryption, and a useful standard would offer solutions designed for different situations, use varied approaches for encryption, and offer more than one algorithm for each use case in the event one proves vulnerable.

That's all very smart. We're beginning to understand how to do these sorts of things. Explaining this for the masses, NIST added:

Encryption uses math to protect sensitive electronic information, including the secure websites we surf and the emails we send. Widely used public-key encryption systems, which rely on math problems that even the fastest conventional computers find intractable, ensure these websites and messages are inaccessible to unwelcome third parties. However, a sufficiently capable quantum computer, which would be based on different technology than the conventional computers we have today, could solve these math problems quickly to defeat today's encryption systems. To counter this threat, the four quantum-resistant algorithms rely on math problems that both conventional and quantum computers should have difficulty solving, thereby defending privacy both now and down the road.

The algorithms are designed for two main tasks for which encryption is typically used: general encryption, used to protect information exchanged across a public network; and digital signatures, used for identity authentication. All four of the algorithms were created by experts collaborating from multiple countries and institutions.

To provide of general encryption, NIST has selected the **CRYSTALS-Kyber** algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation. For digital signatures, NIST has selected the three algorithms: **CRYSTALS-Dilithium**, **FALCON** and **SPHINCS+** (read as "Sphincs plus").

Reviewers noted the high efficiency of the first two, and NIST recommends CRYSTALS-Dilithium to be used as the primary algorithm, with FALCON for applications that need smaller signatures than Dilithium can provide. The third, SPHINCS+, is somewhat larger and slower than the other two, but it is valuable as a backup for the valuable reason that it is based on a different math approach than all three of NIST's other selections. So, again, we've learned that where crypto is concerned there's nothing wrong with using a belt and suspenders.

The first three of the four selected algorithms are based on a family of math problems called structured lattices (which is why the word "CRYSTAL" appears as part of the names of the first two), while SPHINCS+ uses hash functions. The next four algorithms to be chosen, which are still being considered, are designed for general encryption and do not use structured lattices or hash functions in their approaches. NIST wrote:

While the standard is in development, NIST encourages security experts to explore the new algorithms and consider how their applications will use them, but not to bake them into their systems yet, as the algorithms could change slightly before the standard is finalized.

To prepare, users can inventory their systems for applications that use public-key cryptography, which will need to be replaced before cryptographically relevant quantum computers appear. They can also alert their IT departments and vendors about the upcoming change. To get involved in developing guidance for migrating to post-quantum cryptography, see NIST's National Cybersecurity Center of Excellence project page.

All of the algorithms are available on the NIST website.

Now, I obviously don't have any problem with the idea of adopting an advanced post-quantum cryptography system that is named "Dilithium crystals." But in scanning through all of the candidate entries from which these four winners were selected, I'll admit to breathing a sigh of relief when I saw that the algorithm named "Frodo" had not won.

Yubico donated 30,000 Yubikeys to Ukraine

To help Ukraine hold-off Russia's cyber attacks, Yubico donated 30,000 FIDO Yubikeys. So far, more than half of those 30,000, around 16,000 Yubikeys have been deployed to Ukrainian government executives, workers, and employees of private companies in Ukraine's critical sectors.

The initiative is being coordinated by a company named "Hideez", a Ukrainian security firm specializing in identity services and FIDO consultancy. Earlier this spring, Hideez secured a donation of 30,000 Yubikeys from Yubico — nice going Stina! Since then, Hideez's staff has been working with Ukrainian government agencies like the Ministry of Digital Transformation, the National Security and Defense Council, and the State Service of Special Communications and Information Protection of Ukraine (SSSCIP) to ensure the devices can be imported into the country, that government infrastructure is prepared for the keys' rollout, and that recipients receive the necessary training.

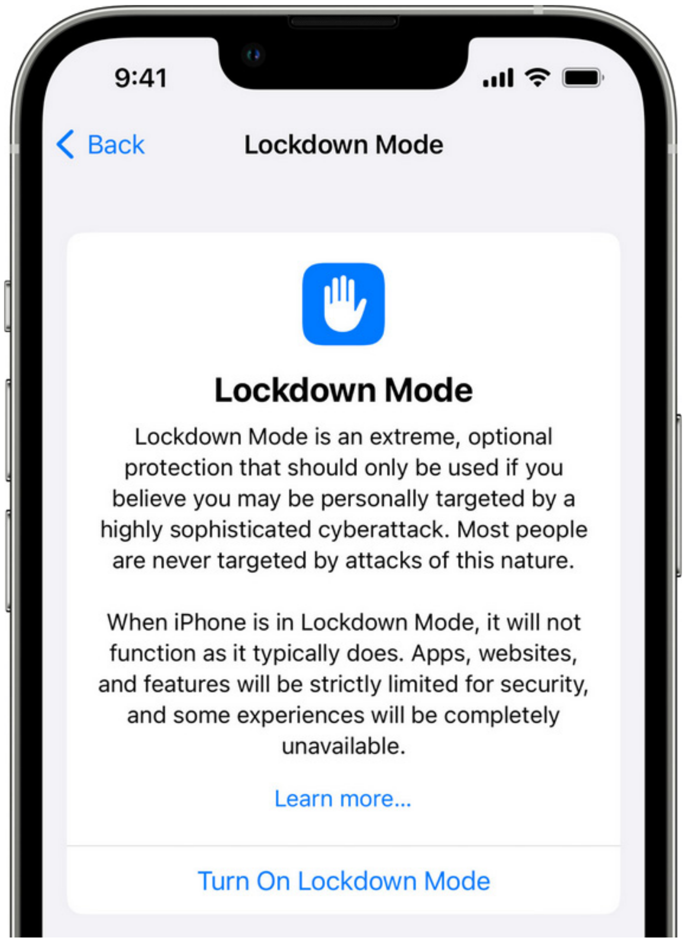
The idea is that once government and critical sector workers have a security key as an extra layer of protection, their accounts would be safe from the onslaught of constant spear-phishing attacks constantly hitting their inboxes every day.

Yuriy Ackermann, VP of War Efforts at Hideez told the publication Ricky Business News that *"We got Yubikey certified, so they are allowed to be deployed into Ukraine instances."* He said: *"We have quite a few ministries that have moved a lot of their stuff to G Suite and Azure—with them is quite easy; we just give them a key. We made instructions in Ukrainian, video instructions, and so on. [...] So, it's really fast. We had a department that pretty much moved to using FIDO, like 500 people, in less than a week because they just needed to understand their policies, read our documentation, and that's it, they just give the keys and roll them, and voila."*

Meanwhile, efforts are also underway to roll out the keys to individuals in other departments, including those without the proper server-side infrastructure. In these cases, Ackermann says Hideez has been providing the government with the company's solution at minimal costs.

Anyway, I just happened upon that nice bit of news and wanted to acknowledge what Yubico had done.

Apple's new extreme "Lockdown Mode"
In a blog posting last Wednesday, Apple took the wraps off of "Lockdown Mode" which will be rolled out later this year in macOS Ventura, iOS 16, and iPadOS 16. This new optional mode will severely restrict some features with the aim of protecting highly-targeted individuals such as human rights workers and researchers by reducing their devices' available attack surface.



Lockdown Mode

Lockdown Mode is an extreme, optional protection that should only be used if you believe you may be personally targeted by a highly sophisticated cyberattack. Most people are never targeted by attacks of this nature.

When iPhone is in Lockdown Mode, it will not function as it typically does. Apps, websites, and features will be strictly limited for security, and some experiences will be completely unavailable.

Learn more...

Turn On Lockdown Mode

Apple's announcement included a screen shot (above) of the Lockdown Mode enablement screen which sounds so ominous that it's unclear who would want to turn it on... (It reads...)

As Apple put it in their announcement:

Apple expands industry-leading commitment to protect users from highly targeted mercenary spyware. Apple is previewing a groundbreaking security capability that offers specialized additional protection to users who may be at risk of highly targeted cyberattacks from private companies developing state-sponsored mercenary spyware.

Apple today detailed two initiatives to help protect users who may be personally targeted by some of the most sophisticated digital threats, such as those from private companies developing state-sponsored mercenary spyware. Lockdown Mode — the first major capability of its kind, coming this fall with iOS 16, iPadOS 16, and macOS Ventura — is an extreme, optional protection for the very small number of users who face grave, targeted threats to their digital security. Apple also shared details about the $10 million cybersecurity grant it announced last November to support civil society organizations that conduct mercenary spyware threat research and advocacy.

Apple's head of Security Engineering and Architecture was quoted: "Apple makes the most secure mobile devices on the market. Lockdown Mode is a groundbreaking capability that reflects our unwavering commitment to protecting users from even the rarest, most sophisticated attacks. While the vast majority of users will never be the victims of highly targeted cyberattacks, we will work tirelessly to protect the small number of users who are. That includes continuing to design defenses specifically for these users, as well as supporting researchers and organizations around the world doing critically important work in exposing mercenary companies that create these digital attacks."

Lockdown Mode offers an extreme, optional level of security for the very few users who, because of who they are or what they do, may be personally targeted by some of the most sophisticated digital threats, such as those from NSO Group and other private companies developing state-sponsored mercenary spyware. Turning on Lockdown Mode in iOS 16, iPadOS 16, and macOS Ventura further hardens device defenses and strictly limits certain functionalities, sharply reducing the attack surface that potentially could be exploited by highly targeted mercenary spyware. At launch, Lockdown Mode includes the following protections:

- *Messages: Most message attachment types other than images are blocked. Some features, like link previews, are disabled.*
- *Web browsing: Certain complex web technologies, like just-in-time (JIT) JavaScript compilation, are disabled unless the user excludes a trusted site from Lockdown Mode.*
- *Apple services: Incoming invitations and service requests, including FaceTime calls, are blocked if the user has not previously sent the initiator a call or request.*
- *Wired connections with a computer or accessory are blocked when iPhone is locked.*
- *Configuration profiles cannot be installed, and the device cannot enroll into mobile device management (MDM), while Lockdown Mode is turned on.*

To my eye, those all sound like very useful and sane restrictions. They would not hugely impact even most users, while they would very clearly and significantly restrict the device's attack surface. So I say "Bravo, Apple!" Nice going. I'm sure that they have looked closely at the history of the way their devices have been compromised and have taken steps to address future threats in a way that will keep their device's useful and useable, while being FAR less easily compromised. Bravo.

Microsoft to re-enable Office Macros
And while we're talking about reducing attack surfaces, let's look at yesterday's big news surrounding Microsoft's announcement that they will soon be deliberately **increasing** the attack surface of all of their Microsoft Office documents by turning VBA Macros back on by default!

And, no, it's not April 1st. Microsoft Office VBA Macro abuse is something we've endured for so many years. So, yes, it was great news when, in February, we talked about Microsoft's final decision to just turn that feature off. Recall the sample UI bars we showed where the way things were it was SO EASY for an unwitting user to click the button to allow the macro to run? And how much better it was going to be? Well... apparently we can't have nice things, or secure software, because Microsoft has decided — while refusing to explain why — to turn VBA macros back on in Office. And no, again, I'm not kidding.

I loved Sophos' coverage of this yesterday, with their headline: *"That didn't last! Microsoft turns off the Office security it just turned on"*. I'm going to share what Paul Ducklin wrote at Sophos because it's so perfect. He wrote:

Remember 1999? Well, the Melissa virus just called, and it's finding life tough in 2022. It's demanding a return to the freewheeling days of the last millennium, when Office macro viruses didn't face the trials and tribulations that they do today.

In the 1990s, you could insert VBA (Visual Basic for Applications) macro code into documents at will, email them to people, or ask them to download them from a website somewhere…
…and then you could just totally take over their computer!

In fact, it was even better/worse than that: If you created a macro subroutine with a name that mirrored one of the common menu items, such as FileSave or FilePrint, then your code would magically and invisibly be invoked whenever the user activated that option.

Worse still, if you gave your macro a name like AutoOpen, then it would run every time the document was opened, even if the user only wanted to look at it. And if you installed your macros into a central repository known as the global template, your macros would automatically apply all the time.

Worst of all, perhaps, an infected document could implant macros into the global template, thus infecting the computer, and the same macros (when they detected they were running from the global template but the document you just opened was uninfected) could copy themselves back out again.

That led to regular “perfect storms” of fast-spreading and long-running macro virus outbreaks.

Simply put, once you’d opened one infected document on your computer, every document you opened or created thereafter would (or could, at least) get infected as well, until you had nothing but infected Office files everywhere.

As you can imagine, at that point in the game, any file you sent to or shared with a colleague, customer, prospector, investor, supplier, friend, enemy, journalist, random member of the public… …would contain a fully-functional copy of the virus, ready to do its best to infect them when they opened it, assuming they weren’t infected already.

And if that wasn’t enough on its own, Office macro malware could deliberately distribute itself, instead of waiting for you to send a copy to someone else, by reading your email address book and sending itself to some, many, or all of the names in there.

The first macro malware, which spread by means of infected Word files, appeared in late 1995 and was dubbed Concept, because at that time it was little more than a proof-of-concept.

But it quickly became obvious that malicious macros were going to be more than just a passing headache. Microsoft was slow to come to the cybersecurity party, carefully avoiding terms such as virus, worm, Trojan Horse and malware, resolutely referring to the Concept virus as nothing more than a “prank macro”. Over the years, however, Microsoft gradually implemented a series of functional changes in Office, by variously:

- Making it easier and quicker to detect whether a file was a pure document, thus swiftly differentiating pure document files, and template files with macro code inside. In the early days of macro viruses, back when computers were much slower than today, significant and time-consuming malware-like scanning was needed on every document file just to figure out if it needed scanning for malware.

- Making it harder for template macros to copy themselves out into uninfected files. Unfortunately, although this helped to kill off self-spreading macro viruses, it didn’t prevent macro malware in general. Criminals could still create their own booby-trapped files up front and send them individually to each potential victim, just as they do today, without relying on self-replication to spread further.

- Popping up a ‘dangerous content’ warning so that macros couldn’t easily run by mistake. As useful as this feature is, because macros don’t run until you choose to allow them, crooks have learned how to defeat it. They typically add content to the document that helpfully “explains” which button to press, often providing a handy graphical arrow pointing at it, and giving a believable reason that disguises the security risk involved.

- Adding Group Policy settings for stricter macro controls on company networks. For example, administrators can block macros altogether in Office files that came from outside the network, so that users can’t click to allow macros to run in files received via email or downloaded the web, even if they want to.

At last, in February 2022, Microsoft announced, to sighs of collective relief from the cybersecurity community, that it was planning to turn on the **"inhibit macros in documents that arrived from the internet"** by default, for everyone, all the time.

The security option that used to require Group Policy intervention was finally adopted as a default setting. In other words, as a business you were still free to use the power of VBA to automate your internal handling of official documents, but you wouldn't (unless you went out of your way to permit it) be exposed to potentially unknown, untrusted and unwanted macros that weren't from an approved, internal source.

Yay!!!! As this podcast also celebrated, Microsoft described the change by saying:

VBA macros obtained from the internet will now be blocked by default.

For macros in files obtained from the internet, users will no longer be able to enable content with a click of a button. A message bar will appear for users notifying them with a button to learn more. The default is more secure and is expected to keep more users safe including home users and information workers in managed organizations.

Sophos was enthusiastic too, though somewhat less so than I was at the time. They wrote:

We're delighted to see this change coming, but it's nevertheless only a small security step for Office users, because: VBA will still be fully supported, and you will still be able to save documents from email or your browser and then open them locally; the changes won't reach older versions of Office for months, or perhaps years, [given that] change dates for Office 2021 and earlier haven't even been announced yet; mobile and Mac users won't be getting this change; and not all Office components are included. Apparently, only Access, Excel, PowerPoint, Visio, and Word will be getting this new setting.

I still see what Microsoft chose to do as a huge step in the right direction since the Tyranny of the Default is SO powerful. So doing anything to help unwitting users not hurt themselves is good. Of course I also agree with Sophos that the whole Macro capability is inherently fraught with danger. But it was all put in place, as Paul wrote, back in the 1990's because it would have been reasonable to consider the future dangers of the Internet. (Though Microsoft sure could have reacted faster once those dangers became as real and obvious as they are today.)

So this all seemed like progress until Microsoft changed direction, writing:

Following user feedback, we have rolled back this change temporarily while we make some additional changes to enhance usability. This is a temporary change, and we are fully committed to making the default change for all users. Regardless of the default setting, customers can block internet macros through the Group Policy settings described in the article Block macros from running in Office files from the Internet. We will provide additional details on timeline in the upcoming weeks.

Again, Microsoft is not telling us why. “*Following user feedback*” What the heck does that mean? If there’s something mission critical that needs Office VBA Macros enabled from the Internet, I’d look into fixing **that** rather than to get Microsoft to lower the boom on the rest of the world.

This Is the Code the FBI Used to Wiretap the World
https://www.vice.com/en/article/v7veg8/anom-app-source-code-operation-trojan-shield-an0m

Last Thursday, Motherboard published an interesting story under the headline *“This Is the Code the FBI Used to Wiretap the World.”* They followed-up that opening with the subheading: *“Motherboard is publishing parts of the code for the Anom encrypted messaging app, which was secretly managed by the FBI in order to monitor organized crime on a global scale.”*

What I thought was interesting was that the approach that Motherboard says the FBI took to pull this off was precisely the solution I have often hypothesized as being the somewhat obvious way in which an end-to-end multi-party messaging system could be compromised. Here's how Motherboard's story begins:

The FBI operation in which the agency intercepted messages from thousands of encrypted phones around the world was powered by cobbled together code. Motherboard has obtained that code and is now publishing sections of it that show how the FBI was able to create its honeypot. The code shows that the messages were secretly duplicated and sent to a “ghost” contact that was hidden from the users’ contact lists. This ghost user, in a way, was the FBI and its law enforcement partners, reading over the shoulder of organized criminals as they talked to each other.

Our listeners will recall that this has been my greatest criticism of any supposedly private and secure system where a user's keys are being, in any way, managed **for** them. The reason that Threema’s approach has always appealed to me is that the user is 100% responsible for their own key management.) As we've often observed, if you're not managing your own keys, someone or something is managing them for you because the one thing any secure and private instant messaging system needs is keys. The point being, key management must be occurring somewhere. So if it's not something you're doing for yourself then you don't have any direct control over what's going on. Now, this is not to say that I think that people **should** be doing their own key management. When today’s podcast is finished, I’ll shoot an iMessage to my wife, Lorrie, to let her know that I’m heading home. My point is, top-level state secrets are not being exchanged. And the fact is, when you get right down to it, no consumer smartphone can really be trusted absolutely. But again, most people don’t need that level of secrecy.

Motherboard continues:

Last year, the FBI and its international partners announced Operation Trojan Shield, in which the FBI secretly ran an encrypted phone company called Anom for years and used it to hoover up tens of millions of messages from Anom users. Anom was marketed to criminals, and ended up in the hands of over 300 criminal syndicates worldwide. The landmark operation has led to more than 1,000 arrests including alleged top tier drug traffickers and massive seizures of weapons, cash, narcotics, and luxury cars.

So, the FBI mounted a good old fashioned sting operation. Good going, guys.

But Motherboard doesn't sound very impressed by the FBI's coders. They write:

Motherboard has obtained the underlying code of the Anom app and is now publishing sections of it due to the public interest in understanding how law enforcement agencies are tackling the so-called Going Dark problem, where criminals use encryption to keep their communications out of the hands of the authorities. [I'm unconvinced that there's any true public interest here, but okay...] *The code provides greater insight into the hurried nature of its development, the freely available online tools that Anom's developers copied for their own purposes, and how the relevant section of code copied the messages as part of one of the largest law enforcement operations ever.*

*The app uses XMPP to communicate, a long-established protocol for sending instant messages. On top of that, Anom wrapped messages in a layer of encryption. XMPP works by having each contact use a handle that in some way looks like an email address. For Anom, these included an XMPP account for the customer support channel that Anom users could contact. Another of these... was "**bot**".*

*Unlike the support channel, "**bot**" hid itself from Anom users' contact lists and operated in the background, according to the code and to photos of active Anom devices obtained by Motherboard. In practice the app scrolled through the user's list of contacts, and when it came across the **bot** account, the app filtered that out and removed it from view.*

*That finding is corroborated by law enforcement files Motherboard obtained which say that **bot** was a hidden or "ghost" contact that made copies of Anom users' messages.*

Authorities have previously floated the idea of using a ghost contact to penetrate encrypted communications. In a November 2018 piece published on Lawfare, Ian Levy and Crispin Robinson, two senior officials from UK intelligence agency GCHQ, wrote that "It's relatively easy for a service provider to silently add a law enforcement participant to a group chat or call," and "You end up with everything still being end-to-end encrypted, but there's an extra 'end' on this particular communication."

[If I may diverge for a moment, this is a perfect example of why I question the value of software and other patent classes. Their solution is the same as mine. I first mentioned this approach before they did. But I'm reasonably sure that they didn't get the idea from me… or that at least they didn't need to, because that's the obvious way to solve the problem. I suppose there is some dividing line between true brilliant invention and simple well-trained and experienced engineering. But finding that line lies well beyond the capabilities of a bureaucracy like the US patent and trademark office, or some random judge in East Texas.]

*The code also shows that in the section that handles sending messages, the app attached location information to any message that is sent to **bot**. On top of that, the AndroidManifest.xml file in the app, which shows what permissions an app accesses, includes the permission for "ACCESS_FINE_LOCATION." This confirms what Motherboard previously* *reported after reviewing thousands of pages of police files in an Anom-related investigation. Many of the intercepted Anom messages in those documents included the precise GPS location of the device at the time the message was sent.*

Motherboard concluded their story by noting that Operation Trojan Shield had been widely successful. And that on top of the wave of 1000's of arrests, authorities were also able to intervene using the intercepted and cloned messages to stop multiple planned murders.

Using a well established open protocol and open-source software allowed the application to be assembled without excessive cost and it got the job done.

Closing The Loop

Mark Thoms / @DyNoMiteT

@SGgrc How should I vet client-side third-party .exe's I use as part of my web application? How could I verify it's not malicious? Example: NeoDynamic.com JSPrintManager (utility to allow silent printing from a website). Thanks, religious SN Listener.

https://www.virustotal.com/gui/home/upload

Isaiah / @boosted37

@SGgrc you often recommend a separate IoT network from the main one. However, devices like Chromecasts require your phone or tablet to be on the same network as the streaming stick to manage a show. How would you recommend separating those from your main network?

Tom Terrific Krauska / @tomterrific1947

You mentioned that you used an Asus router that you really like. Which model is? It's time for me to buy a new router and I always take your advice. Thanks - Tom

[RT-AC68U (V3) - Purchased Oct 10, 2017 - ASUS AC1900 WiFi Gaming Router (RT-AC68U) - Dual Band Gigabit Wireless Internet Router, Gaming & Streaming, AiMesh Compatible, Included Lifetime Internet Security, Adaptive QoS, Parental Control]

Michael Horowitz / Defensive Computing - @defensivecomput

About your Asus router - are you sure the networks are isolated? You may find all the Guest SSIDs share the same subnet. I no longer have an Asus router so can't verify.

https://www.michaelhorowitz.com/
https://defensivecomputingchecklist.com/

When I went over to Michael's Defensive Computing Checklist (.com) I encountered a note that he'll be giving a presentation on Defensive Computing at the HOPE conference in New York City eleven days from today. It's actually called "A New HOPE" where HOPE stands for Hacker On Planet Earth:

NOTE: I will be giving a presentation on Defensive Computing at the HOPE conference in New York City in July 2022. The conference runs from July 22nd thru the 24th, I am scheduled for the 23rd at 1PM ET. Attending in person costs $200 for all three days. You can also stream the entire conference live for $99. More about the talk here.

https://defensivecomputingchecklist.com/HOPEconference.php

But regardless, allow me to urge our listeners to check out Michael's Defensive Computing Checklist at defensivecomputingchecklist.com. It's chock full of really good security advice. I've made it this week's short of the week: https://grc.sc/879

Simon Dearn / @dearn_simon

There are obviously good reasons for a company to use a VPN for allowing staff to connect remotely, but what are your thoughts on domestic use. Do you use a VPN yourself?

I don't. But that's only because during my current daily life I'm never needing to get online using OPNs. "OPNs" is a handy abbreviation everyone should keep in mind. It stands for "Other People's Networks." If my life involved travel, so that I was wanting to be online in airports, in coffee houses and hotels, then there is no question that a VPN would be the only way I would feel comfortable getting online when I was not at home. This is much less of a problem today, where everything is encrypted over TLS with server-authenticating certificates. Back at the start of this podcast, everything was being done over HTTP with a brief switch to TLS only when login forms and credentials were being exchanged. Remember when we used to explain that it was important to verify that the URL of a form's submit button was HTTPS and not just HTTP? Fortunately, we survived that. But it's still true that without universal certificate pinning, which I don't really see ever happening, or DNSSEC being used to publish certificates which we're still a long way from, there's a vulnerability when a malicious man-in-the-middle could have control of our traffic. It's probably uncommon, but it's probably still possible for state-level actors to mint their own certificates that our browsers and operating systems will trust without question. It's true that the bad guys could be operating at the other end of the VPN endpoint. So there's really no way to absolutely guarantee no man-in-the-middle. But something about Hotel WiFi networking, in particular, gives me the creeps. I suppose if I was downloading lots of Torrent content I might want to hide that fact from my Internet provider. But I've found Torrents to be more trouble than they're worth, so I don't care whether COX knows what I do on the Internet. If anyone was watching my Internet use it would just put them to sleep.

The Rolling Pwn

(Do you own a Honda? Are you sure??)

Enterprising security researchers with Star-V Lab, one using the handle "Kevin2600" and the other named Wesley Li, have revealed, as Larry David might say, "a pretty pretty serious" vulnerability in Honda's automobile keyless entry systems. It rather trivially allows attackers to remotely unlock and start potentially "all Honda vehicles currently existing on the market."

It works in a way that's similar to, but even easier to pull off, than the recently discovered Bluetooth replay attack which affected some Tesla vehicles; using readily available off-the-shelf SDR (software defined radio) equipment, the researchers were able to eavesdrop and capture the codes, then broadcast them back to the car in order to gain access.

Recall that we talked about this related attack previously. It was clever: The key fob produces a high-quality cryptographically unpredictable pseudo-random sequence of codes which are derived from a monotonically increasing counter which is then encrypted by an unknown secret key.

The vehicle knows the key, so it's able to decrypt the received code to determine whether it's next or at least forward of the key fob counter's last known position. But the primary security feature is that the vehicle will NEVER accept any code that it has seen previously. In other words, it will only allow its knowledge of the key fob's counter to be moved forward, never backward. This would seem to robustly prevent any possibility of a replay attack. But not so much...

What the clever hackers did before was to receive the fob's code while simultaneously jamming its receipt by the car. The user would think "guess I was too far away, or whatever." So they would press the key fob a second time. Now the hacker would capture the second code while replaying the first code which the auto had never yet received... and the door would unlock, the car would start, etc.

And, here's the important part: The attacker would have obtained and retained a not-yet-seen-by-the-car **next** unlocking code in the unpredictable sequence.

So that was then. Honda is now the target, and their problem is significantly worse since the attack on Hondas is much more passive and will be easier to conduct...

The Honda-attacking researchers have been able to remotely unlock and start the engines of Hondas from as far back as 2012, up to and including this year. The good news is, according to "The Drive", which independently tested and verified the vulnerability on a last year 2021 Honda Accord, the key fob attack does not allow an attacker to drive off with the vehicle, though it will start the engine. (Presumably the car contains some dynamic continuous presence technology that can sense the presence of the key inside the vehicle.)

Okay, so the more active attack against Tesla's worked the way I just recounted. But what Kevin and Wesley discovered was, as I said, significantly worse. They found that the counter in Honda's cars will be **resynchronized** when the car receives lock and unlock commands in a consecutive sequence — even when that sequence is from long ago. This means that Honda's counter can be reset to **any earlier state**. By being induced to move backwards, this will cause the car to accept codes from previous sessions that should have been forever invalidated.

In their write up, Kevin and Wesley said: *"By sending the commands in a consecutive sequence to the Honda vehicles, it will be resynchronizing the counter. Once the counter is resynced, commands from previous cycles of the counter will work again. Therefore, those commands can be used later to unlock the car at will."*

They wrote:

We have successfully tested the 10 most popular models of Honda vehicles from the Year 2012 up to the Year 2022 from the attacker's perspective. Therefore, we strongly believe the vulnerability affects all Honda vehicles currently existing on the market. Tested and known-vulnerable models include:

- *Honda Civic 2012*
- *Honda X-RV 2018*
- *Honda C-RV 2020*
- *Honda Accord 2020*
- *Honda Odyssey 2020*
- *Honda Inspire 2021*
- *Honda Fit 2022*
- *Honda Civic 2022*
- *Honda VE-1 2022*
- *Honda Breeze 2022*

Then we have the always-entertaining FAQfrom which I will excerpt:

1: Why it is called the Rolling-PWN, not a Honda-PWN?
Because this bug may exist in other brands of vehicles too ;)

3: Am I affected by the bug?
As long as a vulnerable Honda vehicle is in use, it can be abused.

4: Is there an assigned CVE for Rolling-PWN?
CVE-2021-46145 is the official reference to this bug.

5: Can I detect if someone has exploited this against me?
Probably not. The exploitation does not leave any traces in traditional log files. But considering the ease of exploitation and attacks leaving no trace, this threat should be taken seriously.

6: Is this a Honda vehicle only Bug?

No. Although the main targets for the research is Honda Automobiles. But we have leads to show the impact of this vulnerability also applies to other car manufacturers. We will release more details in the future.

7: Is the risk real?
We have successfully tested the latest models of Honda vehicles. And we strongly believe the vulnerability affects all Honda vehicles currently existing on the market.

8: What makes this Bug unique or what's the Difference between CVE-2022-27254 and CVE-2019-20626?
During the research, we noticed the other researchers have found similar vulnerabilities in Honda vehicles. Based on the description of: "The remote keyless system on Honda HR-V 2017 vehicles sends the same RF signal for each door-open request, which might allow a replay attack." What they found is a [simpler] **fixed code** vulnerability where an attacker can record the transmission in advance and replay it later to cause the car door to lock or unlock.

However, most modern vehicles including Honda Automobiles implemented a proprietary rolling codes mechanism to prevent fixed code replay attacks. The bug we discovered is in regard to the design flaw of the rolling codes mechanism implemented by Honda Motors... which needs to be taken very seriously.

9: Is there more technical information about Rolling-PWN?
You can follow the author on Twitter [@kevin2600]. However, we will not be releasing any tools required to go out and steal the affected vehicles. At a later stage, we will release technical information in order to encourage more researchers to get involved in car security research.

10: How to patch the modern automobile for Rolling-PWN bug like this?
The common solution requires us to bring the vehicle back to a local dealership as a recall. But the recommended mitigation strategy is to upgrade the vulnerable BCM firmware through Over-the-Air (OTA) Updates if feasible. However, some old vehicles may not support OTA.

11: What does Honda think about this Rolling-PWN Bug?
We have searched through the Honda official website, but we can not find any contact info to report the vulnerability. It appears that Honda Motor DOES NOT have a department to deal with security related issues for their products. And a person who works at HONDA has told us "The best way to report the Honda vulnerability is to contact customer service." Therefore, we filed a report to Honda Customer service, but we have not get any reply yet.

Earlier in March of this year, following similar remote keyless entry attacks on Honda Civics, BleepingComputer reached out to Honda and learned that Honda had no plans to update any of their older model cars. BleepingComputer wrote:

> Honda told us, multiple automakers use legacy technology for implementing remote lock-unlock functionality, and as such may be vulnerable to "determined and very technologically sophisticated thieves."

“At this time, it appears that the devices only appear to work within close proximity or while physically attached to the target vehicle, requiring local reception of radio signals from the vehicle owner's key fob when the vehicle is opened and started nearby,” a Honda spokesperson told BleepingComputer. In their statement, Honda explicitly mentions it has not verified the information reported by the researchers and cannot confirm if Honda's vehicles are actually vulnerable to this type of attack.

And one must imagine that Honda doesn’t want to know, since knowing might make it culpable.

And Honda did add that should the vehicles be vulnerable, “Honda has no plan to update older vehicles at this time.” and “It's important to note, while Honda regularly improves security features as new models are introduced, determined and technologically sophisticated thieves are also working to overcome those features.”

So all of this begs the question, why does this appear to be such a difficult problem to solve? Both the Tesla-style forward-only counter advance and Honda’s bi-directional resettable counter solutions are transparent to their users. But Tesla’s forward-only system is clearly superior from a security perspective. On a Honda, the ability to passively record a series of unlocking codes, which can then be replayed at any time later, seems like a significant oversight that any engineer who was designing this system would have understood. Of course they would. One thought is that there are likely some intellectual property rights issues. There’s no question that the first implementers of such rolling codes would have sought patents. That thought led to me ask the Google, where I immediately found: https://patents.google.com/patent/US6225889B1 titled:

US patent US6225889 titled: “Method of producing rolling code and keyless entry apparatus using the same”. The Abstract of the Patent reads:

A rolling code producing method is provided which may be employed in a keyless entry system for automotive vehicles designed to compare a rolling code derived in a receiver installed in the vehicle and a rolling code derived in a portable transmitter to allow authorized access to the vehicle if both the rolling codes match. The rolling code producing method includes producing different rolling codes, in sequence, using an initial code variable according to a given algorithm and changing the initial code variable in response to insertion of an initial code variable memory card carried by a vehicle operator into the receiver.

The good news is, this patent was issued in 1995 and it expired in 2016. Around the same time, also in 1995, garage door openers were suffering from the same lack of security. So, Brad Farris and James Fitzgibbon “invented” a similar system for their garage door opener employer, The Chamberlain Group and obtained US patent US44688695A.

There has been a lot of litigation over these patents and a long trail of bodies. But that’s what the patent system does. It mostly amounts to a significant source of revenue for intellectual property attorneys. But all of these various patents appear to have finally expired back in 2016. So it’s unclear why Honda would still be using their broken system. They apparently were back on 2012, when they may have had no choice. But it appears that for the past six years at least there has been no reason not to move to a much stronger forward-only counter scheme.

Overall, this is a difficult problem to robustly solve in any one-way only system. The ultimate way to solve the problem is for there to be a full handshake. The user presses the button on the key fob, which sends a fixed-code ping out into the world, identifying itself to any car within range. The car that key is registered to receives the "Hello it's me" ping and replies with a nonce challenge. The key fob receives the nonce challenge and either reversibly encrypts it under its shared secret key, or hashes it with an HMAC keyed with a shared secret. Either way, it returns the result of that cryptographic operation to the car, which verifies that the key fob **must** know the secret they share, so the car performs the requested action.

That system, while ultimately secure, and Internet-proven, is significantly more expensive, since both the key fob and the car must now support bi-directional communications. The key fob must also be able to receive and the car must also be able to transmit. Given the cost and complexity of this full solution, and the comparatively small additional security margin provided above the forward-only counter advancement used by Tesla and presumably other forward thinking automakers, I would say that the small added security is not worth the cost.

But given that forward-only counter technology has been freely available in the public domain, unencumbered by any patent licensing requirements since at least 2016, Honda's continued use of resettable counter protection, since then, can only be ascribed to them just not caring.

Given the popularity of Hondas (and who knows what other car makes that might also have been similarly lazy), the relative ease of collecting key fob codes and the ability to later replay them in an entirely passive attack, likely opens Honda to some future consumer litigation.