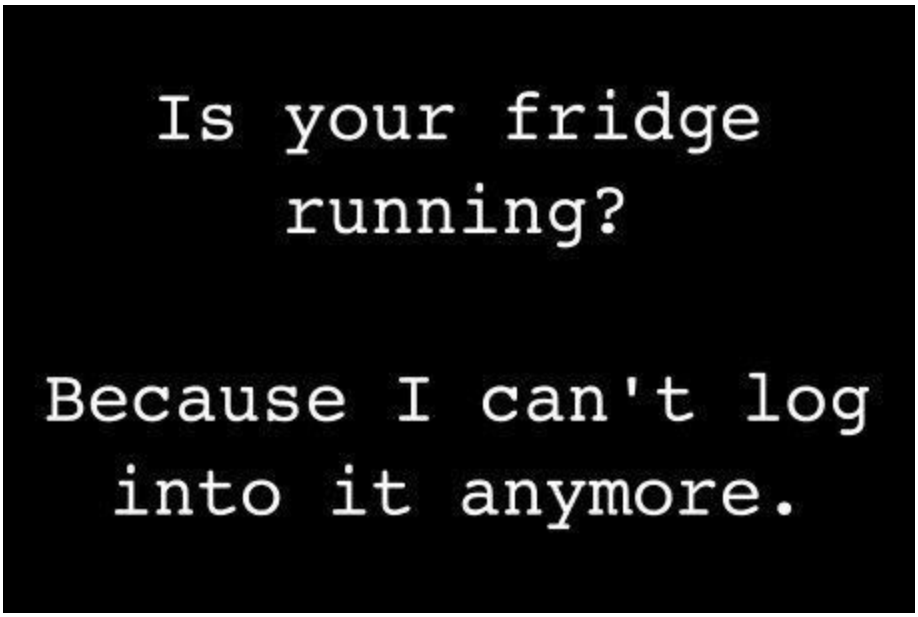
Security Now! #626 - 08-29-17

Shattering Trust

This week on Security Now!

This week we cover a bit of the ongoing drama surrounding Marcus Hutchins, examine a reported instance of interagency hacking, follow the evolving market for 0-day exploits, examine trouble arising from the continued use of a deprecated Apple security API, discover that Intel's controversial platform management engine **can**, after all, be disabled, look into another SMS attack, bring note to a nice looking TOTP authenticator, recommend an alternative to the shutting-down CrashPlan, deal with a bit of errata and miscellany, then we look into an interesting bit of research which invokes "The Wrath of Kahn".

Our World in 2017...



Security News

Fraud Forces WannaCry Hero's Legal Fund To Refund All Donations

The lawyer managing fundraising for Marcus Hutchins' legal defense decided it was easier to refund all donations than figure out which ones were legitimate.
https://www.buzzfeed.com/kevincollier/beset-by-fraud-wannacry-heros-legal-fund-refunds-all

Kevin Collier, who is BuzzFeed's News Cybersecurity Correspondent reported some unfortunate news surrounding the high profile arrest of WannaCry's Marcus Hutchins:

The overwhelming majority of money which had been raised to pay for the legal defense of Marcus Hutchins turned out to have been donated with stolen or fake credit card numbers, and all donations, including legitimate ones, will be returned, according to Tor Ekeland, the attorney who was managing the fund.

Tor reported that at least $150,000 of the money collected came from fraudulent sources, and that the prevalence of fraudulent donations effectively voided the entire fundraiser. He said he'd been able to identify only about $4,900 in legitimate donations, but that he couldn't be certain even of those.

Tor told reporters: “I don’t want to take the risk, so I just refunded everything. One person had five different charges to his account,” Ekeland told BuzzFeed News. “Odd numbers, the kind you’ll glance over when looking at your bill.”

He said he's still determining what to do with a bitcoin wallet set up to take donations for Hutchins, which has received 96 small donations, worth a total of about $3,400.

After Marcus' arrest a shocked community of cybersecurity researchers rallied behind him and tried to raise money for experienced attorneys. At that time, Ekeland stepped in to host the fundraiser after GoFundMe refused to host it, citing conflicts with its terms of service.

Hutchins, who pleaded not guilty to all six charges against him on Aug. 14, has retained Brian Klein, a Los Angeles-based trial lawyer, and Marcia Hofmann, an acclaimed expert on US hacking laws, as his attorneys. A judge told him he could not return to the UK before his trial, scheduled for October, but that he could stay in Los Angeles in the meantime.

Tarah Wheeler, a friend of Hutchins, previously told BuzzFeed News that she planned to set up a new legal defense fund for him, but did not immediately respond to request for comment for this story.

Ekeland said he believes he has refunded every donation to the fund, but that anyone who hasn’t heard from him can email info@torekeland.com for their refund.

And, of course, right on cue we have just had Brian Krebs posting a piece with the headline: "Beware of Hurricane Harvey Relief Scams". Brian writes: "U.S. federal agencies are warning citizens anxious to donate money for those victimized by Hurricane Harvey to be especially wary of scam artists. In years past we've seen shameless fraudsters stand up fake charities and other bogus relief efforts in a bid to capitalize on public concern over an ongoing disaster."

And our listeners may recall that I was automatically cautious about that Marcus Hutchins legal fund donation site, scrutinizing it, before anything else, for signs of legitimacy and illegitimacy. It appeared authentic, but as they say, you can never be too sure.

It occurs to me that we're seeing something of an emerging trend with this: High profile online "things" which attract outsized attention of any sort, are not left alone and allowed to function as intended. By being high profile they become targets of mischief. And if those sites are not adequately prepared to actively fend off mischievous or malicious attacks, disaster results.

When the FCC put up the Net Neutrality feedback site, it was so swamped with fraudulent postings and other nonsense as to become virtually useless. And here we have an entirely well-meaning online fundraiser similarly rendered useless by fraudsters who stole money from others' credit cards to illegally contribute to Marcus' defense fund... And as a result, collapsed the entire thing.

Perhaps in both cases this was sloppy and immature site design that put up a minimally functional web-facing surface that wasn't sufficiently proactive against malicious abuse. We've seen and covered many examples of such short-sighted design on this podcast.

This is sort of a logical extension of the "how do websites store our passwords" question. As we know, there are ridiculously wrong ways and very strong ways to solve the same problem. And the approach taken doesn't really matter if everything always goes well and the strategy never comes under stress which tests its strength. Because when weakly designed and/or implemented systems ARE put under stress, they WILL collapse and fail.

We're operating within an unregulated environment where anyone can do anything they wish. Setup any kind of website they choose using whatever technology occurs to them first. I doubt that imposing regulations upon the industry will help. But I think we do need to make the use of more well thought out solutions easier to deploy. Things like the Libsodium cryptographic library, which has bindings to virtually every language, brings useful drop-in encryption to a much larger audience. The next step will be to incorporate similar libraries into drop-in problem-solving packages so that much of the underlying plumbing can become standardized around solid solutions. In such a future accepting, accepting and storing passwords won't NEED to be written ad hoc because the right way to do it will be built into any platform a developer might choose.

Why did CIA create a bogus software upgrade? To steal data from FBI, NSA

WikiLeaks latest Vault 7 details the CIA's 'spy versus spy' Trojan.
http://www.zdnet.com/article/why-did-cia-create-a-bogus-software-upgrade-to-steal-data-from-fbi-nsa/

Here's one I didn't see coming. But I suppose, sadly, that it should have been obvious in retrospect.

We sort of assume that we're all on the same side. And that, for example, there is automatic inter-agency cooperation and data sharing. But we've also observed the tendency of agencies to protect their own interests, sometimes at the expense of the larger picture and the greater good... doubtless justifying their actions by telling themselves that is what they need to do.

If we are to believe the recently leaked so-called "ExpressLane documents" as part of Wikileaks ongoing Vault 7 document dump, it appears that the CIA embedded it own Trojan into an inter-agency biometrics collection system installed at 'liaison services' and intelligence gathering partners including the NSA, Department of Homeland Security, and the FBI.

Again, if these documents are to be believed, the CIA didn't trust its security service partners to share biometric information with it, so it created a bogus software upgrade to steal the data. This data-stealing Trojan was created as part of a CIA project called ExpressLane, a piece of software installed by CIA Office of Technical Service -- the OTS -- under the guise of upgrading the CIA's biometric collection system.

The CIA installed the biometric system at partner offices around the world and expected them to voluntarily share biometric data with the CIA. But in case they didn't, it installed ExpressLane to "verify that this data is also being shared with the Agency." It also had a feature to cut-off the liaison's access to the system if it didn't provide the CIA with access.

The documents note that: "The systems are provided to Liaison with the expectation for sharing of the biometric takes collected on the systems. Some of these biometric systems have already been given to the Liaison services. OTS/i2c plans to revisit these sites with the cover of upgrading the biometric software to perform a collection against the biometrics acquisitions"

So that OTS agents could install the Trojan in the presence of partner agents, ExpressLane included a "splash screen with a progress bar" to look like an authentic Windows install.

OTS agents would install the software with a USB stick and could set the installation time of the update as well as a kill date before visiting the target.

Once installed the Trojan collects relevant files and stores them in a secret partition on a specially watermarked thumb drive that an OTS agent inserts during a subsequent maintenance visit.

The biometric system itself was provided by US identity management firm CrossMatch. It specifically didn't want the update to reference CrossMatch software.

I suppose that "Spooks will be spooks" -- and that while we ARE all on the same side, the fact that we have any agency divisions inherently creates some degree of inter-agency rivalry and tension. Something like ExpressLane is perhaps the inevitable result.

Zerodium Offers $500K for Secure Messaging App Zero Days
https://threatpost.com/zerodium-offers-500k-for-secure-messaging-app-zero-days/127610/

The discovery and private reporting of exploitable 0-day flaws in mainstream secure messaging apps now brings a payment of half a million dollars!

Last week, Zerodium, a vendor operating in the nebulous exploit acquisition market updated their pricing structure to put a premium on zero-day vulnerabilities in secure messaging applications. Both remote code execution and local privilege elevation zero days in messaging apps including WhatsApp, Signal, Facebook Messenger, iMessage, Telegram and others can fetch as much as $500,000 from the company's program.

As we've covered here before, Zerodium purchases zero days then makes them available in a feed of exploits and defensive capabilities to its customers. And, of course, those attacks and vulnerabilities are never shared with the affected vendor since they would then be promptly patched and made valueless.

So in this dark ecosystem, it is in the interests of everyone within the supply chain -- the discoverer who wants top-dollar, Zerodium who is in this for profit, and the exploit purchasers who want to use the exploits obtained as long as possible until they are patched -- to closely guard and protect the secrecy of the vulnerabilities they are leveraging.

Zerodium's official policy is to sell only to democratic and non-sanctioned governments... but one does wonder how tightly that policy is maintained when cash is dangled, since profit is the underlying motivation.

Zerodium's pricing changes were primarily aimed at mobile platforms. Which suggest that's where the action is. The company is also offering half a million dollar payouts for remote code execution and local privilege escalation (LPE) bugs in default mobile email applications, $150,000 for baseband and media file or document RCE and LPE attacks, $100,000 for sandbox escapes, code-signing bypasses, kernel LPE, Wi-Fi RCE and LPE, and SS7 attacks.

Zerodium's founder, Bekrar, told Threatpost that government customers are in need of advanced capabilities and zero-day exploits that allow them to track criminals using these secure mobile apps. And, of course, governments have the deep pockets required to foot the hefty exploit subscription cost that Zerodium must be levying in order to offer such substantial payouts.

Bekrar said: "The high value of zero-day exploits for such apps comes from both a high demand by customers and a small attack surface in these apps which makes the discovery and exploitation of critical bugs very challenging for security researchers." In other words, they're both rare and valuable.

Zerodium also announced that it would offer $300,000 for Windows 10 remote code execution

zero days, specifically remote exploits targeting default Windows services such as SMB or RDP. Web server zero days, specifically Apache on Linux and Microsoft IIS remote code execution attacks, are now worth $150,000, while a Microsoft Outlook RCE is worth $100,000. Mozilla Thunderbird RCE and VMware ESXi guest-to-host escapes are both worth $80,000.

Threatpost notes that Zerodium also doubled—or nearly doubled—payouts for Chrome, PHP and OpenSSL attacks, while Tor RCEs on Linux and Windows climbed from $30,000 to $100,000 and $80,000 respectively.

Nearly a year ago, Zerodium tripled the bounty it offers for an Apple iOS 10 remote jailbreak to $1.5 million, after previously offering $1 million for iOS 9 zero days.

So what we're seeing here is the emergence and continuing evolution of a predictable for-profit ecosystem created by the tension which currently exists between uncrackable crypto and governments' perceived need to monitor whatever they choose of their own, and others' citizen's activities. This is not a stable state of affairs. One way or another, it is a transient created by the post-Snowden and comparatively sudden rush to encrypt after the world learned exactly how much monitoring was actually going on. It's going to be fascinating to watch how this fundamental tension becomes resolved in our near future because one way or another, it's going to be.

A deprecated, insecure Apple authorization API can be abused to run code at root
https://threatpost.com/deprecated-insecure-apple-authorization-api-can-be-abused-to-run-code-at-root/127618/

Kaspersky's Threatpost also reports that a deprecated and insecure Apple authorization API can be abused to run code at root:

[PARAPHRASING from Threatpost] A deprecated Apple authorization API, invoked by third-party installers, is still developers' preferred choice for updating apps and services on macOS... which is a problem due to a massive security issue that could be abused by a local attacker to elevate privileges to root with a little unwitting help from the user.

The situation is known and was raised again last month during DEF CON by noted Mac security researcher Patrick Wardle, chief security researcher at Synack, due to the ongoing use of the "AuthorizationExecuteWithPrivileges" API that installers for many popular applications including Slack, Google Chrome, Google-owned Dropcam, VMware Fusion, numerous security software updaters, and the open source update library Sparkle... all which call upon this deprecated API during updates.

During DEFCON, Synack's Patrick Wardle explained that the API causes the system to display the familiar authentication dialog box, which is handled by a separate daemon, meaning that the user doesn't have to entrust the application installer with their password. The operating system instead passes trust, and any functionality needing admin or root privileges upon installation may proceed.
"AuthorizationExecuteWithPrivileges", however, does not validate that what is about to be

executed on the machine was not maliciously modified. Therefore, an attacker already present on the computer and running their code, can wait for the third-party installer to call the insecure authorization API and piggyback off the user's credentials as they're entered into the dialog box.

Wardle said: "Normally what happens is these applications ask the operating system to execute something as root, and what they ask to execute is writeable by everyone, something that's in Temp or the downloaded application bundle. But this means that any running (and patient) local code, malware or a local attacker who already has some access to the device can alter what's about to be executed as root -- because by definition and necessity, non-privileged processes have access to what's about to be executed. Since the OS does not verify what the application requested to be executed wasn't modified, when the user puts in their credential and clicks install, the system will execute whatever was requested... even if that has been maliciously modified."

Wardle said the situation extends to Apple's own installer as well, which suffers from a separate, but related issue. He explained that as a user double-clicks on a .pkg file, Apple's installer.app executes and looks for plugins in the file and copies them to Temp, loading those libraries into the process context of the installer application, which is signed by Apple and trusted by the OS. Wardle said that a local attacker can win a race condition and modify those packages before they're installed on the local system. The malicious—and unsigned—dynamic libraries, since they're not verified by the installer app, will be blindly loaded, Wardle said. Those malicious "dylibs" can create their own easily forged authentication popup that will be trusted by the OS. The user will then enter their credentials, unwittingly giving malicious code permission to run as root on the machine.

Apple understood the problem and consequently deprecated the "AuthorizationExecuteWithPrivileges" API four years ago, back in 2013, and now recommends developers use another called SMJobBless. SMJobBless works by copying whatever its being asked to install into a higher privileged directory, and once it's in that secure directory, SMJobBless cryptographically validates that it hasn't been tampered with and only then asks the user to authenticate.

But using the much more secure SMJobBless comes at a cost -- literally -- because developers choosing to use SMJobBless must obtain an Apple developer certificate to perform the required cryptographic signing.

Snyack's Wardle, who is clearly no slouch, said: "The problem is that there is a secure solution, but it costs money and it's incredibly complicated. It took me several hours to get it working. Whereas the deprecated [and insecure] API is literally, like, three lines of code. It's really easy and that's why everybody uses it."

Wardle said he privately disclosed to Apple that its installer was loading unsigned libraries but Apple has not responded. He also he reported the API issue to Google with regard to Chrome and it responded that it was aware, and that no "good replacement exists"—including SMJobBless.

Wardle concluded, saying: “Looking at the secure replacement, it’s such a pain to get [it] working. There’s a massive tradeoff. In an ideal world, everyone would use the secure replacement, or Apple would provide a secure replacement that would be easier to use. When you have the Chromium guys saying this isn’t a valid replacement, they generally know what they’re talking about.”

This is admittedly a difficult problem to solve. There is no replacement for credentialing developers. It's just like credentialing web servers. If we don't do that, then there is no way we can trust their assertions. And this is an example of why Let's Encrypt is an improvement, but not a replacement, for fully credentialing. As we know, Let's Encrypt promises nothing more than reuse of the existing HTTPS TLS system for transiting traffic within encryption privacy. But this IS and has been massively abused by malicious actors. Which is why Let's Encrypt hasn't and doesn't threaten the established certificate authority industry who continue to provide a service that cannot be automated... because if it could be it would be abused.

Similarly, Apple cannot blindly offer free developer certificates to all comers otherwise all will come, some will abuse, and no assurance of enhanced security will result. And just as traditional CA's need reasonable payment for the non-automatable and necessary scrutiny they provide, so too does Apple.

But that said, it does seem as though Wardle's point about the comment from the Chromium developers is a good one. Some money NEEDS to be paid... but then the rest should be easy. And certainly Apple is clever enough to make it so. It feels as though their first pass at this was not sufficiently well thought through. Let's hope they get that fixed.

Disabling Intel ME 11 via undocumented mode
http://blog.ptsecurity.com/2017/08/disabling-intel-me.html

The guys over at Positive Technologies have discovered that it IS, in fact, possible to shutdown the vast majority of Intel's worrisome IME motherboard-integrated subsystem.

The story is the same one we often tell here: Some things CAN be protected, others cannot. The microcode firmware in Intel's preboot environment MUST BE DECOMPRESSED when it is pulled from storage, and that decompressor cannot itself be compressed. Therefore, if the decompressor can be located and reverse engineered, ten the secretly compressed preboot system firmware can be obtain in decompressed form.

The Intel firmware is compressed using the variable-length Huffman coding which every version of SpinRite has employed for the past 30 years to compress its highly-compressible user-interface data. Huffman coding is an incredibly elegant and efficient scheme where the alphabet of all compressible tokens are scanned and counted. Then variable-length bit codes are assigned with the shortest codes going to the most frequently occurring token in the alphabet... and the longest codes going to the least frequently occurring tokens. When there is a large non-uniformity in token occurrence frequency, Huffman coding can provide surprisingly high levels of compression with very little code and computation overhead.

But... the secret to decompressing anything coded with Huffman is having access to the encoding table that was developed by the initial token frequency analysis. THAT is the Positive Technology reverse engineers found. Armed with that table they were able to begin delving deeper into the deliberately hidden and obscured Intel system.

The page linked about provides a very deep and comprehensive dive into the weeds of this. But, in short, what they discovered was something known as "High Assurance Platform (HAP)... and a bit-flag for enabling it. They were unfamiliar with the term. But, as they wrote, a bit of Googling quickly revealed the truth: "The second search result said that the name belongs to a trusted platform program linked to the U.S. National Security Agency (NSA)."

After additional research and lots of experimentation they determined that the setting of this bit flag caused the Intel Management Engine to be shutdown once it had performed all of the necessary work of powering-up, configuring, starting the system and booting the platform's operating system!

In other words... the NSA is no more happy than we are to have this potentially buggy and exploit-prone hidden subsystem always running in the background with full access to the system's network interfaces. But the NSA has a bit more negotiating leverage than we do. So they were able to get Intel to acquiesce and provide them with the undocumented and well-protected option of completely shutting down the IME immediately upon the completion of its power-up work.

Wanting to be absolutely certain of their findings, the Positive Technology guys reached out to Intel for comment and received the following reply:

<INTEL QUOTE> "In response to requests from customers with specialized requirements we sometimes explore the modification or disabling of certain features. In this case, the modifications were made at the request of equipment manufacturers in support of their customer’s evaluation of the US government’s “High Assurance Platform” program. These modifications underwent a limited validation cycle and are not an officially supported configuration."

In other words: Yes, you found it. But we take no responsibility for its use... and we did do everything we could to hide it from everyone to prevent its discovery.

Another SMS-based OTP attack:
https://techcrunch.com/2017/08/23/i-was-hacked/

John Biggs is a writer, consultant, programmer, former East Coast Editor and current contributing writer for TechCrunch.

[John Wrote (and I lightly edited)]: At about 9pm on Tuesday, August 22 a hacker deauthorized my phone's SIM card, replacing it with his, presumably by calling T-Mobile.

This, of course, shut off network services to my phone and, moments later, allowed the hacker

to change most of my Gmail passwords, my Facebook password, and to impersonate me in texts sent under my identity. All of the two-factor notifications went, by default, to my phone number... so I received none of them and in about two minutes I was locked out of my digital life.

I noticed all of this at about an hour later around 10pm when I assessed the damage and called T-Mobile. By 10:30 I had restored my phone's SIM and began the process of changing all of my passwords, hardening my 2-factor accounts and my T-Mobile account... hopefully ensuring that this would not happen again.

Sadly, I worry it will.

My hacker was thorough. In the course a few minutes he or she did a quick search of my Facebook Messenger messages and assessed that I was originally from Ohio and that my Dad was sick. He or she used this information to approach people I knew in the cryptocurrency space with a story that was, arguably, quite ludicrous: the hospital would pull the plug on my Dad if they didn't get payment of a bill and that I, in my anguish, needed to borrow and sell 10 bitcoins immediately and would pay the friend back 15 the next morning. Luckily my friends weren't idiots and immediately texted me and my wife.

The hackers' IP (173.239.232.29), which points to LogicWeb in Plano, Texas, along with a breadcrumb noting a login from Florida suggests that the hackers were from the United States. They clearly had the modus operandi down because they also hacked two other friends of mine in the space of a week.

This all came about, I believe, after another friend in the cryptocurrency space was hacked last week. That hack bore all the same hallmarks as this one except the SIM hijacking. First the hacker grabbed access to my friend's Facebook Messenger and contacted everyone on his list that was interested in cryptocurrency, including me. In the ensuing melee the hacker asked me to send 10 Bitcoin and that he would send me 11 back in the morning. Confused, I told them that I had some Bitcoin but not that much. I then realized the ruse and asked "Did you talk to Wallace Shawn yet? He can help. I think he's having dinner with Andre right now." (Wallace Michael Shawn is a well known actor and comedian who starred in "My Dinner with Andre") The hacker claimed that Wallace wasn't available. I knew I'd been had.

This interaction led to my own subsequent hacking. Once it was clear that I had some bitcoin somewhere, the hackers decided I was their next target.

Ultimately I got away lucky. Nothing major was stolen as of today and I took control of all of my accounts fairly quickly. I had some two-factor set up but because my phone was compromised first I lost access to most of it. I've since activated authentication apps for all of my accounts. The biggest question is how the hackers took control of my SIM card. This is the most troubling and T-Mobile is looking into what happened.

Our takeaway lesson from this is: While it may be unlikely for everyone, targeted attacks CAN be quite powerful and successful. The new hacking method is to arrange to take over the communications channel used for account verification and/or second factor authentication. This is why time-based One Time Passwords, NOT communications-based One Time Passwords, are the only safe choice to make when a choice can be made. And we can hope that as more such instances surface more providers will begin offering TOTP authentication options.

And speaking of Time-Based One Time Passwords: Authenticator Plus
https://www.authenticatorplus.com/
https://authenticatorplus.freshdesk.com/support/solutions/articles/5000541895-technical-details-of-authenticator-plus

Everyone knows that my preference is to print, maintain and manage the QR codes of my OTP authentications in order to manually curate and install them into new devices. But some people may chose the convenience of networked linkage and secure cloud backup.

- *Secure*: 256-bit AES encryption and PIN lock for additional security along with hardware backed encryption which protects your account even in rooted devices

- *Sync Across Devices*: Apps available for Android Phones / Tablets, iPhone / iPad, Android Wear and Apple Watch

- *Organize*: Group accounts with categories and reorder frequently used accounts to top, change themes and displays account logos for easier lookup

- *Automatic backup / restore*: Accounts are automatically backed up to cloud (Google Drive or Dropbox) and can be restored easily.

On iOS it's FREE with in-app purchases. I haven't looked any further than that. But their FAQ says all the right things about their intentions and policies.

CrashPlan has shutdown its "Home" offering.
9to5mac writes in their coverage:

Code42, the company behind CrashPlan has officially announced that as of today they are pulling out of the consumer market. CrashPlan for Home users will have to begin migrating away from the service as it will no longer be available starting in October of 2018.

In a message posted on CrashPlan’s website, the company writes:

Effective August 22, 2017, Code42 will no longer offer new – or renew – CrashPlan for Home subscriptions, and we will begin to sunset the product over several months. CrashPlan for Home will no longer be available for use starting October 23, 2018.

Code42 seems to be handling this situation as delicately as possible by giving users ample time to prepare for the transition and offering discounts on alternative services.

CrashPlan for Home users can transition to the Small Business plan and receive a 75% discount over the next 12 months. If a user wants to completely move away from CrashPlan, the company recommends Carbonite's services.

Many of our listeners have asked what they should do? So I'll repeat my recent advice from my own research and findings:

BoxCryptor got the crypto right, but they have transitioned from a product to a service... so anyone use of BoxCryptor-as-a-service is just as susceptible to them changing their terms of service -- or canceling it outright -- exactly as CrashPlan just has.

This is why I do not seen the benefit of using an integrated bundled service... at least not for our more-sophisticated-than-average listeners. Instead, choose some TNO software whose crypto was done right, then choose among any or all of the available cloud storage providers... and be free to change your mind, or to use different providers for different purposes at any time.

This is why CloudBerry Labs has become my current choice: Their crypto is done exactly right, they offer one-time purchase and lifetime use of a product, not a service. And they support every cloud provider any of us have every heard of and many we haven't.

Errata

- A slip of my tongue:
  Daniel Szmulewicz (@danielszmu)
  "You meant atomicity, not monotonicity, in the section explaining compare-and-swap.
  I hope you don't mind me saying."

- Chris Frederick (@kansaichris)
  @SGgrc You talk quite a bit about "monotonic" operations/functions in the latest podcast.
  I wonder—did you mean to say "atomic" operations?

Miscellany

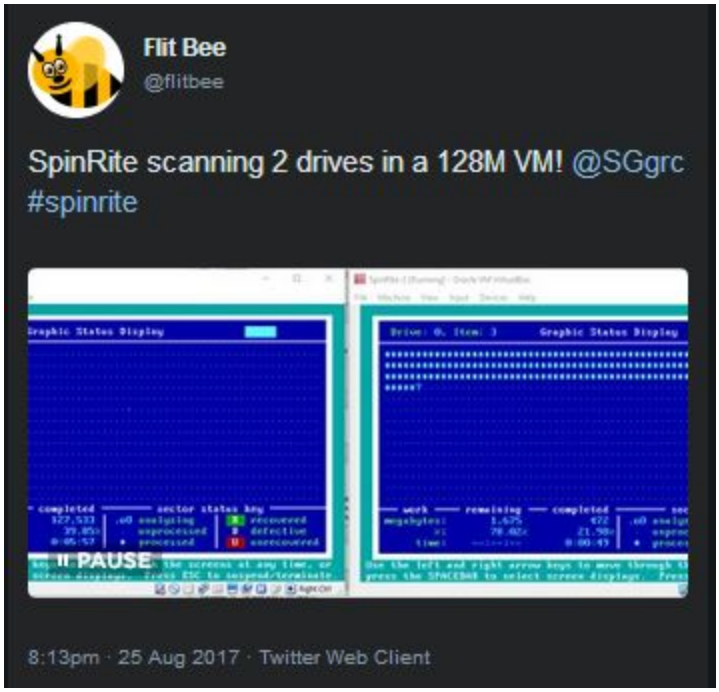
- AspiringLockpicker (@AspiringLockpic)
  I used my AutoIt UDF to break apart Steve's SN pages to get at the minute & episode number of all shows & do the math. A fun few minutes :-)

- Security Now! has, at this moment, ***908.05 hours*** of content spread over 623 episodes (e540 and e436 never recorded)

SpinRite

- xoff00 @xoff00
  A coworker just told me he regularly runs SpinRite on usb thumb drives... Thoughts?

- SpinRite scanning 2 drives in a 128M VM! @SGgrc #spinrite



Closing the Loop

- Pete Blumenthal (@PeteBlumenthal)
  @SGgrc Thoughts on VoicePrint for phone authentication? fidelity.com/security/fidel… or tdbank.com/bank/tdvoicepr… I see no technical details

---

Shattering Trust

https://iss.oy.ne.ro/Shattered.pdf

Shattered Trust: When Replacement Smartphone Components Attack

Abstract
Phone touchscreens, and other similar hardware components such as orientation sensors, wireless charging controllers, and NFC readers, are often produced by thirdparty manufacturers and not by the phone vendors themselves. Third-party driver source code to support these components is integrated into the vendor's source code. In contrast to "pluggable" drivers, such as USB or network drivers, the component driver's source code implicitly assumes that the component hardware is authentic and trustworthy. As a result of this trust, very few integrity

checks are performed on the communications between the component and the device's main processor.

In this paper, we call this trust into question, considering the fact that touchscreens are often shattered and then replaced with aftermarket components of questionable origin. We analyze the operation of a commonly used touchscreen controller.

We construct two standalone attacks, based on malicious touchscreen hardware, that function as building blocks toward a full attack: a series of touch injection attacks that allow the touchscreen to impersonate the user and exfiltrate data, and a buffer overflow attack that lets the attacker execute privileged operations.

Combining the two building blocks, we present and evaluate a series of end-to-end attacks that can severely compromise a stock Android phone with standard firmware. Our results make the case for a hardware-based physical countermeasure.

In other words, there is currently an assumption of trust among the hardware subsystems of Android smartphones.

I am reminded once again of that classic scene from "Star Trek: The Wrath of Kahn" where Kahn was approaching in radio silence, and Kirk had not raised his shields. Spock cited regulations about the need to adopt a defensive posture whenever the intentions of an approaching ship had not been confirmed. Kirk waved Spock off. One of Kahn's minions commented that Kirk was not raising his shields, to which Kahn replied: Of course not, we're all one big happy Federation! Whereupon Kahn blasted the crap out of the Enterprise!

As we have seen many times, it's all about security boundaries, and where they are placed. If a system's security boundaries are all-inclusive -- if it no INTERNAL boundaries -- then any adversary who managed to slip inside the system's outer boundary will potentially obtain unfettered access to the keys to the kingdom.

In their research paper they note:

Counterfeit components have been in existence ever since the dawn of the industrial age. Their effectiveness as attack vectors is also well known. What, then, is unique about the particular setting of a smartphone? We argue that our specific attack model is feasible because we assume only a specific component, with an extremely limited hardware interface, is malicious, while the rest of the phone (both hardware and software) can still be trusted. Furthermore, we assume that the repair technician installing the component is not malicious, and will perform no additional operations other than replacing the original component with an unknown to them, malicious one. Hundreds of millions of devices satisfying this attack model exist in the wild. One can assume that these limitations make this attack vector weaker than complete hardware replacement; we show that it is not. On the contrary, the nature of the smartphone ecosystem makes this attack model both practical and effective.

The authors followed responsible disclosure practices by disclosing the Synaptics device driver

vulnerabilities to Google on Feb. 16, 2017. The disclosure includes the details necessary for understanding, reproducing, and fixing of the issues discovered during the research. Google acknowledged the reported issues and issued a CVE (CVE-2017-0650) with critical severity. The vulnerabilities discovered in the Atmel device driver are being compiled into a responsible disclosure report at the time of submitting this paper.

Conclusions:

The threat of a malicious peripheral existing inside consumer electronics should not be taken lightly. As this paper shows, attacks by malicious peripherals are feasible, scalable, and invisible to most detection techniques. A well motivated adversary may be fully capable of mounting such attacks in a large scale or against specific targets. System designers should consider replacement components to be outside the phone’s trust boundary, and design their defenses accordingly.

~30~