Security Now! #745 - 12-17-19

PlunderVolt

This week on Security Now!

A reminder about Google's still operating SensorVault, inside Google's new "Verified SMS" Messages feature, another salvo in the end-to-end encryption war, a nice authentication feature added to iOS v13.3, some patch Tuesday news, a startling discovery about the weaknesses of RSA at scale, a collection of quick bits about last Friday the 13th, Mozilla 2FA for add-on developers, the surprising hard out for Microsoft's Security Essentials, and two bits about Chrome 79. Then we have a clarification about last week's VPN-geddon Denied discussion, a significant announcement about my new focus, some SQRL news… and then we conclude with a look at yet another interesting new way of compromising Intel processors known as "PlunderVolt".

Coming soon (full screen) to a Windows 7 system near you:

Security News

“SensorVault” ...again...
Google's once secret SensorVault technology was back in the news this past week after some new reporting by Forbes' Thomas Brewster who covers covering cybercrime, privacy, security and surveillance.

Last year and this year, substantial damage was done to Milwaukee, Wisconsin property by arsonists. To help locate the crime's perpetrators, officers from the Bureau of Alcohol, Tobacco, Firearms and Explosives (ATF) demanded Google supply records of user devices in the respective locations at the times the arsons took place. Though federal agents had used the technique before, they’d never received such a data haul back Google.

Two search warrants requested any consumer information in Google's possession covering 29,387 square meters during a total of nine hours for the four separate incidents. As we now know, if Android users, or iOS users running some Google apps, have now turned off “location history,” their physical whereabouts is being continually tracked and archived within Google's giant database called SensorVault.

We've previously discussed how Google uses a series of successive refinement passes to winnow the potentially large number of devices within the targeted geographic region and time window. In this case, Google found 1,494 qualifying device identifiers in SensorVault which it sent to the ATF, pursuant to the subpoena, to comb through.

This sort of "geofencing"-based search warrant has been found to be legal and investigators love them since it provides a potential view into a crime that is unavailable anywhere. Amid the so-called "going dark" outcry, the presence of this sort of completely under-the-radar law enforcement resource argues that in many ways this era of Internet-connected GPS-equipped pocket computers is making detective work easier than ever before.

I wanted to briefly revisit this topic with our listeners just to remind everyone that unless you have explicitly turned it off, your "location history" will be on and monitoring.

I also don't recall that we've ever heard a fulsome explanation from Google about why they are doing this? Is it just because they can? The US Congress sent Google a letter asking many questions, but I've been unable to locate any reply. This whole thing is also a perfect example of why companies who don't wish to be able to provide private intelligence data to others should not keep it in the first place. Secrets are difficult to keep. Google may have been creating this database for their own internal purposes. But once news of its existence leaked, under the US Constitution covering warranted search, anyone who knew to present Google with a search warrant could obtain anything Google had sequestered away.

So... if you really **do** care about your customer's privacy, don't keep any secrets.

Google's explainer reads: "Have safer chats with businesses that send Verified SMS"
Android's Messenger app has just been enhanced to offer its users verified SMS messaging

conversations with supporting companies. At this point the list is rather short: 1-800-Flowers, Banco Bradesco, Kayak, Payback, and SoFi.

I've read through the watered-down explanation and I believe that I know what's going on: Google has, effectively, implemented a separate authentication layer for SMS messages. Both ends of the SMS message connection also need to have data connections to Google for the exchange, with Google, of messaging metadata. Companies must register with Google and securely establish their identities. Once that's done, the company generates a public/private key pair and provides their public key to Google for redistribution.

Each instance of Android's messenger also generates a similar public/private key pair and provides their public key to Google.

When a company wants to send a "Verified by Google" SMS message, they obtain the recipient's public key from Google. They use their private key and their recipient's public key with a key agreement protocol, such as Diffie-Hellman Key Agreement (DHKA), to obtain a shared secret key. This key is used as the key for an HMAC through which they run the SMS message. They send the HMAC back to Google and send the SMS to the Android user.

When the Android user receives the SMS, supposedly from the company, they have the company's apparent phone number. So Messenger asks Google for the public key matching that phone number. After Google returns it, the Messenger app uses its private key and the obtained public key with the same key agreement protocol, such as Diffie-Hellman. This will yield the identical key that the sender obtained. So the recipient uses that to key an HMAC, through which they run the received SMS message. Messenger sends the message's MAC back to Google and, if the two MACs match -- the one from the sender and the one from the recipient, Google sends back a flag that turns on the "Verified Sender" notification along with a bunch of information about the company.

This sure seems like a lot of work to go through, but it does allow senders to send SMS messages to Android recipients in a way that authenticates them as the sending company and also absolutely prevents any tampering with the unencrypted SMS message with its en route.

And notice that US law enforcement would have no complaint about this, since the system is never encrypting the communications... it's only providing robust authentication of the sender and preventing tampering.

Google wrote: As part of this feature, Google attempts to verify all messages that appear to be sent by a business that is registered with Verified SMS. If the authenticity codes don’t match, and Google can't verify the message, the Messages app displays “Message could not be verified.” Because verification requires a data connection, if you have a weak data connection, the Messages app may display “Verifying sender…”. If you have no data connection, the Messages app displays “Waiting for connection to verify sender.” Until the sender of a message has been verified, Google doesn't recommend replying with sensitive info or opening links that you aren't sure you trust.

It seems like a lot of work to go through, but it does provide authentication for registered corporate senders and provides a great deal more security for SMS.

Another Shoe Dropped...
(I originally had "the other shoe dropped", but I think we have several more shoes to go…)
While we were recording last week's Security Now podcast, representatives from Apple and Facebook were testifying during a Senate Judiciary Committee hearing explaining the value of encryption that has not been deliberately weakened. In return, those on the committee informed Apple and Facebook that they had better put backdoors into their end-to-end encryption, or laws would be passed forcing tech companies to do so. The Chairman of the Senate Judiciary Committee, our dear Senator Linsey Graham said, quote:

"You're going to find a way to do this or we're going to do this for you. We're not going to live in a world where a bunch of child abusers have a safe haven to practice their craft. Period. End of discussion." (Linsey has been seeming a bit drunk on power lately, but that's beside the point.)

So this is the latest shot fired in the ongoing war over encryption. The most recent salvos have been launched following the privacy manifesto that Zuckerberg published last March. As we've been noting here, Zuck framed the company's new stance as a major strategy shift that involves developing a highly secure private communications platform based on Facebook's Messenger, Instagram, and WhatsApp services. Facebook's stated plan to leave the three chat services as standalone apps, but to merge their technical infrastructure so that users of different apps can talk "cross-app" to each other more easily.

That merging also would include adding WhatsApp's Signal-based end-to-end encryption to Messenger and Instagram to make them compatible and similarly attack-proof. Currently, Facebook's Messenger supports end-to-end encryption in "secure connections" mode: a mode that's off by default and must be enabled for every chat. And, Instagram has no end-to-end encryption on its chats at all. So ALL of this stands to change once the three are merged into a single Signal-derived messaging triumvirate.

And this past October, three governments warned Facebook that they had better end – or at least pause – that plan. As we covered at the time, US Attorney General William Barr and law enforcement chiefs of the UK and Australia -- those are the three -- signed an open letter calling on Facebook to back off of its "encrypting everything" plan unless it figures out a way to give law enforcement officials backdoor access so they can read messages.

Facebook said "No," with all due respect to law enforcement and its need to keep people safe.

And Monday, Facebook released an open letter responding to Bill Barr. In the letter, the WhatsApp and Messenger heads Will Cathcart and Stan Chudnovsky said that any backdoor access into Facebook's products created for law enforcement would weaken security and let in bad actors who would exploit the access. That's why Facebook has no intention of complying with Barr's request that the company make its products more accessible, they said:

> The 'backdoor' access you are demanding for law enforcement would be a gift to criminals, hackers and repressive regimes, creating a way for them to enter our systems and leaving every person on our platforms more vulnerable to real-life harm. People's private messages would be less secure and the real winners would be anyone seeking to take advantage of that weakened security. That is not something we are prepared to do.

In his opening statement last Tuesday, Linsey Graham told Apple and Facebook representatives that he appreciates "the fact that people cannot hack into my phone," but encrypted devices and messaging create a "safe haven" for criminals and child exploitation.

In Facebook's letter, Cathcart and Chudnovsky pointed out that cybersecurity experts have repeatedly shown that weakening any part of an encrypted system means that it's weakened "for everyone, everywhere." It's impossible to create a backdoor just for law enforcement that others wouldn't try to open, they said. As, as we know, they're not alone in that belief. More than 100 organizations, including the Center for Democracy and Technology and Privacy International, responded to Barr's letter to share their views on why creating backdoors jeopardizes people's safety. Facebook's letter also quoted Bruce Schneier from comments he made earlier this year: "You have to make a choice. Either everyone gets to spy, or no one gets to spy. You can't have 'We get to spy, you don't.' That's not the way the tech works."

Just for the record, because I know that my stance on this confuses some people, =I= don't want backdoors either. I really don't. But too many lawmakers are taking Linsey Graham's and Bill Barr's position. If we're going to be forced to provide law enforcement with the equivalent of a 21st century lawfully warranted wire taps, I would like those taps to be secure. Apple already provides secure group chat with iMessage. It's so utterly obviously possible for an additional "silent listener" to be added to Apple's existing iMessage chats. I don't want that to happen... But saying that it's not possible is not correct. Anywhere you have multi-party chat technology with the system and its keys managed by the provider -- as they all do -- it's obviously possible to add an additional listening party in a secure fashion under lawful warrant. Companies don't want to do that. I get that. But in the US, the right to privacy is NOT an absolute. When a court has been convinced that a limited suspension of an individual's or company's privacy serves the greater public good, privacy can be lawfully breached. The math of cryptography empowers absolute privacy in a way that the US's constitutional framers could have never foreseen and did not intend.

Of course, our Attorney General was unable to resist marching out the kiddie porn during a Wall Street Journal event last Tuesday. He granted that yes, there are benefits to encryption, such as to secure communications with a bank …a financial institution that will, and can, give investigators what they need when served with a warrant. But he said that the growth of consumer apps with warrant-repellent, end-to-end encryption, like WhatsApp and Signal, have aided "terrorist organizations, drug cartels, child molesting rings and kiddie porn type rings."

Unfortunately, whereas the US Congress currently and otherwise seems to be more divided than it's been in quite some time over political matters, in this they are much more united.

Apple's iOS v13.3 adds support for hardware key dongle authentication in Safari.
With last week's update to v13.3 Safari on our iDevices obtained access to FIDO2 compatible authentication hardware, such as Yubico's Yubikey or Google's Titan. All three hardware interfaces can be used: NFC, USB, and Lightning.

After the update to v13.3, users with the proper hardware can authenticate by using the YubiKey 5 NFC or Security Key NFC by tapping the YubiKey on the top of an iPhone from the iPhone 7 on. You can also do physical authentication with the YubiKey 5Ci by plugging it into the Lightning port of an iPhone or iPad. So that's a cool addition.

Last Tuesday was an important Tuesday to Patch...
... Because it foreclosed upon an elevation of privilege vulnerability that was seeing widespread and active exploitation in the wild.

An exploit of the now-patched Windows vulnerability was being deployed in combination with a Chrome exploit to take remote control over vulnerable computers.

In total, this month's December security updates patched a total of 36 vulnerabilities, 7 critical, 27 important, 1 moderate, and one low in severity. The important one to patch was the one rated as important. It was another flaw in the Win32k module enabling privilege escalation. When used in Chrome it facilitated an escape from the Chrome sandbox.

Although Google addressed their side of the flaw in Chrome 78.0.3904.87 with the release of an emergency update last month after Kaspersky disclosed it to them, hackers are still targeting users whose Chrome browsers have not been updated. And, in any event, we've seen many other examples where privilege escalation can hugely empower clever hackers. So closing this Windows bug was a good thing. As Kaspersky explained last month, the Chrome use-after-free exploit was chained together with the now-patched EoP flaw that exists in the way the Win32k component in Windows OS handles objects in memory.

Researchers discover prime factor collisions in active RSA certificates
As we know, an RSA private key is just a very large random prime number. It is hidden inside its matching RSA public key by multiplying it with another very large prime. And the private key is able to remain hidden because none of this planet's best math-magicians have ever been able to figure out any means for breaking those two primes apart again once they've been multiplied.

But, of course, if either of those primes did **not** have sufficient entropy, if either of them could be guessed, the hidden private key inside could be discoverable.

Unfortunately, last Saturday, during the First IEEE Conference on Trust, Privacy, and Security in Intelligent Systems and Applications, held in Los Angeles, California, a team of researchers from "Keyfactor" presented their findings into the current security posture of digital certificates. Their paper carried the chilling title: "Factoring RSA Keys in the IoT Era."

Listen to that again: Factoring RSA Keys in the IoT Era. (What's that about IoT??)

I'll share more in-depth in moment. But the super-short version of their findings is that the certificates being generated by relatively entropy-starved IoT devices turn out to be lacking. The devices lack of super-high-quality random number generators has shown up in weakly randomized primes, which leads to a failure of the fundamental guarantees offered by RSA public key crypto, which asserts that "you can't factor this." As a result of the widespread deployment of IoT devices, today, these Keyfactor researchers claim that one in every 172 RSA certificates in active use is vulnerable to attack (which is not actually true).

https://info.keyfactor.com/factoring-rsa-keys-in-the-iot-era#introduction

ABSTRACT

RSA keys are at risk of compromise when using improper random number generation. Many weak keys can efficiently be discovered and subsequently compromised by finding reused prime factors in a large data set. We collect and analyze 75 million RSA certificates from the Internet, and find that 1 in 172 certificates have keys that share a factor with another.

In contrast, only 5 of 100 million certificates found in a sample from Certificate Transparency logs are compromised by the same technique. The discrepancy in rates of compromise is overwhelmingly due to IoT devices exposed to the Internet, which may be subject to design constraints and limited entropy. The widespread susceptibility of these IoT devices poses a potential risk to the public due to their presence in sensitive settings. We conclude that device manufacturers must ensure their devices have access to sufficient entropy and adhere to best practices in cryptography to protect consumers.

So what exactly is going on here? If turns out that there's sort of a weakness in our cherished and beloved RSA public key system. The result of this weakness is that the entire system, when running at Internet scale, is exquisitely sensitive to the quality of every single prime in use... and that, in turn, makes RSA somewhat vulnerable and brittle.

Here's how these guys describe the attack (with a bit of editorial clarification:

THE ATTACK

RSA is used in the process of encrypting data to send across a network. The server transmits its RSA public key to the client as a part of an SSL or TLS handshake. Part of the RSA public key contains the modulus n = p * q, where p and q are two randomly chosen primes of similar size. Primes p and q are kept secret, as knowing these values allows the private key to be calculated.

Ensuring that p and q are selected with sufficient randomness is a crucial component of keeping the public key secure. Factoring a large modulus n to obtain p and q is not feasible under normal circumstances. However, if keys are generated with poor randomness, then it becomes a concern that two public keys will share a prime factor once enough keys are generated. [Now here's the counterintuitive loophole part:] If two RSA moduli n1 and n2 share precisely one prime factor p, then computing the Greatest Common Divisor (GCD) of those two moduli n1 and n2 will reveal the value of p. The GCD computation is significantly easier than straightforward factoring, and can easily be performed in practice. The other factors of n1 and n2 can then trivially be found by the simple calculations n1 / p and n2 / p, respectively, fully compromising both keys. This GCD computation can be scaled to analyze all pairs of keys in sub-quadratic time in the number of keys.

Selecting the prime factors of appropriate size with uniform randomness should prevent two moduli from ever sharing a factor in practice. However, if there is a flaw in the random number generation when choosing primes, a collision is likely in a sufficiently large dataset. Attackers can use this knowledge to collect a large number of RSA public keys and then look for GCDs between their moduli to search for factors shared by any pair.

Okay, so in other words... a great many of the public keys circulating around the Internet are gathered together into a massive dataset. This is not difficult today since we have cloud everything, massive storage and massive computation. Then a Greatest Common Divisor (GCD) algorithm is run against each pair of keys. That's of course n * (n-1) so it's a very big number. But these days we deal with very big data. If any pair of these public keys turn out to share a common divisor, since each key is the product of a pair of primes, that shared common divisor MUST BE the primes that those two keys share. And in finding that, the keys are instantly broken. Each key's other prime can be found by dividing the now-known prime by the public key modulus, and presto.

So, this means that individual servers can protect themselves by assuring that their primes originate from a source of maximum entropy randomness. So long as your own certificate is made from highly random primes the chance of some other certificate sharing either of those primes is vanishingly small.

But, this also means that certificates we do not ourselves create, such as those used by the IoT devices we use (remember that IoT stands for "Installation of Trojan"), might very well have been lazily generated and could share a prime with another similarly lazily generated certificate on the Internet. And if those two prime-colliding certificates were ever to be tested for a GCD, they would both be compromised.

Quick Bits

Friday the 13th: was true for New Orleans, which was hit by a Ransomware attack. The extent of the attack was not known at the time of its announcement which, get this, was announced over the loudspeaker system in City Hall with all government workers told to immediately turn off and unplug their computers.

City websites are also down. A spokesman for the mayor said the attack started sometime after 11 a.m. The city has activated its Emergency Operations Center and contacted officials from Louisiana State Police, the FBI, the state National Guard and the Secret Service for assistance, according to a tweet from the city's Department of Homeland Security and Emergency Preparedness.

So we're now living in a world where you might receive emergency orders to immediately turn off and unplug your computer. Amazing.

Starting next year, Mozilla will be requiring all Firefox add-on developers to sign in with two factor authentication. Their goal is to help prevent supply-chain attacks when malefactors compromise an add-on author's authentication to inject malware into an add-on.

The primary reason I would have considered obtaining Windows 7 Extended Security Updates (if I had been eligible) would have been for the continuing updates to Microsoft's built-in Security Essentials A/V. But it turns out that all updates for EVERYONE, even corporations who opt for extended security updates is being stopped cold after January 15th.

This seems completely nutso to me since it will drive corporate users to adopt 3rd-party A/V to cover their systems for the duration. It promises to be a windfall for A/V publishers.

With the release of Chrome 79, the option to force that browser to continue displaying what Google considers to be the "trivial" (their word) leading "www" of URLS such as www.grc.com has been removed. The "www" is gone for good and it's not coming back in Chrome.

And speaking for Chrome 79, as planned, Google re-enabled their browser's new code integrity protection feature. But not as planned... the unrecoverable "Aw Snap!" browser crashes immediately resumed, as before. The culprits are many, but include CylancePROTECT, Symantec Endpoint Protection (again), Webroot security solution and more. It's possible to force the feature off when necessary by launching Chrome 79 with the --disable-features=RendererCodeIntegrity switch on the command line.

Clarification

Over in GRC's Security Now newsgroup, Grant Taylor offered some clarification to what I explained during last week's VPN-geddonDenied discussion. **Grant wrote:**

"The VPN, in any form, is /a/ path for the traffic to the victim to take. It happens to be an /encrypted/ path. But it is not the /only/ path. The same traffic can be sent /unencrypted/ to the LAN interface's MAC address destined to the VPN interface's IP address.

By default, many operating systems will accept the traffic on the wrong interface and hand it to the TCP/IP stack which will hand it off to applications. The traffic does NOT /need/ to come into the system through the encrypted VPN."

I replied to Grant, there: "Right. The virtual interface is a "full interface" inasmuch as it can RECEIVE data from other parties with local access to the network. I should have made that more clear, since, as we know, that's the ONLY WAY the attacker, who lacks the VPN's working encryption key, could possibly inject SYNs or SYN-ACKs with test sequence numbers.

SpinRite

Newsgroups: grc.sqrl
Subject: Turning SQRL over to you guys to tend
Date: Mon, 16 Dec 2019 11:23:03 -0800

X-Url: https://www.GRC.com/groups/sqrl:23606

Everyone,

That was a momentous Subject line to write, but I believe it's finally time to do just that. SpinRite is now the more urgent need.

As you saw over this past weekend, I made an intense push to wrap things up with SQRL -- and I did. The published SQRL docs reflect my local WORD doc copies, and nearly 41 thousand copies of the "SQRL Explained" PDF has been downloaded. As we know, anyone who reads that will deeply understand SQRL, just like the Google cloud security guys did. The sqrl.grc.com server is also updated with the latest functional code so it's back up to spec.

I have a list of tweaks for GRC's SQRL client for Windows, but thanks to everyone's deep pre-release testing, the release #1 client doesn't have anything wrong that needs fixing right away. Though some UI bits can be improved, I think that's probably always going to be true, and I do not want to delay my switch to SpinRite any longer.

One thing I absolutely know because I've been watching carefully and I've seen it for myself for some time now, is that I'm leaving SQRL in the hands of an EXTREMELY competent and capable group of developers who understand it every bit as well as I do. And Github has exploded with a wonderful array of SQRL-related clients, servers and middleware.

SQRL no longer needs me. SpinRite does.

Over the next week or two I'll be spending some time at Level3 maintaining the Windows servers that have been neglected for far too long. And I'll deploy a new UNIX server to replace the aging hardware that has been hosting these newsgroups and DNS from the start.

Then I'll switch over to our spinrite.dev newsgroup here, and return to generating test releases of new low-level SpinRite code as we develop the technology that v6.1 will need, and work to bring SpinRite fully back to life.

SQRL

From: Jose C Gomez
Subject: OAuth 2 Provider for SQRL

Hi All
Over the last couple of weeks, I've been working on a functioning OAuth 2 provider that works exclusively with SQRL. This should, in my opinion, allow millions of sites (if they chose to) to adopt SQRL without having to change much on the backend.

I am finally in a pre-alpha release stage and wanted to share it with everyone here and get some input and thoughts on it. Following the SQRL moto, I've made it so you can remain pretty anonymous and still use the service and of course there are really no Secrets to keep.

It currently implements the basic Authorization Code grant flow and works fairly well.

I'm planning on releasing it in Beta sometime this week to let whomever wants to try it play with it. I run a discourse forum like Leo, so I've made sure that it will work with Discourse out of the box so the community at TWiT should be able to start using it (if Leo choses to) pretty easily.

Anyways here's a quick demo of it in my discourse instance. (Again, this is still in alpha / pre-alpha so if you go poking around things may blow up lol but feel free to)

It uses the Ask facility (if available) to act as the Permissions Granting Screen of OAuth, I thought it was a pretty neat way of putting the entire permissions structure in SQRL. We also have the ability if we want to, to make each site have a unique identity though I have that disabled right now, but if you think it would be worth it, I can certainly make it default. The reason for disabling it is that managing the accounts could get cumbersome.

I have to give a BIG thanks to TechLiam and JeffArthur who have been my sounding board over in slack while I slugged through the protocols and fought with the specs. Also, a zillion thanks to Paul F who let me use some of his tools like SQRLView and his command line SQRLClient for troubleshooting. Seriously SQRLView is an amazing piece of software and it should be shouted from the rooftops for anyone writing and or dealing with SQRL. Liam's DotNetCore Middleware is also a great piece of open source engineering and it keeps getting better.

Cheers guys and thanks again, I look forward to some feedback.

Here's a quick Demo Gif: https://sqrloauth.com/images/OAuthSqrlDemo.gif

---

Plundervolt: Software-based Fault Injection Attacks against Intel SGX

https://plundervolt.com/
https://plundervolt.com/doc/plundervolt.pdf

Abstract—Dynamic frequency and voltage scaling features have been introduced to manage ever-growing heat and power consumption in modern processors. Design restrictions ensure frequency and voltage are adjusted as a pair, based on the current load, because for each frequency there is only a certain voltage range where the processor can operate correctly. For this purpose, many processors (including the widespread Intel Core series) expose privileged software interfaces to dynamically regulate processor frequency and operating voltage.

In this paper, we demonstrate that these privileged interfaces can be reliably exploited to undermine the system's security. We present the Plundervolt attack, in which a privileged software adversary abuses an undocumented Intel Core voltage scaling interface to corrupt the integrity of Intel SGX enclave computations.

Plundervolt carefully controls the processor's supply voltage during an enclave computation, inducing predictable faults within the processor package. Consequently, even Intel SGX's memory encryption/authentication technology cannot protect against Plundervolt. In multiple case studies, we show how the induced faults in enclave computations can be leveraged in

real-world attacks to recover keys from cryptographic algorithms (including the AES-NI instruction set extension) or to induce memory safety vulnerabilities into bug-free enclave code. We finally discuss why mitigating Plundervolt is not trivial, requiring trusted computing base recovery through microcode updates or hardware changes.

Affected processors:
- Intel 6th, 7th, 8th, 9th & 10th Generation Core Processors.
- Intel Xeon Processor E3 v5 & v6
- Intel Xeon Processor E-2100 & E-2200 Families.

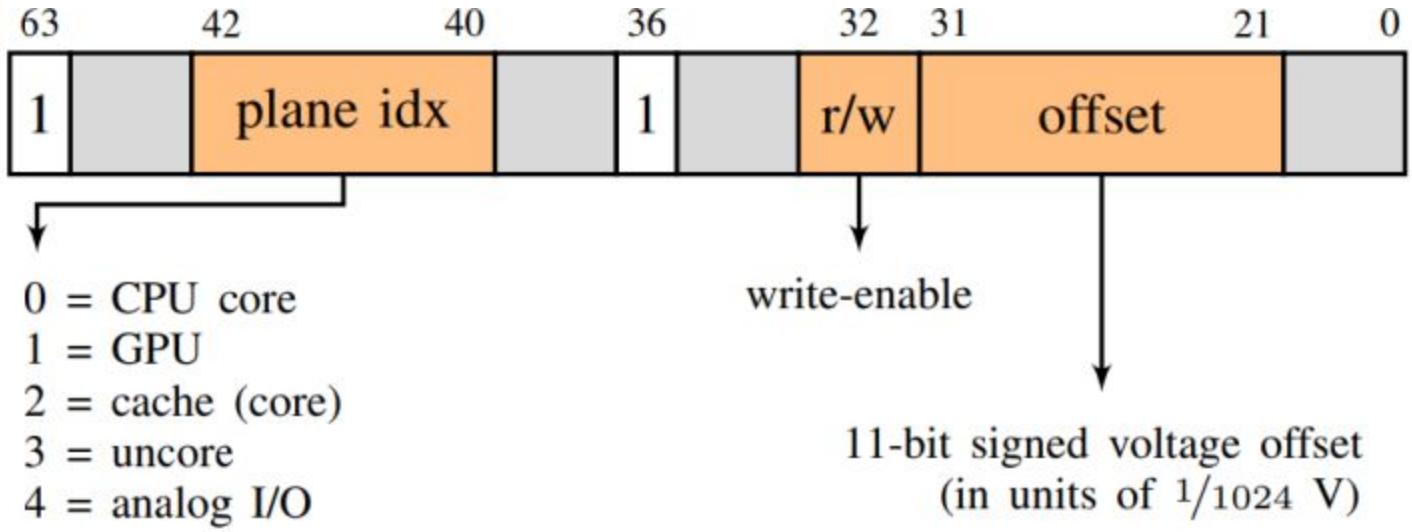

Fig. 1. Layout of the undocumented undervolting MSR with address 0x150.

MSR = Model Specific Registers

```
uint64_t multiplier = 0x1122334455667788;
uint64_t var = 0xdeadbeef * multiplier;

while (var == 0xdeadbeef * multiplier)
{
    var = 0xdeadbeef;
    var *= multiplier;
}
var ^= 0xdeadbeef * multiplier;
```

This code is placed into the secure enclave and clearly should never terminate. But their experiments revealed that undervolting the CPU just before switching to the enclave leads to a bit-flip in ***var***, typically in byte 3 (counting from the least-significant byte as byte 0). This causes the enclave program to terminate. The code outputs the XOR of the erroneous value with the expected value, to highlight only the faulty bit(s). And they consistently observe that in this specific configuration the output is always 0x04 00 00 00.

Whoopsie!

As we have so often seen, what starts off as a benign but unexpected fault in a computer system can often eventually be developed into a working exploit. And they have successfully done that here. Their paper goes into great depth showing how they managed to leverage their undervoltage tweaks into a complete breach of the sequestered processing which is supposed to be hidden inside Intel's secure enclave.

The result was Intel's statement: "When SGX is enabled on a system, a privileged user may be able to mount an attack through the control of CPU voltage settings with the potential to impact the confidentiality and integrity of software assets. Intel has worked with system vendors to develop a microcode update that mitigates the issue by locking voltage to the default settings."

So we have another Intel microcode update affecting all processors since Skylake.

A lesson we have learned during the past two years, since the tip of the iceberg known as Spectre and Meltdown is that choosing a PC vendor who is keeping their past product offerings current with a flow of BIOS updates containing the microcode patches now being produced by Intel on an ongoing basis is one more factor to consider.