Security Now! #689 - 11-13-18

“Self Decrypting Drives”

This week on Security Now!

This week we cover last month's Patch Tuesday this month, we look at a GDPR-inspired lawsuit filed by Privacy International, we ask our listeners to check two router ports to protect against a new Botnet that's making the rounds, we look at another irresponsibly disclosed 0day, this time in Virtual Box, we look at CloudFlare's release of a very cool 1.1.1.1 app for iOS and Android, and in perfect synchrony with this week's main topic, we note Microsoft's caution about the in-RAM vulnerabilities of the BitLocker whole drive encryption. We also cover a bit of miscellany, we close the loop with our listeners, and then we take a deep dive into last week's worrisome revelation about the lack of true security being offered by today's Self-Encrypting SSD Drives.

“Does your car have any idea why my car pulled it over?”

Security News

Patch Tuesday:
Today is November's Patch Tuesday... and today, in addition to November's scheduled security update, Microsoft has re-released last month's major FEATURE update, the October 2018 Update.

However, Microsoft has stated that they are planning to roll out the 1809 feature update slowly, so it may not appear immediately. And my own "zero-holdoff" check for it this morning, which DID reveal the November Cumulative Update for 1803, an update to Defender, and an update for Adobe's Flash Player, did not also offer the feature update to 1809.

Privacy International:
Title: "Privacy International files complaints against seven companies for wide-scale and systematic infringements of data protection law"
https://privacyinternational.org/press-release/2424/privacy-international-files-complaints-against-seven-companies-wide-scale-and

Thursday, November 8, 2018

Today, Privacy International has filed complaints against seven data brokers (Acxiom, Oracle), ad-tech companies (Criteo, Quantcast, Tapad), and credit referencing agencies (Equifax, Experian) with data protection authorities in France, Ireland, and the UK. Privacy International urges the data protection authorities to investigate these companies and to protect individuals from the mass exploitation of their data.

Our complaints target companies that, despite exploiting the data of millions of people, are not household names and therefore rarely have their practices challenged. In tandem with the complaints, we have today launched a campaign to seek to empower people and make it easier to demand that these companies delete our data.

- Our complaints argue that the way these companies exploit people's data, in particular for profiling, is in contravention of the General Data Protection Regulation (GDPR), which took effect on 25 May 2018.

- Our complaints are based on over 50 Data Subject Access Requests to these companies, as well as information that these companies provide in their marketing materials and in their privacy policies. As such, our assertions are based on evidence that represents only the tip of the iceberg. We expect and anticipate the regulators will be able to delve more deeply into our concerns regarding wide-scale and systematic infringements of the GDPR.

- PI is encouraged that the UK's Information Commissioner's Office (ICO) has issued assessment notices to Acxiom, Equifax, and Experian. We are asking the ICO to take into account our submissions in the context of their ongoing investigation and urge the ICO to widen its investigation to include Criteo, Oracle, Quantcast, and Tapad.

- As part of our campaign, PI has made it easier for people to write to companies and demand they delete their data.

The Privacy International complaint argues that:

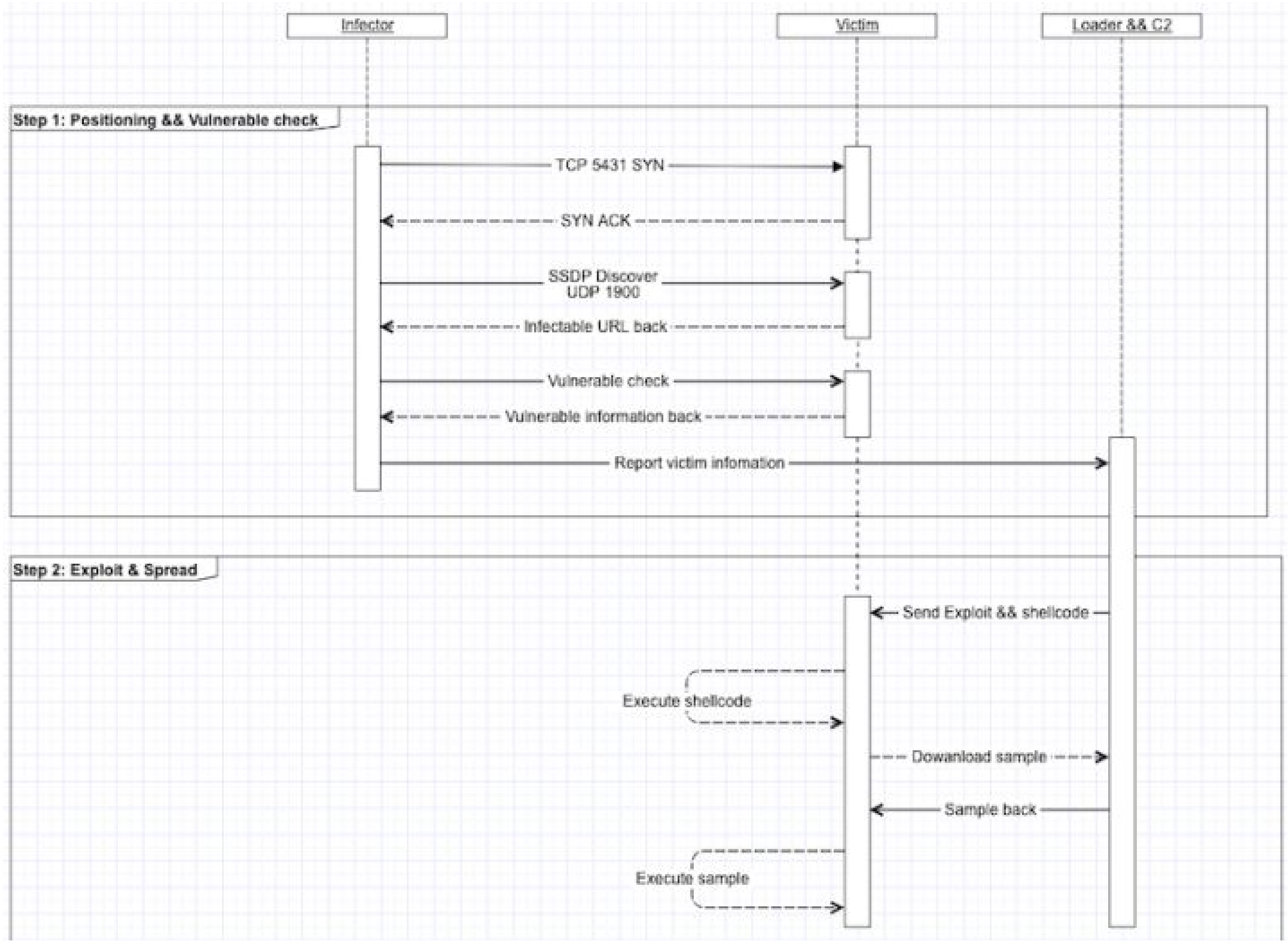
Acxiom, Criteo, Equifax, Experian, Oracle, Quantcast, and Tapad:

- Do not comply with the Data Protection Principles, namely the principles of transparency, fairness, lawfulness, purpose limitation, data minimisation, and accuracy.

- Do not have a legal basis for the way they use people's data, in breach of GDPR. Neither consent nor legitimate interest are satisfactory conditions for processing by these companies. They also do not have a basis for processing special category (sensitive) personal data.

- Should be further investigated as to their compliance with the rights and safeguards in GDPR.

BCMPUPnP_Hunter Botnet:



Probe port 5431 to make sure it's stealth or at least closed.
https://www.grc.com/x/portprobe=5431

Use "GRC's Instant UPnP Exposure Test"
(54,346 positive EXPOSED UPnP ports found so far.)

A massive number of routers all use a very common BroadCom chip which exposes a five year old (since 2013) flaw in UPnP authentication.

ADB Broadband S.p.A,	HomeStation ADSL Router
ADB Broadband,	ADB ADSL Router
ADBB,	ADB ADSL Router
ALSiTEC,	Broadcom ADSL Router
ASB,	ADSL Router
ASB,	ChinaNet EPON Router
ASB,	ChinaTelecom E8C(EPON) Gateway
Actiontec,	Actiontec GT784WN
Actiontec,	Verizon ADSL Router
BEC Technologies Inc.,	Broadcom ADSL Router
Best IT World India Pvt. Ltd.,	150M Wireless-N ADSL2+ Router
Best IT World India Pvt. Ltd.,	iB-WRA300N
Billion Electric Co., Ltd.,	ADSL2+ Firewall Router
Billion Electric Co., Ltd.,	BiPAC 7800NXL
Billion,	BiPAC 7700N
Billion,	BiPAC 7700N R2
Binatone Telecommunication,	Broadcom LAN Router
Broadcom,	ADSL Router
Broadcom,	ADSL2+ 11n WiFi CPE
Broadcom,	Broadcom Router
Broadcom,	Broadcom ADSL Router
Broadcom,	D-Link DSL-2640B
Broadcom,	D-link ADSL Router
Broadcom,	DLink ADSL Router
ClearAccess,	Broadcom ADSL Router
Comtrend,	AR-5383n
Comtrend,	Broadcom ADSL Router
Comtrend,	Comtrend single-chip ADSL router
D-Link Corporation.,	D-Link DSL-2640B
D-Link Corporation.,	D-Link DSL-2641B
D-Link Corporation.,	D-Link DSL-2740B
D-Link Corporation.,	D-Link DSL-2750B
D-Link Corporation.,	D-LinkDSL-2640B
D-Link Corporation.,	D-LinkDSL-2641B
D-Link Corporation.,	D-LinkDSL-2741B
D-Link Corporation.,	DSL-2640B
D-Link,	ADSL 4*FE 11n Router
D-Link,	D-Link ADSL Router
D-Link,	D-Link DSL-2640U

D-Link, D-Link DSL-2730B
D-Link, D-Link DSL-2730U
D-Link, D-Link DSL-2750B
D-Link, D-Link DSL-2750U
D-Link, D-Link DSL-6751
D-Link, D-Link DSL2750U
D-Link, D-Link Router
D-Link, D-link ADSL Router
D-Link, DVA-G3672B-LTT Networks ADSL Router
DARE, Dare router
DLink, D-Link DSL-2730B
DLink, D-Link VDSL Router
DLink, DLink ADSL Router
DQ Technology, Inc., ADSL2+ 11n WiFi CPE
DQ Technology, Inc., Broadcom ADSL Router
DSL, ADSL Router
DareGlobal, D-Link ADSL Router
Digicom S.p.A., ADSL Wireless Modem/Router
Digicom S.p.A., RAW300C-T03
Dlink, D-Link DSL-225
Eltex, Broadcom ADSL Router
FiberHome, Broadcom ADSL Router
GWD, ChinaTelecom E8C(EPON) Gateway
Genew, Broadcom ADSL Router
INTEX, W150D
INTEX, W300D
INTEX, Wireless N 150 ADSL2+ Modem Router
INTEX, Wireless N 300 ADSL2+ Modem Router
ITI Ltd., ITI Ltd.ADSL2Plus Modem/Router
Inteno, Broadcom ADSL Router
Intercross, Broadcom ADSL Router
IskraTEL, Broadcom ADSL Router
Kasda, Broadcom ADSL Router
Link-One, Modem Roteador Wireless N ADSL2+ 150 Mbps
Linksys, Cisco X1000
Linksys, Cisco X3500
NB, DSL-2740B
NetComm Wireless Limited, NetComm ADSL2+ Wireless Router
NetComm, NetComm ADSL2+ Wireless Router
NetComm, NetComm WiFi Data and VoIP Gateway
OPTICOM, DSLink 279
Opticom, DSLink 485
Orcon, Genius
QTECH, QTECH
Raisecom, Broadcom ADSL Router
Ramptel, 300Mbps ADSL Wireless-N Router
Router, ADSL2+ Router
SCTY, TYKH PON Router

Star-Net,	Broadcom ADSL Router
Starbridge Networks,	Broadcom ADSL Router
TP-LINK Technologies Co., Ltd,	300Mbps Wireless N ADSL2+ Modem Router
TP-LINK Technologies Co., Ltd,	300Mbps Wireless N USB ADSL2+ Modem Router
TP-LINK,	TP-LINK Wireless ADSL2+ Modem Router
TP-LINK,	TP-LINK Wireless ADSL2+ Router
Technicolor,	CenturyLink TR-064 v4.0
Tenda,	Tenda ADSL2+ WIFI MODEM
Tenda,	Tenda ADSL2+ WIFI Router
Tenda,	Tenda Gateway
Tenda/Imex,	ADSL2+ WIFI-MODEM WITH 3G/4G USB PORT
Tenda/Imex,	ADSL2+ WIFI-MODEM WITH EVO SUPPORT
UTStarcom Inc.,	UTStarcom ADSL2+ Modem Router
UTStarcom Inc.,	UTStarcom ADSL2+ Modem/Wireless Router
UniqueNet Solutions,	WLAN N300 ADSL2+ Modem Router
ZTE,	Broadcom ADSL Router
ZTE,	ONU Router
ZYXEL,	ZyXEL VDSL Router
Zhone,	Broadcom ADSL Router
Zhone,	Zhone Wireless Gateway
Zoom,	Zoom Adsl Modem/Router
ZyXEL,	CenturyLink UPnP v1.0
ZyXEL,	P-660HN-51
ZyXEL,	ZyXEL xDSL Router
huaqin,	HGU210 v3 Router
iBall Baton,	iBall Baton 150M Wireless-N ADSL2+ Router
iiNet Limited,	BudiiLite
iiNet,	BoB2
iiNet,	BoBLite

Updates are available for many of the more popular and well-supported routers, but shutting off UPnP to the WAN side, which was always a huge mistake, is always better than depending upon your router having no known or currently unknown problems.

Virtual Box has a difficult-to-exploit 0-Day:
A security researcher by the name of Sergey Zelenyuk discovered and irresponsibly disclosed a difficult-to-exploit and not terribly worrisome flaw in the NAT handling code of Virtual Box's Intel E1000 NIC driver code.

https://github.com/MorteNoir1/virtualbox_e1000_0day

On the Github page which beautifully and thoroughly details his work, he write:

> I like VirtualBox and it has nothing to do with why I publish a 0day vulnerability. The reason is my disagreement with contemporary state of infosec, especially of security research and bug bounty:
>
> Wait half a year until a vulnerability is patched is considered fine.
>
> In the bug bounty field these are considered fine:
>
> - Wait more than month until a submitted vulnerability is verified and a decision to buy or not to buy is made.
>
> - Change the decision on the fly. Today you figured out the bug bounty program will buy bugs in a software, week later you come with bugs and exploits and receive "not interested".
>
> - Have not a precise list of software a bug bounty is interested to buy bugs in. Handy for bug bounties, awkward for researchers.
>
> - Have not precise lower and upper bounds of vulnerability prices. There are many things influencing a price but researchers need to know what is worth to work on and what is not.
>
> Delusion of grandeur and marketing B.S.: naming vulnerabilities and creating websites for them; making a thousand conferences in a year; exaggerating importance of own job as a security researcher; considering yourself "a world saviour". Come down, Your Highness.
>
> I'm exhausted of the first two, therefore my move is full disclosure. Infosec, please move forward.

As with the two cases we've seen of "SandboxEscaper" previously releasing 0-days, and given the quantity and quality of the work that both she and Sergey have done (Sergey's is every bit as impressive), it's understandable that they might feel jerked around by the bug bounty system and decide that obtaining their 15-minutes of fame and notoriety is worth more than fighting for some cash. And it's certainly true that much more attention is drawn by a developer who irresponsibly discloses than one who informs the affected software publisher and obtains a footnote of acknowledgement in the eventual security update.

As for the particulars of Sergey's discovery:

- Vulnerable software: VirtualBox 5.2.20 and prior versions.
- Host OS: any, the bug is in a shared code base.
- Guest OS: any.
- VM configuration: default (the only requirement is that a network card is Intel PRO/1000 MT Desktop (82540EM) and a mode is NAT).

<quote> How to protect yourself: Until the patched VirtualBox build is out you can change the

network card of your virtual machines to PCnet (either of two) or to Paravirtualized Network. If you can't, change the mode from NAT to another one. The former way is more secure.

<quote> Introduction: A default VirtualBox virtual network device is Intel PRO/1000 MT Desktop (82540EM) and the default network mode is NAT. We will refer to it E1000.

The E1000 has a vulnerability allowing an attacker with root/administrator privileges in a guest to escape to a host ring3. Then the attacker can use existing techniques to escalate privileges to ring 0 via /dev/vboxdrv.

So it IS a VM containment breach which allows access to the Host OS application (ring3) layer. That's not good, but it's not the end of the world. And Sergey's very impressive full disclosure will make fixing this very simple.

1.1.1.1 comes to iOS and Android:
https://1.1.1.1/

On Sunday, which was 11/11, Cloudflare released their 1.1.1.1 DNS app for iOS and Android.

Significantly, this app not only points any iOS device's DNS to 1.1.1.1 (and 1.0.0.1 for backup), it also encapsulates them inside either an HTTPS or TLS tunnel, thus completely hiding them from anyone who might be passively snooping on the user's DNS queries.

As we recall, Cloudflare first launched their 1.1.1.1 DNS service earlier this year on April 1st.

Their site explains some of the benefits of their offering:

> Privacy First: Guaranteed.
>
> - We will never sell your data or use it to target ads. Period.
> - We will never log your IP address (the way other companies identify you). And we're not just saying that. We've retained KPMG to audit our systems annually to ensure that we're doing what we say.
> - Frankly, we don't want to know what you do on the Internet—it's none of our business—and we've taken the technical steps to ensure we can't.
>
> Faster than anything else.
>
> - 28% faster, in fact.
> - We've built 1.1.1.1 to be the Internet's fastest DNS directory. Don't take our word for it. The independent DNS monitor DNSPerf ranks 1.1.1.1 the fastest DNS service in the world.
> - Since nearly everything you do on the Internet starts with a DNS request, choosing the fastest DNS directory across all your devices will accelerate almost everything you do online.

As we know, using another DNS server such as 1.1.1.1 requires users to modify their local

Internet connection's preferred DNS servers and change their ISP-supplied settings to 1.1.1.1.

Cloudflare's new 1.1.1.1 apps for iOS and Android make this switch easy.

And... you get DNS over HTTPS or TLS. HTTPS is the default.

This means that all of the device's DNS queries are encrypted and hidden and also that DNS spoofing

BitLocker's exposure of keys in RAM:
Microsoft: "Blocking the SBP-2 driver and Thunderbolt controllers to reduce 1394 DMA and Thunderbolt DMA threats to BitLocker"
https://support.microsoft.com/en-us/help/2516445/blocking-the-sbp-2-driver-and-thunderbolt-controllers-to-reduce-1394-d

A BitLocker-protected computer may be vulnerable to Direct Memory Access (DMA) attacks when the computer is turned on or is in the Standby power state. This includes when the desktop is locked.

BitLocker with TPM-only authentication allows for a computer to enter the power-on state without any pre-boot authentication. Therefore, an attacker may be able to perform DMA attacks.

In these configurations, an attacker may be able to search for BitLocker encryption keys in system memory by spoofing the SBP-2 [Serial Bus Protocol] hardware ID by using an attacking device that is plugged into a 1394 port. Alternatively, an active Thunderbolt port also provides access to system memory to perform an attack. Note that Thunderbolt 3 on the new USB Type-C connector includes new security features which can be configured to protect against this type of attack without disabling the port.

This article applies to any of the following systems:

- Systems that are left turned on
- Systems that are left in the Standby power state
- Systems that use the TPM-only BitLocker protector

1394 DMA threats to BitLocker
BitLocker system integrity checks mitigate unauthorized Kernel Debugging status changes. However, an attacker could connect an attacking device to a 1394 port, and then spoof an SBP-2 hardware ID. When Windows detects an SBP-2 hardware ID, it loads the SBP-2 driver (sbp2port.sys), and then instructs the driver to allow for the SBP-2 device to perform DMA. This enables an attacker to gain access to system memory and search for BitLocker encryption keys.

Thunderbolt physical DMA
Thunderbolt is an external bus that allows for direct access to system memory via PCI. This functionality is provided as a performance improvement. It enables large amounts of data to transfer directly between a Thunderbolt device and system memory, thereby bypassing the CPU and software.

Thunderbolt threats to BitLocker

An attacker could connect a special purpose device to a Thunderbolt port and have full direct memory access through the PCI Express bus. This could enable an attacker to gain access to system memory and search for BitLocker encryption keys. Note that Thunderbolt 3 on the new USB Type-C connector includes new security features which can be configured to protect against this type of access.

Bitlocker Countermeasures

https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-countermeasures

Miscellany

GRC's DNS Spoofability System is back online!

- https://www.grc.com/dns/dns.htm
- https://www.grc.com/dns/howthisworks.htm

A website semi-automatic CSP (Content Security Policy) determiner Widget for FF.

- (With thanks to a tweet from Simon Zerafa for spotting it...)
- https://addons.mozilla.org/en-US/firefox/addon/laboratory-by-mozilla/

Peter Hamilton's “Salvation”

- A long read for the setup to an amazing new series.
- Now the long wait begins.
- Fortunately... the 10th book in Ryk Brown's second if five series of 15 book sets was released on October 28th.

SpinRite

Nicholas Kasprinski @nkasprinski

I have a drive that is encrypted with windows bitlocker but is not able to boot up. Can I still run spinrite against it to decrypt and recover the drive's data?

Closing The Loop

Davy Jones [no bcash] @9arsth

Need a quick yes-no answer. Applied the sandbox to my w10 home box; tested, it works. Then applied to w7 pro, cmdline said it was successful but when tested, no child process under MsMpEng. So not possible?

David P. Vallee @Yossarrian

In the SQRL paradigm, what happens if there's a system-wide collapse? People would have hundreds of accounts with no record of their credentials because there was reason to record them when they were created.

SKYNET @fairlane32

Hi Steve, with Windows Defender sandboxing, how do you turn it off if you need to, do you retype the command with a 0 at the end instead of the 1 to disable it???

Edward Evans @EdJoJob

Hi Steve,
Regarding SQRL, how would I be able to have a bot associated with my user account without sharing my core secret? (e.g. a web watcher that is watching on the far side of the auth wall)
Love Security Now. Thanks to you, Leo and the gang I feel that I actually "get" security, now. Thanks for all you do.

Self-Decrypting Drives

The title of the 16-page, well-written, and wonderfully-detailed research paper from two security researchers at the Radboud University in the Netherlands is titled:

Self-encrypting deception: weaknesses in
the encryption of solid state drives (SSDs)

There are two industry standards: ATA Security and “Opal”, a specification produced by the TCG (Trusted Computing Group). ATA security was the original, which essentially locks and unlocks an ATA drive as a whole, based upon a user-supplied password.

The successor to the ATA Security Feature Set is the Trusted Computing Group's “Opal”. Opal defines a much more complex and much more difficult to implement encryption facility.

If the much simpler and more straightforward ATA system could be said to be a “one-to-one” scheme where one password is used to unlock one region (the entire) drive, then the Opal system would be a many-to-many system where a drive's storage space can be freely subdivided into arbitrary bounded regions, each of which can be individually encrypted so that they are accessible under multiple and individually differing passwords so that different passwords can selectively decrypt various subsets of the entire drive.

If the years of this podcast has taught us anything it’s that complexity is the arch enemy of security. And the saddest things about the “Opal” system is that while all modern drive designs have been hugely burdened with its complexity, mainstream operating systems only wish to securely lock and unlock a drive. They have no use for or need for Opal’s complexity, making it a classic and disastrous case of overdesign-by-committee.

In an article written seven years ago in 2011, titled “The Pros and Cons of Opal Compliant Drives” its well-meaning but naive author states:

> “Hardware based encryption is very secure; far more secure than any software-based offering. Software can be corrupted or negated, while hardware cannot. Software runs under an operating system that is vulnerable to viruses and other attacks. An operating system, by definition, provides open access to applications and thus exposes these access points to improper use. Hardware based security can more effectively restrict access from the outside, especially to unauthorized use.”

This charming view of the world predated the Spectre and Meltdown hardware problems by seven years… and it assumes that “hardware” is magically perfect because it is “harder” than software. It would be nice to live in that world.

The #1 lesson we learn from this terrific research is that the fact that a drive sports a given interface, or supports a given security standard, whose marketing brochures boast of its security and encryption… is ENTIRELY DECOUPLED from, and means absolutely NOTHING about, the actual delivered security in the face of a determined effort to obtain the drive's content.

Is the encrypted drive unreadable to the casual passer by? Absolutely. Is it unreadable to someone who is highly motivated to gain access to the drive's contents? That's the interesting question this pair of researchers set out to determine… and initial surprising successes kept them going.

Two skilled researchers from the Netherlands invested significant time and attention to reverse engineering seven mainstream and highly popular SSD's which account for, collectively, nearly half of the market.

The Abstract of their research reads:

> We have analyzed the hardware full-disk encryption of several SSDs by reverse engineering their firmware. In theory, the security guarantees offered by hardware encryption are similar to or better than software implementations. In reality, we found that many hardware implementations have critical security weaknesses, for many models allowing for complete recovery of the data without knowledge of any secret.
>
> BitLocker, the encryption software built into Microsoft Windows will rely exclusively on hardware full-disk encryption if the SSD advertises supported for it. Thus, for these drives, data protected by BitLocker is also compromised.
>
> This challenges the view that hardware encryption is preferable over software encryption. We conclude that one should not rely solely on hardware encryption offered by SSDs.
>
> We have analyzed firmwares from different SSD models offering hardware encryption, focusing on these flaws. The analysis uncovers a pattern of critical issues across vendors. For multiple models, it is possible to bypass the encryption entirely, allowing for a complete recovery of the data without any knowledge of passwords or keys. The situation is worsened by the delegation of encryption to the drive by BitLocker. Due to the default policy, many BitLocker users are unintentionally using hardware encryption, exposing them to the same threats. As such, we should reconsider whether hardware encryption is a true successor to its software counterpart, and whether the established standards actually promote sound implementations.

So what happened? Regular listeners of this podcast will know the CORRECT way to manage a drive's encryption and many of us could state it clearly:

- At the factory, the very first time the drive is powered up, a high quality entropy source -- whether built-in or external -- is used to produce a large (128 or 256-bit) high-entropy symmetric secret; THE key that will forever be used to key an in-line AES algorithm which encrypts and decrypts the drive's contents on the fly.

- If the drive is NOT password protected, it is still encrypted, transparently, under that unique per-drive secret key. Since a hardware implementation of the AES / Rijndael cipher is "in-line" and able to keep up with the performance of the storage medium and the drive's external interface, whichever of the two is the slower, there is no performance penalty.

- Then, if the drive's user or operating system wishes to later protect the drive's contents, a user- or OS- supplied secret is run through a PBKDF2 function which is internal to the drive to produce a password-dependent key which is subsequently used to encrypt the drive's master encryption key. The drive’s original factory-set master key is **physically overwritten** with its password- encrypted version so that the original master key no longer exists anywhere on the drive.

- To allow candidate passwords to be verified before they are applied to the drive, a different PBKDF2 hashing function should also be used to independently verify any would-be decryption password.

As we know... Security is inherently a weakest-link phenomenon and if even one of those things above is not done properly, the system's weakest link will be feasibly broken and the system's security guarantees will fail.

Disturbing though it is, among those seven highly popular drives, failures in every one of those aspects was actually found.

Responsible Disclosure:

Before we go any further, I should note that when the researchers realized just how bad the situation was in this industry they elected to handle its disclosure responsibly. They explained:

> After discovering these vulnerabilities, we followed a process of responsible disclosure. In this case, the National Cyber Security Center (NCSC) of the Netherlands was informed first, which assisted in the responsible disclosure process, by setting up the contact with the manufactures involved. We cooperated with both manufacturers, to fix their products and agreed not to disclose the vulnerabilities for six months. Both vendors have confirmed all the reported issues. For models currently being supported, firmware updates are either released or currently in development.

In the properly designed system I outlined above, we would say that "the key is bound to the password" meaning that the information provided by the password is *absolutely* required to synthesize the key.

Unfortunately, in the majority of the implementations the researchers found that the key was NOT bound to the password, but that it was "protected from ACCESS" by a password, which as we know is not the same thing and which opens that drive to a full password bypass. This is what the researchers were often able to achieve and demonstrate.

In other words, the drive’s master key existed -- hidden -- somewhere on the drive; in its firmware, on internal or external non-volatile memory, somewhere. And the externally supplied password was being used not to CREATE the key, but to unlock its access.

What were the designers of these systems thinking?

The only way to explain their thinking is that the designers **must** have believed that there would **never** be any way to reverse engineer their implementation. Assuming that their code was error free, they must have assumed that the only access anyone would ever have would be through the front door -- by powering up the drive and accessing its standard ATA command interface.

And speaking for a moment about "error free code", in their description of their reverse-engineering of the highly popular Samsung 840 EVO drive the researchers note of some of the drive's power-up logic: *"The key is computed during the drive's boot sequence. However, due to a bug in the firmware, retrieving slot 451 during early boot fails. Therefore, "p" -- a value being hashed -- is in fact a zero buffer. Consequently, the resulting key is constant for all devices."*

So, not only is the fundamental design of the drives flawed, but the flawed design is buggy such that it doesn't even do what its designers intended.

But back on the topic of keeping secrets inside the drive...

Our intrepid security researchers took advantage of two things these designers apparently failed to consider:

- Firmware downloads and
- Microcontroller JTAG debugging interfaces:

First: Most of these devices have readily downloadable firmware.

In some cases the firmware update is provided as a bootable .ISO image incorporating an OS kernel such as a small Linux and the firmware. This can be readily reverse engineered even when the firmware image which is bound into the ISO image has been obfuscated. This is due to the fact that it must be decrypted for upload to the drive.

In other cases, a dedicated firmware upgrade utility is provided but it, too, can be readily reverse engineered to obtain the then reverse engineer the drive's controller code.

And even if neither of those avenues was available, the interface to the drive could be intercepted and the update data captured, since every one of the drives used the industry standard ATA "Upload Firmware" command.

One way or another, once the drive's "secret" internal operation has been exposed, the location of its secrets can be found.

This podcast has spent a great deal of time talking about the keeping of secrets. So we know that the greatest breakthrough which occurred in cryptographic thinking was the innovation of a keyed cipher. Before that innovation, cryptographers created clever unkeyed ciphers... but if that cipher's operation ever became known, EVERY SECRET it had ever been used to protect would simultaneously be divulged. Cryptography's greatest innovation was the concept of a keyed cipher, where its algorithm could be -- and absolutely should be -- freely published and studied. And where the secret that must be kept was only a tiny key. Essentially, the use of a key created a near infinity of individual specific ciphers from a single master general cipher.

The analogy here is to the erroneous thinking that apparently went into the design of these self-encrypting SSDs. Their designers MUST have erroneously believed that they could keep their internal algorithms secret... or perhaps that no one would ever bother looking… because such secrets are impossible to keep.

And beyond their exposure of their own firmware, was the presence of their microcontroller's serial JTAG interface...

JTAG is an industry standard interface which uses only a few wires, and thus only consumes a few pins. It is universally supported in some fashion, by all microcontrollers. The JTAG interface allows the microcontroller to be placed under the control of an external debugging system, and through the JTAG interface it's possible to examine the processor registers and its memory. The JTAG interface can be used to write and execute code on-the-fly in RAM. It gives anyone who can connect total control over the microcontroller.

The JTAG interface **can** usually be disabled after initial testing and manufacture. Many microcontrollers have a one-way fusable link which can be blown to then forever disable a microcontroller's JTAG interface… which is done exactly because it is SUCH a powerful and official standards-based backdoor into the operation of any microcontroller.

Yet only two out of the of the seven SSDs the researchers examined had their JTAG interfaces disabled. And in those that did have it disabled, the other models in the same line from the same manufactures had left it enabled.... So that their common secrets could be obtained.

For example, in one typical instance where one model of a device's firmware was not available for download, its ATA command table could be located through its still-functional JTAG interface. This allowed the researchers to learn of an undocumented "Vendor Specific" ATA instruction -- and the secret parameters it required -- which commanded the drive to export its own firmware. And from there, of course, the drive's secrets were available.

I'm not going to delve into the fundamental design mistakes which were discovered in each of the seven drives which these guys dissected. I've place a link in the show notes for anyone interested in their very nicely assembled 16-page research disclosure:

https://www.ru.nl/publish/pages/909275/draft-paper_1.pdf

But they did have some interesting general things to say about the question of hardware vs software solutions -- and also some salient recommendations which were driven by their discoveries. One the first point they write:

> An argument that is often put forward in favor of hardware encryption is that the secret key is not stored in RAM, and therefore is not vulnerable to the aforementioned attacks. In reality, this argument is invalid for several reasons:

> (i) The software running on the host PC controlling the hardware encryption, typically does keep a secret key in RAM, introducing the same vulnerability. The reason is to support Suspend-to-RAM (S3), a low-power state wherein all peripheral devices are shut down. Since the SSD is [completely] powered down, it must be unlocked again once the system is resumed, and therefore either the operating system must retain a copy of the secret key at all times, or the user must enter it again. In virtually all implementations, including BitLocker, the former approach is chosen.
>
> (ii) The burden of keeping the secret key is moved to the SSD, not eliminated. The SSD typically keeps the key in the main memory of its controller. SSDs are not security hardened devices by any standard. In fact, many have a debugging interface exposed on their PCB, allowing one to attach a debugging device and extract the secret key from the drive. Furthermore, several means of obtaining code execution on the drive exist.
>
> (iii) A memory readout attack against software encryption requires physical access. Given this, the attacker also has the opportunity to carry out a hot-plugging attack against hardware encryption. This has been demonstrated in practice and poses a realistic threat.

In their discussion at the end of the paper they say:

An overview of possible flaws in hardware-based full-disk encryption was given. We have analyzed the hardware fulldisk encryption of several SSDs by reverse engineering their firmware, with focus on these flaws. The analysis uncovers a pattern of critical issues across vendors. For multiple models, it is possible to bypass the encryption entirely, allowing for a complete recovery of the data without any knowledge of passwords or keys. Table I gives an overview of the models studied and the flaws found.

The situation is worsened by the delegation of encryption to the drive if the drive supports TCG Opal, as done by BitLocker. In such case, BitLocker disables the software encryption, relying fully on the hardware implementation. As this is the default policy, many BitLocker users are unintentionally using hardware encryption, exposing them to the same threats.

The results presented in this paper show that one should not rely solely on hardware encryption as offered by SSDs for confidentiality. We recommend users that depend on hardware encryption implemented in SSDs to employ also a software full-disk encryption solution, preferably an open-source and audited one. In particular, VeraCrypt allows for in-place encryption while the operating system is running, and can coexist with hardware encryption. Furthermore, BitLocker users can change their preference to enforce software encryption even if hardware encryption is supported by adjusting the Group Policy setting. However, this has no effect on already deployed drives. Only an entirely new installation, including setting the Group Policy correctly and securely erasing the internal drive, enforces software encryption. VeraCrypt can be an alternative solution for these existing installations, as it offers in-place encryptions.

It is important to ask ourselves what problem SEDs are actually trying to address. As described in Section III, SEDs do not offer any meaningful mitigations in situations where software encryption falls short. However, as demonstrated, in situations where software encryption offers full data confidentiality, hardware encryption often does not. Hence, at best, the security *guarantees of SEDs are similar to that of software encryption, and often much less. Finally, nowadays since the AES-NI extension on x86 CPUs has become mainstream, it seems legitimate to question the supposed performance and side-channel susceptibility benefits of SEDs as well.*

Hardware encryption currently comes with the drawback of having to rely on proprietary, non-public, hard-to-audit crypto schemes designed by their manufacturers. Correctly implementing disk encryption is hard and the consequences of making mistakes are often catastrophic. For this reason, implementations should be audited and subject to as much public scrutiny as possible. Manufacturers that take security seriously should publish their crypto schemes and corresponding code so that security claims can be independently verified.

A pattern of critical issues across vendors indicates that the issues are not incidental but structural, and that we should critically assess whether this process of standards engineering actually benefits security, and if not, how it can be improved. The complexity of TCG Opal contributes to the difficulty of implementing the cryptography in SEDs. From a security perspective, standards should favor simplicity over a high number of features. The requirements as specified by the Opal standard, having a many-to-many relation between passwords and keys, and allowing for multiple independent ranges with adjustable bounds, makes it very hard to implement it correctly.

Finally, TCG should publish a reference implementation of Opal to aid developers. This reference implementation should also be made available for public scrutiny. It should take into account that wear-leveling is applied for non-volatile storage. Opal's compliance tests should cover the implementation of the cryptography and these tests should be independently assessed.

So where does this leave us?

I would feel better using one of these drives once it's firmware can been audited and fixed than any OTHER drive that has not yet been scrutinized like this. And all of the drives which were subjected to their scrutiny HAVE now been updated. So it you have a Crucial MX100, MX200 or MX300, a Samsung 840 EVO or 850 EVO, or an external Samsung T3 or T5, significantly more secure firmware awaits you.

And we should note that these were the seven drives these guys examined and every one of them was found to be fatally flawed. So there's no reason to believe that the same situation is not universal… except in these drives after being updated.

Given what we know, until such time as self-encrypting drive firmware IS open, accessible, auditable... and audited... Anyone who is truly interested in uncrackable whole-drive encryption WOULD probably be best served by layering VeraCrypt into their system… or by switching Microsoft's BitLocker from hardware to software encryption.

Microsoft:
ADV180028 | Guidance for configuring BitLocker to enforce software encryption
https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV180028
Elevated Command Prompt:

"BDE: BitLocker Drive Encryption"
manage-bde.exe -status

<quote> Microsoft is aware of reports of vulnerabilities in the hardware encryption of certain self-encrypting drives (SEDs). Customers concerned about this issue should consider using the software only encryption provided by BitLocker Drive Encryption™. On Windows computers with self-encrypting drives, BitLocker Drive Encryption™ manages encryption and will use hardware encryption by default. Administrators who want to force software encryption on computers with self-encrypting drives can accomplish this by deploying a Group Policy to override the default behavior. Windows will consult Group Policy to enforce software encryption only at the time of enabling BitLocker.

To check the type of drive encryption being used (hardware or software):

- Run ‘manage-bde.exe -status’ from elevated command prompt.

- If none of the drives listed report "Hardware Encryption" for the Encryption Method field, then this device is using software encryption and is not affected by vulnerabilities associated with self-encrypting drive encryption.

For drives that are encrypted using a vulnerable form of hardware encryption, you can mitigate the vulnerability by switching to software encryption using Bitlocker with a Group Policy.

Note: After a drive has been encrypted using hardware encryption, switching to software encryption on that drive will require that the drive be unencrypted first and then re-encrypted using software encryption. If you are using BitLocker Drive Encryption, changing the Group Policy value to enforce software encryption alone is not sufficient to re-encrypt existing data.

IMPORTANT: You do NOT need to reformat the drive or reinstall any applications after changing BitLocker settings.

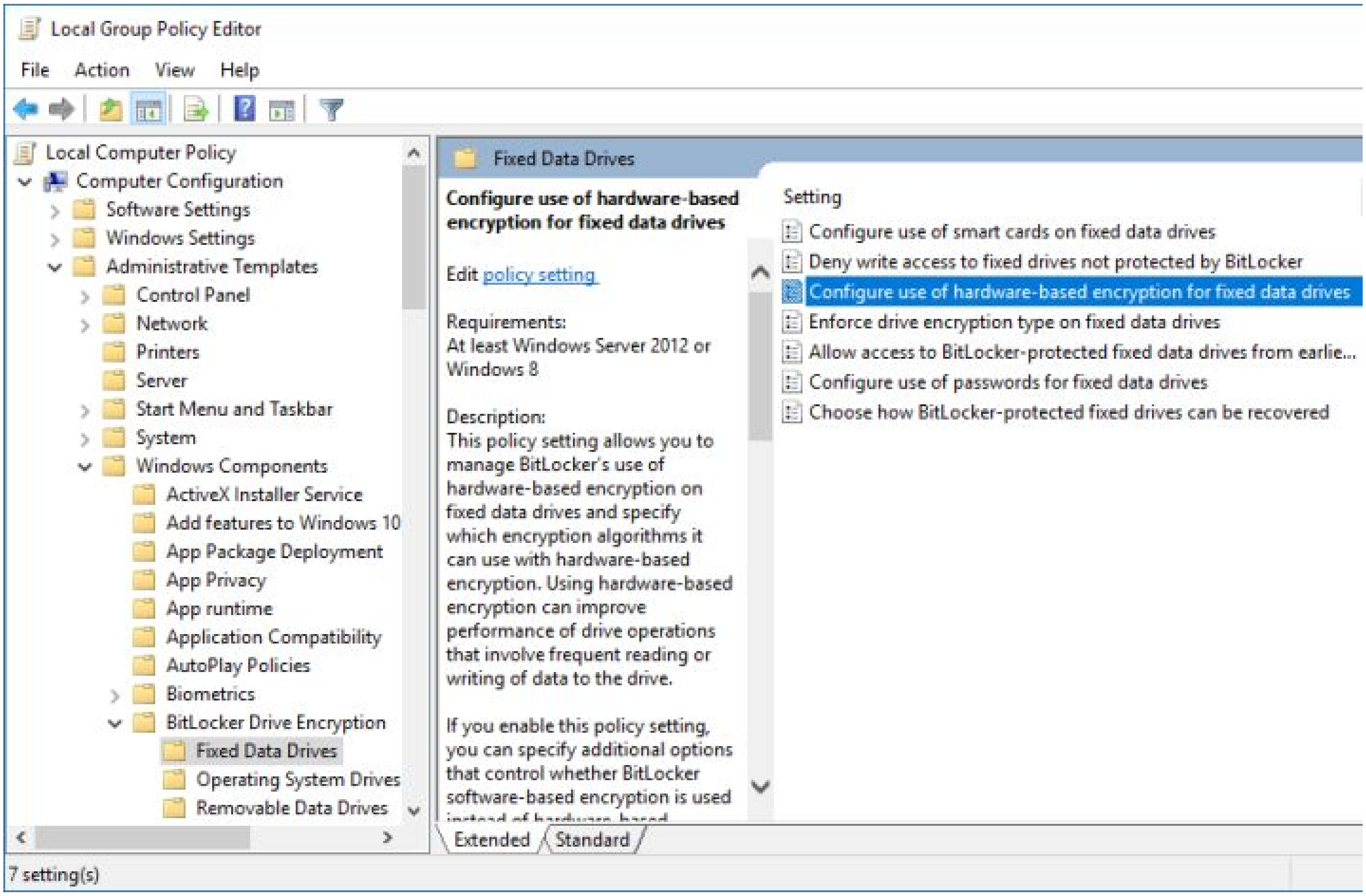
To mitigate vulnerabilities associated with self-encrypting drives on Windows systems:

- Configure and deploy a Group Policy to enable forced software encryption.
- Fully turn off BitLocker to decrypt the drive.
- Enable BitLocker again.

For more information on Bitlocker and Group Policy settings to enforce software encryption:

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj679890(v=ws.11)#configure-use-of-hardware-based-encryption-for-fixed-data-drives

Note that the required Group Policy settings are NOT available under Windows 7, only from Windows 8 on…



- Local Computer Policy
  - Computer Configuration
    - Software Settings
    - Windows Settings
    - Administrative Templates
      - Control Panel
      - Network
      - Printers
      - Server
      - Start Menu and Taskbar
      - System
      - Windows Components
        - ActiveX Installer Service
        - Add features to Windows 10
        - App Package Deployment
        - App Privacy
        - App runtime
        - Application Compatibility
        - AutoPlay Policies
        - Biometrics
        - BitLocker Drive Encryption
          - Fixed Data Drives
          - Operating System Drives
          - Removable Data Drives

Fixed Data Drives

Configure use of hardware-based encryption for fixed data drives

Edit policy setting

Requirements:
At least Windows Server 2012 or Windows 8

Description:
This policy setting allows you to manage BitLocker's use of hardware-based encryption on fixed data drives and specify which encryption algorithms it can use with hardware-based encryption. Using hardware-based encryption can improve performance of drive operations that involve frequent reading or writing of data to the drive.

If you enable this policy setting, you can specify additional options that control whether BitLocker software-based encryption is used instead of hardware-based

Setting
Configure use of smart cards on fixed data drives
Deny write access to fixed drives not protected by BitLocker
Configure use of hardware-based encryption for fixed data drives
Enforce drive encryption type on fixed data drives
Allow access to BitLocker-protected fixed data drives from earlie...
Configure use of passwords for fixed data drives
Choose how BitLocker-protected fixed drives can be recovered

7 setting(s)

The last page of this week's show notes shows the path through the Group Policy Tree and the setting that needs to be changed.