Security Now! #599 - 02-14-17

TLS Interception INsecurity

This week on Security Now!

Patch Tuesday DELAYED (and we may know why!), our favorite ad-blocker embraces the last major browser, a university gets attacked by its own vending machines, PHP leaps into the future, a slick high-end Linux hack, the rise of fileless malware, some good advice for tax time, it's not only Android's pattern lock that's vulnerable to visual eavesdropping, what happens with you store a huge pile of Samsung Note 7's in one place?, some fun miscellany, a MUST NOT MISS science fiction TV series, a look at the growing worrisome security implications of uncontrolled TLS interception.

The Best Picture Ever

Security News

Patch Tuesday DELAYED (and we may know why!)

- Recall from last week:
  - Security researcher Laurent Gaffie gave Microsoft 90-days to patch a flaw he found in Windows client processing of SMB (Server Message Block - Windows File and Printer Sharing) protocol handling.
  - Microsoft didn't.
  - So Laurent went public, releasing a Python Proof of Concept (PoC).
  - https://github.com/lgandx/PoC/blob/master/SMBv3%20Tree%20Connect/Win10.py
  - US-CERT warned that the vulnerability could do more than crash a system... it could also be exploited to execute arbitrary code with Windows kernel privileges.
  - It affects Windows 10 and v8.1 client systems and Windows Server 2012 & 2016.
  - The vulnerability is a "malicious server" denial of service where, when an innocent Windows client connects to a malicious SMB server, the server can return a malformed structure to crash the client.
- MSFT: February 2017 security update release
  - https://blogs.technet.microsoft.com/msrc/2017/02/14/february-2017-security-update-release/
  - MSRC TeamFebruary 14, 2017
    Our top priority is to provide the best possible experience for customers in maintaining and protecting their systems. This month, we discovered a last minute issue that could impact some customers and was not resolved in time for our planned updates today.

    After considering all options, we made the decision to delay this month's updates. We apologize for any inconvenience caused by this change to the existing plan.

uBlock Origin coming to Safari

- https://github.com/el1t/uBlock-Safari
- Slogan: "uBlock Origin -- YOU decide what enters your browser."
- Chromium:
  - Chrome & Opera (and other Chromium-based browsers)
- Firefox:
  - Firefox, FF for Android, SeaMonkey, Pale Moon, and other's based on FF.
  - Note that the Firefox version of uBlock Origin has an extra feature known as "Script tag filtering" which is currently not yet available on Chromium-based browsers due to Chromium API limitations. It is of great help to foil attempts by many web sites to circumvent blockers.
  - https://github.com/gorhill/uBlock/wiki/Inline-script-tag-filtering

    - Thanks to Debian contributor Sean Whitton, users of Debian 9 or later or Ubuntu 16.04 or later may simply apt-get install xul-ext-ublock-origin.

- Microsoft's Edge

- Now also Safari on MacOS

- Gorhill: uBlock Origin is NOT an "ad blocker": it is a wide-spectrum blocker -- which happens to be able to function as a mere "ad blocker". The default behavior of uBlock Origin when newly installed is to block ads, trackers and malware sites -- through EasyList, EasyPrivacy, Peter Lowe's ad/tracking/malware servers, various lists of malware sites, and uBlock Origin's own filter lists.

- Relies heavily upon volunteer curated block lists.

- SN #523 - uBlock Origin

University attacked by its own vending machines, smart light bulbs & 5,000 IoT devices

- http://www.verizonenterprise.com/resources/reports/rp_data-breach-digest-2017-sneak-peek_xg_en.pdf
- Verizon published a 4-page PDF describing their incident response in what they term their "Data Breach Digest"
- Written in 1st-parson narrative, it begins...

Senior members of my university's IT Security Team rotated weekly as on-call "Incident Commanders" in the event that a response was needed. This week was my turn and as I sat at home, my phone lit up with a call from the help desk. They had been receiving an increasing number of complaints from students across campus about slow or inaccessible network connectivity. As always seemed to happen, the help desk had written off earlier complaints and it was well after 9 PM when I was finally pulled in.

I joined the conference bridge and began triaging the information. Even with limited access, the help desk had found a number of concerns. The name servers, responsible for Domain Name Service (DNS) lookups, were producing high-volume alerts and showed an abnormal number of sub-domains related to seafood. As the servers struggled to keep up, legitimate lookups were being dropped—preventing access to the majority of the internet. While this explained the "slow network" issues, it raised much more concerning questions. From where were all these unusual DNS lookups coming from? And why were there so many of them? Were students suddenly interested in seafood dinners? Unlikely. Suspecting the worst, I put on a pot of coffee and got to work.

Now that I had a handle on the incident in general, I began collecting and examining network and firewall logs. The firewall analysis identified over 5,000 discrete systems making hundreds of DNS lookups every 15 minutes. Of these, nearly all systems were found to be living on the segment of the network dedicated to our IoT infrastructure. With a massive campus to monitor

and manage, everything from walkway light bulbs to vending machines had been connected to the network for ease of management and improved efficiencies. While these IoT systems were supposed to be isolated from the rest of the network, it was clear that they were configured to use DNS servers in a different subnet.

Of the thousands of domains requested, only 15 distinct IP addresses were returned. Four of these IP addresses and close to 100 of the domains appeared in recent indicator lists for an emergent IoT botnet. This botnet was known to spread from device to device by brute forcing default and weak passwords. Once the password was known, the malware had full control of the device and would check in with command infrastructure for updates and change the device's password – locking us out of the 5,000 systems.

This was a mess. Short of replacing every soda machine and lamp post, I was at a loss for how to remediate the situation. We had known repeatable processes and procedures for replacing infrastructure and application servers, but nothing for an IoT outbreak. Fortunately a less drastic option existed than replacing all the IoT devices on campus.

Analysis of previous malware samples had shown that the control password, used to issue commands to infected systems, was also used as the newly updated device password. These commands were typically received via Hypertext Transfer Protocol (HTTP) and in many cases did not rely on Secure Sockets Layer (SSL) to encrypt the transmissions. If this was the case for our compromise, a full packet capture device could be used to inspect the network traffic and identify the new device password. The plan was to intercept the clear text password for a compromised IoT device over the wire and then use that information to perform a password change before the next malware update. If conducted properly and quickly, we could regain control of our IoT devices.

While we waited for the full packet capture solution to be set up, I instructed the network operations team to prepare to shut down all network access for our IoT segments once we had intercepted the malware password. Short lived as it was, the impact from severing all of our IoT devices from the internet during that brief period of time was noticeable across the campus—and we were determined never to have a repeat incident.

With the packet capture device operational, it was only a matter of hours before we had a complete listing of new passwords assigned to devices. With these passwords, one of our developers was able to write a script, which allowed us to log in, update the password, and remove the infection across all devices at once. The whole process took a matter of minutes and I made a mental note to save that script for later–although I prayed that we would never need it again. Now that the incident had been contained, we looked towards ways to prevent it from happening again.

Lessons Learned:

- Don't keep all your eggs in one basket; create separate network zones for IoT systems;
- air-gap them from other critical networks where possible.
- Don't allow direct ingress or egress connectivity to the internet; don't forget the importance of an in-line proxy or content filtering system.
- Change default credentials on devices; use strong and unique passwords for device accounts and Wi-Fi networks.

- Regularly monitor events and logs; hunt for threats at endpoints, as well as at the network level; scan for open remote access protocols on your network and disable commonly unused and unsecured features and services (such as, UPnP, RTSP) that aren't required.
- Include IoT devices in IT asset inventory; regularly check manufacturer websites for firmware updates.

PHP 7.2: To add state-of-the-art cryptography to its standard library

- https://dev.to/paragonie/php-72-the-first-programming-language-to-add-modern-cryptography-to-its-standard-library

- Last week, the voting phase closed on an RFC to add libsodium to PHP 7.2. The result was unanimous (37 in favor, 0 against). When version 7.2 releases at the end of the year, PHP will incorporate state of the art cryptography in its standard library.

- Libmcrypt hasn't been touched in eight years (last release was in 2007), leaving OpenSSL as the only viable option for PHP 5.x and 7.0 users.

- By comparison, Libsodium is a modern state-of-the-art cryptography library offering authenticated encryption, high-speed elliptic curve cryptography, and much more. Unlike other cryptography standards (which are a potluck of cryptography primitives; i.e. WebCrypto), libsodium is comprised of carefully selected algorithms implemented by security experts to avoid side-channel vulnerabilities.

Wipe and reinstall a running Linux system via SSH, without rebooting.

- GitHub - marcan/takeover.sh: You know you want to.
- https://github.com/marcan/takeover.sh

- "takeover.sh" is a shell script to completely take over a running Linux system remotely, allowing you to log into an in-memory rescue environment, unmount the original root filesystem, and do anything you want, all without rebooting. Replace one distro with another without touching a physical console.

- NOT for the faint of heart…

- <QUOTE> This is experimental. Do not use this script if you don't understand exactly how it works. Do not use this script on any system you care about. Do not use this script on any system you expect to be up. Do not run this script unless you can afford to get physical access to fix a botched takeover. If anything goes wrong, your system will most likely panic.

  That said, this script will not (itself) make any permanent changes to your existing root filesystem (assuming you run it from a tmpfs), so as long as you can remotely reboot your box using an out-of-band mechanism, you should be OK. But don't blame me if it eats your dog.

This script does not have any provisions for exiting out of the new environment back into something sane. You will have to reboot when you're done. If you get anything wrong, your machine won't boot. Tough luck.

This is not a guide for newbies. I'm deliberately not giving you commands you can copy and paste. If you can't figure out what to do exactly without handholding, this script is not for you.

A rash of invisible, fileless malware is infecting banks around the globe

- https://arstechnica.com/security/2017/02/a-rash-of-invisible-fileless-malware-is-infecting-banks-around-the-globe/

- Dan Goodin writing for ArsTechnica:
  Two years ago, researchers at Moscow-based Kaspersky Lab discovered their corporate network was infected with malware that was unlike anything they had ever seen. Virtually all of the malware resided solely in the memory of the compromised computers, a feat that had allowed the infection to remain undetected for six months or more. Kaspersky eventually unearthed evidence that Duqu 2.0, as the never-before-seen malware was dubbed, was derived from Stuxnet, the highly sophisticated computer worm reportedly created by the US and Israel to sabotage Iran's nuclear program.

  Now, fileless malware is going mainstream, as financially motivated criminal hackers mimic their nation-sponsored counterparts. According to research Kaspersky Lab plans to publish Wednesday, networks belonging to at least 140 banks and other enterprises have been infected by malware that relies on the same in-memory design to remain nearly invisible. Because infections are so hard to spot, the actual number is likely much higher. Another trait that makes the infections hard to detect is the use of legitimate and widely used system administrative and security tools—including PowerShell, Metasploit, and Mimikatz—to inject the malware into computer memory.

  "What's interesting here is that these attacks are ongoing globally against banks themselves," Kaspersky Lab expert Kurt Baumgartner told Ars. "The banks have not been adequately prepared in many cases to deal with this." He went on to say that people behind the attacks are "pushing money out of the banks from within the banks," by targeting computers that run automatic teller machines.

  The 140 unnamed organizations that have been infected reside in 40 different countries, with the US, France, Ecuador, Kenya, and the UK being the top five most affected nations. The Kaspersky Lab researchers still don't know if a single group of individuals is behind the attacks, or if they're being carried out by competing hacker gangs. The use of the fileless malware and command-server domains that aren't associated with any whois data makes the already difficult task of attribution almost impossible.

In general we're seeing an increase in "in memory" compromise.

- Firmware may not be safely writable without crashing the device, so it lives "opportunistically" in RAM.
- Many systems are "appliances" that are rarely, if ever, power cycled.
- Servers typically remain up between patch cycles.
- Routers may remain up for years.

Scanning any of these devices' file systems will reveal nothing.

Network behavior monitoring is the only viable solution... and that's no easy on today's incredibly noisy networks.

Encrypt Your Tax Documents BEFORE You Send Them

- http://cantus.us/encrypt-your-tax-documents-before-you-send-them/

- It's that time. We all are doing our taxes online now. Some of us need some help and that usually ends up with digitally sending tax documents.

  Do not ever, under any circumstance, email a document that has your SSN, home address, phone number, bank information, etc. unencrypted. That's a good way to lose all of your money and even your identity.

  I'm going to use two free services to send my encrypted documents out: AESCrypt and Dropbox.

- Step 1: Install AESCrypt
  Get AESCrypt installed on your computer. Go here and install for your platform.

- Step 2: Zip up all of those files
  You want to send one encrypted blob of data. Put all of your files into a single folder and zip that up. On the Mac, select the file and go to File >> Compress "Sensitive Docs".

- Step 3: Encrypt
  Follow the directions for your platform to encrypt your zip file. On the Mac, drag the file onto the dock icon and enter a strong password when prompted.

  I suggest using a secure password generator like the one over at GRC.com. You can also use Lastpass or whatever you use to manage your passwords. Make a note of that password though. You can keep a copy of the password in TextEdit or NVAlt – the documents are already on your computer.

- Step 4: Drop it in Dropbox*
  Using DropBox makes sharing files easy. Put the encrypted file into your dropbox folder and click Share Link.

  You can send them the file directly through the DropBox website or get the link and send it separately.

- Step 5: Send the Password via a Separate Method
  If you send an email with a link to your encrypted document, it doesn't make sense to put the password into that email. Additionally, you might not want to send the password from your account to that same account. Try to send the password over iMessage, Signal, or another instant messenger, or to a secondary email address the person has.

  You could also make the password something you could read to them over the phone, "uppercase J, lowercase z, six, five …"

- Step 6: The Takedown
  Once the recipient has confirmed that they have the documents and have decrypted them on their own machine you will remove the encrypted documents from DropBox. Sure they are safe, but there is no reason to keep that online.

- About AESCrypt
  - I recently referred to AxCrypt when I meant to say AEScrypt.
  - AxCrypt has gone commercial with a subscription plan... for an encryption utility.
  - Bye bye, AxCrypt.
  - AEScrypt is free and open source and available on WIndows, Mac, Linux, iOS, Android, PHP and Java.
  - https://www.aescrypt.com/

Sam Cox (@LordZarano) / 2/8/17, 8:40 AM

- @SGgrc Numeric passcodes are also entered on a 3x3 grid, so aren't they also vulnerable to the android pattern unlock attack?

Samsung battery factory bursts into flame in touching Note 7 tribute • The Register
https://www.theregister.co.uk/2017/02/08/samsung_battery_factory_catches_fire/
No one was hurt.

Takeaway: When you are recalling a huge number of batteries because they are prone to spontaneous combustion, it would be best not to store them in what amounts to a large Fukushima pile.

Miscellany

Nomikos Zografakis (@nzogra) / 2/12/17, 9:09 AM

- @SGgrc #pfSense allows one to activate upnp and allow only user-specified IP addresses on the LAN to open ports.
- UPnP Access Control Lists
    - These entries control access to the UPnP service. Client systems may be granted or denied access based on several criteria.

    - Format: [allow or deny] [ext port or range] [int ipaddr or ipaddr/CIDR] [int port or range]
    - Example: allow 1024-65535 192.168.0.0/24 1024-65535
    - + Add

Awful pun of the week...

- My 103 year old grandfather:
    - "I opened the windows and influenza."
    - "If the rain keeps up it won't come down."
- Chris Heilmann (@codepo8) / 12/21/16, 1:33 PM
    - You can't use "beefstew" as a password. It isn't Stroganoff.

Metaphysics and Scrodenger's Cat meets Cryptography...

- "The sound of one hand clapping" and "If a tree falls in the woods..."
- If you encrypt data with a key that is so strong that would take more than all of the energy in the universe to crack, and the key is then destroyed... does the data continue to exist?

Chris Fowlkes (@MyGhostWorld) / 2/9/17, 8:37 PM

- @SGgrc After watching new episodes of the Expanse season 2,
- VERY realistic space combat...makes Star Trek look very silly and childish!
- Season One on Amazon Prime.
- Season Two -- stunning MUST SEE!!

SpinRite

From: David Goldenberg

Hi Steve,

I've been a happy owner of SpinRite for a few years now, it's my secret weapon in the technology trenches. I am the family tech guy, I help out at my kids school, and have my own part time business fixing PC's, training and networking. SpinRite is always within reach, and never lets me down.

Last week I had been preparing a laptop for a presentation for my ARES Amateur Radio group, I volunteered to get a new program running to send text messages and email type communications over the radio. After several days, I had everything working great, and spent several hours getting screenshots for the PowerPoint I was to prepare.

I was getting together with another HAM to go over what I had and to get his machine working, so I started up my laptop and got the dreaded BSOD and an un-mountable boot volume. I did not break a sweat, or even worry, as I knew from experience that SpinRite would save the day, and needless to say, two hours later the drive scanned, several sectors were repaired, the laptop booted perfectly and everything I needed was ready to go!

You're great! THANKS!

David Goldenberg
KJ6MCQ

The Security impact of TLS interception

https://jhalderm.com/pub/papers/interception-ndss17.pdf

University of Michigan, University of Illinois, Mozilla, Cloudflare, Google, UC Berkeley, International Computer Science Institute.

Abstract:
As HTTPS deployment grows, [corporate] middlebox and antivirus products are increasingly intercepting TLS connections to retain visibility into network traffic. In this work, we present a comprehensive study on the prevalence and impact of HTTPS interception. First, we show that web servers can detect interception by identifying a mismatch between the HTTP User-Agent header and TLS client behavior.

[In other words, current TLS interceptors are not bothering to mask their presence... though they certainly could.]

We characterize the TLS handshakes of major browsers and popular interception products, which we use to build a set of heuristics to detect interception and identify the responsible product. We deploy these heuristics at three large network providers: (1) Mozilla Firefox update servers, (2) a set of popular e-commerce sites, and (3) the Cloudflare content distribution network.

We find more than an order of magnitude more interception than previously estimated and with dramatic impact on connection security.

To understand why security suffers, we investigate popular middleboxes and clientside security software, finding that nearly all reduce connection security and many introduce severe vulnerabilities. Drawing on our measurements, we conclude with a discussion on recent proposals to safely monitor HTTPS and recommendations for the security community.

Introduction:
When it comes to HTTPS, the security community is working at cross purposes. On the one hand, we are striving to harden and ubiquitously deploy HTTPS in order to provide strong end-to-end connection security. At the same time, middlebox and antivirus products increasingly intercept (i.e., terminate and re-initiate) HTTPS connections in an attempt to detect and block malicious content that uses the protocol to avoid inspection. Previous work has found that some specific HTTPS interception products dramatically reduce connection security; however, the broader security impact of such interception remains unclear. In this paper, we conduct the first comprehensive study of HTTPS interception in the wild, quantifying both its prevalence in traffic to major services and its effects on real-world security.

We begin by introducing a novel technique for passively detecting HTTPS interception based on handshake characteristics. HTTPS interception products typically function as transparent proxies: they terminate the browser's TLS connection, inspect the HTTP plaintext, and relay the HTTP data over a new TLS connection to the destination server. We show that web servers can detect such interception by identifying a mismatch between the HTTP User-Agent header and the behavior of the TLS client. TLS implementations display varied support (and preference order)

for cipher suites, extensions, elliptic curves, compression methods, and signature algorithms. We characterize these variations for major browsers and popular interception products in order to construct heuristics for detecting interception and identifying the responsible product.

Next, we assess the prevalence and impact of HTTPS interception by applying our heuristics to nearly eight billion connection handshakes. In order to avoid the bias inherent in any single network vantage point, we analyzed connections for one week at three major Internet services: (1) Mozilla Firefox update servers, (2) a set of popular e-commerce websites, and (3) the Cloudflare content distribution network. These providers serve different types of content and populations of users, and we find differing rates of interception: 4.0% of Firefox update connections, 6.2% of e-commerce connections, and 10.9% of U.S. Cloudflare connections were intercepted. While these rates vary by vantage point, all are more than an order of magnitude higher than previous estimates.

To quantify the real-world security impact of the observed interception, we establish a grading scale based on the TLS features advertised by each client. By applying the metric to unmodified browser handshakes and to the intercepted connections seen at each vantage point, we calculate the change in security for intercepted connections. While for some older clients, proxies increased connection security, these improvements were modest compared to the vulnerabilities introduced: 97% of Firefox, 32% of e-commerce, and 54% of Cloudflare connections that were intercepted became less secure. Alarmingly, not only did intercepted connections use weaker cryptographic algorithms, but 10–40% advertised support for known-broken ciphers that would allow an active man-in-the-middle attacker to later intercept, downgrade, and decrypt the connection. A large number of these severely broken connections were due to network-based middleboxes rather than client-side security software: 62% of middlebox connections were less secure and an astounding 58% had severe vulnerabilities enabling later interception.

Finally, we attempt to understand why such a large number of intercepted connections are vulnerable by testing the security of a range of popular corporate middleboxes, antivirus products, and other software known to intercept TLS. The default settings for eleven of the twelve corporate middleboxes we evaluated expose connections to known attacks, and five introduce severe vulnerabilities (e.g., incorrectly validate certificates). Similarly, 24 of the 26 client-side security products we tested reduce connection security, and two thirds introduce severe vulnerabilities. In some cases, manufacturers attempted to customize libraries or re-implement TLS, introducing negligent vulnerabilities. In other cases, products shipped with libraries that were years out of date. Across the board, companies are struggling to correctly deploy the base TLS protocol, let alone implement modern HTTPS security features.

Our results indicate that HTTPS interception has become startlingly widespread, and that interception products as a class have a dramatically negative impact on connection security. We hope that shedding light on this state of affairs will motivate improvements to existing products, advance work on recent proposals for safely intercepting HTTPS and prompt discussion on long-term solutions.

Takeaway points:

- We need community consensus.
  There is little consensus within the security community on whether HTTPS interception is acceptable. Discussions over protocol features that facilitate safer interception have been met with great hostility within standards groups. These communities need to reach consensus on whether interception is appropriate in order to develop sustainable, long-term solutions.

- Antivirus vendors should reconsider intercepting HTTPS.
  Antivirus software operates locally and already has access to the local filesystem, browser memory, and any content loaded over HTTPS. Given their history of both TLS misconfigurations and RCE vulnerabilities, we strongly encourage antivirus providers to reconsider whether intercepting HTTPS is responsible.

- There have been some efforts to develop protocols that require explicit client-side notification when a "middlebox", A/V, or TLS proxy of any kind is intercepting.

Remember... so long as the intercepting proxy does NOT have the ability to on-the-fly synthesize certificates trusted by the publicly trusted root store -- and NO CA willingly (or at least publicly) allows this or THEY would become untrusted -- this can ONLY be done with the knowledge of the user's browser client which is trusting the public key installed into it by the proxy's vendor. So notification could be given whenever a non-globally trusted certificate is being used for any connection.