Security Now! #791 - 11-03-20

Chrome's Root Program

This week on Security Now!

This week we examine a serious newly revealed Windows 0-day flaw, a public service reminder from Microsoft, Google's newly announced plan to get into the VPN service business, CERT's unappealing plan for automatic vulnerability naming, and a real mess that WordPress just made of an incremental security update to 455 million sites. Then we'll close a loop, I'll update about SpinRite, and we'll finish by examining Google's new plan to go their own way with a new Chromium browser certificate Root Store.

Security News

A new 0-day in Win7 through Win10

Last week we talked at some length about the bug Google found in the FreeType library which had been in use since June 19th, 2015, so for more than 5 years. What we knew then was that this flaw, which was patched by that update to Chrome, was a 0-day because it was being actively exploited.

What we did not learn until the end of last week, was that there was a previously secret, second part to this 0-day. The FreeType flaw that was being exploited through Chrome only opened the door to the attacker. But as is often the case thanks to modern operating system design, the damage that can be done by an aberrant application running under the non-root user account is deliberately minimal. And all of today's web browsers are careful to run under the user's deliberately-limited account's privileges. This is why successful attacks and attackers often need to chain two or more exploits together to accomplish their nefarious ends. IF the FreeType library can in the kernel then a single exploit of it might have been sufficient. But FreeType was also properly designed to run in user space.

So, exploiting the FreeType flaw opened the door. But the attacker then needed to elevate their privilege on the system to root or kernel level. The week before, Google had seen the whole picture. They saw a second phase which was leveraging a previously completely unknown and quite potent 0-day flaw in Windows to achieve privilege elevation. This was allowing the attackers to do some real damage. The privilege elevation they discovered, by watching it in action, existed — or I should say EXISTS because it still does today — within the Windows kernel-based cryptographic services API. And because that kernel-based module exports an API that's callable from userland, the bad guys had arranged to run their malicious code with full system permissions.

Google's Project Zero folks immediately reached out to Microsoft to inform them of what they had found and also to explain that since this was an active vulnerability being exploited in the wild, Project Zero's normal, patient, 90-day disclosure window would be reduced to just one week. And that's why the industry subsequently learned of this late last week. Seven days after Google told Microsoft.

https://bugs.chromium.org/p/project-zero/issues/detail?id=2104

The Project Zero Day disclosure starts off with: “NOTE: We have evidence that the following bug is being used in the wild. Therefore, this bug is subject to a 7 day disclosure deadline.”

And they begin their write up by explaining: “The Windows Kernel Cryptography Driver (cng.sys) exposes a \Device\CNG device to user-mode programs and supports a variety of IOCTLs with non-trivial input structures. It constitutes a locally accessible attack surface that can be exploited for privilege escalation (such as sandbox escape).”

Microsoft, for their part, doesn't see this as an emergency. Google has already closed and locked the front door through which attackers were able to reach the crypto API vulnerability. And November's Patch Tuesday is next Tuesday, at which time everyone expects that Microsoft will have this fixed. Unfortunately, the detailed nature of this flaw is no secret, since the Project Zero

guys have published proof-of-concept code which can be used, as a demonstration, to immediately crash any Windows system.

The other unfortunate reality is that this is known to also afflict Windows 7 machines which will not, as we know, ever receive a patch for this, unless they're within an enterprise that is now purchasing security for their older machines from Microsoft. The rest of Windows 7 users will be left to fend for ourselves.

To that end, although it hasn't happened yet, we can hope and expect that the micro-patch folks a 0patch.com have been watching and will again be able to offer one of their cute little "patchlets" for this. What we also know is that there will also be a massive inventory of Windows 8 and 10 machines which, even in the face of readily available patches, will also somehow remain unpatched for the longer term.

Which brings us back to ZeroLogon...

A public service reminder from Microsoft

In a public service reminder from Microsoft last Thursday, responding to the ongoing exploitation of the very serious ZeroLogon vulnerability — which, we noted, had been quickly adopted and was being leveraged in our "Anatomy of a Ryuk Attack" podcast two weeks ago, Microsoft Security Response tweeted:

> Reminder to all our Windows customers to deploy at least the August 2020 update or later and follow the original, published guidance to fully resolve the vulnerability, CVE-2020-1472. For further information, see our blog post: https://t.co/br77bEP0mu
> — Security Response (@msftsecresponse) October 29, 2020

"Google One" adding an Android VPN

Google has just announced that they'll be getting into the VPN business. They'll be adding a no-additional-charge VPN facility, first for Android users only, to their existing paid "Google One" service. For those who don't know, Google One is the $10/month or $100/year expansion of Google cloud storage from 15 GB to 2 TB.

Because Google is Google, the announcement of them doing a VPN has been met with well-deserved eye rolls. But they really are saying all the right things, and they are bringing some new technology to bear to back it up. Here's a bit of how they're thinking about this, their mission statement, and how they intend to be different:

> Demand for VPNs is growing, with evidence that it's becoming more mainstream -- up to 25% of all Internet users accessed a VPN within the last month of 2019. Unfortunately, not all VPN providers have been proven to be trustworthy: some services are vulnerable , others request unnecessary access to their users' data or monetize the same data that users are utilizing the VPN to keep private and secure, while others fail to deliver on the promise of not logging their users' online activity.

With growing demand for better privacy in a mixed landscape of solutions, we have used our expertise in privacy, cryptography, and infrastructure to build a Google-grade VPN that provides additional security and privacy to online connectivity without undue performance sacrifices. With VPN by Google One, users' online activity is not identifiable to the VPN and not logged by the VPN. We believe a VPN must be transparent, and robust. That's why we have open sourced our client and will provide a third party audit of the end-to-end solution to make them externally verifiable.

Privacy is at the core of the products and services we build. With VPN by Google One, we will never use the VPN connection to track, log, or sell your online activity. Some minimum logging is performed to ensure quality of service, but your network traffic or IP associated with the VPN is never logged. To demonstrate how our design works, we have open sourced the code that runs on a user's device and in the coming months we will be open sourcing the server side user authentication mechanism as well as providing the results of a third party audit, currently underway. These will provide further assurances of how user data is handled and how robust the VPN's security is.

Open sourcing our VPN and providing an audit are just some of the steps we are taking to ensure user privacy. While building VPN by Google One we realized it was important to strengthen some of the systems that are often attacked or compromised in order to access users' personal data. Traditional VPNs can sometimes compromise a users' identity or online activity by linking the usage of their service to the activity they conduct by means of a session ID. This ID could allow VPN operators, or attackers that compromise their infrastructure, to "eavesdrop" and identify users' and their activity.

We wanted to eliminate that vulnerability by separating the authentication of the subscriber from their use of the service. By employing a cryptographic blinding step between user subscription validation and connecting to the VPN, we give users a stronger guarantee that their online activity won't be tied back to their identity.

'RSA Blind Signing' which Google will be using is a real thing. It provides a means for Google to verify that a Google One account holder has the right to use their service without revealing who that account holder is. We've never discussed blind signing technology, but I think it would make for a terrific topic. And although Google hasn't said so yet publicly, the word on the street is that this service will be extended beyond Android and eventfully offered widely.

I love the idea that their VPN itself is blinded to its user's identity as an account holder. And, frankly, I think that Google probably had no choice but to do that if they wanted to provide a VPN service. But we all know that a user's browser's Google-tagged cookies, upon emerging from the Google VPN endpoint, will immediately scoot over to the nearest Google server to report in. So it's not as though the VPN provides any additional anonymity compared with any other VPN service. But... and here's the key, I think: Google's use of cryptographic account authentication blinding means that at least it doesn't provide any less anonymity than the use of any other VPN service. There's a lot more to be said about this. We'll be covering it in much greater detail soon.
https://www.gstatic.com/subscriptions/marketing_page/vpn/white_paper_4f995ab5d7c7edc3d3f14f2e0593f790.pdf

What's in a name?
Introducing our new "Dumb Idea of the Week" section, we have a proposal from CERT.

I think that giving catchy, sometimes humorous and descriptive names — with matching graphic images — to security flaws is part of the fun of this industry. We've had Spectre, Meltdown, Dirty Cow, ZeroLogon, Heartbleed, BlueKeep, BLESA, SIGRed, BLURTooth, DejaBlue, Stagefright and who could ever forget the Honey Monkeys? (Though, Honey Monkeys was more a technique than a vulnerability.) Naming thing is, I would contend, important. If nothing else, having at least nominally topical names makes them easy to remember. But not everyone agrees. In a blog posting Friday, the original CERT, operating out of Carnegie Mellon University, which now collaborates and partners with the DHS' official US-CERT team, has proposed spoiling this particular bit of fun.

https://insights.sei.cmu.edu/cert/2020/10/vulnonym-stop-the-naming-madness.html

I first noted that CERT's own blog posting was titled: "Vulnonym: Stop the Naming Madness!" But "Vulnonym" is, itself, a fun and memorable name which they gave to their newly proposed auto-vulnerability-naming service... even though, now, they don't want the rest of us to use fun names. So anyway... here's what they said:

> Spectre. Meltdown. Dirty Cow. Heartbleed.
>
> All of these are vulnerabilities that were named by humans, sometimes for maximum impact factor or marketing. Consequently, not every named vulnerability is a severe vulnerability despite what some researchers want you to think. Sensational names are often the tool of the discoverers to create more visibility for their work. This is an area of concern for CERT as we attempt to reduce any fear, uncertainty, and doubt for vendors, researchers, and the general public.
>
> [I suppose that I would argue that the general public has relatively low exposure to these names. No one knows what Meltdown is. Sounds like something that once happened in Chernobyl or when you leave your ice cream cone unattended. Anyway...]
>
> Software vulnerabilities are currently catalogued by number, primarily the Common Vulnerabilities and Exposures (CVE) ID, which makes it very easy for computer analysis and storage. However, humans aren't well conditioned to remember numbers. Instead, humans prefer names because we find them easier to remember. [Now that I certainly agree with. As a podcaster, CVE's are awful.] We don't remember IP addresses, but do easily remember domain names to browse to our favorite websites. We also name things like hurricanes, snow storms, operating system updates, particular geographic locations like cities or states, and so on. They all are named because it's easier to remember Mojave instead of Mac OS 10.14, or Pittsburgh instead of 40.4406° N by 79.9959° W.
>
> Names of vulnerabilities, in particular, are matriculating into important spheres of influence. Case and point, on July 11, 2018, congressional testimony weighed the impacts of the "Meltdown" and "Spectre" vulnerabilities. The CVE-IDs, CVE-2017-5753, CVE-2017-5715 and CVE-2017-5754, were never mentioned, only the sensational names were.

We aren't arguing that vulnerabilities shouldn't have names, in fact, we are encouraging this process! Our goal is to create neutral names that provide a means for people to remember vulnerabilities without implying how scary (or not scary) the particular vulnerability in question is. Our neutral names are generated from the CVE IDs to provide a nice mapping between name and number. CERT decided that if we can come up with a solution to this problem, we can help with discussions about vulnerabilities as well as mitigate the fear that can be spread by a vulnerability with a scary name. We plan to name the vulnerabilities with a phrase of adjective noun, for example, Arbitrary Albatross. [Yeah, or how about: "Stupid Idea"]

When tackling this problem, we considered several lists of words to ensure no sensational, scary, or offensive names were included. We created the list of both adjective and nouns using the combined resources of the wikitionary and categories of words such as animals, plants, objects in space, and more. Next, we created the method by which we map the CVE-IDs to the pair of adjective names. After much consideration, we used the Cantor Depairing Function, which is a bijection between the natural numbers and a pair of natural numbers. This means that each natural number can be mapped to two natural numbers uniquely.

To test out this idea, we're operating @vulnonym on Twitter to publish the neutral names associated with CVE IDs as they are issued. Follow @vulnonym and let us know if this naming experiment is useful! And in case anyone considers a word or name to be offensive, we have a simple process to remove it from the corpus and re-generate a name.

So I checked out @Vulnonym. I'm not sorry that I brought all this up, since we may start seeing some very poorly named vulnerabilities and we should understand what happened there. Here are three tweets made by the @Vulnonym tweet bot:

- CVE-2020-4785 is called Whacking Mouflon.

- Hi, I'm CVE-2020-4649. I was never good with numbers though, so you can call me Unmatched Cwm.

- Let the annals of the day show that CVE-2014-0160... has been granted the moniker Cyclic Hyrax.

Wow! The good news is that vulnerability discoverers can just ignore all this and will not be compelled to use CERT's system which provides "Unmatched Cwm's."

I looked up "Cwm". It is a word, kinda. It's used predominantly in Wales: It means "a steep-sided hollow at the head of a valley or on a mountainside; a cirque. Maybe that word comes in handy when one's in Wales.

WordPress fumbles an important update

Last Friday, WordPress patched 10 security bugs as part of the release of v5.5.2. The most severe problem patched would have allowed a remote unauthenticated attacker to take over a targeted website through what was described as a "narrowly tailored" denial-of-service attack... whatever that means. WordPress write that: "The vulnerability allows a remote attacker to

compromise the affected website. The vulnerability exists due to improper management of internal resources within the application, which can turn a denial of service attack into a remote code execution issue."

The researcher who found the bug described it as interesting but likely difficult to carry out in the wild. He said: "You have to be able to produce a very accurate DoS attack. The principle is to trigger a denial of service on MySQL so that WordPress will think that it's not installed and then un-DoS on the DB under the same execution thread.

Version 5.5.2 also brought a bunch of feature enhancements. WordPress described the update as a "short-cycle security and maintenance release" to fill-in before the next major release, which will be v5.6. With the update, all versions since WordPress 3.7 will also be current.

So that was how WordPress hoped things would go with the update they were beginning to push out to [get this!] 455 Million sites!

But it was soon discovered that v5.5.2 was badly broken... which is really not what you want to have from an update that's being automatically pushed out to 445 million installations. It turned out that 5.5.2 was causing new WordPress installs — based upon that release — to fail. As soon as WordPress became aware of the mistake that put the brakes on its rollout... but they screwed that up too by inadvertently triggering an unreleased and not debugged Alpha version of WordPress to be downloaded to some customers.

The first thing that happened was that WordPress site operators began reporting that new WordPress installs were failing, and others complained about broken admin login pages.

WordPress said: "v5.5.2 caused an issue with installing ZIP packages available on WordPress.org for new versions of 5.5.x, 5.4.x, 5.3.x, 5.2.x, and 5.1.x. The issue only affected fresh WordPress installations without an existing wp-config.php file in place."

WordPress further explained that "While work was being done to prepare for WordPress 5.5.3, the release team attempted to make 5.5.2 unavailable for download on WordPress.org to limit the spread of the issue noted, the one that only affected new installations. But this action resulted in some installations being updated to a pre-release '5.5.3-alpha' version."

Unfortunately, the alpha release mis-installation brought with it the old default "Twenty" themes and the "Akismet" plugin as part of the pre-release 5.5.2-alpha package. So that messed things up. And admins, who had not asked for any pre-release install were suddenly being greeted with the message that: "BETA TESTERS: This site is set up to install updates of future beta versions automatically." What!?!

In the wake of all this un-asked-for auto-update mess, many WordPress admins were vocally worried and upset about the whole not-in-their-control issue. We, of course, were just talking about this last week. I took the position that anything hooked to the Internet needed to have the capability of being automatically updated. But in light of all this, and standing back from it a bit, perhaps Microsoft's semi-compromise stance with Windows 10 is the best we can do. Notify everyone of updates. Allow them to be deferred until a more convenient time. But ultimately force the issue if necessary.

The problem is, an operating system CAN notify its users. There's no clear way for someone's router, or most other IoT devices, to notify them. I think we're going to need to have some standards defined and adopted so that we can move this whole dilemma into history.

As we know, my real complaint with Microsoft and Windows 10 is that they have taken a path where they are now never leaving it alone. They keep breaking things that have worked for years. If you never leave a massively complex software system alone, so that it can settle down and stabilize, it's going to be a constant patch fest.

Closing the Loop

Spencer Salmon @SpencerTSalmon
Hello! I just finished last week's podcasts and you mentioned IE and Edge. Interesting fact, I work in the financial industry and our core provider's web app is **only** compatible with IE (ActiveX). Through group policy, I'm able to force that web app to open in Edge's IE mode and I have an XML file that Edge references for using IE mode. I'm happy to see Microsoft get proactive about getting everyone over. Love the show, I've never missed an episode.

SpinRite

The past week went into overhauling a bunch more of the earlier work on the pre-AHCI IDE/ATA driver code. The way I had written that first-cut exploratory code back in 2013, it wasn't ever intended to be production ready. But the way this project is evolving, the sooner I can make it production ready the sooner we'll have SpinRite. So, as I said last week, I've been realizing that I have an opportunity to pre-test pretty much all of what will become SpinRite's new foundation within the context of an interesting and surprising low-level benchmark. So I'm taking that opportunity now to get this code as ready for release as I can.

Chrome's Root Program

A little less than two weeks ago, a new page appeared at Chromium.org titled: "Chrome Root Program." It was not met with universal joy.

https://www.chromium.org/Home/chromium-security/root-ca-policy

This podcast has covered the operation of SSL/TLS web server trust certificates at great length because its proper operation and functioning is crucial to enabling users and their web browsers to establish a trust relationship with otherwise unknown remote web servers. As we well know, the system is far from perfect with a myriad of well-known failure modes. It relies upon several different actors, each performing their jobs perfectly, where failure to do so results in a local breach of the privacy and security which is the system's entire purpose. But, for better or for worse, it's the system we have today.

From the beginning, Chrome and most other browsers have all relied upon the connection security and certificate verification provided by whatever operation system they were running on top of. The notable exception to this has been Mozilla and Firebox. To their credit, Netscape with their Netscape Navigator invented SSL and the concept of using public key cryptography to provide both server authentication and connection-contents privacy. Over time, this evolved into NSS, Mozilla's Network Security Services. NSS is the SSL/TLS library which Firefox uses to provide all of its connection security. Since NSS is cross-platform and a freestanding component of Firefox, it includes its own root certificate store which anchors the validation chain of any certificate received from a connecting web server.

All other web browsers, which inherently have a shallower history than Netscape's Navigator and Mozilla's Firefox, rely, as I mentioned, upon the hosting OS's platform for connection security. But how many times have I noted that today's modern cryptography is a solved problem? What was once decidedly regarded as highly complex "better not mess with it" magic crypto from some ivory tower guru is now run-of-the-mill. So, the barrier to entry of bringing up a new TLS communications foundation from scratch is as low as it's ever been... and that's quite low.

Against that backdrop, Google's new "Chrome Root Program" shouldn't surprise anyone. Putting it simply, Google also wants control over this aspect of Chrome's operation which it has, until now, delegated to the hosting OS. It was able to operate on the sidelines, blacklisting and pinning certificates. But it's never been in the position to directly and completely manage the Root Store which underlies its browser's trust. And knowing Google, that's gotta chafe.

However, running a Root Store program is a significant responsibility, since who you need to very very carefully decide who you'll let in, who you won't, and who you may eventually need to kick out. These decisions directly affect your customers' security and their ability to get to where they want to go. In years past, we've covered some of the torturous decisions that browser and OS vendors have had to make when, for example, deciding to ban StartCOM certs after, by any measure, they were not being sufficiently responsible. From its position on the sidelines, Google could sniff the certificate exchange and block connections using certificates which chained up to the StartCOM root. But they could not directly remove the StartCOM cert from the underlying OS since the OS's Root Store was not theirs to manage. So, really, does it come as a surprise to anyone that Google plans to now take over this aspect of connection security for Chromium?

Their low-key announcement of their intention to develop and run their own Root Store program establishes the important of the browser's root store with this short description:

> *When Chrome presents the connection to a website as secure, Chrome is making a statement to its users about the security properties of that connection. Because of the CA's critical role in upholding those properties, Chrome must ensure the CAs who issue certificates are operated in a consistent and trustworthy manner. This is achieved by referring to a list of root certificates from CAs that have demonstrated why continued trust in them is justified. This list is referred to as a Root Store. The policies and requirements for participating and being included in a Root Store are known as a Root Program.*

For long standing, tried and true certificate suppliers with a time-proven track record and impeccable credentials—like my own favored provider, DigiCert—inclusion in Google's, Mozilla's,

or any other operating system's Root Store is pretty much a proforma no brainer. But the world has a great many certificate authorities of questionable provenance. And deciding whom to trust can become very political very quickly.

Since the security of the Chromium project's Root Store is of crucial importance, I want to share the inclusion policies and their underlying philosophy...

The explanations below describe the Chrome Root Program, and policies and requirements for CAs to have their certificates included in a default installation of Chrome, as part of the transition to the Chrome Root Store.

Historically, Chrome has integrated with the Root Store provided by the platform on which it is running. Chrome is in the process of transitioning certificate verification to use a common implementation on all platforms where it's under application control, namely Android, Chrome OS, Linux, Windows, and macOS. Apple policies prevent the Chrome Root Store and verifier from being used on Chrome for iOS. This will ensure users have a consistent experience across platforms, that developers have a consistent understanding of Chrome's behavior, and that Chrome will be better able to protect the security and privacy of users' connections to websites.

For CAs that already participate in other public Root Programs, such as the Mozilla Root Program, many of these requirements and processes should be familiar.

During this transition, the Chrome Root Store contains a variety of existing Certification Authorities certificates that have historically worked in Chrome on the majority of supported platforms. This promotes interoperability on different devices and platforms, and minimizes compatibility issues. This should ensure as seamless a transition as possible for users.

In addition to compatibility considerations, CAs have been selected on the basis of past and current publicly available and verified information, such as that within the Common CA Certificate Database (CCADB). CCADB is a datastore run by Mozilla and used by a variety of operating systems, browser vendors, and Certification Authorities to share and disclose information regarding the ownership, historical operation, and audit history of CA certificates and key material.

For Certification Authorities that have not been included as part of this initial Chrome Root Store, questions can be directed to chrome-root-authority-program@google.com. Priority is given to CAs that are widely trusted on platforms that Chrome supports, in order to minimize compatibility issues.

For the inclusion of new CA certificates, priority is given to CAs in the following order, in order to minimize disruption or risks to Chrome users:

1. CAs that are widely trusted, and which are replacing older certificates with certificates and key material created within the past five years, and have an unbroken sequence of annual audits where these certificates and key material are explicitly listed in scope.

2. CAs whose certificates and certificate hierarchy are only used to issue TLS server certificates, and do not issue other forms of certificates.

3. CAs that have undergone a widely-recognized public discussion process regarding their CP, CPS, audits, and practices. [CP = Certificate Policy. CPS=Certification Practices Statement] At this time, the only discussion process recognized as acceptable is the discussion process operated by Mozilla on behalf of the open-source community at mozilla.dev.security.policy.

> Note: If you're curious to see how the sausage is made, you really need to check out the "Mozilla.dev.security.policy" group:
> https://groups.google.com/g/mozilla.dev.security.policy
> It is SO EASY for us to under appreciate all the hard and thankless work that's going on, to our immeasurable benefit, behind the scenes, by real people we'll never know to thank.

4. CAs that maintain sole control over all CA key material within their CA certificate hierarchy, and include their entire certificate hierarchy within a single audit scope.

5. CAs that have been annually audited according to both of the "WebTrust Principles and Criteria for Certification Authorities" and the "WebTrust Principles and Criteria for Certification Authorities - SSL Baseline With Network Security". Other audit criteria may be accepted on a discretionary basis, but Chrome will prioritize audits conducted according to both of these criteria.

Certification Authorities who do not meet all of the above criteria will be dealt with on a case-by-case basis. Note that the requirements above are illustrative only; Google includes CAs in its Root Program, and includes or removes CA certificates within its Root Store as it deems appropriate for user safety. The selection and ongoing membership of CAs is done to enhance the security of Chrome and promote interoperability; CAs that do not provide a broad service to all browser users are unlikely to be suitable.

As this transition occurs, CAs should continue to work with the relevant vendors of operating systems where Chrome is supported to additionally request inclusion within their root certificate programs as appropriate. This will help minimize any disruption or incompatibilities for end users, by ensuring that Chrome is able to validate certificates from the CA regardless of whether it is using the Chrome Root Store or existing platform integrations.

The Chrome Root Store Policy will be updated to more fully detail the set of formal ongoing requirements for working with Google in order to be distributed and included in a default installation of Chrome, as well as additional steps for applying or updating existing included certificates. Any questions regarding this policy can be directed to <<... @google.com>>

Chrome requires that CAs included in the Chrome Root Program abide by their Certificate Policy and/or Certification Practices Statement, where the CA describes how they operate, and must incorporate CA/Browser Forum's Baseline Requirements. Failure of a CA to meet their commitments, as outlined in their CP/CPS and the Baseline Requirements, is considered an incident, as is any other situation that may impact the CA's integrity, trustworthiness, or compatibility.

Chrome requires that any suspected or actual compliance incident be promptly reported and publicly disclosed upon discovery by the CA, along with a timeline for an analysis of root causes, regardless of whether the non-compliance appears to the CA to have a serious or immediate security impact on end users. Chrome will evaluate every compliance incident on a case by case basis, and will work with the CA to identify ecosystem-wide risks or potential improvements to be made in the CA's operations or in root program requirements that can help prevent future compliance incidents.

When evaluating a CA's incident response, Chrome's primary concern is ensuring that browsers, CAs, users, and website developers have the necessary information to identify improvements, and that the CA is responsive to addressing identified issues.

Factors that are significant to Chrome when evaluating incidents include (but are not limited to): a demonstration of understanding of the root causes of an incident, a substantive commitment and timeline to changes that clearly and persuasively address the root cause, past history by the CA in its incident handling and its follow through on commitments, and the severity of the security impact of the incident. In general, a single compliance incident considered alone is unlikely to result in removal of a CA from the Chrome Root Store.

Chrome expects CAs to be detailed, candid, timely, and transparent in describing their architecture, implementation, operations, and external dependencies as necessary for Chrome and the public to evaluate the nature of the incident and depth of the CA's response.

When a CA fails to meet their commitments made in their CP/CPS, Chrome expects them to file an incident report. Due to the incorporation of the Baseline Requirements into the CP and CPS, incidents may include a prescribed follow-up action, such as revoking impacted certificates within a certain timeframe.

If the CA doesn't perform the required follow-up actions, or doesn't perform them in the expected time, the CA should file a secondary incident report describing any certificates involved, the CA's expected timeline to complete any follow-up actions, and what changes the CA is making to ensure they can meet these requirements consistently in the future.

When a CA becomes aware of or suspects an incident, they should notify chrome-root-authority-program@google.com with a description of the incident. If the CA has publicly disclosed this incident, this notification should include a link to the disclosure. If the CA has not yet disclosed this incident, this notification should include an initial timeline for public disclosure. Chrome uses the information on the public disclosure as the basis for evaluating incidents.

So there's nothing too surprising there. But it also feels as though this is a bit of a placeholder. Perhaps largely cribbed from other root store program guidelines. The document indicates that it will be refined over time. So more specific requirements might be expected to evolve.

I mentioned at the top that this announcement was met with a moderate level of grumbling on some fronts. We end user consumers just happily click away on links, trusting that all of the plumbing underneath works correctly. But there are those in the enterprise — wearing well-worn

plumber's overalls — who have identified that the proactive management of certificate root stores, being a crucial anchor of trust throughout the enterprise, can form an important and powerful management firewall. If software needs to be signed by certs issued by recognized CA's, and the same for software, keeping a tight reign over the precise content of an enterprise's trusted root stores can form another potent line of defense.

So, in light of Google's announcement that they would be splitting from the root stores of Windows, Mac and Linux, to go their own way, the grumbling took the form of ... "Oh, that's just great. Now we'll have another root store to deal with."

On one hand this will, indeed, make life a bit more complex. But on the other hand, this **is** the way Firefox has always operated. So it's likely not a big deal. And you can so totally see Google wanting their own cert root store.

One question that's not clear to me is whether any given Chromium-based browser will be able to choose which store it uses — the Chromium store or the underlying OS's. I suppose Microsoft might be willing to run their new and shiny Chromium-based Edge browser with the Chromium root store. But it would be sitting on top of Window's own perfectly good root store. Seems sort of odd. I guess we'll see.