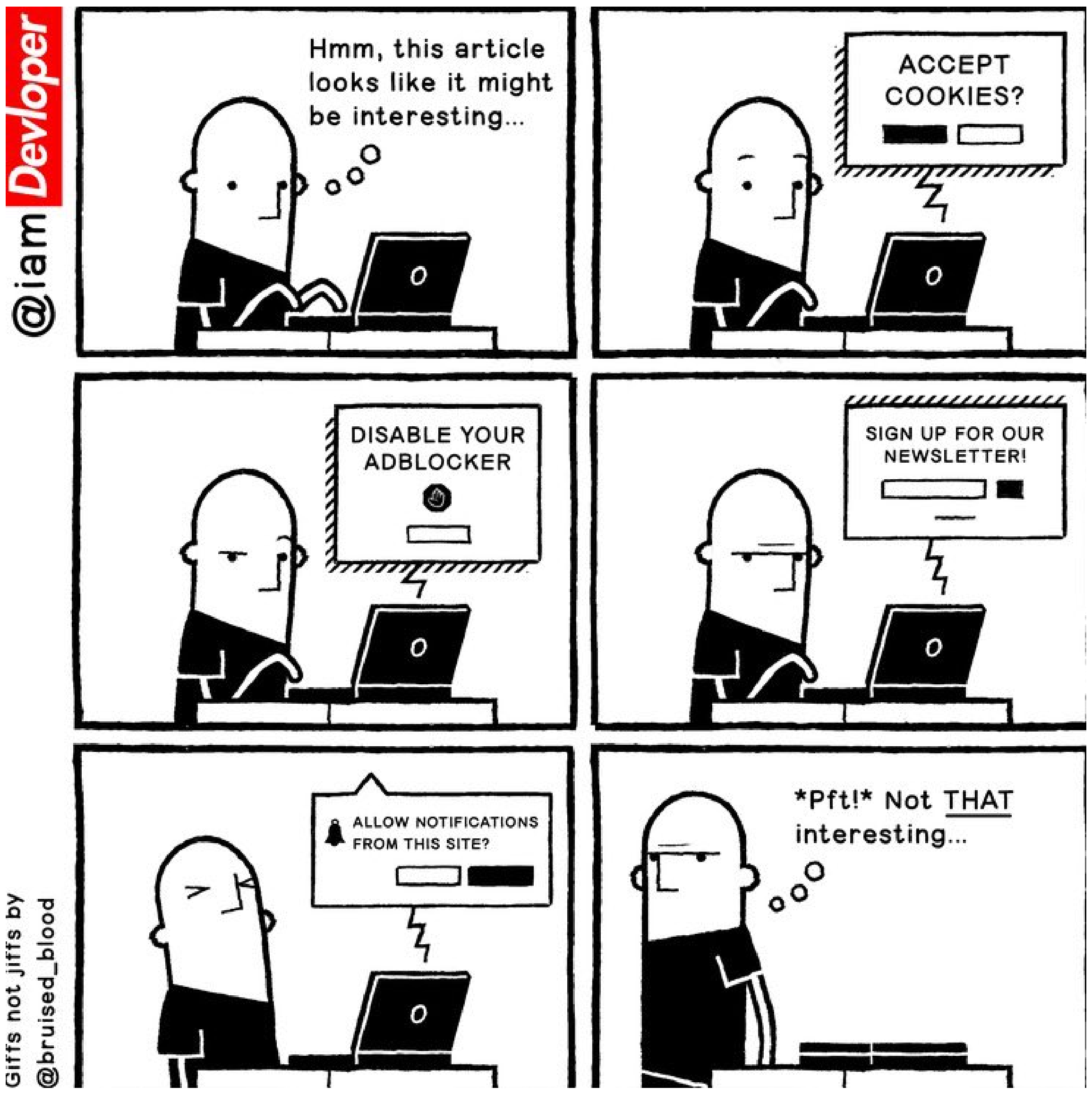
Security Now! #822 - 06-08-21
Extrinsic Password Managers

This week on Security Now!

This week I want to start off with a calm rant to summarize why today's computer security is so atrocious. I think it's worth a bit of a reality check on that. Then we're going to look at a new feature in Firefox and at Firefox's apparent jump in performance. We'll touch on three new ransomware victims, look at what's been learned about how Colonial Pipeline was breached, and at the curious news that the FBI somehow managed to snatch all of DarkSide's Bitcoins. We'll look at the latest good and bad news regarding WordPress, and at Github's updated policy regarding posting proofs-of-concepts for ongoing attacks. I've finished Project Hail Mary, so I have a comment to make there, and I want to address the surprisingly controversial question of NAT vs IPv6. Then we'll wrap up by examining the question of whether password managers should be intrinsic to our browsers or extrinsic. I think we're going to have some fun!


@iamDevloper
Hmm, this article looks like it might be interesting...
ACCEPT COOKIES?
DISABLE YOUR ADBLOCKER
SIGN UP FOR OUR NEWSLETTER!
ALLOW NOTIFICATIONS FROM THIS SITE?
Pft! Not THAT interesting...
Giffs not jiffs by @bruised_blood

The Great CyberSecurity Awakening of 2021

Everywhere I've turned for the past week, the talk has been about the threat from state sanctioned transnational cybercrime. Ransomware, which we've been talking about here for so long that I have repeatedly tried to promise that I would stop talking about it, has, at long last, finally bubbled up into the collective consciousness of the general public and press.

Last Monday, Lorrie and I attended a small dinner gathering, and as the resident computer security guy I found myself attempting to explain why most of what we were hearing about how this or that group was going to be appointed or created to get to the bottom of this was wrong and impossible but understandable, because no one wants to hear or believe that there is no simple fix for this, or more truthfully, no fix at all, simple or otherwise.

87.8% of the world's desktop and laptop machines are running an operating system which is so riddled with bugs that they needed to stop releasing them as they became ready or no one would have been able to get any work done. So now they're clumped up into monthly releases of between 50 to 150, where every month a handful are rated CRITICAL and "patch first with highest priority" because those are known to be currently in use compromising untimming people's. Every month. Every Single Month. Month after month with no diminishment or apparent end. And, as we all know, Microsoft was informed of a horribly serious vulnerability in their Exchange Server product — all of them, forever — late last year... yet it took until March to produce a patch, which they then did as an emergency because, they believe, news of this oh-so-juicy latest in a never ending line of vulnerabilities had somehow gotten loose.

And, of course, it's not just Microsoft. Many of the largest players have synchronized their monthly vulnerability patch-release cycles to Microsoft's. So a monthly patch-fest has evolved because we are apparently unable to produce software without serious exploitable flaws.

And just last week we noted how the latest ransomware — mostly just a handful of PowerShell scripts — was getting into people's computers when they clicked on a link in a perfectly authentic looking and specifically targeted eMail that loaded and rendered an HTML web page which contained and ran some JavaScript, which secretly downloaded some malware and then politely explained to its user that they needed to click once more to open the document.

Whose fault is that, exactly? Our eMail clients finally — but only after a long siege of exploitation — have started to actively refuse to have anything to do with executable content. Thank goodness for that. But, of course, there's a workaround. Since we want the power and flexibility of having web apps, and, since a web app might need to save something to our local computer, we've given JavaScript the ability to do so. Repeat after me: "What could possibly go wrong?" So, now the bad guys bootstrap. A benign looking eMail loads up an HTML page containing a little, hidden, benign-looking bit of JavaScript, and when given permission by its unwitting user, **it** downloads the executable file on behalf of the eMail that was originally received. How do we robustly solve **that** problem within the framework of computing that we've been making up as we go along? I have no idea... because though no one wants to hear it, all of our systems are fundamentally and deeply broken. Why? Because the economics are all wrong and those economics create perverse incentives. We reward new features because those can be seen. But security is invisible. No one gets credit for making something more secure, because you can't prove a negative.

Before the release of Windows XP, Microsoft's Steve Ballmer famously jumped around on stage, declaring that XP would be the most secure operating system they had ever created. History shows that it was the worst by far. But Ballmer's assertion was unchallengeable at the time because, as we've often observed, true security is only proven over time when, as, and if that unprovable negative is gradually proven by never happening. How the heck does one reward that?

So, we have some problems which are due to errors appearing in systems that are so complex that we've become tangled up in our own code. All we can do there is frantically patch and patch and patch all of the mistakes that we've made as we find them. And, separately, we have problems like that eMail-to-JavaScript malware download bypass which arise from the abuse of the **deliberate** design of our systems. It's not a bug, it's a feature... yet it's costing companies many millions of dollars in lost revenue, lost reputation and ransom payments.

Everyone wants to assign blame somewhere. Out of sheer desperation, enterprises are now sending their own employees baited eMails trying to trick them into clicking on a link that they shouldn't in order to pounce on them *"Ah Hah!! Your prior training has apparently worn off! Back to the reeducation camp with you!!"* What is wrong with us that we are now blaming the user for behaving in a perfectly normal and understandable fashion by responding to an eMail that appears to be in every way perfectly legitimate? We would like to shoot the messenger, but we can't find them. So we're going to shoot the recipient instead. We have to shoot somebody!

And as for features versus bugs, we were once assured that Windows 10 would be the last Windows ever. Someone, somewhere, decided that change was bad for security. And they were right, of course. Now we hear rumors of Windows 11. Oh joy, that's what we need. Apparently Windows is going to get lovely rounded corners for its rectangles to distract us from the minor detail that all of our desktop's icons have just disappeared. We're all so terrified now to click a link in an eMail that perhaps not having those sharp pointy corners will calm us down a bit.

During that dinner party, where everyone lost their appetite and stopped eating, I pointed at an AC wall plug that's controlled by a cloud-based service. I explained that the plug was connected back to servers in China and that the software running inside that itty-bitty computer contained in the plug was known to have a handful of remotely exploitable vulnerabilities such that, if at any point someone in China wished to infiltrate the house's network to snoop around it could be done. And I noted that the exploitation of a vulnerability in the plug's firmware would only be necessary if the plug had not come pre-loaded with a deliberate backdoor which would simply open when asked to allow foreign access. How would we know? There's no certification process. There's no qualification process. We click "Buy Now" on Amazon and that little miracle is on our front doorstep the next day. It may have a UL Seal of Approval to attest that we won't be electrocuted when we plug it in, but that does nothing to regulate the foreign packets that flow right back from our household's internal networks to China... a country with whom the US has a very complex relationship.

Today, we are frantically deploying millions, if not billions, of Internet of Things devices throughout our lives because they are shiny, incredibly inexpensive and do neat stuff. But there's ZERO oversight anywhere in the design, implementation and delivery of these services. They are cute little time bombs just waiting to go off.

Accountability is another problem in the computer industry. If your car's brakes fail unexpectedly, or its wheels fall off when you take a corner, there are consequences for its manufacturer. They are accountable. Software companies are not. There are no consequences for Microsoft when they wait three months before patching a horrific vulnerability that had been demonstrated to them, the exploitation of which has without any doubt been incredibly expensive for their users. But not for Microsoft. Microsoft requires everyone touching its obviously defective software to explicitly waive all expectations of their software's performance or fitness. It's in every license agreement. They say: *"It may work. It may not. We don't know either. But either way, the risk is* ***all*** *yours because we did our best and everyone knows that software, despite its name, is hard."*

So what have the natural consequences of all of this wrought? We have insecure hardware processors and memory, running insecurely designed software written in insecure languages implementing insecure protocols and APIs by people who require no formal training or certifications of any kind to create any of this. It's a black box in a black box in a black box.

But it is also one other thing that we apparently prize more than anything and everything else: It is astonishingly inexpensive. That Chinese wall plug has been working flawlessly for a year and it cost $5. I love it!! You can now purchase a breathtakingly powerful and useful laptop with a free Windows operating system (soon to have rounded corners) for a few hundred dollars. And storage? Oh my god. And what operating system am I sitting in front of right now as I write this? Yes, the same as 87.8% of the world: Windows. I'm completely satisfied with it. It works great. Overall, what it manages to do is a miracle. Okay, so yes, last week when I fired up the Win10 box to record this podcast all of the icons had disappeared from the desktop. I waited a while to see whether they would reappear, and when they didn't I rebooted. Now they're back.

Since most people have a very limited understanding of how their own bodies work, we rely upon highly trained medical professionals who have obtained extra education and certification. The system has been designed to remove all possible sources of error, so the resulting medical care is astonishing. But it can also be astonishingly expensive. And no one who hasn't been trained in the law should attempt to write a complex legally binding contract. But sometimes we need one. So we train-up attorneys in the law, we require them to prove that they know how to properly write complex contracts by passing the bar. But now that contract will really cost ya.

But software, which may have been written by someone in his mother's basement? Hey, it's free!! ... and you get what you pay for.

Throughout this little rant I've attempted to touch on a number of points. Looping back to the original issue, all of a sudden people are saying that **now** they want security. But that ship has sailed... and it sunk. Much as we might wish we could, we can't just dust off Steve Ballmer and have him jump around on stage to declare today's problems solved. You can't slap a fresh coat of paint on top of the rickety & flaky computer systems and technology we have collectively and deliberately built, placing features before security, and expect to suddenly have actual security.

And more importantly, NO ONE — NO ONE — would actually be willing to pay the cost to obtain true computer security even if we knew how — and there's no reason to believe we do. The systems we have are not secure and at this rate they're never going to be. But my god are they inexpensive, and soon they're going to have really nice rounded corners.

Web Browser News

Firefox will soon auto-update on Windows even when it's not running

The new feature is in the Firefox 90 beta. Mainstream users are currently running 89. So the next major release should bring this new functionality to everyone. Kirk Steuber, Mozilla's Platform Engineer said “Until now, Firefox has only downloaded and installed updates when the user runs it. This means that users who only use Firefox infrequently may be well out-of-date. It also means that if they open Firefox again in response to a [Firefox] marketing campaign, they may not immediately get the features [Mozilla] advertised. Background Update aims to address this problem by allowing updates to be downloaded and installed, even when the user is not running Firefox.”

Firefox background updates will be scheduled for the default profile every 7 hours when the browser is not running to check for new updates, download them, and install them without user interaction. Firefox also installs an optional service named Mozilla Maintenance Service that only launches in the background when updates are downloaded. This allows updates to be installed without requiring the user to acknowledge the Windows User Account Control (UAC) dialog since Windows Services can run with system privileges.

Anyone wishing to disable Firefox's background auto-update behavior can go to the "Updates" section "about:preferences" to find the UI for controlling auto updates.

And, for the time being this all only applies to Windows. Mozilla has announced no immediate plans to implement the feature for either macOS and Linux.

And speaking of Firefox...

My podcast prep workflow is to assemble a collection of many stories of the week in Firefox using its left-hand vertical column of tabs. Then I open Chrome to edit the show notes doc. I also open the Desktop outliner I use to do most of the writing. So I’m bouncing around among all three tools. And yesterday evening, after updating the instance of Firefox to 89 — the one with the more polished UI — I am consistently noticing that it is running FAR FASTER than it ever has. I wasn’t looking for or expecting anything to change. But this latest Firefox is greased lightning. So... I’m just saying.

Edge takes its own approach to HTTPS switching

As we know, Google's Chrome now, finally, defaults to HTTPS for “schemeless” URLs typed in the address bar. And Firefox has also added an HTTPS-Only Mode designed to secures web browsing by rewriting URLs to use the HTTPS protocol. At the moment it's still disabled by default. But hopefully that will soon change. Until then it can be enabled from the browser's settings... but it's not we who are most endangered.

https://blogs.windows.com/msedgedev/2021/06/01/available-for-preview-automatic-https-helps-keep-your-browsing-more-secure/

Here's the way they describe their approach:

Starting with Microsoft Edge 92, users can preview the Automatic HTTPS feature, which automatically switches your connections to websites from HTTP to HTTPS.

When sites are loaded over HTTP, attackers can view or change page content in transit, or redirect you to a different location than you had expected. Most websites now support HTTPS, which can help protect against these man-in-the-middle attacks. However, too many of these sites aren't configured to require HTTPS, leaving open a short window of opportunity for attackers before the site can redirect to the more secure protocol. Some sites may not redirect visits from HTTP to HTTPS at all, leaving some visitors with a less secure connection. To help protect your information as you browse, we are introducing a feature called Automatic HTTPS: now available for preview in Canary and Developer channels with Microsoft Edge 92.

Automatic HTTPS switches your connections to websites from HTTP to HTTPS on sites that are highly likely to support the more secure protocol. The list of HTTPS-capable websites is based on Microsoft's analysis of the web, and helps enable a more secure connection on hundreds of thousands of top domains. Automatic HTTPS upgrades your connection only on HTTPS-capable domains by default in order to prevent connection errors and potential performance issues.

*[Many sites like GRC have been publishing a **"Strict-Transport-Security" header with a max-age=31536000; preload** for years. TWiT's max-age=604800000; — Very nice Leo! :)]*

This protocol switch process happens automatically and without intrusive notifications, so that you don't have to think about your connection to websites—simply browse as usual! If you'd like even tighter security and don't mind potentially encountering connection errors more frequently, you can opt to switch all navigations from HTTP to HTTPS using the toggle on edge://settings/privacy:

If you want to test it right now, you have to open edge://settings/privacy and turn on "Automatically switch to more secure connections with Automatic HTTPS."

If the experiment hasn't reached you yet, you can enable it by going to "edge://flags/#edge-automatic-https", toggling on the 'Automatic HTTPS' experimental flag, and restarting the browser.

The HTTPS upgrades will be automatic with no alerts to allow you to browse the web just as you usually do, but over a secure connection wherever possible.

While, by default, Automatic HTTPS will only switch to HTTPS on sites likely to support this secure protocol, you can also choose to have all connections switched, which will likely lead to connection errors if the website is missing HTTPS support.

Ransomware

Three new ransomware victims

1. FujiFilm acknowledged last Wednesday that it had been hit by a ransomware attack, probably launched by the QBot Trojan group who have recently been teaming up with the REvil group.

2. The Massachusetts Steamship Authority, Massachusetts' largest ferry service, was hit by a ransomware attack on Wednesday which led to ticketing and reservation disruptions.

3. The University of Florida Health, also known as UF Health, is a healthcare network of hospitals and physician practices that provide care to countries throughout Florida. They suffered a ransomware attack that forced two of their hospitals to shut down portions of their IT network. So they're back to using pen and paper until their systems are restored.

We believe we know how Colonial Pipeline was breached

Bloomberg reported on Friday some of the findings by Mandiant, the group within FireEye who has been working with Colonial Pipeline to figure out how this happened.

As always, attribution and post-attack forensics is difficult. But there's strong evidence to support the theory that the attackers used a compromised VPN account password. The VPN login in question, which lacked any multi-factor authentication protection, was not in use, but it had been left active and it was at the time of the attack. The account's password was discovered inside a batch of leaked passwords on the dark web. This suggests that an employee of the company may have reused the same password on another account that was previously breached.

The takeaways are a little late. And it's always easy to admonish with “told you so's” after the fact. But unused accounts should be disabled. Authentication should require multiple factors. And I suppose that, while I've never been a fan of forced password changing, so long as the new passwords are unique and not shared, forcing a change might have prevented this entire mess.

I was recently informed that my logon to the management portal for Level3 would be expiring, since I hadn't logged onto it in six months. That's annoying, but it's good policy. And for things that are mission critical, like remote VPN access to a corporate network, the pain is probably worth the gain.

The FBI strikes back... but how, exactly?

Yesterday's press from the US Department of Justice was victoriously titled: “Department of Justice Seizes $2.3 Million in Cryptocurrency Paid to the Ransomware Extortionists Darkside”

I've excerpted the interesting new parts, removing the remedial “what is ransomware” bits…

WASHINGTON - The Department of Justice today announced that it has seized 63.7 bitcoins currently valued at approximately $2.3 million. These funds allegedly represent the proceeds of a May 8, ransom payment to individuals in a group known as DarkSide, which had targeted Colonial Pipeline, resulting in critical infrastructure being taken out of operation. The seizure warrant was authorized earlier today by the Honorable Laurel Beeler, U.S. Magistrate Judge for the Northern District of California.

Deputy Attorney General Lisa O. Monaco for the U.S. Department of Justice said: "Following the money remains one of the most basic, yet powerful tools we have. Ransom payments are the fuel that propels the digital extortion engine, and today's announcement demonstrates that the United States will use all available tools to make these attacks more costly and less profitable for criminal enterprises. We will continue to target the entire ransomware ecosystem to disrupt and deter these attacks. Today's announcements also demonstrate the value of early notification to law enforcement; we thank Colonial Pipeline for quickly notifying the FBI when they learned that they were targeted by DarkSide."

FBI Deputy Director Paul Abbate added: "There is no place beyond the reach of the FBI to conceal illicit funds that will prevent us from imposing risk and consequences upon malicious cyber actors. We will continue to use all of our available resources and leverage our domestic and international partnerships to disrupt ransomware attacks and protect our private sector partners and the American public."

Acting U.S. Attorney for the Northern District of California Stephanie Hinds said: "Cyber criminals are employing ever more elaborate schemes to convert technology into tools of digital extortion. We need to continue improving the cyber resiliency of our critical infrastructure across the nation, including in the Northern District of California. We will also continue developing advanced methods to improve our ability to track and recover digital ransom payments."

On or about May 7, Colonial Pipeline was the victim of a highly publicized ransomware attack resulting in the company taking portions of its infrastructure out of operation. Colonial Pipeline reported to the FBI that its computer network was accessed by an organization named DarkSide and that it had received and paid a ransom demand for approximately 75 bitcoins.

As alleged in the supporting affidavit, by reviewing the Bitcoin public ledger, law enforcement was able to track multiple transfers of bitcoin and identify that approximately 63.7 bitcoins, representing the proceeds of the victim's ransom payment, had been transferred to a specific address, for which the FBI has the "private key," or the rough equivalent of a password needed to access assets accessible from the specific Bitcoin address. This bitcoin represents proceeds traceable to a computer intrusion and property involved in money laundering and may be seized pursuant to criminal and civil forfeiture statutes.

[And then the press release concludes with several self congratulatory paragraphs.]

I have a theory. First of all, I picked up somewhere else that DarkSide actually wanted their payment via Monero — which, unlike Bitcoin's quite visible public transaction ledger, was deliberately designed to be untraceable. But, the DarkSide group indicated that ransom could also be paid via Bitcoin for an additional 10%. We know that Colonial didn't bother using their decryptor — which was given to them for free — claiming that it was too slow and buggy. But that they did, nevertheless, pay the ransom. Some have speculated that the ransom was paid, along with that 10% more via Bitcoin, specifically so that the FBI could track the coin flow.

What's also odd is that the Eclipse folks who analyzed the Bitcoin transaction ledger to first identify the DarkSide wallet claimed that the ransom paid to DarkSide was immediately transferred out of the group's wallet. That would mean that any later capture of bitcoins were not technically those paid by Colonial. In any event, there are people who know exactly what transpired and they're not talking.

But I'm extremely skeptical that this was an actual breach of DarkSide's wallet. As we know, a Bitcoin wallet is an abstraction. It’s simply a public key to which Bitcoins have been virtually transferred. And this virtual wallet’s owner holds the matching secret private key. And that’s just too easy to keep secret. No technical Kung Fu will somehow magically liberate the private key from its holder. So I'd be willing to bet my dinner that social/political pressure was brought to bear directly on DarkSide and that they were instructed by the sorts of Russian enforcers who you don't say 'no' to, to immediately transfer the entire content of their Bitcoin wallet to the following Bitcoin address... which was a wallet created by and under the control of the US Federal Bureau of Investigation.

In other words, this was not a technical win, this was pure and simple behind-the-scenes political pressure applied. It’s the only theory that aligns with the information we have available.

Security News

WordPress force installs Jetpack security update on 5 million sites

Last Tuesday the Jetpack folks who create and maintain one of the most popular plug-ins for WordPress — JepPack — acknowledged that they had quietly fixed and pushed out an update to resolve a privately reported vulnerability that, had ITT been exploited in the wild, would have been “not good.”

https://jetpack.com/2021/06/01/jetpack-9-8-engage-your-audience-with-wordpress-stories/

Their disclosure for Jetpack v9.8 said:

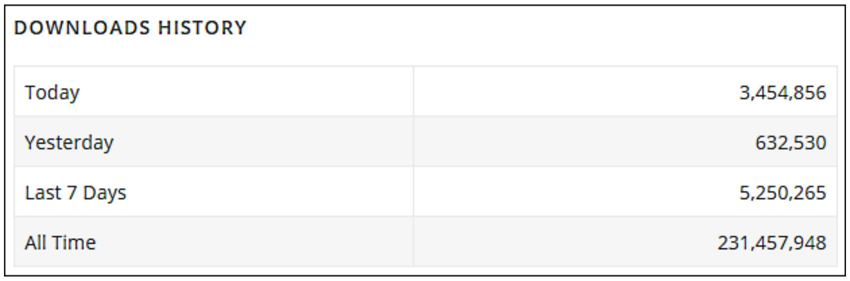
> We found a vulnerability in the Carousel feature and its option to display comments for each image. Thank you to nguyenhg_vcs for disclosing this issue to us in a responsible manner.
>
> We have no evidence that this vulnerability has been exploited in the wild. However, now that the update has been released, it is only a matter of time before someone tries to take advantage of this vulnerability.
>
> We consequently invite you to update your version of Jetpack as soon as possible. To help you in this process, we worked with the WordPress.org Security Team to release patched versions of every version of Jetpack since 2.0. Most websites have been or will soon be automatically updated to a secured version.

Versions released today include 2.0.8, 2.1.6, 2.2.9, 2.3.9, 2.4.6, 2.5.4, 2.6.5, 2.7.4, 2.8.4, 2.9.5, 3.0.5, 3.1.4, 3.2.4, 3.3.5, 3.4.5, 3.5.5, 3.6.3, 3.7.4, 3.8.4, 3.9.8, 4.0.5, 4.1.2, 4.2.3, 4.3.3, 4.4.3, 4.5.1, 4.6.1, 4.7.2, 4.8.3, 4.9.1, 5.0.1, 5.1.2, 5.2.3, 5.3.2, 5.4.2, 5.5.3, 5.6.3, 5.7.3, 5.8.2, 5.9.2, 6.0.2, 6.1.3, 6.2.3, 6.3.5, 6.4.4, 6.5.2, 6.6.3, 6.7.2, 6.8.3, 6.9.2, 7.0.3, 7.1.3, 7.2.3, 7.3.3, 7.4.3, 7.5.5, 7.6.2, 7.7.4, 7.8.2, 7.9.2, 8.0.1, 8.1.2, 8.2.4, 8.3.1, 8.4.3, 8.5.1, 8.6.2, 8.7.2, 8.8.3, 8.9.2, 9.0.3, 9.1.1, 9.2.2, 9.3.3, 9.4.2, 9.5.3, 9.6.2, 9.7.1.

If you are running any of these versions, your website is not vulnerable to this issue.

Why are sites not vulnerable to this issue? Because by the time this announcement was made public, Automattic had already pushed out their "force-installed" patched versions on all websites running vulnerable Jetpack versions and most sites had already been updated. Automattic said: "To help you in this process, we worked with the WordPress.org Security Team to release patched versions of every version of Jetpack since 2.0. Most websites have been or will soon be automatically updated to a secured version."

Bleeping Computer provided the current download stats which are available on the WordPress Plugins site. The stats confirm that the security updates have been pushed to most if not all exposed websites.


DOWNLOADS HISTORY

Today	3,454,856
Yesterday	632,530
Last 7 Days	5,250,265
All Time	231,457,948

I understand that the idea of updates being force-installed makes many people queasy. I received some well-considered blowback from my suggestion a few weeks ago that allowing our routers to auto-update should be welcomed with open arms and celebrated. A number of people felt that this would also open us to supply-chain attacks if malicious firmware was ever allowed to get into all of some manufacturer's routers. Point taken. But I think it's a matter of the least bad of two options. Allow routers to continue using known-defective and vulnerable firmware, thus exposing their unwitting users to significant danger — which will never be cured. Or allow improvements to that firmware to be pushed when needed.

On balance, I suspect that we're headed toward a more "pushy" future.

WordPress Fancy Product Designer
The plug-in is named "Fancy Product Designer" but perhaps a more fitting name would be "Fancy Site Destroyer." The WordFence security guys have observed active scanning to exploit a CRITICAL 0-day remote code execution (RCE) flaw which allows for complete WordPress and WooCommerce site takeover.

The problem lies in the plug-in known as “Fancy Product Designer” which is a visual product configurator plugin for WordPress, WooCommerce, and Shopify. It allows its users to customize products using their own graphics and content. Sales statistics show that Fancy Product Designer has been sold and installed on more than WordPress 17,000 websites.

WordFence's Ram Gall said: “The WordPress version of the plugin is the one used in WooCommerce installations as well and is [also] vulnerable.” The Shopify version of the plug-in mitigates and likely blocks the vulnerability because Shopify uses stricter access controls for sites hosted and running on its platform.

However, on WordPress and WooCommerce, successful exploitation of the Fancy Product Designer flaw allows attackers to bypass built-in checks which would otherwise prevent uploading of malicious executable PHP files. And, as we know, once you can get a malicious PHP file placed into a directory from which an HTML query will invoke it, it's game over. In short, the flaw allows attackers to completely take over vulnerable sites.

And, as the WordFence folks noted, the hunt for those 17,000 sites hosting that plug-in is underway. Ram Gall added that “This attacker appears to be targeting e-commerce sites and attempting to extract order information from site databases.” And he added: “Due to this vulnerability being actively attacked, we are publicly disclosing minimal details to alert the community to take precautions to keep their sites protected and to update.” Attacks targeting the thousands of sites running the Fancy Product Designer plugin started more than four months ago, first being seen on January 30, 2021.

Since the vulnerability is under active exploitation and was justifiably rated as critical severity, customers are advised to immediately install the Fancy Product Designer 4.6.9 patched version released last Wednesday on June 2.

WordFence will be holding off on releasing additional details about this vulnerability until more sites running Fancy Product Designer update to the latest version given that the zero-day can be exploited "in some configurations" even after its deliberate deactivation. Detailed info on how to update the plugin (which does not include an automatic update mechanism) and indicators of compromise, including IP addresses used to launch these ongoing attacks, are available in WordFence's report.

GitHub Updates its formal posting policy

And speaking of WordFence withholding details until more sites have patched — and a big bravo to them for having the self-control to forestall a bit of lime light — Github has officially announced a series of updates to the site's policies to address the recently controversial questions surrounding how it will handle malware and proof-of-concept exploit code that's uploaded to its service.

https://docs.github.com/en/github/site-policy/github-acceptable-use-policies

Under Section 2 of Github's Acceptable Use Policies, Github states:

Under no circumstances will Users upload, post, host, execute, or transmit any Content that directly supports unlawful active attack or malware campaigns that are causing technical harms — such as using our platform to deliver malicious executables or as attack infrastructure, for example by organizing denial of service attacks or managing command and control servers — with no implicit or explicit dual-use purpose prior to the abuse occurring.

They clearly wanted to be very clear about this, so this is further expanded upon separately in a section titled "Active malware or exploits" where they explain:

https://docs.github.com/en/github/site-policy/github-community-guidelines#active-malware-or-exploits

Being part of a community includes not taking advantage of other members of the community. We do not allow anyone to use our platform in direct support of unlawful attacks that cause technical harms, such as using GitHub as a means to deliver malicious executables or as attack infrastructure, for example by organizing denial of service attacks or managing command and control servers. Technical harms means overconsumption of resources, physical damage, downtime, denial of service, or data loss, with no implicit or explicit dual-use purpose prior to the abuse occurring.

Note that GitHub allows dual-use content and supports the posting of content that is used for research into vulnerabilities, malware, or exploits, as the publication and distribution of such content has educational value and provides a net benefit to the security community. We assume positive intention and use of these projects to promote and drive improvements across the ecosystem.

In rare cases of very widespread abuse of dual-use content, we may restrict access to that specific instance of the content to disrupt an ongoing unlawful attack or malware campaign that is leveraging the GitHub platform as an exploit or malware CDN. In most of these instances, restriction takes the form of putting the content behind authentication, but may, as an option of last resort, involve disabling access or full removal where this is not possible (e.g. when posted as a gist). We will also contact the project owners about restrictions put in place where possible.

Restrictions are temporary where feasible, and do not serve the purpose of purging or restricting any specific dual-use content, or copies of that content, from the platform in perpetuity. While we aim to make these rare cases of restriction a collaborative process with project owners, if you do feel your content was unduly restricted, we have an appeals process in place.

To facilitate a path to abuse resolution with project maintainers themselves, prior to escalation to GitHub abuse reports, we recommend, but do not require, that repository owners take the following steps when posting potentially harmful security research content:

- Clearly identify and describe any potentially harmful content in a disclaimer in the project's README.md file or source code comments.

- Provide a preferred contact method for any 3rd party abuse inquiries through a SECURITY.md file in the repository (e.g. "Please create an issue on this repository for any questions or concerns"). Such a contact method allows 3rd parties to reach out to project maintainers directly and potentially resolve concerns without the need to file abuse reports.

GitHub considers the npm registry to be a platform used primarily for installation and run-time use of code, and not for research.

Miscellany

NAT vs IPv6 — I inadvertently set off a bit of a firestorm with my strongly-worded statements two week ago about NAT and IPv6. One person tweeting via DM wrote:

> You're completely conflating NAT and firewalls, you can certainly have a firewall without NAT (if you are not out of addresses) and have an easier to understand environment.

And over in GRC's newsgroups someone posted:

> Instead of being scared of IPv6 why not learn a bit about it and how security is provided for IPv6 connected devices?
>
> During Security Now 650 Steve Gibson said:
> \> The nutty IP purists, with their heads well positioned far up their own you-know-what's
> \> where the sun don't shine, have always decried the use of NAT.
>
> They're right. NAT is an annoyance that provides nothing other than reducing the use of valuable IPv4 addresses.
>
> You do not need NAT to provide security. All you need is a stateful firewall that only allows packets in, that are part of, or are related to an outgoing connection. You DO NOT need to be translating between different IP addresses to achieve that. This provides EXACTLY the same security that NAT does.

Okay. First of all... **Duh!!!** For the record, I never suggested otherwise. I never, in any way, intimated that NAT was the only way to obtain the equivalent of a stateful firewall. In fact, my chosen personal firewall for the PC, which I endorsed 21 years ago in 2000, was ZoneAlarm. A bunch of embarrassingly old pages are still up on GRC talking about it. ZoneAlarm's claim to fame, along with being application-centric, was that it was inherently stateful. It tracked connections and only allowed incoming packets for connections that had been previously established. There was no NAT involved, anywhere.

But the entire world desperately needed the security provided by stateful packet inspecting firewalls back then, 21 years ago. While personal home networks were growing like weeds and needing many many of their own IP addresses on the LAN — when ISPs only had the one to give. Where was IPv6? Nowhere. And where is it today? Still mostly nowhere. Oh, sure, some ISPs are now, 21 years later, finally beginning to deploy IPv6. Was the entire world supposed to wait for IPv6 before we could have more than one device attached to our ISP's single-IP DSL or cable modem? NAT was a godsend and it still is today. Someday, we won't **need** NAT but I'll wager that we're still going to be using it. And in any event, most users still have no choice because today, in 2021, IPv6 is STILL not ready for prime time.

And one last point. The guy who posted over in the grc.securitynow newsgroup wrote, in CAPS, about IPv6 that "This provides EXACTLY the same security that NAT does." — which is also incorrect. Anyone who understands security appreciates the concept of multiple layers of protection. Having a stateful firewall provides one good layer, whether it's translating IP addresses or not. But having a local network of private and not-publicly-routable IP addresses is another massively useful layer of security. Everything that happens inside your network is local and "off the Net" until and unless something wants to reach out onto the public internet.

Getting everything to work through NAT has been a pain in the butt. We've had to develop robust STUN and TURN protocols and deploy a publicly accessible rendezvous server to directly interconnect two devices which wish to peer when both are sequestered behind their respective NAT routers. But that's all been figured out and it works. And that sequestration also helps to keep us safe the rest of the time.

Sci-Fi

Project Hail Mary — I finished reading the book this weekend and I very much enjoyed it.

One thing I can say without any doubt or question, is that any possible movie that's shorter than about eight hours will necessarily utterly fail to do the book justice. It's just not going to be possible to make a faithful movie out of this book. A movie titled "Project Hail Mary" might still be great, but those who have read the book will need to prepare themselves for an immensely watered down story. And that's a shame, since Andy's works are dense with delicious detail. And those details won't be left on the cutting room floor because no one will have bothered to include them in the first place. I really don't know how one makes a movie out of this book. So that alone is something I'll be intensely curious about. It's always the case that the book is far better than the movie. But in this case the movie would need to be a miniseries. So, if you imagine that this might interest you — and I sincerely think it should — please either read the book or have Audible read the book to you, before you see the movie.

SpinRite

The Curious Data Recovery Adventure —

Extrinsic Password Managers

The name "Tavis Ormandy" is one we've often mentioned on this podcast because Tavis is a prolific security researcher at Google. He's constantly finding problems in this industry's security designs and implementations, posting to Google's Zero-Day project and starting timers to require companies to fix their stuff... or else!

So, when Tavis posts on the topic of Password Managers to his own informal blog, it comes to the attention of many who are interested in topics of security. And many of those people listen to this podcast and wonder what I think of what Tavis says.

Tavis Ormandy on "Password Managers": https://lock.cmpxchg8b.com/passmgrs.html

Tavis was not overly wordy. So any attempt I might make to summarize would likely be longer than what he explained. After removing his examples and pointers to specifics, I think that the best way for me to share his points, so that I can then coherently respond, is to simply share what Tavis wrote. The link to his posting is here in the show notes:

I've spent a lot of time trying to understand the attack surface of popular password managers. I think I've spent more time analyzing them than practically anybody else, and I think that qualifies me to have an opinion!

First, let's get a few things out of the way. For some reason, few subjects can get heated faster than passwords. Maybe politics and religion, but that's about it. It's okay if you don't like my opinion.

Second, everyone needs to be using unique passwords. You don't have to use a password manager to do that, whatever system works for you is fine. If you want to use a notebook in a desk drawer, that's totally acceptable.

Okay, let's begin.

Conceptually, what could be simpler than a password manager? It's just a trivial key-value store. In fact, the simplest implementations are usually great. Good examples of simple and safe password managers are keepass and keepassx, or even pass if you're a nerd.

Things start to go wrong when you want integration with other applications, or when you want data synchronized by an untrusted intermediary. There are safe ways to achieve this, but the allure of recurring subscription fees has attracted businesses to this space with varying degrees of competence. I'm generally skeptical of these online subscription password managers, and that's going to be the focus of the rest of this article.

I often say that "use a password manager" is bad advice. That's because it's difficult to tell the difference between a competent implementation and a naive one. The tech press can review usability and onboarding experience, but can't realistically evaluate any security claims, so how do you propose users tell the difference? For that reason, I think “use a password manager” is so vague that it’s dangerous.

A good analogy is telling someone with a headache to pop any pills they find in the medicine cabinet. Maybe they’ll get lucky and find an aspirin, or maybe they won’t and you’ll be making a call to poison control.

Advice on this topic needs to be specific. It’s better to recommend implementations that are well designed, rather than general product categories. This position is surprisingly contentious, many people argue any password manager is acceptable and that I’m sowing fear by actually evaluating vendor claims. I remain unconvinced.

My primary area of interest is how remote attackers can interact with your password manager.

I’m not interested in things like testing how resistant encrypted blobs are to offline cracking. This might be a valid concern for some, but in most cases if an attacker is in a position to access or tamper with encrypted state, then you were in trouble whether you used a password manager or not.

There are two common issues I run into, the first is that trusted user interface elements are injected into potentially hostile websites. The second is that different components ipc over web-accessible channels (e.g. WebSockets, postMessage, etc.) without adequate mutual authentication.

Lets discuss user interface elements first.

Most online password managers use content scripts, javascript that is inserted into every website you visit. It’s really easy to write content scripts, but really tough to make them tamper resistant. That’s kind of a problem, because they’re going to be hosted in hostile environments.

How isolated worlds interact is complicated enough, but password managers make matters even worse by blurring the distinction between user interface and content.

We’ve already established that one component of online password managers must be injected into potentially hostile environments. How can those components communicate with other components?

One naive solution would be to just use XHR or WebSockets to a local HTTP endpoint. This sounds appealing to developers, they’re the native way to communicate on the web. The problem with this solution is it’s very difficult to differentiate between your content script, and a hostile script running on the same page but a different world.

Essentially every implementation I’ve looked at has got this wrong, resulting in critical game-over vulnerabilities.

Vendors come up with all kinds of hacky solutions to this, often involving inherently racy background scripts that try to verify a tab's origin.

Another gripe I have with online password managers is that they render browser sandboxes less effective. Modern browsers use a sandbox architecture to isolate components that can go wrong.

The problem is that online password managers effectively inject privileged components into these sandboxed processes with extensions. The purpose of sandboxing is to isolate potentially compromised components from each other, but if you stuff all your most valuable secrets inside the sandbox - then what's the point?

I worry that people don't understand the tradeoff they're making here.

Despite what your vendor says, if their network is compromised, the attacker can read your passwords. Here are some selected marketing claims from password manager vendors:

- "No one apart from you, not even [us], has access to your passwords."
- "[We keep] your information private, secure and hidden (even from us)."
- "Your data is secured in a way that only you can view and manage it. [Our] employees can't" ... etc, etc.

These claims are all nonsense. An attacker (or malicious insider) in control of the vendor's network can change the code that is served to your browser, and that code can obviously access your passwords. This isn't farfetched, altering the content of websites (i.e. defacement) is so common that it's practically a sport.

The reality is that you have to trust your vendor to maintain their infrastructure and keep it safe. The existence of encryption ("bank grade" or not) does not alter this.

Perhaps you think this isn't a big deal, you already trusted them when you installed their software. Fine, but these claims are front and center in all marketing, so vendors must believe their customers care about it. I think these claims are bending the truth to assuage legitimate concerns.

It's easy to poke holes in marketing fluff, but here are some other fun ones I noticed from real password manager vendors.

- *"Keystroke encryption protects everything you type from being read by cybercriminals."*
  Oh, okay.
- *"Many of the .NET assemblies [...] are obfuscated, so even using a disassembler users are unable to view critical areas of methods/functions/classes."*
  Well, I certainly feel safer. ... etc, etc.

If you want to use an online password manager, I would recommend using the one already built into your browser. They provide the same functionality, and can sidestep these fundamental problems with extensions.

I use Chrome, but the other major browsers like Edge or Firefox are fine too. They can isolate their trusted UI from websites, they don't break the sandbox security model, they have world-class security teams, and they couldn't be easier to use.

No doubt there will be many people reading this who don't like this advice. All I can say is I've heard all the arguments, and stand by my conclusions.

So, we obviously have a Google-centric person. And we have a Chrome-centric person. And we have an extremely smart and security-conscious person.

I find no fault with anything Tavis has said. And yet I'm proudly and happily using, **not** an intrinsic password manager, but an extrinsic add-on password manager.

Why? Because I'm polyamarous. I move among multiple browsers to suit my various needs. I use Safari on iOS (and Leo, I heard you mention yesterday that you do too.) I use both Firefox and Chrome, bouncing back and forth as needed, and I'm often depending upon a single bridge between them. What we might term "intrinsic password manager lock-in" is a thing. I don't want to be prevented from moving to another browser if I choose. I'm not only polyamarous, I'm also fickle. And I want the freedom to use whichever browser I want.

Also, I have not kept up with all of the many additional features offered by the best add-on solutions, but I know they offer a whole host of extras. I do take advantage of the secure enclave style synchronized storage for random notes which allow non-password things to be kept somewhere safe and synchronized.

So, I guess my conclusion here would be that IF someone has NO need to EVER bridge browser families, nor any use for the extra goodies that are offered by 3rd-party add-on extrinsic password managers... then, yeah, such a person's needs would be amply met just sticking with the password managers which have finally been added to our browsers.

And I cannot really speak to the theoretical loss of security which Tavis argues will necessarily accompany the use of any 3rd-party tool. I have no doubt whatsoever that there are many horrifically insecure and badly designed password managers. Just as with anything else. But what I do know is that we're not seeing **any** evidence that the most carefully and well-designed 3rd-party password managers are introducing exploitable vulnerabilities. It would be such a disaster if they were that I'm sure we'd know.