

# Security Now! #866 - 04-12-22

## Spring4Shell

### This week on Security Now!

We'll wrap up this week's podcast by revisiting Spring4Shell. Last week, when we first mentioned it, it was just a questionable itch. Now, a week later, it's a full blown outbreak deserving of today's podcast title. But before we roll up our sleeves for that we're going to examine credible reports of a 0-day in the Internet's most popular web server platform. We're going to take a look at Microsoft's newly announced "Autopatch" system, and the rapidly approaching end-of-security life of some Windows 10 editions. We have another instance of an NPM protest-ware modification of a highly used library, and I want to share a bit of miscellany and listener feedback. Then we'll finish by looking at what one week has done to Spring4Shell.

What Follows "Patch Tuesday" ???



# Security News

## Could NGINX have a 0-day?

F5 is investigating reports of a 0-day in NGINX.

For those who don't know, Nginx, spelled "N G I N X" is a web server—and more—that's been steadily growing in popularity. When I installed GRC's GitLab instance on a FreeBSD UNIX box, NGINX was, and is, providing that platform's web services. Apache is no longer the leader. Nginx is now the de facto web platform for new installations and it's now the most commonly used web server on the Internet with a 33.2% share overall and a 45.2% share of the top 1,000 sites. It can also serve as a reverse proxy, a load balancer, a mail proxy and an HTTP cache.

Over time, large projects tend to get pushed to do things that they were not originally designed to do. New chunks get added here and there. And what might have started out as an elegant and straightforward architecture becomes awkward, riddled with special case exceptions and it becomes increasingly difficult to maintain. In other words, they almost inevitably begin to show their age. An example of one such is OpenSSL. It has become so old, cumbersome and creaky that various lean and streamlined alternatives have been built. OpenSSL remains an amazing toolkit, but if all that one wants is fast, clean, simple and secure TLS connections, OpenSSL may no longer be the best choice.

So what about web servers? The official Apache.org site claims that *"Co-founder Brian Behlendorf first came up with the name "Apache" for the server. The name "Apache" was chosen out of reverence and appreciation for the people and tribes who refer to themselves as "Apache".*" Now, I think that indigenous native Americans are a great and noble people. But those of us who have been around a while will recall that that, is a crock. The "Apache" web server got its name because it was "A patchy web server." And the Internet archive doesn't lie. Question #4 on the Apache's FAQ from July 9th, 1997 asks: *"Why the name "Apache"?"* and it provides the answer: *"A cute name which stuck. Apache is "A PATCHy server". It was based on some existing code and a series of "patch files".*" Sorry about that, Org!

<https://web.archive.org/web/19970106233141/http://www.apache.org/docs/misc/FAQ.html>

As I recall, Apache was completely rewritten at some point along the way. But even that was a long time ago. So as with OpenSSL, Apache's age was once again showing. The new kid in town is Nginx. A newer and cleaner and much more recently written open source web server platform.

And F5 Networks, which focuses upon web application security, needed a platform. So, just over three years ago in March of 2019, they purchased NGINX for \$670 million. As a result, it is they who are today investigating a credible-appearing report of a 0-day in NGINX. Let me repeat that. There are credible-appearing reports, including of successful breaches, using a 0-day in the Internet's most popular web server which is in use by nearly half of the top 1,000 Internet sites.

A spokesperson for F5 told the press on Monday, yesterday: *"We are aware of reports of an issue with NGINX Web Server. We have prioritized investigating the matter and will provide more information as quickly as we can."* Well, that's good.

The problem first surfaced last Saturday, when a Twitter account connected to a US-based group known as "BlueHornet" tweeted about an experimental exploit for NGINX v1.18 — the current release. The group tweeted: *"As we've been testing it, a handful of companies and corporations have fallen under it."* They didn't respond to requests for further comment. But a different researcher shared a conversation they had with the people behind BlueHornet about the issue. The group explained that the exploit has two stages and starts with an LDAP injection. LDAP is the Lightweight Directory Access Protocol and LDAP Injection is an attack used to exploit web based applications that construct LDAP statements based on user input.

BlueHornet said they would share the issue with the Nginx security team through HackerOne for a bounty (which seems more than fair) or through F5's internal platform. And BlueHornet later created a GitHub page where they explained in detail how they discovered the issue and how it works: <https://github.com/AgainstTheWest/NginxDay> They wrote:

We had been given this exploit by our sister group, BrazenEagle, who had been developing it for some weeks. Or at least since Spring4Shell came out. We are still in the early stages of usage and understanding it, as we are working on another vendor vulnerability.

Regarding the finite details about it, we recommend you check out this tweet:

<https://twitter.com/Gi7w0rm/status/1512789855197093891>

GitWorm was allowed to share that information with permission from our DMs. We were initially confused, as LDAP doesn't interact much with Nginx, however, there is an ldap-auth daemon used alongside Nginx, which allows for this to be used. It primarily is used to gain access to private Github, Bitbucket, Jenkins & Gitlab instances. Further testing is required in due time.

**Update 1:** As some further analysis is ongoing, the module relating to the LDAP-auth daemon within nginx is affected greatly. ;) Anything that involves LDAP optional logins works as well. This includes Atlassian accounts. Just working out if we can bypass some common WAFs. Default nginx configs seem to be the vulnerable type, or common configs.

We highly recommend disabling the ldapDaemon.enabled property. If you plan on setting it up, be sure to change the ldapDaemon.ldapConfig properties flag with the correct information and don't leave it on default. This can be changed until Nginx (fucking) respond to their emails and DMs.

**Update 2:** Been talking to some infosec people about this, some mixed responses. Some are saying it's a problem with LDAP itself and not Nginx, while ldapDaemon isn't always used. The exact quote is "CI/CD pipeline hardens the instance, one of the steps is to completely strip out the LDAP module.". This is partially correct. In fact, it is an option when compiling nginx. However, it could be a problem with LDAP itself.

The issue with this, is that it only works with nginx instance using LDAP, such as any login

portal that supplies that authentication method.

Further analysis and testing is required. Looks to only be affecting this version. If it affects updated versions of the LDAP protocol, then we'll see what comes of that.

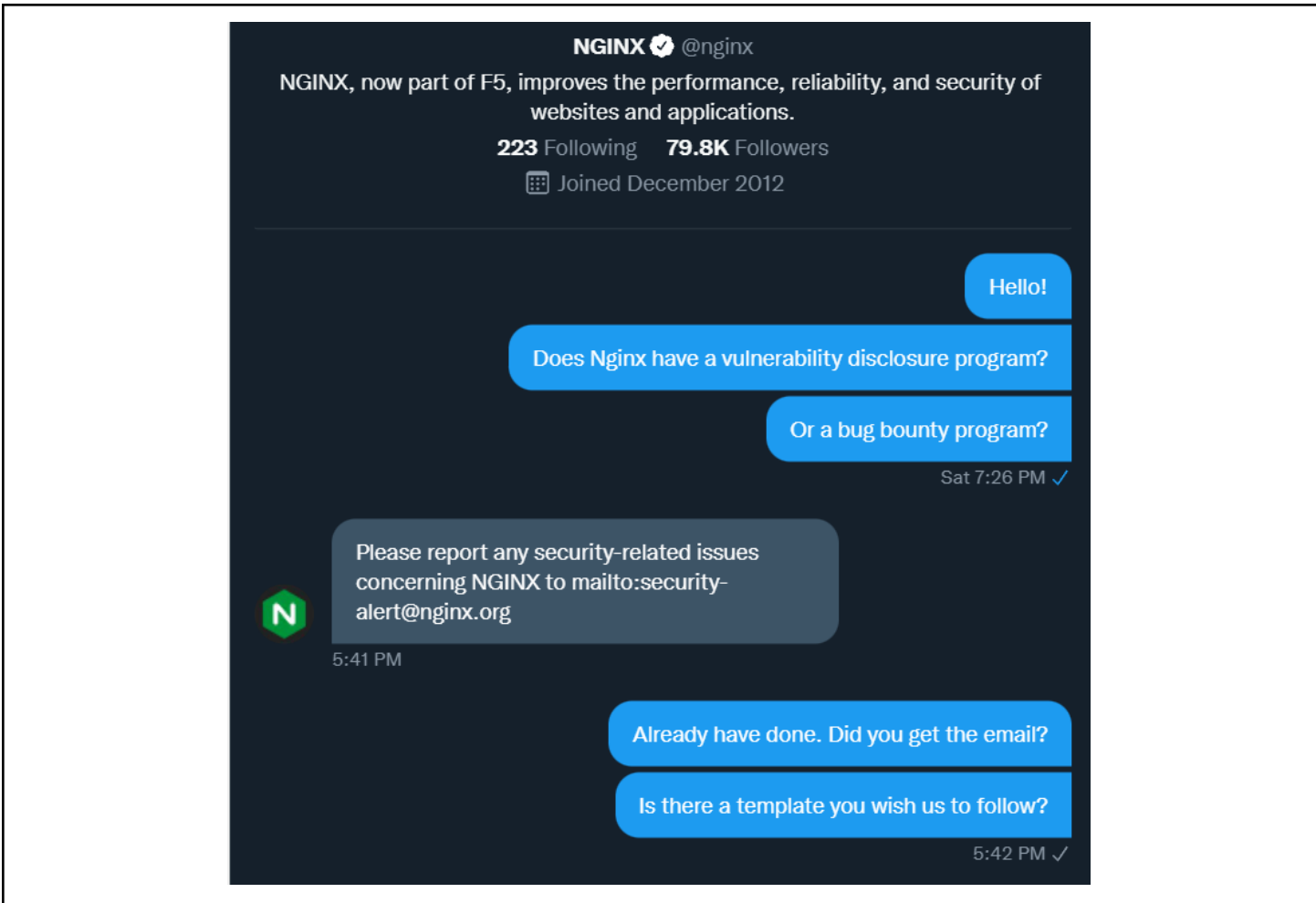
**Update 3:** I (as I'm still in ATW, but I'm just the only one online) have forwarded our own questions, community concerns, further testing questions to BrazenEagle via email. They have yet to respond, as they are US-based. Hopefully, they can provide some further answers. Shouldn't leak it, but their main contact email is brazeneagle@riseup.net (duh!).

While I'm still skeptical on the workings of this issue, it would explain the companies that were breached in under an hour during testing by BrazenEagle. They stated that they had passed this exploit to us as they were "working on more lucrative exploits". What that means, I'm not so sure. Some few individuals were (clearly) told about this being in the works some months ago via Telegram chats, which maybe perhaps some Twitter infosec people were talking about it and ATW.

**Update 4:** Regarding what people have been suggesting on twitter and on the issues page about this only being a LDAP issue, the problem with this is, during testing phases, it's only worked on Nginx, not on Apache or other web servers. Also, Nginx have still not replied. DMs or E-mail. We've emailed some companies that are affected that we've not breached (since that's heavily against our ideals) for support on the matter regarding security around this exploit.

**Update 5:** So, we've been followed by an employee of Nginx on Twitter. Threw our a DM asking about the situation, no response yet. We've been working on another exploit for MongoDB and another database management framework. Looking to have the PoC out in a week's time. Video as well. Will be working on it with @Gi7w0rm on Twitter.

**Update 6:** :O We got a DM from Nginx on twitter regarding the issue:



Then **Update 7** was a Twitter link to NGINX's Tweet:



**Update 8:** As Nginx have now released a blog post about the public releases of information, we've emailed them with a description, some familiarities of the issue that they highlighted over and assets affected. However, people are quick to jump on the "This is fake" or "This isn't anything" bandwagon. As we got no answer to if there is any bounty offered by Nginx for the findings, we've not shared any deeper information about this. If there is no bounty or even reward, we've looked at the other option that would be to sell the exploit on either breached.co, exploit.in or other sites. (We've been offered about 200K in XMR for the exploit).

If you're thinking that we're "Only interested in money", yes, what do you expect? We're a threat group, lol

For their part, NGINX is playing this as though they don't think that this is much of a threat. Their posting about this is titled: "Addressing Security Weaknesses in the NGINX LDAP Reference Implementation" and it starts out saying:

<https://www.nginx.com/blog/addressing-security-weaknesses-nginx-ldap-reference-implementation>

*On 9 April 2022, security vulnerabilities in the NGINX LDAP reference implementation were publicly shared. We have determined that only the **reference implementation** is affected. NGINX Open Source and NGINX Plus are not themselves affected, and no corrective action is necessary if you do not use the reference implementation.*

[In other words: "You weren't dumb enough to actually use the sample code of the way this should be used, were you?" Boy, have we seen this over and over. The classic example from early in this podcast was when Intel published a reference implementation for UPnP which all router vendors naturally copied and pasted into their code. Then, when it was later found to be horribly defective Intel said: "We didn't mean for you to actually use it! We just offered it as a reference!" Right.]

*The NGINX LDAP reference implementation uses LDAP to authenticate users of applications being proxied by NGINX. It is published as a Python daemon and related NGINX configuration at <https://github.com/nginxinc/nginx-ldap-auth>, and its purpose and configuration are described in detail on our blog: <https://www.nginx.com/blog/nginx-plus-authenticate-users/>*

*Deployments of the **LDAP reference implementation** are affected by the vulnerabilities if any of the following [three] conditions apply. Below, we further discuss the conditions and how to mitigate them:*

- *Command-line parameters are used to configure the Python daemon*
- *There are unused, optional configuration parameters*
- *LDAP authentication depends on specific group membership*

*Note: The LDAP reference implementation is published as a reference implementation and describes the mechanics of how the integration works and all of the components required to verify the integration. It is not a production-grade LDAP solution. For example, there is no encryption of the username and password used for the sample login page, and security notices call this out.*

At this point we'll have to see how this all plays out. NGINX has a corporate interest in downplaying this and appears to be sliding the responsibility for this mess onto the shoulders of those who actually implemented their reference implementation.

But for all of us here, there's a larger takeaway lesson to be learned. It's a variation on the tyranny of the default. And that is: *"Never, for the sake of simplicity or clarity, offer an insecure example or reference implementation that you're not prepared to have pretty much everyone use, for real, exactly as it is offered without modification in a full production environment."* Because that's exactly what's going to happen. Everyone's in a hurry. No one's as much in love with your stuff as you are. And no one else knows it as well as you do. They don't want to make a career out of setting it up. They just want to get it going and then move on to the next thing. So, assume that all of the default settings are going to be left as-is, including any code or examples that are provided as samples. Because that's exactly what's going to happen.

On Sunday, the hacking group claimed that they had tested the zero day on the Royal Bank of Canada but didn't explain whether the bank had actually been breached. It later said it breached the systems of the Chinese branch of UBS Securities. Neither of those institutions responded to requests for comment. But none of us should be surprised if we learn in the coming weeks of web site's breached by leveraging this newly uncovered 0-day in NGINX's reference implementation of LDAP-based authentication.

### **Microsoft's new Autopatch system**

A bunch of the tech press covered the news of Microsoft's new autopatch system. And some noted that it was going to make Patch Tuesdays much less exciting. Because, you know, it's excitement that you're hoping for whenever Microsoft updates your system. Like "Where did all my desktop icons go?" No one ever said that using Windows was boring. That's not something you hear often.

Lior Bela, a Senior Product Marketing Manager at Microsoft said *"This service will keep Windows and Office software on enrolled endpoints up-to-date automatically, at no additional cost. The second Tuesday of every month will be 'just another Tuesday'."* Right, unlike the Pandora's Box that it's been recently.

It's going to be interesting to see how this goes. Microsoft explained that *"Windows Autopatch manages all aspects of deployment groups for Windows 10 and Windows 11 quality and feature updates, drivers, firmware, and Microsoft 365 Apps for enterprise updates. It moves the update orchestration from organizations to Microsoft, with the burden of planning the Update process (including rollout and sequencing) no longer on the organization's IT teams."*

But the whole point of giving control to IT teams was to allow them to carefully roll out Windows updates to enterprise machines after first carefully vetting those changes to make sure they don't break anything mission critical. And yes, it's a big pain in the butt. But it's proven to be necessary over time since Windows updates have established such a track record of breaking things.

Okay, so how does Autopatch work? What does it do? Microsoft plans to automate the process that IT teams have been performing in-house: The service automatically divides an organization's population of Windows machines into four groups known as testing rings. Microsoft likes their rings. The test ring, the first ring, the fast ring and the broad ring. So: Test, First, Fast and Broad. The 'test ring' will contain a minimum number of devices. The 'first ring'

will contain around 1% of all endpoints that need to be kept up-to-date. The 'fast ring' will have around 9% and the 'broad ring' will have 90% of all devices. So, a few in the test ring, then 1%, 9% and 90%.

Lior Bela said that *"The population of these rings is managed automatically, so as devices come and go, the rings maintain their representative samples. Since every organization is unique, though, the ability to move specific devices from one ring to another is retained by enterprise IT admins."* Once the testing rings are set up, updates will be deployed progressively, beginning with the test ring and moving to larger sets of devices following a validation period through which device performance is monitored and compared to pre-update metrics.

Huh. *"All I'm getting now is a blue screen. I don't think this compares favorably to my pre-update metrics."*

Microsoft announced that this new Windows Autopatch service will be released this summer in July. Either way, it won't bother non-enterprise end users since it will be a new managed service offered for free to all Microsoft customers who already have a Windows 10 & 11 Enterprise E3 or above license. (Whatever that is — if you have one you probably know.)

The good news is, Autopatch includes "Halt" and "Rollback" features that will automatically block updates from being applied to higher test rings or rolled back automatically. That's good. But listen to this one: The product manager said something I had to decode: *"Whenever issues arise with any Autopatch update, the remediation gets incorporated and applied to future deployments, affording a level of proactive service that no IT admin team could easily replicate. As Autopatch serves more updates, it only gets better."*

Okay. What I think he said was that when Autopatch breaks something, it learns about that breakage and doesn't do it again. What, on that one machine? Or on any of that enterprise's other machines? Or on similar machines globally? This is beginning to feel like more of that "We don't know for sure where Windows 11 will run" hocus pocus. Like, where did all of the actual computer science go? It sounds like Microsoft is using their telemetry feedback, and the fact that updating their operating system has become so problematic, that they're going to turn all of the machines owned by all of their Enterprise customers into a gigantic neural network of *"let's try this and see what happens."* Anyway... I've never felt so happy to be a lowly end-user. This is going to be interesting.

### **Another instance of Russian Protest in JavaScript's repository**

On March 17th, the Russia-based developer Viktor Mukhachev aka "Yaffle" (that's much easier, so we'll call him Yaffle) altered his popular NPM library 'event-source-polyfill'. This change, which was introduced into v1.0.26 of 'event-source-polyfill', will cause web applications built with this now-latest version of the library to display anti-war messages protesting the "unreasonable invasion" of Ukraine, to Russia-based users 15 seconds after a webpage which incorporates this code is displayed.

"Polyfill packages" (or we might call them "backfill packages") implement sets of newer JavaScript features on web browsers that do not yet support them. In this case, the 'event-source-polyfill' package that has been deliberately polluted by its developer implements



the very useful JavaScript “EventSource” API. This API allows a webpage to open a persistent connection back to an HTTP server, which sends events to the browser. And the connection remains open until it's explicitly closed by calling `EventSource.close()`.

What's interesting is why anyone would need to backfill this particular API, since it's been present in all major browsers for quite a while. It was first adopted by Chrome and Firefox in their respective version 6's, and as we know Firefox is now at version 98 and Chrome is at 100. But the chart showing the API's adoption profile had a single interesting and glaring exception: Internet Explorer. And it is probably the case that Russia remains the last surviving bastion of Internet Explorer use. So, in order for Russia's inventory of IE to be able to run web applications that rely upon the EventSource API, that support needs to be provided by a “polyfill.”

But, given how pervasive the use of Yaffle's EventSource polyfill package is, and given that it is only needed by IE, since all other browsers have incorporated the API natively for many years, it must mostly be due to other developers NOT having yet proactively removed it from their own package dependencies. It is currently used by more than — get this — 135,000 GitHub repositories and it's downloaded more than 600,000 times every week on NPM for incorporation into those other packages.

Of course, the bigger concern here is the use of what should be a rigorously politically neutral software API for the purpose of injecting its author's political sentiment into the use of their software package. The users who receive this sudden anti-war protest pop-up have no idea where it's coming from. They don't know that it was buried in some inter-package API dependency and that it wasn't put up by and reflective of the website or web app they're using. In fact, since everything else they see IS coming from the website or web app they're using, that's exactly what they are going to think.

It's the abuse of the implicit trust by the developers who have chosen to use and depend upon this package that's the problem. Over time and with repeated incidents — this is the 3rd one, recently — the abuse of this trust is going to weaken this entire ecosystem. And maybe that's not such a bad outcome. Perhaps it should be weakened. The very fact that a package's author and maintainer was able to cause their package to behave in a way that its dependent users may well disapprove of should serve as further demonstration of just how rickety, from a security standpoint, this entire package repository ecosystem has become.

In the case of NPM and the browsers that run this code, they're going to start needing to not trust the code that's being sourced by the same-origin server. That's a game changer.

And, of course, NPM is only one instance from the world of open source public repository supply chains. What has been built is not robust in the face of an active adversary.

### **End-of-service life for some popular Windows editions**

It's April 2022 which makes next month May of 2022. And that means that several editions of Windows 10 20H2 and 1909 are reaching the end of their service life after May's Patch Tuesday — their last Patch Tuesday — on May 10th.

Win10 20H2, once also known as the October 2020 Update, reaches EOS for Home, Pro, Pro Education, and Pro for Workstations users. Enterprise, Education, and IoT Enterprise editions get one additional year, reaching their end of service on May 9, 2023.

And this also means that next month the already EOS Win10 1909 which previously ended for Home, Pro, Pro Education, and Pro for Workstations users will also finally also be ending for Enterprise, Education, and IoT Enterprise editions.

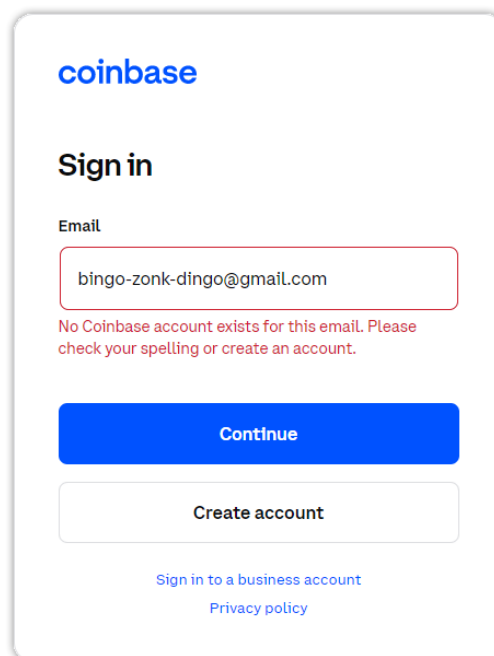
So, for those users who have been holding back and holding off moving forward to a newer edition, two months from now it'll be necessary if you want to stay current with updates.

And today is April's patch Tuesday. So I'm sure we'll have some news next week.

## Miscellany

We have a neighbor whose son uses Coinbase to manage and retain all of his Crypto currency and he's been urging them to buy and hold some Bitcoin. Independent of the value of any cryptocurrency as a buy and hold asset, I made a comment about the general inadvisability of leaving any sizable investment in crypto online, noting that many exchanges had been breached and that Coinbase had not escaped from that. They suffered a breach back in 2019. And exactly a year ago, between March and May of 2021, more than 6,000 of their customers were hacked in a large-scale email phishing campaign which tricked their customers into giving up the email addresses, passwords, and phone numbers associated with their accounts. I explained that the only safe practice was to remove any Crypto from any online exchange pool and to transfer it to an offline wallet.

That discussion made me a bit curious. So I went over to [coinbase.com](https://coinbase.com) and attempted to sign in without an account. And I discovered in two seconds that they had made one of the cardinal mistakes of online security: showing an attacker when they have guessed wrong about an account's eMail:



The screenshot shows the Coinbase sign-in interface. At the top is the Coinbase logo. Below it is the heading "Sign in". There is an "Email" label above a text input field containing "bingo-zonk-dingo@gmail.com". Below the input field is a red error message: "No Coinbase account exists for this email. Please check your spelling or create an account." There are two buttons: a blue "Continue" button and a white "Create account" button. At the bottom, there are links for "Sign in to a business account" and "Privacy policy".

It's fine to take the eMail or Account Name first and separately. But **never** tell the user that their account is unknown. **Always** ask for their password regardless of whether or not the account is known. That prevents the attacker from being able to probe for existing accounts separately from providing a known account's password. I understand that from a customer service standpoint, it's MUCH less confusing to a user to provide them with immediate feedback when they've mis-entered their eMail address or username at the first stage. But the reason it's much less confusing is exactly why it provides an advantage to any attacker who is now able to probe that service for its base of existing accounts. In the case of Coinbase, an attacker might know someone's eMail address and wish to know whether they have an account on Coinbase. Coinbase lets them know immediately.

## Closing The Loop

**Mementh / @Mementh**

*Time based port knocking ??? You have an authenticator app and port knocker gets that and generates the ports to knock.*

That's kinda brilliant! I asked "Mementh" whether he'd come up with this on his own or may have seen it somewhere, since I wanted to give him credit for a brilliant and clever solution. It solves and resolves the static-knock replay attack and brute force knock guessing problems in the same way a one-time password solves the same problems for passwords. Assuming that the client and server are able to both obtain an awareness of the time of day, the port listening server could be continuously receiving incoming port knocks into a ring buffer and whenever it adds a knock to the buffer it would scan the buffer for the current knock sequence all coming from the same IP address. Very neat.

Something else occurred to me as I was writing this up: One nice bit of client-controllable data that's also logged in the typical firewall log, is not only the source IP, but also the source port. Although ICMP-based knocking doesn't have any port, both UDP and TCP do. So manipulating the knocking packet's source port doubles the number of entropy bits per packet from 16 to 32 without any other complexity.

**vitrapemli / @vitrapemli**

*RE: Port Knocking (ep 865) I do something a little bit different but I think it's just as cool. I have IPTables log all the packets just above the drop. This is a little messy in the logs but I have Fail2Ban watching that log. If someone is port knocking or scanning my host, 3 failed attempts on any closed port, I block that IP for 1+ week. The idea is, the people I want to talk to my services, on random ports, know what port they're on so they have no reason to try other closed ports.*

I agree, that's nifty, too. A mature port knocking system ought to definitely have an IP-based lock-out. That pretty convincingly

**Lay the proud usurper low. / @ehastings**

*Dear Steve: Regarding SpinRite 6.1 and successors, is there any spot on your timeline for a version that is Apple Silicon native? Before the new systems I had hoped that there would be a Mac native release. That would seem to be far off, at best. What can you report? Thanks. Gene*

# Spring4Shell

So, as I noted at the top of the podcast, Spring4Shell is no longer theoretical. The attacks have begun.

Last week we introduced this latest Java-based flaw that had been found in VMWare's Spring.io web framework which, at the time, was still only theoretical. Recall that there had been some questioning about just how bad this potential RCE exploit would turn out to be. Flashpoint had said "Current information suggests in order to exploit the vulnerability, attackers will have to locate and identify web app instances that actually use the DeserializationUtils, something already known by developers to be dangerous." I was and still am skeptical that any Java developers appreciate that Java object deserialization is dangerous. But, okay. And Rapid7 said that despite the public availability of PoC exploits, "it's currently unclear which real-world applications use the vulnerable functionality." and "Configuration and JRE version may also be significant factors in exploitability and the likelihood of widespread exploitation." But CERT's Will Dormann tweeted that "The Spring4Shell exploit in the wild appears to work against the stock 'Handling Form Submission' sample code from spring.io. If the sample code is vulnerable, then I suspect [Will tweeted, that] there are indeed real-world apps out there that are vulnerable to RCE." And also recall that this is one of those 9.8's.

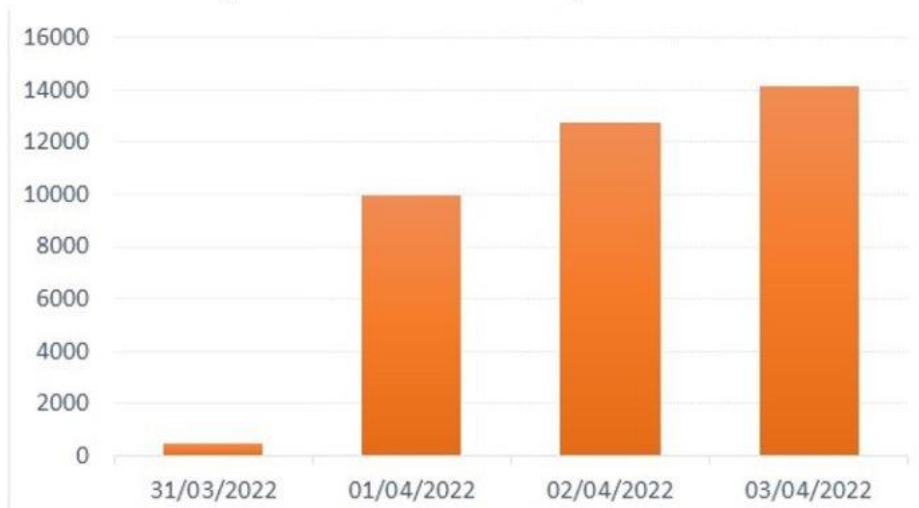
Now, CISA is warning of active exploitation of the CRITICAL Spring4Shell vulnerability. So it appears that its 9.8 was prescient and is being earned. CISA has added it to its Known Exploited Vulnerabilities Catalog based on "evidence of active exploitation."

Praetorian researchers Anthony Weems and Dallas Kaman noted that "Exploitation requires an endpoint with DataBinder enabled — in other words, an HTTP POST request that decodes data from the request body automatically and depends heavily on the servlet container for the application." The automatic decoding of a POST fits in with Will Dormann's observation that Spring.io's sample code for handling form submission is, itself, vulnerable. Although details of in-the-wild abuse are still a bit unclear, the information security company SecurityScorecard said "active scanning for this vulnerability has been observed coming from the usual suspects like Russian and Chinese IP space." (I guess Russia's still connected to the Internet.)

Spring4Shell vulnerability scanning activities have also been spotted by Akamai and Palo Alto Networks' Unit42, with the attempts leading to the deployment of a web shell for backdoor access and to execute arbitrary commands on the server with the goal of delivering other malware or spreading within the target network. No big surprise there. So Spring4Shell has created another new way in by jimmying the front door lock.

Check Point Research said: "During the first four days after the vulnerability outbreak, 16% of organizations worldwide were impacted by exploitation attempts" and they added that they had detected 37,000 Spring4Shell-related attacks over the weekend:

## Vulnerability Allocation Attempts Since Outbreak



Microsoft 365 Defender Threat Intelligence Team chimed in, stating it has been “tracking a low volume of exploit attempts across our cloud services for Spring Cloud and Spring Core vulnerabilities.” And according to statistics released by Sonatype, potentially vulnerable versions of the Spring Framework account for 81% of the total downloads from Maven Central repository since the issue came to light on March 31. So, SINCE the issue came to light at the end of March, 4 out of 5 of all downloaded were of the potentially vulnerable version.

Cisco, which quickly jumped to investigate its own line-up to determine which of its products might be impacted confirmed that three of its products are affected: The “Cisco Crosswork Optimization Engine”, “Cisco Crosswork Zero Touch Provisioning (ZTP)” and “Cisco Edge Intelligence.”

VMware, Spring.io's parent company, has said that three of its products are vulnerable: Their “Tanzu Application Service for VMs”, the “Tanzu Operations Manager”, and “Tanzu Kubernetes Grid Integrated Edition (TKGI).” They have made patches and workarounds available as needed. VMware said: “A malicious actor with network access to an impacted VMware product may exploit this issue to gain full control of the target system.”

Look how quickly we now move from “there may be a problem here” —to— “Oh Crap! Cancel Christmas!” This is today's reality.

To flesh this out a bit further, SecurityScorecard wrote that on Thursday, March 31st a patch for a widely used Java framework called the Spring Framework was given the designation CVE-2022-22965 with a CVSS Score of 9.8. That’s bad news for a lot of companies that make use of this framework for delivery of their web applications, services and APIs. This is a remote code execution (RCE) vulnerability and the ease of exploitation is partly why it has earned a 9.8 out of 10 on the CVSS Score.

Way back in 2010 there was an RCE for the Spring Framework v2.5 which fixed the vulnerability discovered then about unsafe `class.classLoader.URLs`. This new RCE is related to that

vulnerability. The fix 12 years ago was to forbid jumping from a class to classLoader and the fix this time is to forbid jumping from a class to module. Once again it seems that being a major version behind is a saving grace because `getModule()` is only present in JDK9+.

Spring has acknowledged the vulnerability and released v5.3.18 and v5.2.20 to patch the issue as well as version v2.6.6 for spring-boot. And, no surprise, SecurityScorecard recommends an immediate upgrade for all users.

They go deeply into remediation details, exploitation details and detection details. I have a link in the show notes for anyone who might want that level of nitty-gritty:

<https://securityscorecard.com/blog/spring4shell-12-year-old-vulnerability-springs-back-to-life>

And in their conclusion they note that: "If this feels all-too-familiar and is reminding you of the Equifax hack that was due to an exploitation of the Apache Struts2 framework, then your instinct is spot on. This is the same kind of vulnerability."

And, on top of everything else, the Spring4Shell vulnerability is also now, since the start of April, being actively exploited by threat actors to execute the Mirai botnet malware, focusing upon the Singapore region. In their posting titled: "Analyzing the Exploitation of Spring4Shell Vulnerability in Weaponizing and Executing the Mirai Botnet Malware" Researchers at Trend Micro said that "The exploitation allows threat actors to download the Mirai sample to the '/tmp' folder and execute them after making a permission change using 'chmod'." They wrote that they began seeing malicious activities at the start of April and they they also found the malware file server with other variants of the sample for different CPU architectures. That makes sense, of course, since Java is a multi-architecture language which is executed by its own JVM.

Trend Micro's write-up is by far the most in-depth and detailed analysis that I've encountered so far. I have a link in the show notes to their page for anyone who's interested:

[https://www.trendmicro.com/en\\_us/research/22/d/cve-2022-22965-analyzing-the-exploitation-of-spring4shell-vulner.html](https://www.trendmicro.com/en_us/research/22/d/cve-2022-22965-analyzing-the-exploitation-of-spring4shell-vulner.html)

And it makes sense that botnets would be quick to jump on this time-limited vulnerability. It's not the first time we've seen this. In December of last year, multiple botnets including Mirai and Kinsing were found to be leveraging the Log4Shell vulnerability to breach susceptible servers on the Internet. As we know, Mirai, meaning "future" in Japanese, is the name given to the Linux-hosted malware which has continued to target networked smart home devices such as IP cameras and routers and link them together into a network of infected devices. And the leaking of Mirai's source code back in October 2016 has given birth to many variants including Okiru, Satori, Masuta, and Reaper, making it an ever-mutating threat.

Intel 471 researchers said last month that "The Mirai code is so influential that even some of the malware offshoots are starting to have their own code versions released and co-opted by other cybercriminals." And in January, CrowdStrike noted that compared to 2020, malware targeting Linux-based systems had increased by 35% during 2021. Intel said that: "The primary purpose of these malware families is to compromise vulnerable internet-connected devices, amass them

into botnets, and use them to perform distributed denial-of-service (DDoS) attacks.” And we know all too well just how powerful and prevalent DDoS attacks have become today.

