Security Now! #847 - 11-30-21

Bogons Begone!

This week on Security Now!

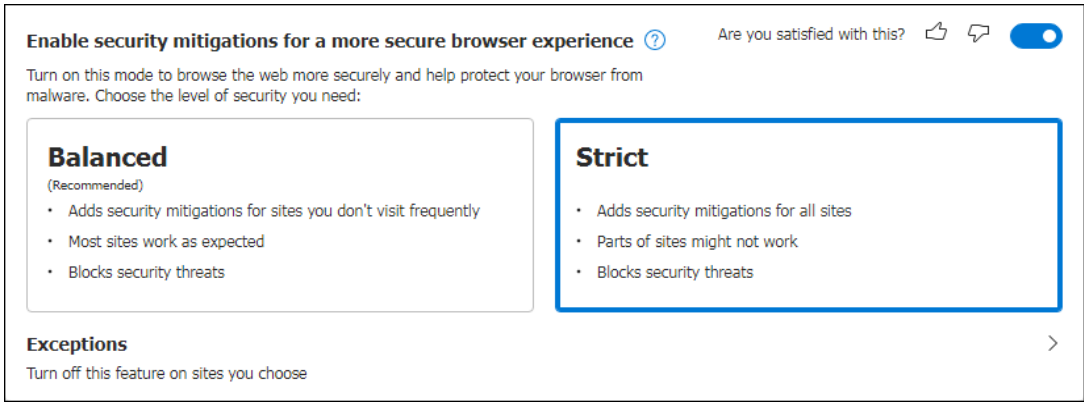
This week we'll note that the new Edge browser's Super Duper Secure Mode has been deployed and can be enabled by security conscious users. We also have more than one third — 37% — of the world's Smartphones vulnerable to audio monitoring and recording flaws in their MediaTek firmware. We have an important reminder about clicking links in eMail, and wonder how that can still be a problem? And the entirely predictable evolution of a Windows 0-day vulnerability which is latent no longer. We have some interesting Closing the Loop feedback from our terrific listeners, and a Sci-Fi book update. Then we take another and much broader look at the recent efforts to cleanup IPv4 — but this time from the perspective of those working to do so.

Security News

“Super Duper Secure Mode”

We previously discussed the experiment Microsoft was conducting with their Edge branch of the Chromium browser. In recognition that a disproportionate percentage of security troubles arise from the most extreme measures being used to push browser performance to the limits, and the recognition that underlying system processor performance has advanced so far recently that pushing the browser so hard may be producing diminishing and even negative returns in terms of security, Microsoft began experimenting with a “Super Duper Secure Mode” for Edge which pulls back on the most historically troublesome performance optimizations in favor of improved security.

We're talking about it today because, without any ballyhoo, Microsoft recently quietly added “Super Duper Secure Mode” to Edge. We all have it already. It appeared in v96.0.1054.29. It's currently disabled by default. To enable it as I did, you'll need to go to: **edge://settings/privacy** then scroll down to the Security section. At the bottom of that section on the right is a switch which you need to flip. If you wish you can then change “Balanced” security to “Strict” security:



This removes Chromium's Just-In-Time compilation (JIT) from the V8 processing pipeline and also enables Intel's Control-flow Enforcement Technology (CET), which is a hardware-based exploit mitigation that provides enhanced security. Based upon evidence from historical exploits, it's believed that this will significantly reduce the browser's attack surface. Microsoft describes Super Duper Secure Mode as "a browsing mode in Microsoft Edge where the security of your browser takes priority, providing you an extra layer of protection when browsing the web."

So what's “Balanced” vs “Strict”?

“Balanced” adaptively learns which sites are visited often and eventually trusts those. Strict is strictly TNO — Trust No One — mode. And in either case, exceptions can be added manually.

Microsoft's research revealed that around 45% of all security vulnerabilities found in the V8 JavaScript and WebAssembly engine were related to the JIT engine, accounting for over half of all 'in the wild' Chrome exploits abusing JIT bugs. So by disabling JIT, the attack surface is drastically reduced by removing almost half of the V8 bugs that should be fixed. Microsoft said: “This reduction in attack surface kills half of the bugs we see in exploits and every remaining bug becomes more difficult to exploit.”

Moving forward, Microsoft plans to include support for Arbitrary Code Guard (ACG) in Super Duper Secure Mode. ACG is another security mitigation that would block attackers from loading malicious code into memory. Hmmmm. That sounds like a good thing. I'll take two, please!

The Android and macOS editions of Edge will soon also be obtaining these new vulnerability mitigation features. And the screen shot above shows how I immediately set my Edge.

37% of the world's smartphones are vulnerable.
Chips by MediaTek are installed in roughly 37% of the world's smartphones and CheckPoint Research recently reverse engineered the firmware of those proprietary chips.

CheckPoint's highly detailed technical report showed that malicious apps installed on a device would be able to interact with the MediaTek-based audio driver. Such apps could send maliciously-crafted messages to the MediaTek firmware to gain control over the driver and steal any audio flow going through the device, turning more then one-third of the world's Smartphones into audio spying devices. And since the MediaTek subsystem is deep in the system, exploitation of the vulnerability allows audio from phone calls, WhatsApp calls, browser videos, and video players to be recorded.

The fact that MediaTek chips are installed on roughly 37% of the world's smartphones means that this creates a large attack surface for any malicious app and malware creator. Devices from Xiaomi, Oppo, Realme, and Vivo are known to use MediaTek chipsets.

Three issues were patched last month in October and a fourth will be fixed next month. CheckPoint explained that the MediaTek chips contain a special AI processing unit (APU) and audio Digital signal processor (DSP) to improve media performance and reduce CPU usage. But both the APU and the audio DSP use custom microprocessor architectures which makes the MediaTek DSP a unique and challenging target for security research. But CheckPoint grew curious about the degree to which the MediaTek DSP could be used as an attack vector for threat actors. So they managed to reverse engineer the MediaTek audio processor and discovered a handful of security flaws.

These flaws can be updated with firmware. So keeping Android devices current continues to be as important as ever. While the chips remained a proprietary mystery, the likelihood of their exploitation remained low. But CheckPoint's write-up is extremely detailed and now offers a readily available roadmap to any technically competent miscreant who might want it with topics such as:

- Classic heap overflow in the AUDIO_DSP_TASK_MSGA2DSHAREMEM message handler.
- Classic heap overflow in the init_share_mem_core function.
- Improper validation of array index in the audio_dsp_hw_open_op function.

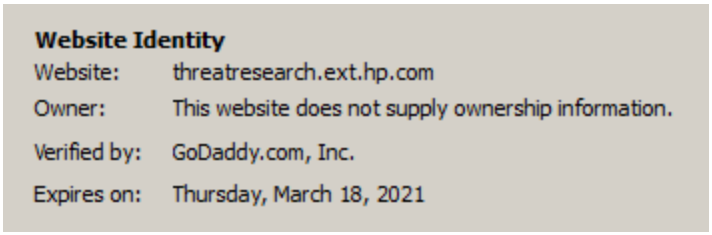
The problem, of course, is that off-brand or unmaintained smartphones are far less likely to ever obtain updates to their MediaTek firmware. Their original vendors won't even bother even if their users did. So, on top of the already overwhelming number of known and never-to-be-patched vulnerabilities from the past, CheckPoint has just carefully uncovered and documented another handful.

The RAT Dispenser

It's probably worth taking just a moment to reinforce the need to never, and I really mean never, click to open an attachment received in an eMail — any eMail — even if it's from your mother.

"Cybersecurity experts from HP" said they discovered a new strain of JavaScript malware that criminals are using as a way to infect systems and then deploy dangerous remote access trojans (RATs).

I should mention that I placed "Cybersecurity experts from HP" in quotes because it was quite difficult to get to the HP Wolf Security Blog due to the fact that the site's TLS certificate had expired back in March. I received all manner of flashing red warnings which I needed to push past in order to get to their research. And I tried with both Firefox and Chrome. Same result. It's wonderful that they are using TLS1.3, but I really cannot understand how they could be using a certificate that expired eight months ago.



In any event, if anyone from HP is listening to this, checkout https://threatresearch.ext.hp.com

Anyway, HP Explains...

Threat actors are always looking for stealthy ways of delivering malware without being detected. In this article, we describe how attackers are using an evasive JavaScript loader, that we call RATDispenser, to distribute remote access Trojans (RATs) and information stealers. With [only] an 11% detection rate, RATDispenser appears to be effective at evading security controls and delivering malware. In total, we identified eight malware families distributed using this malware during 2021. All the payloads were RATs [remote access Trojans], designed to steal information and give attackers control over victim devices.

As with most attacks involving JavaScript malware, RATDispenser is used to gain an initial foothold on a system before launching secondary malware that establishes control over the compromised device. Interestingly, our investigation found that RATDispenser is predominantly being used as a dropper (in 94% of samples analyzed), meaning the malware doesn't communicate over the network to deliver a malicious payload. The variety in malware families, many of which can be purchased or downloaded freely from underground marketplaces, and the preference of malware operators to drop their payloads, suggest that the authors of RATDispenser may be operating under a malware-as-a-service business model.

The infection chain begins with a user receiving an email containing a malicious attachment. It's the classic double file extension "OrderInformation.txt.js" — which we've known about for how long? Yet it still works? So the unwitting user simply needs to double-click the file to run the malware.

What I want to know, is how is it that any of today's eMail clients will run such a file with a simple double click? I just had to promise my first born child just to view this report from HP because their once-legitimate TLS certificate had expired. How is it that clicking on **any** link in **any** eMail is allowed to run JavaScript? That's just insane! The industry clearly has its priorities backwards!

HP notes that network defenders can prevent infection by blocking executable email attachment file types from passing through their email gateways, for example JavaScript or VBScript. Defenders can also interrupt the execution of the malware by changing the default file handler for JavaScript files, only allowing digitally signed scripts to run, or disabling Windows Script Host (WSH). But I ask why any of that's even necessary. It won't protect anyone outside of those boundaries.

When the malware runs, the JavaScript decodes itself at runtime and writes a VBScript file to the %TEMP% folder using cmd.exe. To do this, the cmd.exe process is passed a long, chained argument, parts of which are written to the new file using the echo function. Then the VBScript file runs to download the malware payload. If it was downloaded successfully, it is executed, and the VBScript file is deleted.

The initial JavaScript downloader is obfuscated and contains several eval functions. One of the eval calls is a function that returns a long string, which is decoded by another function. And it's clearly effective since only 1 out of every 10 instances are now being detected after many months of successful exploitation.

Over the past three months, HP said the malware had been used to drop at least eight different RAT strains, such as STTRAT, WSHRAT, AdWind, Formbook, Remcos, Panda Stealer, GuLoader, and Ratty.

So as I started out saying, it's worth just refreshing the strength of the prohibition against ever clicking on anything received in eMail.

0-Day Watch

The Entirely Predictable 0-Day Windows Exploit

Right on schedule Cisco’s Talos group discovered the active exploitation of a 0-day elevation of privilege vulnerability in Microsoft Windows Installer. This vulnerability allows an attacker with a limited user account to elevate their privileges to become an administrator. This vulnerability affects every version of Microsoft Windows, including fully patched Windows 11 and Server 2022. Talos detected malware in the wild taking advantage of this vulnerability.

What was entirely predictable about this? We’ve been tracking this one for some time. Microsoft was first informed of the fundamental underlying problem by security researcher Abdelhamid Naceri who discovered this elevation of privilege vulnerability and worked with Microsoft to address it. However, when Naceri examined Microsoft’s supposed patch for this following this month’s patch Tuesday, he discovered that Microsoft had merely patched against the proof-of-concept that he had provided, rather than addressing and repairing the underlying flaw.

To demonstrate this, Naceri then published proof-of-concept exploit code on GitHub on Nov. 22 which works despite the fixes implemented by Microsoft. The code Naceri released leverages the discretionary access control list (DACL) for Microsoft Edge Elevation Service to replace any executable file on the system with an MSI file, allowing an attacker to run code as an administrator to gain full control over the compromised system, including the ability to download additional software, and modify, delete, or exfiltrate sensitive information stored in the machine.

Independent security researcher, Kevin Beaumont, tweeted: "Can confirm this works, local priv esc. Tested on Windows 10 21H2 and Windows 11. The prior patch MS issued didn't fix the issue properly." Naceri noted that the latest variant of CVE-2021-41379 is "more powerful than the original one," and that the best course of action would be to wait for Microsoft to release a security patch for the problem "due to the complexity of this vulnerability."

So, a security researcher responsibly and privately reports a serious problem to Microsoft including a proof-of-concept demonstration. Microsoft responds, not by fixing the problem, but by breaking the security researcher's proof-of-concept claiming that the problem has been fixed. Annoyed security researcher then publicly posts another proof-of-concept to demonstrate that Microsoft actually fixed nothing. Somewhere, this is seen as good news as malware authors jump on this now-public and well-documented unpatched vulnerability in Windows, using it to obtain administrative rights on Windows machines. We're all now living with the consequences of Microsoft's deliberate de-emphasis of Windows' pre-release testing. Unfortunately, it appears that post-release vulnerability patching has also been de-emphasized.

Sci-Fi

"The Frontiers Saga: Fringe Worlds", the 1st of 15 books in Ryk Brown's 3rd, 15-book story arc, launched on Thanksgiving. Ryk is no fan of Kindle Unlimited, feeling that it doesn't fairly reward authors for their work. In his announcement e-mail he said that the book would not be on Kindle Unlimited but would be for sale for $3 for at least a few months, after which he would put it on Kindle Unlimited. And I would have happily paid $3 for all the hours of pleasant relaxation I obtain from his writing. But I just checked and there it was on Kindle Unlimited.

https://www.amazon.com/dp/B09MJXZ8K7

I'm finishing up my complete re-read of Ryk's first 30 books. I'm on the last one. But then I plan to finally see what the Bobiverse series is all about. So many of our listeners have recommended it, that it's next.

However, my recent reading progress has been slow because I've truly become fully engaged in the work on SpinRite. I go to sleep thinking about it. I awaken thinking about it and I'm thinking about it right now. It's all I want to do because I'm becoming very excited about what it is becoming. For one thing, it's guaranteed highly engaging full employment for the next several years as I publish successively more capable versions. Watching its new benchmarks run, the non-uniformity of performance across both spinning **and** solid-state memory surfaces is quite apparent. The only thing **that** can represent, is trouble... and SpinRite can fix that.

I can't wait to get to the point where SpinRite will have the ability to zero-in on spots that it has discovered are reluctant to be read. In the case of solid state memory, the electrostatic charge of individual bits are leaking, losing their certainty, thus requiring more work and time to be read. SpinRite will be able to detect that and selectively re-write just those trouble spots to recharge them before any data is lost. And if there's actual media damage, SpinRite will demonstrate that to the media's controller, enabling it to relocate the data to keep it safe.

There's a lot of work yet to be done before we'll be there. But I love the work. I need to get v6.1 out to satisfy the immediate needs while I keep working. Then SpinRite needs to be moved away from DOS over to the OnTime RTOS-32 kernel for operation on either BIOS or UEFI. Then native drivers for USB and NVMe need to be added to that environment. And at that point I'll be in a position to develop an entirely new media surface analysis system. I can't wait to get there!

Closing the Loop

David Wright / @wright_de

Another month, another Microsoft balls-up! Windows Server updates this month kick Exchange in the teeth. Some part of the update doesn't have the correct privileges, so a certificate doesn't get updated and you can't access Exchange control panel or OWA etc. afterwards. That was a great start to Monday morning, after updating at the weekend - everything seemed to go through smoothly (mails going in and out after the update, but only through automated SMTP), once the users started coming into work, things went downhill!

Bob Thomas / @btcomp

@SGgrc Any word on Microsoft plans on fixing network printing. I can't print any other way. I got notice to update & it was horrible, no printing and I could not tell which update, of three, was responsible. Until all uninstalled... and then the printer printed! P.O.ed at Microsoft!

I assume Bob meant that he had no choice other than to print over a network. There is news that the promised fix for printing has been released into broader testing and, assuming that all goes well, it should be December 14th. Since tomorrow is December 1st, we have another of those latest-possible Patch Tuesdays of the month — on the 14th.

Rick Nyman / @ricknyman

In Microsoft's defense regarding JavaScript in Excel, they are playing catch up to Google, who has had JavaScript based Google Apps Script in Sheets for a long time. Microsoft needs to replace VBA with something current. But yes, I hope they can secure it.

Donn Edwards / @donnedwards

Hi Steve, happy Thanksgiving
Is the IETF going to allocate the 0.0.0.0 address space as well?
Windows uses 0.0.0.0 in the hosts file the same way it uses 127.0.0.1
What could possibly go wrong?

418: Tea Ready? / @ramriot

> ***Replying to @nixcraft***
> *It appears you have a copy of Spinrite by @SGgrc relabelled as Norton Disk Doctor, how did that happen?*

Ken Mix / @ken_mix

> *Hello Steve -- As I listened to SN846, I had to pause and pull up the IETF draft for the 127/8 unicast proposal. As a frequent reader of RFCs for my day job as a network engineer, I often find myself checking the authors section at the bottom to try and glean some insight into their motivations by seeing which companies or organizations they are associated with. In the draft you referenced (https://www.ietf.org/id/draft-schoen-intarea-unicast-127-00.html), all of the authors reference the "IPv4 Unicast Extensions Project". Google led me to this Github page:*
>
> *https://github.com/schoen/unicast-extensions*
>
> *So, it looks like this is a longer-term effort these gentlemen are working towards.*
>
> *Professionally speaking, I would NOT want to be assigned an address out of this pool if it were released to the RIRs for unicast use, as I imagine there would be some hosts or even entire ISPs that would never implement this standard, ensuring that any services hosted on those IPs would have flakey connectivity, at best.*
>
> *One whopper of an understatement in the "Compatibility and Interoperability" section of the draft gave me a bit of a chuckle:*
>
> *"Since deployed implementations' willingness to accept 127/8 addresses as valid unicast addresses varies, a host to which an address from this range has been assigned may also have a varying ability to communicate with other hosts."*

An understatement, indeed! And that leads us into today's discussion of the broader goals of the IPv4 Cleanup Project...

Bogons Begone!

Before we begin, since we're necessarily going to be talking about network addressing, I want to make sure everyone is on the same page about the nomenclature for describing IPv4 networks.

One of the many brilliant innovations made by the designers of the Internet's routing architecture was the idea of dividing the 32-bits of IPv4 address space into a network number, and the number of a specific machine within that network. So when, for example, we talk about the "10 dot" network, we mean any IP whose first, leftmost 8-bits (or octet) of network address is 10. But the clearer and more formal way of describing it is 10/8, where the '8' refers to the number of bits, counting from the left, which will be used to designate the network number, and the rest of the bits to the right will be used to designate a specific machine within that network.

Last week I prefaced our discussion of the IETF's stated intention to redefine the entire currently unroutable 127 into two pieces: 127.0/16 would remain unroutable and be used as a localnet containing the default localhost IP of 127.0.0.1 plus 65,535 other 127.0 IPs. Then, the rest of 127/8, which would be from 127.1/16 through 127.255/16, would be made available to the IANA and regional Internet registries as blocks of newly routable IPv4 addresses.

But it turns out that this IETF draft proposal is only the tip of the iceberg. And since our discussion of this bit of the iceberg drew so much interest and response last week, I decided that while we were on the topic, I ought to share the rest.

So we need to talk about Bogons.

By definition, “Bogons” are IP packets having unroutable source or destination IPs which should, therefore, never appear on the public Internet.

For a formal and fun definition of Bogons we can turn to Wikipedia:

> *The term bogon stems from hacker jargon, with the earliest appearance in the Jargon File in version 1.5.0 (dated 1983). It is defined as the quantum of bogosity, or the property of being bogus. A bogon packet is frequently bogus both in the conventional sense of being forged for illegitimate purposes, and in the hackish sense of being incorrect, absurd, and useless. These unused IP addresses are collectively known as a bogon, a contraction of "bogus logon", or a logon from a place you know no one can actually logon.*

And for the record, despite the similarity in sound, Bogons should never be confused with Vogons. Vogons are, of course, the distasteful alien race created by Douglas Adams for his Hitchhiker's Guide to the Galaxy. As we learned in his first novel, Vogons take on very large construction projects and specialize in demolition.

Last week, before introducing the localnet 127/8 network, we talked about the concept of non-routable networks by reminding everyone of the most common non-routable private networks that most of us probably have: 192.168.0/24 or 192.168.1/24. These are small pieces of the larger 192.168/16 network that was set aside by RFC 1918. And, as we know, that RFC also defined 10/8 and 172.16/12 to be similar set aside non-routable networks to be used to

number the machines inside of private LANs. So all of the IPs within those ranges are Bogons.

But there are also other long-standing set-asides, and they, too, have come under the scrutiny of a group which has founded the so-called “IPv4 Cleanup Project”. And they're quite serious. Before we consider what they have to say, let's take a look at these other Bogon addresses. The original RFC 3330, dated September 2002, was later obsoleted by RFC 5735 in 2010 to reflect a few changes, and both RFCs were titled: “Special Use IPv4 Addresses”.

Abstract:

This document describes the global and other specialized IPv4 address blocks that have been assigned by the Internet Assigned Numbers Authority (IANA). It does not address IPv4 address space assigned to operators and users through the Regional Internet Registries, nor does it address IPv4 address space assigned directly by IANA prior to the creation of the Regional Internet Registries. It also does not address allocations or assignments of IPv6 addresses or autonomous system numbers.

Introduction:

Throughout its history, the Internet has employed a central Internet Assigned Numbers Authority (IANA) responsible for the allocation and assignment of various identifiers needed for the operation of the Internet [RFC1174]. In the case of the IPv4 address space, the IANA allocates parts of the address space to Regional Internet Registries (RIRs) according to their established needs. These RIRs are responsible for the registration of IPv4 addresses to operators and users of the Internet within their regions.

On an ongoing basis, the IANA has been designated by the IETF to make assignments in support of the Internet Standards Process [RFC2860]. Section 4 of that document describes that assignment process.

Small portions of the IPv4 address space have been allocated or assigned directly by the IANA for global or other specialized purposes. These allocations and assignments have been documented in a variety of RFCs and other documents. This document is intended to collect these scattered references and provide a current list of special use IPv4 addresses.

This document is a revision of RFC 3330 [RFC3330], which it obsoletes; its primary purpose is to reflect the changes to the list of special IPv4 assignments since the publication of RFC 3330. It is a companion to [RFC5156], which describes special IPv6 addresses.

Global and Other Specialized Address Blocks

0.0.0.0/8 - (Any IP whose first octet is 0) Addresses in this block refer to source hosts on "this" network. Address 0.0.0.0/32 may be used as a source address for this host on this network; other addresses within 0.0.0.0/8 may be used to refer to specified hosts on this network ([RFC1122], Section 3.2.1.3).

10.0.0.0/8 - This block is set aside for use in private networks. Its intended use is documented in [RFC1918]. As described in that RFC, addresses within this block do not legitimately appear

on the public Internet. These addresses can be used without any coordination with IANA or an Internet registry.

127.0.0.0/8 - This block is assigned for use as the Internet host loopback address. A datagram sent by a higher-level protocol to an address anywhere within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32 for loopback. As described in [RFC1122], Section 3.2.1.3, addresses within the entire 127.0.0.0/8 block do not legitimately appear on any network anywhere.

169.254.0.0/16 - This is the "link local" block. As described in [RFC3927], it is allocated for communication between hosts on a single link. Hosts obtain these addresses by auto-configuration, such as when a DHCP server cannot be found.

172.16.0.0/12 - This block is set aside for use in private networks. Its intended use is documented in [RFC1918]. As described in that RFC, addresses within this block do not legitimately appear on the public Internet. These addresses can be used without any coordination with IANA or an Internet registry.

192.0.0.0/24 - This block is reserved for IETF protocol assignments. At the time of writing this document, there are no current assignments. Allocation policy for future assignments is given in [RFC5736].

192.0.2.0/24 - This block is assigned as "TEST-NET-1" for use in documentation and example code. It is often used in conjunction with domain names example.com or example.net in vendor and protocol documentation. As described in [RFC5737], addresses within this block do not legitimately appear on the public Internet and can be used without any coordination with IANA or an Internet registry. See [RFC1166].

192.88.99.0/24 - This block is allocated for use as 6to4 relay anycast addresses, in [RFC3068]. In contrast with previously described blocks, packets destined to addresses from this block do appear in the public Internet. [RFC3068], Section 7, describes operational practices to prevent the malicious use of this block in routing protocols.

192.168.0.0/16 - This block is set aside for use in private networks. Its intended use is documented in [RFC1918]. As described in that RFC, addresses within this block do not legitimately appear on the public Internet. These addresses can be used without any coordination with IANA or an Internet registry.

198.18.0.0/15 - This block has been allocated for use in benchmark tests of network interconnect devices. [RFC2544] explains that this range was assigned to minimize the chance of conflict in case a testing device were to be accidentally connected to part of the Internet. Packets with source addresses from this range are not meant to be forwarded across the Internet.

198.51.100.0/24 - This block is assigned as "TEST-NET-2" for use in documentation and example code. It is often used in conjunction with domain names example.com or example.net in vendor and protocol documentation. As described in [RFC5737], addresses within this block do not legitimately appear on the public Internet and can be used without any coordination with IANA

or an Internet registry.

203.0.113.0/24 - This block is assigned as "TEST-NET-3" for use in documentation and example code. It is often used in conjunction with domain names example.com or example.net in vendor and protocol documentation. As described in [RFC5737], addresses within this block do not legitimately appear on the public Internet and can be used without any coordination with IANA or an Internet registry.

224.0.0.0/4 - This block, formerly known as the Class D address space, is allocated for use in IPv4 multicast address assignments. The IANA guidelines for assignments from this space are described in [RFC3171].

240.0.0.0/4 - This block, formerly known as the Class E address space, is reserved for future use; see [RFC1112], Section 4. The one exception to this is the "limited broadcast" destination address 255.255.255.255. As described in [RFC0919] and [RFC0922], packets with this destination address are not forwarded at the IP layer.

There's no way we can mess with the 3 private and widely used RFC 1918 networks. They are clearly safe. 169.254.0.0/16 is safe too. Anyone who has ever turned on a networked Windows machine without an explicitly configured IP and no DHCP server around may have discovered that Windows assigned itself an IP within the 169.254/16 range. I've never been curious enough to look into it. But I would guess that any network interface configured for "Obtain IP Address Automatically" when DHCP doesn't answer, probably picks a provisional IP within that range and sends out an ARP broadcast asking all network interfaces on the LAN whether they have that IP. And if no one answers, the machine uses that as its own. In any event, that's not going away.

The "zero net" is interesting. And it's a /8 ... so it's taken 16.77 million IPv4's (1/256th of the entire IPv4 space) out of service.

The three TEST-NETs are all /24's, so they aren't worth bothering with.

The 198.18.0.0/15 sets aside and ties up 128K IPs ostensibly for use in benchmark tests of network interconnect devices so their packets don't get loose on the Internet. That one's sort of difficult to appraise.

So, aside from the "Zero net", this leaves us with the two monster allocations: The 224.0.0.0/4 which has been set aside for use in IPv4 multicast assignments. And the 240.0.0.0/4 which the RFC simply states has been reserved for future use.

And this brings us to the "IPv4 Cleanup Project"
https://github.com/schoen/unicast-extensions

About

The IPv4 unicast extensions project - Making class-e (240/4), 0/8, 127/8, 225/8-232/8 generally usable - adding 419 million new IPs to the world, and fixing various other slightly broken pieces of the IPv4 world.

Fixing the odd nooks and crannies still mildly broken in IPv4, by:

- Making class-e (240/4), 0/8, 127/8, 224/4 more usable
- Adding 419 million new IPs to the world
- Fixing zeroth networking
- Improving interoperability with multiple protocols and tunnelling technologies
- Supplying tested patches and tools that address these problems

NANOG is the "North American Network Operators' Group" — and their NANOG listserve is pretty much where most of the Internet evolves. Thursday before last, on November 18th, John Gilmore responded at some length to a NANOG posting.

John Gilmore (one of the people in the IPv4 Cleanup Project) is one of the founders of the Electronic Frontier Foundation (the EFF). He created the Cypherpunks mailing list, and co-founded Cygnus Solutions. He created the alt.* hierarchy in Usenet (boy, what a sewer that was back in the day!) and John is also a major contributor to the GNU Project. He's a civil libertarian who once famously quipped that "the Internet interprets censorship as damage and routes around it." So I suppose he's also something of an idealist. Anyway, Steven Bakker, a network engineer of some repute posted:

> *The ask, is to update every IP stack in the world (including validation, equipment retirement, reconfiguration, etc)...*

John took this in good spirit and appeared unruffled. He makes a number of valid points while presenting and stating a well thought out case for hugely reducing the Internet's current bogon bloat. So I felt that it was worth hearing John out. The purpose of the preamble was to create a foundation and context for understanding John's reply. Here's what John Gilmore wrote:

> *This raises a great question.*
>
> *Is it even *__doable__*? What's the *__risk__*? What will it *__cost__* to upgrade every node on the Internet? And *__how long__* might it take?*
>
> *We succeeded in upgrading every end-node and every router in the Internet in the late '90s and early 2000's, when we deployed CIDR. It was doable. We know that because we did it! (And if we hadn't done it, the Internet would not have scaled to world scale.)*
>
> *So today if we decide that unicast use of the 268 million addresses in 240/4 is worth doing, we can upgrade every node. If we do, we might as well support unicast on the other 16 million addresses in 0/8, and the 16 million in 127/8, and the other about 16 million reserved for 4.2BSD's pre-standardized subnet broadcast address that nobody has used since 1985. And take a hard look at another hundred million addresses in the vast empty multicast space, that have never been assigned by IANA for anybody or anything. Adding the address blocks around the edges makes sense; you only have to upgrade everything once, but the 268 million addresses becomes closer to 400 million formerly wasted addresses. That would be worth half again as much to end users, compared to just doing 240/4!*

That may not be worth it to you. Or to your friends. But it would be useful to a lot of people -- hundreds of millions of people who you may never know. People who didn't get IP addresses when they were free, people outside the US and Europe, who will be able to buy and use them in 5 or 10 years, rather than leaving them unused and rotting on the vine.

We already know that making these one-line patches is almost risk-free. 240/4 unicast support is in billions of nodes already, without trouble. Linux, Android, MacOS, iOS, and Solaris all started supporting unicast use of 240/4 in 2008! Most people -- even most people in NANOG -- didn't even notice. 0/8 unicast has been in Linux and Android kernels for multiple years, again with no problems. Unicast use of the lowest address in each subnet is now in Linux and NetBSD, recently (see the drafts for specifics). If anyone knows of security issues that we haven't addressed in the drafts, please tell us the details! There has been some arm-waving about a need to update firewalls, but most of these addresses have been usable as unicast on LANs and private networks for more than a decade, and nobody's reported any firewall vulnerabilities to CERT.

Given the low risk, the natural way for these unicast extensions to roll out is to simply include them in new releases of the various operating systems and router OS's that implement the Internet protocols. It is already happening, we're just asking that the process be adopted universally, which is why we wrote Internet-Drafts for IETF. Microsoft Windows is the biggest laggard; they drop any packet whose destination OR SOURCE address is in 240/4. When standards said 240/4 was reserved for what might become future arcane (variable-length, anycast, 6to4, etc) addressing modes, that made sense. It doesn't make sense in 2021. IPv4 is stable and won't be inventing any new addressing modes. The future is here, and all it wants out of 240/4 is more unicast addresses.

By following the normal OS upgrade path, the cost of upgrading is almost zero. People naturally upgrade their OS's every few years. They replace their server or laptop with a more capable one that has the latest OS. Laggards might take 5 or 10 years. Peoples' home WiFi routers break, or are upgraded to faster models, or they change ISPs and throw the old one out, every 3 to 5 years. A huge proportion of end-users get automatic over-the-net upgrades, via an infrastructure that had not yet been built for consumers during the CIDR transition. "Patch Tuesday" could put some or all of these extensions into billions of systems at scale, for a one-time fixed engineering and testing cost.

We have tested major routers, and none so far require software updates to enable most of these addresses (except on the lowest address per subnet). At worst, the ISP would have to turn off or reconfigure a bogon filter with a config setting. Also, many "Martian address" bogon lists are centrally maintained (e.g. https://team-cymru.com/community-services/bogon-reference/) and can easily be updated. We have found no ASIC IP implementations that hardwire in assumptions about specific IP address ranges. If you know of any, please let us know, otherwise, let's let that strawman rest.

Our drafts don't propose to choose between public and private use of the newly usable unicast addresses (so the prior subject line that said "unicast public" was incorrect). Since the kernel and router implementation is the same in either case, we're trying to get those fixed first.

There will be plenty of years and plenty of forums (NANOG, IETF, ICANN, IANA, and the RIRs) in which to wrestle the public-vs-private questions to the ground and make community decisions on actual allocations. But if we don't fix the kernels and routers first, none of those decisions would be implementable.

Finally, as suggested by David Conrad, there is a well understood process for "de-bogonizing" an address range on the global Internet, once support for it exists in OS's. Cloudflare used it on 1.1.1.1; RIPE used it on 128.0/16 and on 2a10::/12. You introduce a global BGP route for some part of the range, stand up a server on it, and use various distributed measurement testbeds to see who can reach that server. When chunks of the Internet can't, an engineer figures out where the blockage is, and communicates with that ISP or vendor to resolve the issue. Lather, rinse and repeat for a year or more, until reachability is "high enough".

Addresses that later end up allocated to private address blocks would never need 100% global reachability, but global testing would still help to locate low-volume OS implementations that might need to be updated. Addresses purchased to number retail cellphones need not be as reachable as ones used on public-facing servers, etc. The beauty of a market for IP addresses, rather than one-size-fits-all allocation models, is that ones with different reachability can sell for different prices, at different times, into different niches where they can be put to use.

John

I like John's argument. The gist of it is that we have nothing to lose by immediately changing some of today's long-established IPv4 standards. IPv4 has outgrown its original design which deliberately incorporated a lot of waste that once seemed insignificant. That waste is not insignificant today. IP stack vendors will be formally notified that the IETF is changing a bunch of definitions which affect a handful of mostly unused IPv4 space. All that's needed are some tweaks to their OS's default routing table.

Engineers simply need clear guidance and specification to do whatever they need to. Last week we noted that F5 Networks BIG-IP systems were currently using some of the space under 127 that's now slated to become routable. But the entire 127.0/16 network remains local and non-routable. So all F5 needs to do is migrate those arbitrarily scattered blocks they're currently using, which are outside the lower 64K IPs, down into that range. Once that's done they'll know that if the routability of 127/8 should ever change, they'll already be compatible. And there's no reason for them not to change that today since it will be completely compatible with our current system.

The other thing I appreciate about John's post is his sense of time. Everyone's always in a hurry, and 10 years sounds like forever. But it'll be here before we know it. And when we get there it would be nice to find that 419 million more IPv4 addresses have been widely usable for some time. I think the point is: It's entirely possible that it might not ever happen. But it can never happen if we never make it possible. And making it possible is not difficult.