Security Now! #793 - 11-17-20

SAD DNS

This week on Security Now!

This week the Chrome 0-days just keep on coming, and we contemplate what it means for the future. We have two interesting bits of ransomware meta-news including a new tactic. We update after last week's Super Tuesday patch marathon, and examine new research into the most common source of Android malware to see where most unwanted apps come from — and it's not what we would likely guess. We'll share a bit of listener feedback and an update on my work on SpinRite. Then we look at the new "SAD DNS" attack which successfully regresses us 12 years in DNS cache poisoning and spoofing attack prevention!

Ah... PHP. How we love to hate thee.

Browser News

Two more new 0-days revealed in Chrome.
Last week we had three 0-days patched in the previous two weeks.
Today, we have five 0-days patched in the previous three weeks.

Last Wednesday the 11th, Chrome announced the Stable Channel Update for Windows, writing that the stable channel has been updated to 86.0.4240.198 for Windows, Mac & Linux which will roll out over the coming days/weeks. [Although I think that was conservative.]

Under "Security Fixes and Rewards" they noted with their standard boilerplate that details would be kept restricted until the majority of users would no longer be affected. They indicated that both of the in-the-wild 0-days were discovered and reported by "Anonymous", the first on the 7th and the second on the 9th. So the update was pushed out to our desktops very quickly after it was reported to Google and the bounty reward was "$TBD." The first flaw was another "Inappropriate implementation in V8" which was the exact language used to describe the previous week's vulnerability. And the other flaw was a "Use after free flaw in site isolation."

This is a model for the way we need to be doing security moving forward: Researchers spot problems, either doing static research or by catching something happening in the wild. They report them privately to the responsible party. That responsible party rewards them for their discovery and for keeping their report private, and then quickly updates the affected software and pushes it out to all affected parties and/or devices.

Something we know with absolute certainty today, is that "Cyberwar" and "Cybercrime" — ad hoc or organized — are real things. They exist today and probably will, now, forever. As with encryption, once that genie is out of the bottle it cannot be put back. 15 years ago, when this podcast was launched, some cyberwar may have been underway by nation states. I may have mentioned before that in the early days I was once approached by a three letter agency and asked whether I might be interested in developing networked cyber technology for offensive purposes. I declined, since that doesn't really appeal to me and I naively failed to appreciate that my country might actually be under attack, then or in the future. But I'm sure that plenty of others did not decline, which suggests that this has been percolating for decades. Stuxnet showed us what was possible. Edward Snowden filled in many suspicions and revealed a range of sobering truths about what our U.S. intelligence services had been, and were, up to.

During the 15 year life of this podcast, we've watched it happen: Cyberwar and cybercrime have gone from "Really?" to "Oh my God! Take us off the Net now!!!" It has become front of mind for many in the Information Technology industry. And CSO — Chief Security Officer — is now a "C suite" position.

Of course, not all software is in the same position. The firmware controlling an electric toothbrush just needs to manage its rechargeable battery, run a timer and control its motor. If it's not "connected" no one can get to it. But **any** software that **is** "connected" — even if it's a coffee pot, a thermostat or a printer — is exposed to what is obviously an increasingly hostile world. Nothing connected to the Internet can afford the luxury of being naïve to the very real possibility that it might be used as an entry point for cybercrime. And no one using connected devices can afford not to exercise some caution over their deployment within a network.

As I noted above, the Chrome project has evolved the right model. The browser is on the front lines and it needs to be updateable autonomously and with a super-short cycle. But Google has not done it right everywhere. As we know, Android is a catastrophe. Google is working hard to fix their earlier mistakes, making more of the OS push-updatable. But the vast majority of today's Android devices will never be updated. We can only hope, for the sake of their owners, that those devices die as soon as possible, to then be replaced by Google's newer autonomously updatable operating systems.

As for the rest of the consumer and enterprise industry, the same goes: The only responsible way to produce anything that's connected to the Internet is to provide a built-in means for allowing the party responsible for its maintenance an autonomous means for pushing out critical updates. We're not there yet, but it's where we need to go.

And speaking of Cybercrime...

Ransomware News

Intel421 Report:
https://public.intel471.com/blog/ransomware-as-a-service-2020-ryuk-maze-revil-egregor-dopplepaymer/

We have, unfortunately, been observing that the ransomware "business" is booming. However, it's easy to lose sight of the forest when examining its individual trees. When we pull back and take in the entire scope of what's been evolving, somewhat unseen, what we find is a bit startling.

How many Ransomware-as-a-Service (RaaS) operations are there? In the past, we've touched on the coming and going of a few. But a comprehensive review finds that there are presently more than twenty-five separate RaaS service portals now in the business of renting ransomware for use by other criminal gangs.

The group Intel421 published their report yesterday titled: "Ransomware-as-a-service: The pandemic within a pandemic." There report was not highly detailed, but it was illuminating. I dislike legitimizing this by calling it a business, but a business it is. And it's not really surprising that there would be a hierarchy of players at the top who are pulling down the lion's share of the extortion revenue, while there's a much larger base of wannabe players hoping to make their mark and get cut in for a piece of the action.

So, I'll just note that this podcast has not missed anything. The big players noted by Intel421's research are **DopplePaymer**, **Egregor**, **Ragnar Locker**, **REvil** and **Ryuk**. They're the ones in the news and the ones making the lion's share of the money.

But one thing this research does show is that there are around 20 other me-too players who are working to move from also-ran to "pay us now or else". They might not remain 2nd tier forever.

Ragnar Locker takes out a Facebook ad

And speaking of the Ragnar Locker ransomware gang: Last week we mentioned their successful Ransomware attack on the Italian distiller Compari and their publication of the contract between Wild Turkey and Matthew McConaughey. It appears that this gang is into shaming their victims as a means of inducing payment. I suppose this is the logical next step once companies become more able to recover from the original simpler local ransomware encryption attacks.

So get this: After obtaining two terabytes of sensitive data stolen during their November 3rd attack exactly two weeks ago, today, and demanding a $15 million ransom paid in Bitcoin, the Ragnar Locker gang have used a compromised Facebook account to take out public Facebook ads threatening to public release the sensitive Compari data unless the ransom is paid. Apparently, this new tactic is intended to public shame and pressure the Ransomware victim into paying.

The compromised account belonged to a Chicago-based deejay by the name of Chris Hodson. Chris believed that all of his online accounts were protected by 2-factor authentication. But it turned out that he had missed one — Facebook. Brian Krebs, who reported on this, wrote that Chris said a review of his account showed the unauthorized campaign reached approximately 7,150 Facebook users, and generated 770 clicks, with a cost-per-result of 21 cents. Chris said Facebook billed him $35 for the first part of the campaign, but apparently detected the ads as fraudulent before his account could be billed another $159 for the campaign.

So we have a new wrinkle added to the ransomware scenario: Ransomware gangs who get into a corporate network are going to first exfiltrate as much data as they can, then they will deny its owner's access by encrypting it in their wake. Then they will then contact the victim and demand payment — not only for providing the master decryption key, but also to threaten the release of their victim's sensitive corporate data, with the newly added wrinkle that they might widely and publicly employ an advertising pressure campaign to up the ante and coerce payment.

And given that these attackers have already shown their true colors, there's no reason to expect, nor means to enforce, the permanent deletion of the sensitive stolen data. It's a mess.

Security News

Last Tuesday's Windows Patching

... and speaking of messes... Last Tuesday, Microsoft fixed 112 known vulnerabilities in Microsoft products. 17 of those 112 were rated CRITICAL, 93 were Important and 2 were rated moderate. In terms of remote code execution, 24 of those flaws created pathways for attacker-supplied code to be executed in vulnerable systems using Excel, Sharepoint, Exchange Server, Windows Network File System, Windows GDI+ component, the Windows printing spooler service, and Microsoft Teams.

We did receive the expected and hoped-for patch for the 0-day privilege escalation vulnerability in the Windows Kernel Cryptography Driver (cng.sys) we talked about previously. This was the one used in an attack chain with the earlier Chrome 0-day.

I checked over at 0patch.com, since I'm wondering whether they'll be offering a fix for Windows 7 and Server 2008 R2, but so far, nothing.

In addition to the repaired 0-day, and those many other repaired remote code execution vulnerabilities, an unspecified "security bypass flaw" was fixed in Windows Hyper-V. But calling something a "security bypass" says absolutely nothing about it, which I suppose was Microsoft's intention, since what cannot be described as a security bypass? Last week's picture of that locked gate standing by itself out in a field? Yep... that's sure vulnerable to a security bypass!

One of the RCE's, that Network File System remote code execution is rated 9.8 out of 10. And one place you really never want a remote code execution is in your network file system. Microsoft has also stated that the attack complexity is low. Which is only good news only when you're the attacker. Since NFS runs over TCP and UDP port 2049, expect to be seeing an increase in port scanning targeting 2049. A Shodan query reported 38,893 servers with port 2049 exposed to the Internet; however, exploitation requires that an NFS share be configured for anonymous write access, thus no authentication required. Doing something like that is not all that uncommon. Once upon a time, FTP servers would often allow for anonymous file uploads into specific directories. The idea being that such files received there would be treated with caution, but that there was some valid need for receiving such. So, of those 38,893 servers answering on port 2049 we don't know what percentage of them are actually NFS shares and also configured for anonymous write access. But I'd put some money down on a bet that attackers are working to determine exactly that right now.

Microsoft also fixed critical memory corruption vulnerabilities in their Scripting Engine (CVE-2020-17052) and IE (CVE-2020-17053), and once again, multiple RCE flaws in HEVC Video Extensions Codecs library. And, as we know, all of these are exposed to the Internet.

So, yeah... as always, updating when you can would be wise.

Where do most malicious Android apps come from?
It turns out, not from where it's easiest and most comforting to imagine. **Not** from unauthorized 3rd-party app sources and repositories.

In a recently published 17-page report titled: "How Did That Get In My Phone? Unwanted App Distribution on Android Devices" NortonLifeLock (formerly known as Symantec) introduced the result of their careful 4-month study by writing:

> Android is the most popular operating system with billions of active devices. Unfortunately, its popularity and openness makes it attractive for unwanted apps, i.e., malware and potentially unwanted programs (PUP). In Android, app installations typically happen via the official and alternative markets, but also via other smaller and less understood alternative distribution vectors such as Web downloads, pay-per-install (PPI) services, backup restoration, bloatware, and IM tools. **This work** performs a thorough investigation on unwanted app distribution by quantifying and comparing distribution through different vectors. At the core of our measurements are reputation logs of a large security vendor, which include 7.9 million apps observed in 12 million [Android] devices between June and September 2019. As a first step, we measure that between 10% and 24% of users devices encounter at least one unwanted app, and compare the prevalence of malware and PUP.

An analysis of the who-installs-who relationships between installers and child apps reveals that **the Google Play market** is the main app distribution vector, responsible for 87% of all installs and **67% of unwanted app installs**, while also providing the best defense against unwanted apps.

Alternative markets distribute instead 5.7% of all apps, but over 10% of unwanted apps.

Bloatware is also a significant unwanted app distribution vector with 6% of those installs. And, backup restoration is an unintentional distribution vector that may even allow unwanted apps to survive users' phone replacement. We estimate unwanted app distribution via pay-per-install to be smaller than on Windows. Finally, we observe that Web downloads are rare, but provide a riskier proposition even compared to alternative markets.

So, this interesting research demonstrates that it's really just a numbers game.

We've often said that installing from non-Google Play sources is dangerous. And that's true, as a percentage of per-install sources. The Google Play store does a far better job of keeping the crap off people's phones on a percentage-of-crap basis. But when viewed overall, the Google Play store, being the source of 87% of all Android app installs, simply means that, as their research showed, 2/3rds of all unwanted apps which find their way onto Android devices come directly from the Google Play store.

And, interestingly, this supports and doesn't change our longstanding advice: Don't install stuff just because it's there. Resist the temptation to say "Oooooo! Look at that!" and then click "Yeah, gimmie!" Some percentage **will** be crap — even from the Google Play store. And, in fact, as the research also shows, since there's so much more stuff there overall, the absolute amount of crap on the Google Play store is that much higher, even though as a percentage, it's lower.

The research paper is terrific and any Android user interested in a deep dive should check it out!
https://arxiv.org/pdf/2010.10088.pdf

Listener Feedback

Kevin Morris / @KevMorris_Today
Hi Steve. I've been experimenting with various flavors of Linux and once I burned an ISO to my Flash drive, I always had a devil of a time being able to use it again. InitDisk solved the problem! Thank you for that. Kevin Morris, Santa Clara, CA

/dev/jake / @jakefromstf4rm
@SGgrc playing devil's advocate, but isn't the #letsencrypt issue you spoke of last week a good reason to force users of Android devices older than 7.1 to upgrade if they can or abandon those insecure devices?

SpinRite

Everyone,

Working with millQ, through a series of tests we finally tracked down the cause for drives appearing and disappearing depending upon whether, even though the system was booted from diskette, an unused USB stick also in the machine was causing some hard drives not to be seen.

A register within the PCI RAM space contained bogus data that my code had been relying upon. Since only 4 of that register's 32 bits should ever be non-zero, the addition of a sanity test to that register's contents now prevents it from being trusted when it should not be. This resolved the "drive coming & going" that millQ was seeing, and it might also have fixed other known and as yet unknown problems from ever occurring.

A good day for the future of SpinRite. :)

I'm switching to podcast prep mode until it's behind me.

THANK YOU EVERYONE for the patient testing and feedback. We're making some last improvements that are very useful.

millQ still has some weird crashing behavior on that one very old laptop triggered by his setting of LASTDRIVE in config.sys, and Chris has a hang problem on one machine. But we're getting closer and closer!

And although this is technically listener feedback, it made more sense here:

Dear Steve,

I'm a regular follower of Security Now and a VERY happy SpinRite 6 user. It is my GO-TO tool to test new drives as I purchase them. I find that IF they successfully pass SpinRite's Level 4 testing, they usually give many good years of service.

I'm writing to you today to ask you a favor. With your blessing, I would like to join your testing group for the latest SpinRite. Since it's nearly ready-to-go, what could one more tester hurt? I just purchased 1 of what will be 4, 4TB WD Drives, and found to my dismay, that SpinRite doesn't do GPT in its current version. Duh?? I suppose I should have known this but didn't.

If you could see fit to allow me into the program and use of the current Spinrite Beta, I promise not to blame you if I should ever develop an allergy to asparagus.

For verification purposes, below is my original SpinRite purchase information. The original email address is no longer preferred as the company is defunct (Who could compete with TiVO?), but you can reach me here on Gmail.

With anticipation, Thanks!

SAD DNS

https://www.cs.ucr.edu/~zhiyunq/SADDNS.html

A team of researchers from the University of California at Riverside, and the Tsinghua [Ching Whaa] University in Beijing, China, presented their paper at the ACM Conference on Computer and Communications Security (CCS '20) which was held last week. And it won the Distinguished Paper Award.

Before we go any further, let's review DNS cache poisoning, which this podcast covered in great detail when it first happened 12 years ago:

When a DNS resolver needs to look up the IP address of a domain, it issues a query using the lightest weight Means possible, which is a single UDP packet containing the query. The UDP packet is addressed, in turn, to a more authoritative DNS server, asking it about some component of the DNS domain name path. The replying packet, to be a reply, must be returned to the same port on the first DNS server from which it was sent, and the reply's internal DNS transaction ID must match a query that the DNS server has outstanding.

This system is admirably lean and hyper efficient. But it inherently suffers from a series of now quite well known vulnerabilities. For one thing, UDP is trivial to spoof. Unlike TCP where the famous 3-way handshake verifies the IPs of each endpoint in the conversion, a UDP packet is simply launched from the source IP it claims to whatever destination IP it chooses. And that packet can contain whatever it wishes.

So, 12 years ago, during the summer of 2008's Black Hat conference, Security Researcher Dan Kaminsky presented his discovery of a massively widespread and critical DNS vulnerability that would have allowed attackers to send users to malicious websites, hijack email, and get up to all sorts of mischief. By allowing for the poisoning of a DNS resolver's cache, the exploit would empower attackers to impersonate any legitimate website and steal data, etc. In today's world of HTTPS with certificate protection this is more difficult. But if a DNS cache is poisoned anywhere upstream of where LetsEncrypt does its lookups, then it's trivial to obtain certificates for any desired website. The flaw that Dan found allowed for arbitrary DNS cache poisoning which, at that time, affected nearly every DNS server on the planet. The one brand of DNS server that was not vulnerable had been created by our friend Daniel J. Bernstein who had, nine

years earlier back in 1999 told everyone else not to make the mistake they did, but no one paid attention.

When Dan Kaminski realized the extent of the vulnerabilities — and what it meant for the integrity of the Internet which depended, as it still does today, upon the integrity of DNS — Dan called Paul Vixie, the creator of several DNS protocol extensions and applications used throughout the Internet to tell him about the bug he found. They then called an emergency summit to Microsoft's headquarters to discuss how to address the wide-spread problem, bringing in Paul's contacts and vendor representatives.

This coalition worked silently to create a fix for the vulnerability before it was disclosed to the public, with all vendors simultaneously releasing patches for their products on July 8, 2008. However, according to Dan, they didn't actually repair DNS, they just took a 16-bit transaction identifier (which could not be expanded) and added random UDP source ports as a hack to expand the query entropy from 16 bits to 32 bits. In other words, DNS queries used to be issued from the fixed DNS service port 52, and the query contained a 16-bit transaction ID whose reply needed to match. Even if the transaction ID was randomized, that's only 64K possible IDs. So an attacker could stuff 64K replies into an awaiting DNS server, and one of them would be accepted as valid, thus poisoning that server's cache for anyone who then asked for that reply's DNS IP.

By simply randomizing the DNS query's source port, any successful reply would need to match BOTH the port and the transaction ID, thus expanding the query entropy from 16 bits to 32 bits. Once that had been done, an attack that used to take 10 seconds would take hours, days or weeks. Though it did not eliminate the possibility of attack, it did eliminate the instantaneously successful attack. And the industry breathed a collective sigh of relief about the bullet they had dodged.

As we know, just wanting to have systems upgraded doesn't make it so. Therefore, coincident with that, in the spirit of the ShieldsUP! service, I created GRC's "DNS Spoofability" service. It allows users to quickly and easily check the DNS servers being used by their own Internet connection — originally their ISP's, but more recently any of the other popular third-party providers — to analyze the DNS query transaction entropy of those servers. Back then, many users were discovering that their DNS provider was still using vulnerable DNS resolvers that were generating easily spoofable DNS queries. Fortunately, that's much less so today.

Okay. So that was 12 years ago, in 2008... and things have been relatively quiet since that flurry of frenzied activity to dodge that bullet. Until now. This new group of researchers have titled their award-winning paper:

"DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels"

Their paper explains their accomplishment with its introduction:

> In this paper, we report a series of flaws in the software stack that leads to a strong revival of DNS cache poisoning --- a classic attack which is mitigated in practice with simple and effective randomization-based defenses such as randomized source port. To successfully poison a DNS cache on a typical server, an off-path adversary ["off-path" meaning an

adversary that is not participating in the local network traffic] would need to send an impractical number of 2^32 [4.3 Billion] spoofed responses, simultaneously guessing the correct source port (16-bit) and transaction ID (16-bit).

Surprisingly, we discover weaknesses that allow an adversary to "divide and conquer"' the space by guessing the source port first, followed by the transaction ID (leading to only 2^16 + 2^16 spoofed responses). Even worse, we demonstrate a number of ways an adversary can extend the attack window which drastically improves the odds of success.

This should remind our listeners of the horrible vulnerability that was discovered in the WPA protocol. Remember that it used an 8-digit PIN. That would normally be 100 million combinations and impractical to guess. But it was discovered that the first 4 digits could be guessed first, independently (thus 10,000) and that since the last digit was a check-digit, the final three could then be guessed (thus 1000) so that the guessing space was reduced from 100 million to 11 thousand. In other words... divide and conquer.

The attack affects all layers of caches in the DNS infrastructure, such as DNS forwarder and resolver caches, and a wide range of DNS software stacks, including the most popular BIND, Unbound, and dnsmasq, running on top of Linux, and potentially other operating systems. The major condition for a victim being vulnerable is that an OS and its network is configured to allow ICMP error replies. From our measurement, we find over 34% of the open resolver population on the Internet **are** vulnerable (and in particular 85% of the popular DNS services including Google's 8.8.8.8). Furthermore, we comprehensively validate the proposed attack with positive results against a variety of server configurations and network conditions that can affect the success of the attack, in both controlled experiments and a production DNS resolver.

Their attack disclosure page provides a link to test one's own DNS servers. So while preparing this report I clicked the link and received...

Your DNS server IP is **172.68.191.21**
It seems your DNS server is running **Linux > 3.18**
Since it is running the vulnerable version of OS that has not been patched yet, your DNS server is **vulnerable**. The test currently only takes the side channel port scanning vulnerability into consideration. A successful attack may also require other features in the server (e.g., supporting cache).
The test is conducted on 2020-11-16 21:52:21.854949379 UTC
Disclaimer: This test is not 100% accurate and is for test purposes only.

Okay, so what does this mean?

Separate from their report, their attack's description web page lays it out:

"SAD DNS" is a revival of the classic DNS cache poisoning attack (which no longer works since 2008) leveraging novel network side channels that exist in **all** modern operating systems, including Linux, Windows, macOS, and FreeBSD. This represents an important milestone — the first weaponizable network side channel attack that has serious security impacts. The attack allows an off-path attacker to inject a malicious DNS record into a DNS cache (e.g., in BIND, Unbound, dnsmasq).

The **SAD DNS** attack allows an attacker to redirect any traffic (originally destined to a specific domain) to his own server and then become a man-in-the-middle (MITM) attacker, allowing eavesdropping and tampering of the communication.

A bit deeper into their paper they layout in some greater detail what they have done...

Historically, the very first widely publicized DNS cache poisoning attack was discovered by Kaminsky in 2008, who demonstrated that an off-path attacker can inject spoofed DNS responses and have them cached by DNS resolvers. This has led to a number of DNS defenses being deployed widely, including source port randomization and other defenses such as DNSSEC. Unfortunately, due to reasons such as incentives and compatibility, these defenses are still far from being widely deployed. To summarize, source port randomization becomes the most important hurdle to overcome in launching a successful DNS cache poisoning attack.

Indeed, in the past, there have been prior attacks that attempt to derandomize the source port of DNS requests. As of now, they are only considered nice conceptual attacks but not very practical. Specifically, one requires an attacker to bombard the source port and overload the socket receive buffer, which is not only slow and impractical (unlikely to succeed in time) but also can be achieved only in a local environment with stringent RTT requirements. It is assumed that a resolver sits behind a NAT which allows its external source port to be derandomized, but such a scenario is not applicable to resolvers that own public IPs.

In contrast, the vulnerabilities we find are both much more serious and generally applicable to a wide range of scenarios and conditions. Specifically, we're able to launch attacks against all layers of caches which are prevalent in the modern DNS infrastructure, including application-layer DNS caches (e.g., in browsers), OS-wide caches, DNS forwarder caches (e.g., in home routers), and the most widely targeted DNS resolver caches. The vulnerabilities also affect virtually all popular DNS software stacks, including BIND, Unbound, and dnsmasq, running on top of Linux and potentially other OSes, with the major requirement being the victim OS is allowed to generate outgoing ICMP error messages. Interestingly, these vulnerabilities result from either design flaws in UDP standards or subtle implementation details that lead to side channels based on a global rate limit of ICMP error messages, allowing derandomization of source port with great certainty.

To demonstrate the impact, we devise attack methods targeting two main scenarios, including DNS forwarders running on home routers, and DNS resolvers running BIND and Unbound. With permission, we also tested the attack against a production DNS resolver that serves 70 million user queries per day, overcoming several practical challenges such as noise, having to wait for cache timeouts, multiple backend server IPs behind the resolver frontend, and multiple authoritative name servers. In our stress test experiment we also evaluate the attack in even more challenging network conditions and report positive results. In this paper, we make the following contributions:

- We systematically analyze the interaction between application and OS-level behaviors, leading to the discovery of general UDP source port derandomization strategies, the key one being a side channel vulnerability introduced by a global rate limit of outgoing ICMP error messages.

- We research the applicability of the source port derandomization strategies against a variety of attack models. In addition, to allow sufficient time to conduct the derandomization attack, we develop novel methods to extend the attack window significantly, one of them again leveraging the rate limiting feature (this time in the application layer).

- We conduct extensive evaluation against a wide variety of server software, configuration, and network conditions and report positive results. We show that in most settings, an attacker needs only minutes to succeed in an end-to-end poisoning attack. We also discuss the most effective and simple mitigations.

The crux of what they have done, is to arrange to eliminate the entropy-creating query source port randomization. If the DNS query source port can be determined, then the effectiveness of DNS spoofing and cache poisoning attacks return to their quite tractable pre-Kaminsky severity.

The reason ICMP comes into the picture is that an incoming UDP packet hitting a closed port will evoke an ICMP "port unreachable" error response, whereas the same packet hitting an open port will not. Thus it's possible to probe a DNS server's ports by monitoring ICMP replies.

But it's not that easy, since all modern operating system IP stacks have evolved to deliberately rate-limit ICMP error replies. Remember those good old days where it was possible to ping a server ("ping" — formally known as an echo request) with a spoofed IP and it would dutifully reply with an "echo reply" to the apparent incoming source IP. Those days are long gone. These days operating systems limit on both a per-IP basis and also on a global rate limited basis.

So, get a load of how clever these guys have been...

Even though a source port can be directly probed by any attacker IP in this case, e.g., as in unbound and dnsmasq, it is imperative to bypass the per IP rate limit (present in Linux primarily) to achieve faster scan speed. We develop three different probing methods that can overcome the ICMP rate limit challenge.

1. If the attacker owns multiple IP addresses, either multiple bot machines or a single machine with an IPv6 address, then it is trivial to bypass the per IP limit. IPv6 address allocation states that eachLAN is given a /64 prefix, effectively allowing any network to use 2^64 public IP addresses. We have tested this from a machine in a residential network that supports IPv6 and picked several IPs within the /64 to send and receive traffic successfully.

2. If an attacker owns only a single IPv4 address, it is still possible to ask for multiple addresses using DHCP. We verified that multiple private IPv4 addresses can be obtained in a home network. In addition, we have tested this in an educational network where a single physical machine is able to acquire multiple public IPv4 addresses through this method as well.

3. If an attacker owns a single IPv4 address and the above method fails for some reason (e.g., statically assigned IPs), then the last method is to leverage IP spoofing to bypass the per IP rate limit, and the global rate limit **as a side channel to infer whether the spoofed probes have hit the correct source port or not**, i.e., with or without ICMP responses.

As has been shown in the context of TCP recently, global rate limiting can introduce serious side channels. Here we leverage the ICMP global rate limit to facilitate UDP port scans which we describe next.

In observing the maximum globally allowable burst of 50 ICMP packets in Linux, the attacker first sends 50 spoofed UDP probe packets each with a different source IP (thus bypassing the per-IP rate limit). If the victim server does not have any source port open among the 50, then 50 ICMP port unreachable messages will be triggered (but they are not directly observable to the attacker since each of the 50 will be returned to one of the spoofed source IPs). If the victim server does have open ports, then only 5 ICMP packets will be triggered (as the UDP probing packets will be silently discarded at the application layer). Now, the attacker sends a verification packet using its **real IP address**,e.g., a UDP packet destined to a known closed port, such as 1. It will either get no response (if the global rate limit has been hit and is drained), or an ICMP reply otherwise. Thus, even without receiving replies, the presence of an open port within a range of 50 can be directly determined by inference. If no port is found in the first batch, the attacker waits for at least 50ms for the rate limit counter to recuperate and reset, and then starts the next round.

Effectively, the scanning speed will be capped at 1,000 per second. It therefore takes a little more than 60 seconds to enumerate the entire port range of 65,536 ports. Nevertheless, it is a winning battle as the attacker can simply repeat the experiment and the probability that one experiment will succeed increases dramatically.

So, what these guys have done is to figure out a way to use modern ICMP rate-limiting as a side-channel to probe a DNS server for an open port. Thus moving the DNS spoofability clock back 12 years to a point where DNS cache poisoning sent the industry into a frenzy.

As I mentioned above, these days things are less panic-prone because the entire industry has switched over to HTTPS and a certificate-based system to add authentication to privacy. But as I also noted, automated certificate services already exist and more are coming online. So the practical vulnerability, while not hair on fire, is still there. We haven't yet secured our domain name system and it doesn't appear that that's going to happen anytime soon.

So this is less urgent than it is supremely clever, so I wanted to share it with everyone for its cleverness.

In their Q&A...

Q: Am I affected by the vulnerability?
A: Likely, as long as you are using a vulnerable DNS service (e.g., 8.8.8.8 or 1.1.1.1). Most public resolvers have been checked to be vulnerable. If you are using private DNS services (i.e., those provided by your ISP or your organization), we do not have sufficient data but there is a good chance that it is vulnerable as well.

Q: What applications are affected?
A: Any networking application using DNS to retrieve the IP address of peers/servers are affected. Besides, vulnerable/affected DNS software includes but is not limited to BIND, Unbound and dnsmasq.

Q: How widespread is this?
A: According to our measurements, 35% of open resolvers are vulnerable to the attack. We also found 12/14 public resolvers and 4/6 routers made by well-known brands are vulnerable. In theory, any DNS server running the newer version of popular operating systems without blocking outgoing ICMPs (only Windows blocks it by default) is also vulnerable.

Q: Has SAD DNS been patched?
A: Yes, we have worked with the Linux kernel security team and developed a patch that randomizes the ICMP global rate limit to introduce noises to the side channel. Please refer to Security Advisories for more recent updates.

Q: Which version of operating systems are affected?

- Linux 3.18-5.10
- Windows Server 2019 (version 1809) and newer (we did not test older versions)
- macOS 10.15 and newer (we did not test older versions)
- FreeBSD 12.1.0 and newer (we did not test older versions)

The patch for Linux is integrated into 5.10 and backported to many stable versions. However, we don't know how and when Windows/macOS/FreeBSD will patch this vulnerability.

Q: Is this a man in the middle attack?
A: No, this is a totally off-path attack. The attacker is not required to sniff the traffic between the DNS servers, but a key requirement is that the attacker has the IP spoofing capability to impersonate the legitimate server. IP Spoofing is still feasible today.

Q: Does DNS over HTTPS (DoH) mitigate the attack?
A: No, currently DoH only encrypts the traffic between the DNS client and DNS resolvers. However, SAD DNS attacks the link between resolvers and authoritative name servers, which is not protected by DoH.

https://dl.acm.org/doi/pdf/10.1145/3372297.3417280