Security Now! #792 - 11-10-20

“Slipstream” NAT Firewall Bypass

This week on Security Now!

This week we look at the dilemma of Let's Encrypt's coming root expiration, new Chrome and Apple 0-day vulnerabilities, some new high-profile ransomware victims, China's Tianfu Cup pwning competition, the retirement of a PC industry insider, the continuing Great Encryption Dilemma, police monitoring of consumer's video, more ongoing pain for WordPress, a note about a Sci-Fi book event one week from now, and Samy Kamkar's tricky “Slipstream” attack and its mitigations.

These just never get old

(And, boy do they beg a few questions!)

Note that in the center is a padlock... because, you know, you wouldn’t want anyone to open the gate.

Browser News

Let's Encrypt's cross-signed root expires next year

Let's Encrypt's blog posting Friday is titled: "Standing on Our Own Two Feet"
https://letsencrypt.org/2020/11/06/own-two-feet.html

The issue that has come up is so interesting and fundamental that I want to share what Let's Encrypt has said — with ample in-line editorializing thrown in...

When a new Certificate Authority (CA) comes on the scene [which, I'll note, does not happen very often], it faces a conundrum: In order to be useful to people, it needs its root certificate to be trusted by a wide variety of operating systems (OSes) and browsers. However, it can take years for the OSes and browsers to accept the new root certificate, and even longer for people to upgrade their devices to the newer versions that include that change. The common solution: a new CA will often ask an existing, trusted CA for a cross-signature, to quickly get it into being trusted by lots of devices.

Five years ago, when Let's Encrypt launched, that's exactly what we did. We got a cross-signature from IdenTrust. Their "DST Root X3" had been around for a long time, and all the major software platforms trusted it already: Windows, Firefox, macOS, Android, iOS, and a variety of Linux distributions. That cross-signature allowed us to start issuing certificates right away, and have them be useful to a lot of people. Without IdenTrust, Let's Encrypt may have never happened and we are grateful to them for their partnership. Meanwhile, we issued our own root certificate ("ISRG Root X1") and applied for it to be trusted by the major software platforms.

Now, those software platforms have trusted our root certificate for years. And the DST Root X3 root certificate that we relied on to get us off the ground is going to expire - on September 1, 2021. Fortunately, we're ready to stand on our own, and rely solely on our own root certificate.

However, this does introduce some compatibility woes. Some software that hasn't been updated since 2016 (approximately when our root was accepted to many root programs) still doesn't trust our root certificate, ISRG Root X1. Most notably, this includes versions of Android prior to 7.1.1. That means those older versions of Android will no longer trust certificates issued by Let's Encrypt.

Android has a long-standing and well known issue with operating system updates. There are lots of Android devices in the world running out-of-date operating systems. The causes are complex and hard to fix: for each phone, the core Android operating system is commonly modified by both the manufacturer and a mobile carrier before an end-user receives it. When there's an update to Android, both the manufacturer and the mobile carrier have to incorporate those changes into their customized version before sending it out. Often manufacturers decide that's not worth the effort. The result is bad for the people who buy these devices: many are stuck on operating systems that are years out of date.

Google no longer provides version numbers on its Distribution Dashboard, but you can still get some data by downloading Android Studio. Here's what the numbers looked like as of September 2020:

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.8%
4.2 Jelly Bean	17	99.2%
4.3 Jelly Bean	18	98.4%
4.4 KitKat	19	98.1%
5.0 Lollipop	21	94.1%
5.1 Lollipop	22	92.3%
6.0 Marshmallow	23	84.9%
7.0 Nougat	24	73.7%
7.1 Nougat	25	66.2%
8.0 Oreo	26	60.8%
8.1 Oreo	27	53.5%
9.0 Pie	28	39.5%
10. Android 10	29	8.2%

Currently, 66.2% of Android devices are running version 7.1 or above. The remaining 33.8% of Android devices will eventually start getting certificate errors when users visit sites that have a Let's Encrypt certificate. In our communications with large integrators, we have found that this represents around 1-5% of traffic to their sites. Hopefully these numbers will be lower by the time DST Root X3 expires next year, but the change may not be very significant.

What can we do about this? Well, while we'd love to improve the Android update situation, there's not much we can do there. We also can't afford to buy the world a new phone. Can we get another cross-signature? We've explored this option and it seems unlikely. It's a big risk for a CA to cross-sign another CA's certificate, since they become responsible for everything that CA does. That also means the recipient of the cross-signature has to follow all the procedures laid out by the cross-signing CA. It's important for us to be able to stand on our own. Also, the Android update problem doesn't seem to be going away. If we commit ourselves to supporting old Android versions, we would commit ourselves to seeking cross-signatures from other CAs indefinitely.

It's quite a bind. We're committed to everybody on the planet having secure and privacy-respecting communications. And we know that the people most affected by the Android update problem are those we most want to help - people who may not be able to buy a new phone every four years. Unfortunately, we don't expect the Android usage numbers to change much prior to ISRG Root X1's expiration. By raising awareness of this change now, we hope to help our community to find the best path forward.

As of January 11, 2021, we're planning to make a change to our API so that ACME clients will, by default, serve a certificate chain that leads to ISRG Root X1. However, it will also be possible to serve an alternate certificate chain for the same certificate that leads to DST Root X3 and offers broader compatibility. This is implemented via the ACME "alternate" link relation. This is supported by Certbot from version 1.6.0 onwards. If you use a different ACME client, please check your client's documentation to see if the "alternate" link relation is supported.

There will be site owners who receive complaints from users and we are empathetic to that being not ideal. [Yeah. 1/3 of all Android devices will be unable to connect — at all — after next

August. Does anyone think that might be a problem?] We're working hard to alert site owners so you can plan and prepare. We encourage site owners to deploy a temporary fix (switching to the alternate certificate chain) to keep your site working while you evaluate what you need for a long-term solution: whether you need to run a banner asking your Android users on older OSes to install Firefox, stop supporting older Android versions, drop back to HTTP for older Android versions, or switch to a CA that is installed on those older versions.

If You Get Let's Encrypt Certificates Through Your Hosting Provider, your hosting provider may be serving the DST Root X3 until September 2021, or they may decide to switch to the certificate chain that leads to ISRG Root X1 after January 11, 2021. Please contact them if you have any questions!

If you're on an older version of Android, we recommend you install Firefox Mobile, which supports Android 5.0 and above as of the time of writing.

Why does installing Firefox help? [And/or why did Google recently decide to have Chrome eventually switch-over to its own certificate root store?] For an Android phone's built-in browser, the list of trusted root certificates comes from the operating system - which is out of date on these older phones. However, Firefox is currently unique among browsers - it ships with its own list of trusted root certificates. So anyone who installs the latest Firefox version gets the benefit of an up-to-date list of trusted certificate authorities, even if their operating system is out of date.

We appreciate your understanding and support both now and over the years as we continue to grow as a CA, making sure people everywhere have access to encryption.

Chrome updates to remove another 0-day vulnerability

While we were recording last week's podcast, unbeknownst to us, Googe was busy releasing another emergency update to Chrome for Windows, MacOS and Linux, bringing it up to version 86.0.4240.183. We don't know much yet about the 0-day flaw that was found being actively used in the wild, except that CVE-2020-16009 is a v8 (so a bug in Chrome's V8 JavaScript engine) which is being used to enable remote code execution.

When I went to check on my Chrome, that version was already obsolete for me since my Chrome took the opportunity of my taking a peek to jump itself forward to 86.0.4240.193. There's no information, yet, about that one.

A Chrome update for Android

Before we move one, I'll also note that Android smartphone users should also be sure to poke their Chromes to verify that they're now running 86.0.4240.185 or later to close another Android-specific 0-day that Google has found being exploited in the wild. This was a heap buffer overflow vulnerability in the Chrome for Android UI component which had been given the CVE number: 2020-16010. It was being exploited to allow attackers to bypass and escape from Chrome's security sandbox on Android devices to then run code on the underlying OS.

So at this point, Google's internal Threat Analysis Group (the TAG team) has discovered not only

this 0-day, but also the two previous 0-days, bringing their total to three 0-days in just the past two weeks. Thus, it appears that Chrome is finally receiving the attention from the attacker community that was once the province of IE. Let's all hope Chrome fares better.

Ransomware News

Mattel

To the rapidly growing list of embarrassed, inconvenienced and annoyed victims of ransomware attacks, we add the famous toy manufacturer, Mattel, which, last Wednesday revealed in their 10-Q quarterly filing that they had been hit by a ransomware attack back at the end of July. No data was stolen and they were able to restore the impacted services quickly.

Compal

Also, the world's second largest laptop manufacturer, Compal, located in Taiwan, who builds laptops for Apple, Acer, Lenovo, Dell, Toshiba, HP, and Fujitsu, suffered a ransomware attack this past weekend. The attackers are believed to by the DoppelPaymer ransomware gang. The incident was discovered Sunday morning and is believed to have impacted approximately 30% of Compal's computer fleet. Since Sunday, Compal's IT staff has been reinstalling encrypted workstations and was expected to be back to 100% by yesterday (Monday). And, in any case, a spokesman from the company said that the attack had only affected the company's internal office network and that Compal's production lines, which build laptops for other companies, were never impacted.

Capcom

Also, last week the well-known Japanese game developer, Capcom, was hit by the "Ragnar Locker" ransomware in an attack that also exfiltrated around 1 terabyte of the company's data. The data was reportedly from Capcom's corporate networks in the US, Japan, and Canada. The gamers among us will know Capcom from their popular game franchises which include Street Fighter, Resident Evil, Devil May Cry, Monster Hunter, and Mega Man.

Campari

I got a kick out of ThreatPost's headline for this one: "Campari Site Suffers Ransomware Hangover." Given that Compari is a famous producer of alcoholic beverages including the brands SKYY, Grand Marnier and Wild Turkey, "a hangover" seemed appropriate. (Especially following Grand Marnier. OMG, that stuff is evil!) Anyway... while they may have ended up feeling hungover, I doubt that they had any fun getting into that condition. They've restored their servers after being hit by the Ragnar Locker ransomware and having received the attackers' demand of $15 million in Bitcoin.

The attackers left behind the notice: "We have BREACHED your security perimeter and accessed every server of the company's network in different countries across all your international offices," the note goes on to detail the types of data compromised, including accounting files, bank statements, employee personal information and more. The note said that they had obtained a total of 2 terabytes of data and said: "If no deal is made then all your data will be published and/or sold through an auction to any third parties." And as proof of theft, the group posted a copy of the contract between Wild Turkey and Matthew McConaughey.

So there's four more recent attacks.
The gang behind the Ryuk ransomware alone reportedly averages 20 such attacks per week. I managed to pick up a bit of intelligence about the Ryuk gang's recent successes. The average payment received by this Russian-speaking gang is 48 bitcoins, ($720,000) and since 2018 they have netted at least $150 million. They are exceedingly tough during negotiations, rarely show leniency or compassion, and the largest confirmed payment they are known to have received was 2,200 bitcoins, currently valued at $33 million. (Bitcoin is on the rise again, by the way. It's back up to $15K per.)

Since I like to sleep at night, I adhere to the old adage that "Crime doesn't pay." And I'm sure that most of us are not tempted by the lure of the dark side. But neither is it difficult to imagine why the ransomware business is booming as it clearly is.

Security News

Apple moved iOS to 14.2
If you're an iOS user, as I am, you may have noticed that your iOS devices were recently bumped from 14.1 to 14.2. And there's a story behind that. The very short version is that this was done to close three 0-day vulnerabilities that were discovered in-use in attacks against iOS users.

The much longer and more interesting version is that, according to Shane Huntley, the Director of Google's Threat Analysis Group (that same TAG team), the three iOS 0-days are "related" (unquote) to those recent three Chrome 0-days and the Windows 0-day we covered last week. So it was a big, well-coordinated multi-platform campaign. Since Google and Apple have clamped down on any details, we don't know whether the 0-days were being used against selected targets or sprayed, but all iOS users should update to iOS 14.2, regardless.

These same three vulnerabilities have also been closed in the most recent iPadOS and watchOS updates, and they have also been backported to older generation iPhones as iOS 12.4.9.

What little we know from Ben Hawkes' Project Zero team, is that the three iOS zero-days are:

1. CVE-2020-27930 — a remote code execution issue in the iOS FontParser component that lets attackers run code remotely on iOS devices.

2. CVE-2020-27932 — a privilege escalation vulnerability in the iOS kernel that lets attackers run malicious code with kernel-level privileges.

3. CVE-2020-27950 — a memory leak in the iOS kernel that allows attackers to retrieve content from an iOS device's kernel memory.

All three bugs are believed to have been used together to form a highly sophisticated exploit chain, allowing attackers to compromise iPhone devices remotely. And think for a moment about how difficult that is to pull off on any locked-down iOS device. With Windows, Linux or Android it's so much easier. All the code is just there to be poked and prodded. But not on iOS. As we know, Apple has their devices so locked down, encrypted and hack-proof that it takes some

serious effort just to get a peek under the covers, let alone perform the sort of deep reverse engineering that's required first to find a problem, then turn it into anything like a reliable exploit.

Almost all vulnerabilities are far easier to find and would be vastly easier to exploit once found. But this feels like a world-class piece of work. For that reason, I would bet that this was not being widely sprayed around the Internet. While it was there and secret, it would have been incredibly valuable for enabling highly targeted information compromise. It may well have been a nation state.

Introducing the "Tianfu Cup" (tea-AN-foo)
The Tianfu Cup is China's version of the West's Pwn2Own hacker competition. It was created two years ago, 2018 following the Chinese government's regulation which barred their security researchers from participating in international hacking competitions over national security concerns. Our listeners may recall that we talked about the absence of the Chinese hackers — who were among the most talented and often successful.

So, China's third ever, Tianfu Cup 2-day event was just held over the weekend. Contestants from 15 different teams participated to deploy and demonstrate their discoveries of original vulnerabilities to break into widely used software and mobile devices. Each team was allotted 5 minutes and three attempts.

The requirement was to use various web browsers to navigate to a remote URL or use a software flaw to obtain control of the browser or the underlying operating system. The targets was software from Adobe, Apple, Google, Microsoft, Mozilla, and Samsung, all which were successfully pwned utilizing previously unknown exploits. The event's organizers said that "Many mature and hard targets have been pwned during this year's contest. 11 out of 16 targets were cracked with 23 successful demos."

The hacking competition showed off hacking attempts against a number of platforms, including:

1. Adobe PDF Reader
2. Apple iPhone 11 Pro running iOS 14 and Safari browser
3. ASUS RT-AX86U router
4. CentOS 8
5. Docker Community Edition
6. Google Chrome
7. Microsoft Windows 10 v2004
8. Mozilla Firefox
9. Samsung Galaxy S20 running Android 10
10. TP-Link TL-WDR7660 router
11. VMware ESXi hypervisor

Qihoo 360's Enterprise Security and Government (ESG) Vulnerability Research Institute [there's a mouthful] came out top winning $744,500. They were followed by Ant-Financial Light-Year Security Lab ($258,000) and a security researcher named Pang ($99,500).

Patches for all the demonstrated bugs demonstrated are expected to be released in the coming days.

November's Patch Tuesday

Today, and in days following, many Windows machines will have the somewhat mixed blessing of receiving Microsoft's latest monthly insult of updates and new breakages. Sadly, as we've been seeing all year, updating promptly appears to be of increasing urgency every month as the stakes in this game have risen. So, I expect that by our next podcast I'll have a readout on some outcomes of this month's Windows 10 continuing update adventure.

Something I recently encountered was apropos of this: Woody Leonard, a longtime valued participant and commentator in the PC industry, announced his retirement from our industry two days ago, last Sunday. For those who are unfamiliar with Woody, two short bits from the Internet say: "Woody Leonhard is a columnist at Computerworld and author of dozens of Windows books, including "Windows 10 All-in-One for Dummies." and elsewhere: "Woody Leonhard has covered Windows Dummies foibles and fantasies since the days of Windows XP. With more than a million regular readers, he's Senior Contributing Editor at InfoWorld, and Senior Editor at Windows Secrets, where he weighs in daily on all things Windows."

Given his deep background, I thought that his parting characterization was interesting, if only to further assure us that we're not all crazy. When announcing this retirement, Woody wrote:

> Life's changed in extraordinary ways since my first "meatspace" book "Windows 3.1 Programming for Mere Mortals" appeared 28 years ago. Windows has evolved from a rickety infrastructure built on top of a wobbly operating system to a wobbly operating system in its own right.
>
> I don't miss the original bug-ridden incarnations of Windows. But I do miss the fire and vision that drove the unqualified success of Windows XP and Win7. And I'll continue to rail against the flaws that are introduced — and sometimes re-introduced — with every round of updates.
>
> Microsoft has a long history of Windows patching issues. Some things never change, eh?

So, a tip of the hat to Woody. Enjoy your retirement. The rest of us are still having too much fun to quit!

The Great Encryption Dilemma

The Council of the European Union last week published a short 5-page draft resolution with the title: "Draft Council Resolution on Encryption - Security through encryption and security despite encryption." I read through the resolution and there's nothing new there. But since "The Great Encryption Dilemma" remains outstanding and unresolved, I wanted to just stick a pin in this, to note that the issue remains alive and well. And also that I suspect it always will. I want to take this occasion to be more clear and definitive about this than I've been previously:

Cryptographers — and this podcast's audience — know with absolute clarity that this is a problem without a solution. Such things exist, and this is one of them. Bureaucrats, who are not

cryptographers are unable to accept the simple math of this fact: Once unbreakable encryption was created, that was the end of it. Game over. Enciphering algorithms without known weaknesses now exist. They cannot ever be made not to exist. Once something is encrypted with them, the ONLY KNOWN WAY to reverse the encryption is with the key. Period. Sure, a deliberately weakened encryption system could easily be created. We know how to do that. But that doesn't mean that the existing systems, the ones without any known weaknesses, can be uncreated. They **cannot** be uncreated. Governments and law enforcement may not be happy about what's been created. But it has been. Already. The cat's out of the bag. The horses have left the stable. The chickens have flown coop. The train has left the station. The ship has sailed.

So, hopefully, this will forever remain a stalled issue, with academia and industry patiently explaining over and over as many times as necessary to any government or law enforcement agency who asks, whenever it comes up, as it certainly will, that there's no safe way to add a deliberate backdoor into existing encryption. And even if there were, that would not spontaneously uncreate any of the existing uncrackable encryption technologies that anyone could easily and freely use.

Ring Doorbells to be tapped in a trial by local Police

So this one is maybe a little creepy. In Jackson, Mississippi, a small trial has been initiated and is being conducted for 45 days to explore the feasibility of allowing private citizens to have their video doorbells participate in police dragnet monitoring.

The rationale is that while on the one hand, municipalities might install video cameras pointing down all four directions of every intersection — which is, by the way, exactly what I've noticed being done here in Southern California — that doesn't provide as much granular video coverage as might also be afforded if all of the video doorbells in residential neighborhoods were also tied into a much larger surveillance network.

We've talked about pervasive video monitoring before, and it's a little creepy. But the EFF feels somewhat more strongly. The EFF noted that handing over control of live streams to law enforcement may not only allow the covert recording of a willing participant's comings-and-goings but neighbors, too.

The EFF wrote: “The footage from your front door includes you coming and going from your house, your neighbors taking out the trash, and the dog walkers and delivery people who do their jobs in your street. In Jackson, this footage can now be live-streamed directly onto a dozen monitors scrutinized by police around the clock. Even if you refuse to allow YOUR footage to be used that way, your neighbor's camera pointed at your house may still be transmitted directly to the police.”

Interestingly, just this past August, Jackson city officials voted to pre-emptively ban police forces from using facial recognition technology to identify potential suspects on city streets. Although this is not that, it's getting close. And the month before, in September, an analysis leaked from the FBI highlighted how smart doorbells could also be turned against law enforcement, as live feeds could warn suspected criminals of police presence, alert them to incoming visits from police, and might show suspects where officers are, which could pose a safety risk to law enforcement conducting property raids.

This is, increasingly, feeling like a Brave New World.

WordPress "Ultimate Member" — or ultimate dismember
If you haven't yet been convinced to sequester any WordPress instance so that its takeover cannot harm you further, here's another reasons to consider either sequestration or eviction:

Three, separate super-critical flaws exist within another highly popular WordPress add-on known as "Ultimate Member." They have CVSS severity ratings of 10, 10 and 9.9, each out of 10. And the Ultimate Member plug-in is installed on more than 100,000 WordPress sites. Each of the three critical security bugs allows for privilege elevation leading to full control over a WordPress site.

The plugin allows web admins to add user profiles and membership areas to their sites and, according to Wordfence researchers, the flaws make it possible for both authenticated and unauthenticated attackers to elevate their privileges during registration to obtain admin status. And, of course, once an attacker has admin access to a WordPress site, they have effectively taken over the entire site and can perform any action, from taking the site offline to further infecting the site with additional malware.

I'm not going to delve into detail about each of the three vulnerabilities because I think that a broader point needs to be made:

We've seen that the hacker community tends to focus on one category or another from time to time. For a while RDP is under attack. Then it's router botnets attacking HTTP authentication, and tomorrow it'll be something else. But the recent evidence suggests that WordPress plug-ins have been enjoying a period of relative quiet and under-examination by that nefarious community. But that the community has recently awakened to just how much low-hanging fruit has been growing while their attention has been directed elsewhere.

Last week, a security vulnerability in the Welcart e-Commerce plugin was found to be opening WordPress sites to code injection. This led to payment skimmers being installed, crashing of the site, or information retrieval via SQL injection.

Last month, two high-severity vulnerabilities were disclosed in Post Grid, another WordPress plugin with more than 60,000 installations. It opened the door to site takeovers. And in September, a high-severity flaw in the "Email Subscribers & Newsletters" plugin by Icegram was found to affect more than 100,000 WordPress site.

In August, a plugin that adds quizzes and surveys to WordPress patched two critical vulnerabilities which could be exploited by remote, unauthenticated attackers to launch a variety of attacks including full site takeover. And also in August, "Newsletter," a WordPress plugin with more than 300,000 installations, was discovered to have a pair of vulnerabilities leading to code-execution and site takeover.

And before that, in July, researchers warned of a critical vulnerability in a WordPress plugin called "Comments – wpDiscuz," which is installed on more than 70,000 websites. The flaw gave unauthenticated attackers the ability to upload arbitrary files (including PHP files) and ultimately

execute remote code on vulnerable website servers.

I said before that WordPress is demonstrably a PHP-coded disaster, and that the tantalizing WordPress plug-in ecosystem, which is, I'm sure, a large part of WordPress' allure, is a hot mess. It's impractical to tell people not to use it. I get that. But don't run a WordPress instance on your Drobo or on any machine that has access to anything else. Depending upon how many tasty-looking goodies you add to your WordPress installation over time, there's a high likelihood of local site compromise. That means that containment is the best you can hope for. Please consider it.

Sci-Fi

"The Saints of Salvation"

Exactly one week from today, on November 17th, the third tome of Peter Hamilton's "Salvation" trilogy will drop. I've been awaiting it with great impatience. Like all of Peter's work, it is a truly remarkable work of science fiction. In a world filled with lazy and largely derivative fiction, Peter somehow always arranges to create entire, fully realized believable worlds and characters. And with this Salvation trilogy he has done so again.

It is annoying to wait for long periods between installments of his multi-book series. And when the seconds book dropped I re-read the first book again to refresh my memory of what had happened so far. So for anyone who loves his work and who waits for all of the books of a story to be published, it is now safe to embark on the journey into "Salvation." So far it's been really fun.

"SlipStream" NAT Firewall Bypass

"***SlipStream***" is a perfect name for this new bit of cleverness. And let's all be thankful that Samy KamKar, the security researcher who invented this, had the freedom to name his invention whatever he wished, rather than needing to dip into @Vulnonym to receive a name. I'll confess to being morbidly curious yesterday, so I went over and took a look at CERT's feed at that moment. I was greeted with "Putative Loon", "Pungent Pronghorn", "Discordant Screamer" and "Feckless Mongrel." It's not clear to me that being told I've been a victim of the "Feckless Mongrel" demonstrates the achievement of CERT's intentions with regard to naming.

Okay, so what did Samy come up with? **https://samy.pl/slipstream/**

His NAT Slipstreaming page states: "NAT Slipstreaming allows an attacker to remotely access any TCP/UDP service bound to a victim machine, bypassing the victim's NAT/firewall (arbitrary firewall pinhole control), just by the victim visiting a website."

So what does that mean? We've come to treat our NAT routers as smart firewalls which block all unsolicited incoming traffic by default. They do this for us automatically by operating a system of stateful packet inspection where any incoming packets must be replies to recent outgoing

packets. This super-elegant and simple scheme has served us very well so far.

I'll quote from Samy's Summary, even though his terminology is opaque and confusing. But don't worry, it will all become very clear. Samy explained:

> NAT Slipstreaming exploits the user's **browser** [don't forget that part] in conjunction with the Application Level Gateway (ALG) connection tracking mechanism built into NATs, routers, and firewalls, by chaining internal IP extraction via timing attack or WebRTC, automated remote MTU and IP fragmentation discovery, TCP packet size massaging, TURN authentication misuse, precise packet boundary control, and protocol confusion through browser abuse. As it's the NAT or firewall that opens the destination port, this bypasses any browser-based port restrictions.
>
> This attack takes advantage of arbitrary control of the data portion of some TCP and UDP packets without including HTTP or other headers; the attack performs this new packet injection technique across all major modern (and older) browsers, and is a modernized version to my original NAT Pinning technique from 2010 (presented at DEFCON 18 + Black Hat 2010). Additionally, new techniques for local IP address discovery are included.
>
> This attack requires the NAT/firewall to support ALG (Application Level Gateways), which are mandatory for protocols that can use multiple ports (control channel + data channel) such as SIP and H323 (VoIP protocols), FTP, IRC DCC, etc.

Okay. So let's first talk about Application Layer Gateways.

They exist because not all Internet protocols are as simple as HTTP, where we make an outbound query over a connection and receive a returning reply over the same connection. For example, the old timers among us may recall that the FTP — File Transfer Protocol — has both so-called "active" and "passive" modes, with "active" being the preferred original default. With the original FTP protocol — which was designed pre-firewall — the user's FTP client would reach out to a new FTP server to initiate a connection at that server's port 21. They would exchange logon username and password over this so-called "control" channel. And upon agreeing on the transfer of data, that transfer would occur NOT over the current connection to the server's port 21, but from its port 20 — the so-called "data" channel. And, most significantly, the FTP server would be the initiator of that data channel connection.

This meant that an FTP client would need to open a high-numbered listening port at its end. Remember that as a client of an operating system, applications are restricted from opening low-numbered ports below 1024. Those are reserved for privileged OS processes. So, over the original connection to the remote server's FTP port 21, the FTP client would say "My OS has given me port xxxx, where I'm now listening for your incoming return data channel connection. Please open a TCP connection to me at that port and I'll answer."

So, in summary, the client sets up a high-numbered listening port, then initiates an outbound connection to a remote FTP server's port 21 and, among other things, requests the remote server to call back to it at that high-numbered port.

The bad news is that the operation of NAT is totally hostile to this wacky old active FTP protocol. This was recognized early on by the implementers of NAT and it was "solved" or at least resolved with a horrific kludge: NAT routers would notice when an outbound connection was being made to a server at port 21. They would then begin sniffing the outbound traffic over the FTP control channel looking for the client's command to open the reverse data channel. And the router would then on-the-fly patch in a WAN port for the remote server to query and establish a NAT mapping from that WAN port back to the client's listening port.

If you think about that for a moment you'll notice that this effectively, deliberately and necessarily penetrated the NAT firewall for this connection instance. Since those early pre-firewall days, a number of other protocols have been designed which are similarly NAT-hostile. And the need for a means of opening incoming ports has been generalized under the umbrella term "Application Layer Gateway" since the control side of this resides not at the packet level but, as with our FTP example, within the application's data layer.

Wikipedia explains it this way:

> Application-level gateway (also known as ALG, application layer gateway, application gateway, application proxy, or application-level proxy) is a security component that augments a firewall or NAT, employed in a computer network. It allows customized NAT traversal filters to be plugged into the gateway to support address and port translation for certain application layer "control/data" protocols such as FTP, BitTorrent, SIP, RTSP, file transfer in instant messaging applications, etc. In order for these protocols to work through NAT or a firewall, either the application has to know about an address/port number combination that allows incoming packets, or the NAT has to monitor the control traffic and open up port mappings (firewall pinhole) dynamically as required. Legitimate application data can thus be passed through the security checks of the firewall or NAT that would have otherwise restricted the traffic for not meeting its limited filter criteria.

So, most clearly stated:

The problem is that Application Layer Gateways attempt to be completely transparent to the application protocols they're proxying for. They're sitting there in our routers, enabled by default, hidden, powerful, and automatic. And, because they are automatic and trusting, they can be readily spoofed by any inside agent pretending to need their help. "Spoofing" in this context means tricking the user's border NAT router into opening a packet return path through to any internal IP and port of the attacker's choice. This subjects any services existing anywhere on a LAN to remote external access and abuse.

The second part of Samy's work was to clearly demonstrate that JavaScript code running in any of today's web browsers — even in a malicious malvertisement on any unwitting site — is all that's necessary to launch just such a spoofing attack. In other words, our web browsers themselves can serve as unwitting Application Layer Gateway spoofing agents.

During his research, Samy explored all of the various ALG's, looked at Linux's Netfilter ALG support, he reverse engineered router firmware and finally settled upon the abuse of the SIP — Session Initiation Protocol.

Samy Wrote:

> "While we've found some FTP functions, we're more interested in ports that we can use. Modern browsers prevent outbound HTTP(S) connections to a number of restricted ports, including FTP, so abusing the FTP ALG is likely a no-go.
>
> In 2010, when I first demonstrated NAT Pinning, I used port 6667 (IRC) via the DCC CHAT/FILE messages. Quickly, browser vendors blocked port 6667."

And the same thing has happened this time, in quick reaction to Samy's revelations, all web browser vendors are planning to block the TCP SIP ports 5060 and 5061, used in Samy's demonstrated attacks by adding them to the browsers' existing restricted lists.

Chromium developer Adam Rice said: "As a workaround for the 'Slipstream' NAT bypass attack, we will be blocking HTTP and HTTPS connections to the SIP ports 5060 and 5061. This will mean that connections to servers on those ports will fail."

Once those ports have been added to the restricted ports list, Rice expects some impact to be observed by browser users. For example, connections to servers on those ports, for example, http://example.com:5060/ or https://example.com:5061/ will no longer work. probably not a big problem. But any tests that might spin up a server on an arbitrary high-numbered port will need to avoid any of the browsers' slowly growing list of restricted server ports.

The development teams behind Firefox, Safari, and Blink (the Chromium rendering engine) have all expressed their intent to implement the mitigations needed to block these NAT Slipstreaming attacks.