https://www.rapid7.com/research/report/nicer-2020/

141 min read

National / Industry / Cloud Exposure Report (NICER) 2020

To provide even more actionable data, the findings are broken down by country, industry sector, and protocol, with special sections on the effect of the pandemic and technological shifts like the movement to the cloud. This helps diagnose what is vulnerable, what is improving or getting worse, and what solutions are available for policymakers, business leaders, and innovators to make the internet more secure.

By the middle of 2020, the entire world has been coping with a virus outbreak of the sort we don't usually cover in computer science fields: the biological pandemic of COVID-19[1] and the nearly immediate economic recession following the resultant lockdown. At Rapid7, we were already planning on producing another survey of the internet and the state of security worldwide, but we now have a unique opportunity to capture this unprecedented period of tumult as it reshapes our world in sudden, chaotic ways.

The first question we tasked ourselves with answering was, "How did the pandemic, lockdown, and job loss affect the character and composition of the internet?" We expected to see a renaissance of poorly configured, hastily deployed, and wholly insecure services dropped on the public internet, as people scrambled to “just make things work” once they were locked out of their offices, coworking spaces, and schools, suddenly shifted to study-at-home models of work and study. Fears of thousands of new Windows SMB services for file sharing between work and home, rsync servers collecting backup data across the internet, and unconfigured IoT devices offering Telnet-based consoles haunted us as we started to collect the April and May data for this project.

But, the year 2020 is nothing if not full of surprises—even pleasant ones! Indeed, we found that the populations of grossly insecure services such as SMB, Telnet, and rsync, along with the core email protocols, **actually decreased** from the levels seen in 2019, while more secure

alternatives to insecure protocols, like SSH (Secure Shell) and DoT (DNS-over-TLS) increased overall. So, while there are regional differences and certainly areas with troubling levels of exposure—which we explore in depth in this paper—the internet as a whole seems to be moving in the right direction when it comes to secure versus insecure services.

This is a frankly shocking finding. The global disasters of disease and recession, along with the uncertainty they bring, appear to have had no obvious effect on the fundamental nature of the internet. It is possible that this is because we have yet to see the full impact of the pandemic, recession, and greater adoption of remote working. Rapid7 will continue to monitor and report as things develop.

Meanwhile, the cloud is home to more “internet stuff” than ever before. Platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS) providers enable even small organizations to launch professionally managed and maintained server infrastructure to run every sort of internet-based venture you can think of. Given the good news of a decrease of insecure-by-design services on the internet, we expected to find that cloud providers were responsible for this decrease. Perhaps naively, we imagined that the global leaders in cloud provisioning were constructing glittering silver cities in the sky, resplendent with the perfect architecture of containerized software deployments, all secure by design.

This was not the case. It turns out that cloud providers are often dogged by the same problems that traditional, on-premises IT shops struggle with; the easy path is not necessarily the secure path to getting internet services configured and up and running, even in those providers’ own documentation and defaults. Furthermore, there seems to be a fairly common “set it and forget it” mindset with many users of these services given the number of unpatched/out-of-date Ubuntu (and other Linux distributions) instances and associated installed services in these networks. This finding was a sobering reminder that the security of the internet still trails the desire to just get things working, and working quickly.

While network operators may know exactly what the autonomous system “AS4809” refers to, most of the rest of us do not. As such, it is much easier to just refer to the computers there as simply being “in China” or “in networks attributed or allocated to China.” These network assignments, along with other characteristics, are what IP geolocation services use to attribute a given IP address to a given location. We’ve chosen to use “country” as the attribution feature since it is the most accurate out of all possible fields.

While we use terms like “nation” and “country” interchangeably to describe the regions discussed in this report, we occasionally refer to other named regions on Earth as well, without making any distinction as to their political statuses as special administrative regions, scientific preserves, or claimed territories of other countries. The rule for inclusion is simply, “Does the region have an ISO 3166 3-alpha country code?”[2] If so, it's a “country” or “nation” for our research purposes, regardless of the political realities of the day. For example, regions like “Hong Kong” (HKG) can and do appear in this research (Hong Kong has some interesting features when it comes to exposure), while “California” does not, even though it is home to plenty of poorly configured internet-connected devices.

In case you're reading this report and wondering only what the "most exposed" countries in the world are, we hope to save you the trouble by providing you with the discussion below and the findings in Table 1.

Most Exposed Countries

Table 1 shows the most exposed countries, calculated by weighting a country higher on the list (i.e., more bad exposure) by:

- **Total attack surface** (i.e., number of total IPv4s in use exposing something during the study period). Rationale: More stuff = more stuff to attack.
- **Total exposure of selected services.** Specifically SMB, SQL Server, and Telnet. Rationale: These should never be exposed. Ever.
- **Distinct number of CVEs present across all services.** Rationale: More known vulnerabilities = more exposure.
- **The center of the distribution of vulnerability rates.** Vulnerability rate is defined as the number of exposed services with vulnerabilities/exposed services. Rationale: Higher vulnerability concentration across all exposed services should contribute more to the rank penalty.
- **Maximum vulnerability rate.** Rationale: To break any ties that remain after the previous steps, penalize a nation state with the highest vulnerability rate.

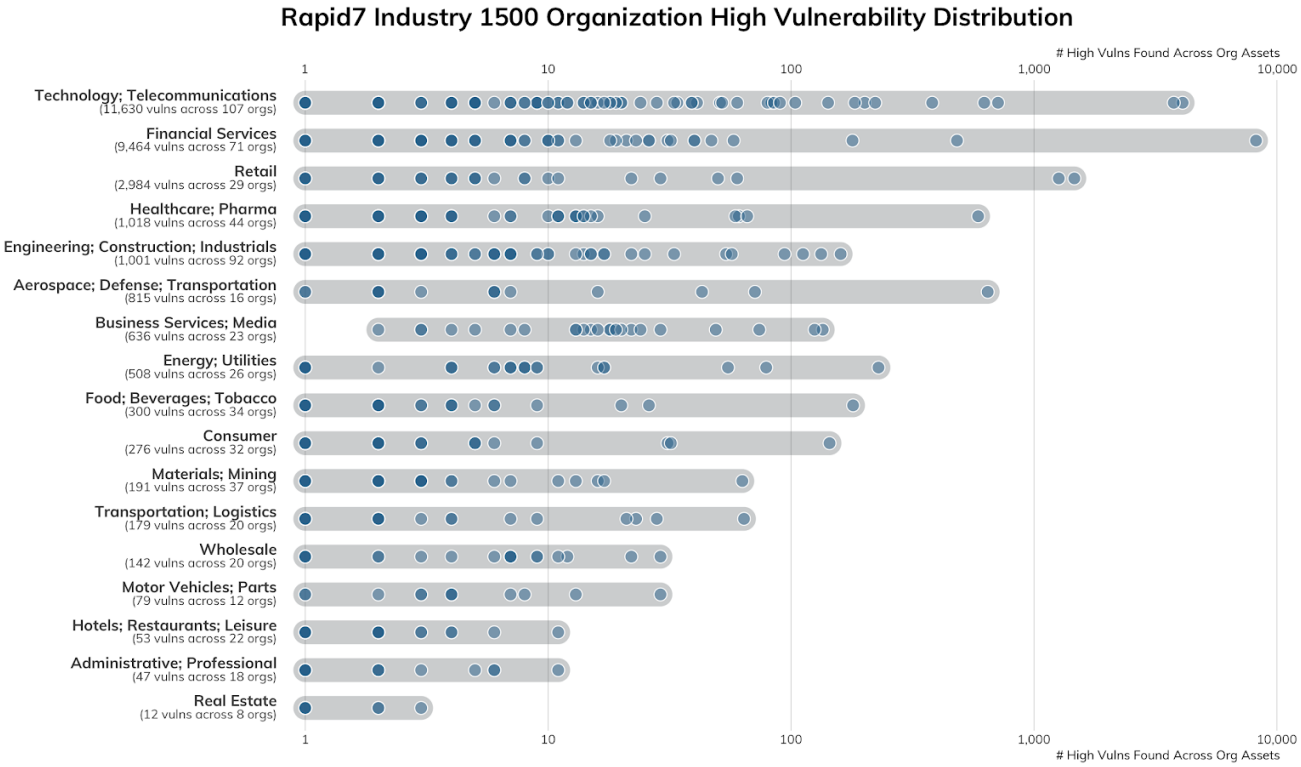
As one might expect, our ranking methodology highlights countries with large swaths of IP space, so the U.S. and China being in the No. 1 and No. 2 spots of “most exposed” shouldn't be too surprising. The interesting, and perhaps more surprising, findings are found from No. 3 through the remainder of the 50 most-exposed regions. For example, while Canada and Iran

both have sophisticated, extensive internet presence, Canada has less than half the human population of Iran. Even so, Canada and Iran have very similar exposure rates, with Canada edging out Iran for the No. 9 exposure spot.

More to Come

In the coming weeks, we expect to produce supplemental reports that take a closer look at certain countries and groups of countries, such as Japan and Latin America. You can look here for those updates, or just keep an eye out for them on the Rapid7 blog. If you have a favorite region you'd like to visit with this particular magnifying glass, please let us know!

In 2019, Rapid7 produced a series of reports, the Industry Cyber-Exposure Reports (or ICERs), which looked at the security posture of the Fortune 500 in the United States, the FTSE 250+ in the United Kingdom, the Deutsche Börse Prime Standard 320 in Germany, the ASX 200 in Australia, and the Nikkei 225 in Japan. Because we've already gone to the trouble of identifying these 1,500 or so highly successful, well-resourced, publicly traded companies, we've also taken a moment to grade and rank these industries against each other.



In the above chart, each dot represents one organization, and the position on the X axis represents how many high-severity vulnerabilities were discovered in components of their internet-facing attack surface. These components range from web servers to caching servers to

the operating systems these services run on. The majority of the weakness are in poorly maintained Apache HTTP web servers with a smattering of “the usual suspects” (e.g., DNS, SMTP, SSH) quickly trailing behind:

Apache components—in this case, the HTTPD web server and Java servlet runner—have the distinct disadvantage of being much older than, say, nginx, and also being part of “appliances” that many organizations have little visibility into or control over (making a great case for the need of some type of “software bill of materials”[3] so IT and security teams can better manage their exposure).

While the chart shows the overall distribution, we’ve approached ranking these industries in a slightly different way than we did country ranks, since we know a bit more about the population that is exposing these services. These are well-financed organizations; most of them must adhere to at least one regulatory framework; they employ dedicated information technology and information security experts; most have been around for at least 10 years; and, all of them should have some basic IT and cybersecurity hygiene practices in place.

With all this in mind, we defined our own cyber-version of grime and rodent infestations as the number of “high” (CVSS 8.5 or higher) vulnerabilities in discovered services for each industry grouping[4], and assigned a letter grade similar to what New York City does for restaurants. The assigned letter is determined by the logarithmic mean high exposure per-industry (since some industries have far fewer member organizations than others and certain industries lend themselves to exposing hundreds or even thousands of assets), which results in the following scores, shown in Table 3.

Looking back at just the past 12 or so months, all of the industries with a “D” grade have garnered the majority of breach and ransomware headlines, so it’s not too surprising to see them trailing behind the others.

However, none of these industries come close to “perfect,” and our previous ICER work showed each regional list of companies had a lot of work to do when it comes to cyber-hygiene. This is underscored by the fact that 80% of the Apache HTTPD high exposure lies in versions ranging from three to 14 years old, as seen in Table 4.

However, the news is not *all* bad. We’ve mapped the public IPv4 space of just under 1,500 organizations across the ICERs, and only a minority slice—611 (~40%)—had high-severity

exposure. We hope to see that percentage drop as more organizations replace aging IT applications and infrastructure with more modern components.

While we are bolstered by the evidence that progress is being made in cybersecurity, and the trend is heading in the right direction, progress continues to be slow. Arguably, progress is happening far too slowly when considered in the context of the exploding adoption of, and reliance on, connected technologies; the prevalence of both automated, untargeted attacks, and highly sophisticated and well-resourced attackers; and the increasing complexity of our systems.

With that in mind, let’s examine which aspects contribute to the overall security posture of this advanced global telecommunications network we all depend on to conduct commerce, express our culture, and, more than ever, live our daily lives. We believe that it comes down to four considerations, all measured throughout this paper across the 24 protocols we assessed: the design, deployment, access, and maintenance of internet-based services.

Cryptographic Design

The internet was invented in the 1960s, exploded in the 1990s, and has continued to grow and change throughout the 21st century. Along the way, thousands of universities, companies, and individuals have developed novel solutions for transmitting and receiving data across a public network, yet only recently are services designed with at least some security concepts in mind. The most important design element of any new service on the internet is the incorporation of modern cryptography, which serves two functions. First, cryptography ensures that communications and data maintain a high degree of confidentiality and cannot be easily eavesdropped upon or altered while traversing the public internet. Second, cryptographic controls establish the identities of the machines and people involved in secure conversations—cryptography ensures that you know with certainty that the machine that purports to be part of your bank is actually your bank’s, and your bank can know that you are, in fact, you. Whenever cryptography isn't used—known as “cleartext” communications—neither of these assertions of confidentiality nor identity can be guaranteed in any meaningful way.

- **Key finding:** A technical assessment of the 24 service protocols surveyed finds that, on the whole, **unencrypted, cleartext protocols are still the rule**, rather than the exception, on how information flows around the world, with 42% more plaintext HTTP servers than

HTTPS, 3 million databases awaiting insecure queries, and 2.9 million routers, switches, and servers accepting Telnet connections.
- **Key recommendation:** If it touches the internet, it should be encrypted. This goes for data, service identification, everything.

Service Deployment

This area is the one we tend to spend the most time and effort in measuring—what services are actually available on the internet, and who can reach those services? The internet was created with a decentralized and open design, but this openness is a double-edged sword. On the one hand, it makes the internet, as a whole, very difficult to destroy. When parts of the internet are damaged and destroyed, the internet tends to simply route around the damage, continuing to function for other services and their users. On the other hand, it means that every computer can normally reach and exchange information with any other computer, even if some of those computers are operated by malicious attackers. Therefore, when we find services that should not be reachable by just anyone, such as databases, command and control consoles, or other services that external strangers have no business talking to in the first place, we mark them as dangerously exposed.

- **Key finding:** One bit of positive news was that we found **the population of insecure services has gone down** over the past year, with an average 13% decrease in exposed, dangerous services such as SMB, Telnet, and rsync, crushing the predicted doom-and-gloom jump of newly exposed insecure services such as Telnet and SMB, despite the sudden shift to work-at-home for millions of people and the continued rise of Internet of Things (IoT) devices crowding residential networks.
- **Key recommendation:** Keep up the good work! ISPs, enterprises, and governments should continue to work together to kick these inappropriate services off the internet.

Console Access

The first job of the internet was to provide remote system operators with a capability to log on to far-flung computers, run commands, and get results, and securing those remotely accessible consoles was second to connectivity. Today, when we find console access, security professionals tend to take a dim view of that entry point. At the very least, these Telnet, SSH, RDP, and VNC consoles should be hardened against internet-based attackers by incorporating some type of second-factor authentication or control. Usually, incorporating a virtual private

network (VPN) control removes the possibility of direct access; potential attackers must first breach that cryptographically secure VPN layer before they can even find the consoles to try passwords or exploit authentication vulnerabilities. Services lacking this first line of defense are otherwise open to anyone who cares to “jiggle the doorknobs” to see how serious the locks are, and automated attacks can jiggle thousands of doorknobs per second across the world.

- **Key finding:** We have discovered nearly **3 million Telnet servers** still active and available on the internet, and many of those are associated with **core routing and switching gear**. This is 3 million too many. While remote console access is a fundamental design goal of the internet, there is no reason to rely on this ancient technology on the routers and switches that are most responsible for keeping the internet humming.
- **Key recommendation:** Telnet, SSH, RDP, and VNC should all enjoy at least one extra layer of security—either multifactor authentication, or available only in a VPN-ed environment.

Software Maintenance

We often talk about the internet in terms of the machines that swap the bits of information around the world, but recall that these machines are remarkably durable. They are not designed with physical pistons or gears that can wear out, but instead are largely made of solid-state, unmoving electronics, with the most mechanical part of a machine being a cooling fan. These machines can run, without incident or human attention, for many years. While this may seem like a feature, it can, in fact, lead to complacency. The complex software that runs on this hardware absolutely does require routine maintenance in the form of patches, upgrades, and eventual replacements. As our knowledge and understanding of security and secure design concepts expand, new software must be deployed to satisfy the same business, academic, or culture requirements. Unfortunately, we have found evidence that this routine maintenance is largely absent in many areas of internet service, as we find decades-old software running with decades-old software vulnerabilities, just waiting for exploitation.

- **Key finding: Patch and update adoption continues to be slow**, even for modern services with reports of active exploitation. This is particularly true in the areas of email handling and remote console access where, for example, 3.6 million SSH servers are sporting vulnerable versions between five and 14 years old. More troubling, the top publicly traded companies of the United States, the United Kingdom, Australia, Germany, and Japan are hosting a surprisingly high number of unpatched services with known

vulnerabilities, especially in **financial services** and **telecommunications**, which each have ~10,000 high-rated CVEs across their public-facing assets. Despite their vast collective reservoirs of wealth and expertise, this level of vulnerability exposure is unlikely to get better in a time of global recession.

- **Key recommendation:** Given that all software has bugs and patching is inevitable, enterprises should bake in regular patching windows and decommissioning schedules to their internet-facing infrastructure.

With these aspects of internet engineering in mind, we are certain that there is much more work to do. While the internet has gotten incrementally "better" in the past year, and so far appears to be resistant to biologically and economically influenced disaster, vulnerability and exposure abound on the modern internet. These exposures continue to pose serious risk through the continued use of poorly designed, outmoded software, through an over-availability of critical "backend" services, through uncontrolled operating system console access points, and through a lack of routine patch and upgrade maintenance.

Furthermore, we know that internet-based attackers are well aware of these weaknesses in design, deployment, access, and maintenance, and are already exploiting soft internet targets with essentially no consequences. We know there are no effective "internet police" patrolling the internet, so it is up to IT and cybersecurity professionals around the world to secure their internet infrastructure, with the assistance of those who teach, train, advise, and pay them.

This is a fairly weighty tome of observations. Specific, actionable security advice is given for each of the 24 protocols and their seven service categories of console access, file sharing, email, graphical remote access, database services, core internet infrastructure, and the web. If you take nothing else away from this research report, we hope we can convince you of this: The internet is not an automatic money- and culture-generating machine; it depends on the heroic efforts of thousands and thousands of professionals who are committed to its well-being, even in the face of daily attacks from a wide array of technically savvy criminals and spies.

Policymakers—those folks in governments around the world who promulgate and enforce regulations, determine public budgets and procurement priorities, and oversee tax-funded grants—should and do play a crucial role in the security of the internet. Security is not manifested through technology alone, but also through the right mix of incentives, information sharing, and leadership, ideally informed by up-to-date and accurate information on the evolving

security landscape and expert guidance on potential solutions to complex security challenges. Where could policymakers find some research like that? Why, right here!

The NICER 2020 outlines the risks and multinational prevalence of protocols that are inherently flawed or too dangerous to expose to the internet—such as FTP, Telnet, SMB, and open, insecure databases—which have direct bearing on public- and private-sector security. Policymakers can review this corpus of current internet research to help inform decisions regarding online privacy, security, and safety. Policymakers should consider using this information to encourage both public- and private-sector organizations to reduce their collective dependence on high-risk protocols, mitigate known vulnerabilities and exposures, and promote the use of more secure protocols. For example:

- Issue guidance on the use of specific high-risk protocols and software applications, such as in the context of implementing security and privacy regulations;
- Evaluate the presence of high-risk protocols and applications in government IT, and transition government systems to more secure and modernized protocols;
- Conduct oversight into agencies' efforts to mitigate known exposures in government IT;
- Consider exposure to dangerous protocols in inquiries, audits, or other actions related to private-sector security;
- Use data regarding the regional prevalence of known exposures and dangerous protocols to enhance multinational cooperative efforts to share threat information;
- Direct additional studies into the impact of cleartext, dangerous, and inherently insecure protocols and applications on national security, national economies, and consumer protection, and the challenges organizations face in mitigating exposure.

The remainder of this report is concerned with in-depth analysis of 24 protocols across seven families of services—everything from Telnet to DNS-over-TLS. First, we compare the raw number of services providing each protocol compared to counts from the same time last year, in 2019. Then, each protocol section will have a brief "Too Long, Didn't Read" (or TLDR) overview of that protocol with an executive summary of:

- What the protocol is for
- How many of these services we found
- The general vulnerability profile of that protocol
- Alternatives to that protocol for its business function
- Our advice regarding that protocol

- Whether things are getting better or worse in terms of deployment

Following this overview, we provide detailed data and analysis about these protocols, including the per-country, per-cloud, and per-version breakdowns of what we saw. Each one of these analyses would make for fine papers in and of themselves, but lucky you, you get all of them at once. So, strap on your cyber-scuba rebreather, and let's dive deep!

Providing a point-in-time exposure view has tons of merit, but the Rapid7 Labs team was curious as to how April 2020 compares to April 2019 since *quite a bit has happened since then*.

We approached the report with some hypotheses regarding the impact of the global pandemic and a looming worldwide recession, such as “we will probably see more RDP servers” as people are forced to work from home, but still need access to their GUI consoles, or, “we'll probably see a spike in SMB” as people hastily reconfigured Windows workflows to accommodate a remote workforce. It turns out that we were *woefully wrong*. Table 5 shows the rise and fall—and mostly fall—of the protocols covered in these 2020 studies compared to a sampling of 2019 studies around this time last year.

Table 5

Sonar Study	Port	2019	2020	Change	Percentage
SMB	445	709,715	594,021	-115,694	-16.30%
Telnet	23	3,250,417	2,830,759	-419,658	-12.91%
rsync	873	233,296	208,882	-24,414	-10.46%
POP3	110	4,818,758	4,335,533	-483,225	-10.03%
SMTP	25	6,439,139	5,809,982	-629,157	-9.77%
IMAP	143	4,296,778	4,049,427	-247,351	-5.76%
SMTP	587	4,220,184	4,011,697	-208,487	-4.94%
RDP	3389	4,171,666	3,979,356	-192,310	-4.61%
POP3S	995	3,887,033	3,717,883	-169,150	-4.35%

Sonar Study	Port	2019	2020	Change	Percentage
IMAPS	993	4,008,577	3,852,613	-155,964	-3.89%
MS SQL Server	1434	102,449	98,771	-3,678	-3.59%
SMTPS	465	3,592,678	3,497,791	-94,887	-2.64%
DNS (Do53)	53	8,498,166	8,341,012	-157,154	-1.85%
FTP	21	13,237,027	13,002,452	-234,575	-1.77%
NTP	123	1,653,599	1,638,577	-15,022	-0.91%
FTPS	990	443,299	460,054	16,755	3.78%
SSH	22	15,890,566	18,111,811	2,221,245	13.98%
DNS over TLS (DoT)	853	1,801	3,237	1,436	79.73%

Before we break down the table, we need to emphatically state that **the internet is getting safer**, at least where these core services are concerned. As certified security curmudgeons, that sentence was extremely difficult to type. But the survey data does not lie, so let's dig into the results.

First, we've left a few services without a highlight color in the "percent change" column. Scanning the internet is fraught with peril, as noted in the **Methodology** appendix, and we've collected sufficient scan data over the course of six years to understand how "off" a given scan might be (it's become a bit worse over the years due to the increase in dynamic service provisioning in cloud providers, too). Each Sonar study has its own accuracy range, but across all of them, any results differing +/- 4% means we cannot make definitive statements about whether anything's "gone up" or "gone down." All that said, it turns out that most services had numbers well outside that range.

Microsoft SMB has seen almost a 20% drop in exposure, mostly due to ISPs blocking SMB ports. There are still over half a million Windows and Linux systems exposing SMB services, so let's not break out the champagne *just yet*.

Telnet being down nearly 15% warms our hearts, since we've spent the last four or so years begging y'all to stop exposing it. Still, a sizable chunk of those remaining 2.8 million devices with Telnet exposed are routers and switches, so just because those services are stuck in 1999 configuration mode does not mean we can party like it is that year, either.

It's great to see folks giving up on hosting their own mail infrastructure. While this transition to professionally hosted email services like Outlook 365 and G Suite *may* have seen a more accelerated adoption because of the sudden shutdown, we believe that this transition to more centralized email was already in progress and will continue.

Seeing a near 13% increase in SSH leaves us a bit concerned (see the **SSH** section to see how seriously vulnerable the SSH service and their host operating systems are). We're hoping to see a reduction in SSH over time as more Linux distributions mimic Ubuntu and adopt WireGuard[6] for secure remote system connections, as it is much safer and as or more secure than SSH, and provides even better connectivity.

Finally, we're not surprised that the number of DoT servers nearly doubled over the past year, and expect to see another doubling (or more) in 2021.

So, while we have seen a major shift in the way people work and use the internet in this time of pandemic lockdown and economic uncertainty, we haven't detected a major shift in the nature and character of the internet itself. After all, the internet was designed with global disaster in mind, and to that end, it appears to be performing admirably.

That said, let's jump into the per-protocol analysis. Don't worry, we'll get back to our usual cynical selves.

In the beginning, the entire purpose of the capital-I Internet was to facilitate connections from one text-based terminal to the other, usually so an operator could run commands "here" and have them executed "there." While we usually think of the (now small-i) internet as indistinguishable from the (capital W's) World Wide Web, full of (essentially) read-only text, images, audio, video, and eventually interactive multimedia applications, those innovations came much later. Connecting to the login interface of a remote system was the foundational feature of the fledgling internet, and that legacy of remote terminal access is alive and not-so-well today.

Telnet (TCP/23)

It wasn't the first console protocol, but it's the stickiest.

Discovery Details

Way back in RFC 15,[7] Telnet was first described as “a shell program around the network system primitives, allowing a teletype or similar terminal at a remote host to function as a teletype at the serving host.” It is obvious from this RFC that this was intended to be a temporary solution and that "more sophisticated subsystems will be developed in time," but to borrow from Milton Friedman, there is nothing quite so permanent as a temporary solution. Remote console access is still desirable over today's internet, and since Telnet gets the job done at its most basic level, it persists to this day, 50 years later.

Exposure Information

Technically, despite its half-century of use, Telnet (as a server) has suffered relatively few killer vulnerabilities; a quick search of the CVE corpus finds only 10 vulnerabilities with a CVSS score of 5 or higher. However, Telnet suffers from a few foundational flaws. For one, it is nearly always a cleartext protocol, which exposes both the authentication (username and password) and the data to passive eavesdropping. It's also relatively easy to replace commands and responses in the stream, should attackers find themselves in a privileged position to man-in-the-middle (MITM) the traffic. Essentially, Telnet makes little to no security assurances at all, so paradoxically, the code itself tends to remain relatively vulnerability-free.

The bigger issue with Telnet is the fact that, in practice, default usernames and passwords are so common that it's assumed to be the case whenever someone comes across a Telnet server. This is the central hypothesis of the Mirai worm of 2016, which used an extremely short list of common default Telnet usernames and passwords and, for its trouble, managed to take down internet giants like Twitter and Netflix, practically by accident.

The chart to the right shows that China alone has a pretty serious Telnet problem, followed by Argentina and the United States.

While we would prefer to see no Telnet among cloud service providers, we see that good progress seems to have been made here. Microsoft Azure tops the list with about 7,000 exposures, which is a little weird since Windows platforms don't normally have a built-in Telnet server.

Of the Telnet instances found on the internet that we could confidently identify by vendor, Table 6 describes those services from vendors that appear in Sonar with at least 10,000 responsive nodes.

Table 6

What's interesting about these figures is that we can see right away that the vast majority of Telnet services exposed to the internet are strongly associated with core networking gear vendors. Cisco and Huawei, two of the largest router manufacturers on Earth, dominate the total count of all Telnet services. Furthermore, sources indicate that about 14% of Cisco devices and 11% of Huawei devices are accessible, today, using default credentials. This lack of care and maintenance of the backbone of thousands of organizations the world over is disappointing, and every one of these devices should be considered compromised in some way, today.

While Telnet is still prevalent far and wide across the internet, we can see that some ISPs are more casual about offering this service than others. The table below shows those regional ISPs that are hosting 10,000 or more Telnet services in their network. Furthermore, the vast majority of these exposures are not customer-based exposures, but are hosted instead on the core routing and switching gear provided by these ISPs. This practice (or oversight), in turn, puts **all** customers' network traffic originating or terminating in these providers at risk of compromise.

Attacker’s View

Telnet’s complete lack of security controls, such as strong authentication and encryption, makes it totally unsuitable for the internet. Hiding Telnet on a non-standard port is also generally useless, since modern Telnet scanners are all too aware of Telnet lurking on ports like 80 and 443, simply because firewall rules might prohibit port 23 exposure. Today, Telnet is most commonly associated with low-quality IoT devices with weak to no security hardening, so exposing these devices to the internet is, at the very least, a signal that there is no security rigor in that organization.

Judging from our honeypot network, attackers continue to actively seek out Telnet services, even four years after the Mirai attacks that took several hundred thousand offline. Approximately 90% of the traffic shown below represents basic access attempts using common usernames and passwords. The spikes represent concentrated credential stuffing attacks, using new username and password combinations pulled from publicly available dumps of other credentials.[8]

Our Advice

IT and IT security teams should prohibit Telnet traffic both to and from their networks, both through blocking standard Telnet ports and through deep packet inspection (DPI) solutions, whenever possible. They should continuously monitor their network for Telnet services, and track down offending devices and disable Telnet connectivity. There is no reason to expose Telnet to the modern internet, ever. In the rare case a network operator wants someone to log in to a text-based console over the internet, a well-maintained SSH server will be far more reliable, flexible, and secure.

Cloud providers should operate similarly to the above organizations. They should, by default, prohibit all Telnet traffic through automatic, technical means. It's possible that some customers are relying on Telnet today, but those customers should be gently, but firmly, moved to SSH-based alternatives. After all, Telnet clients are not available by default on modern versions of Windows, OSX, or Ubuntu Linux for a reason, so it would seem that few future clients will suffer from its absence on their hosted machines.

Government cybersecurity agencies should actively pursue those ISPs that are offering an egregious amount of Telnet access to their core network operations hardware, and give those operators a stern talking-to, along with migration documentation on how to set up SSH, if needed. ISPs, at this point, should know better.

Secure Shell (SSH) (TCP/22)

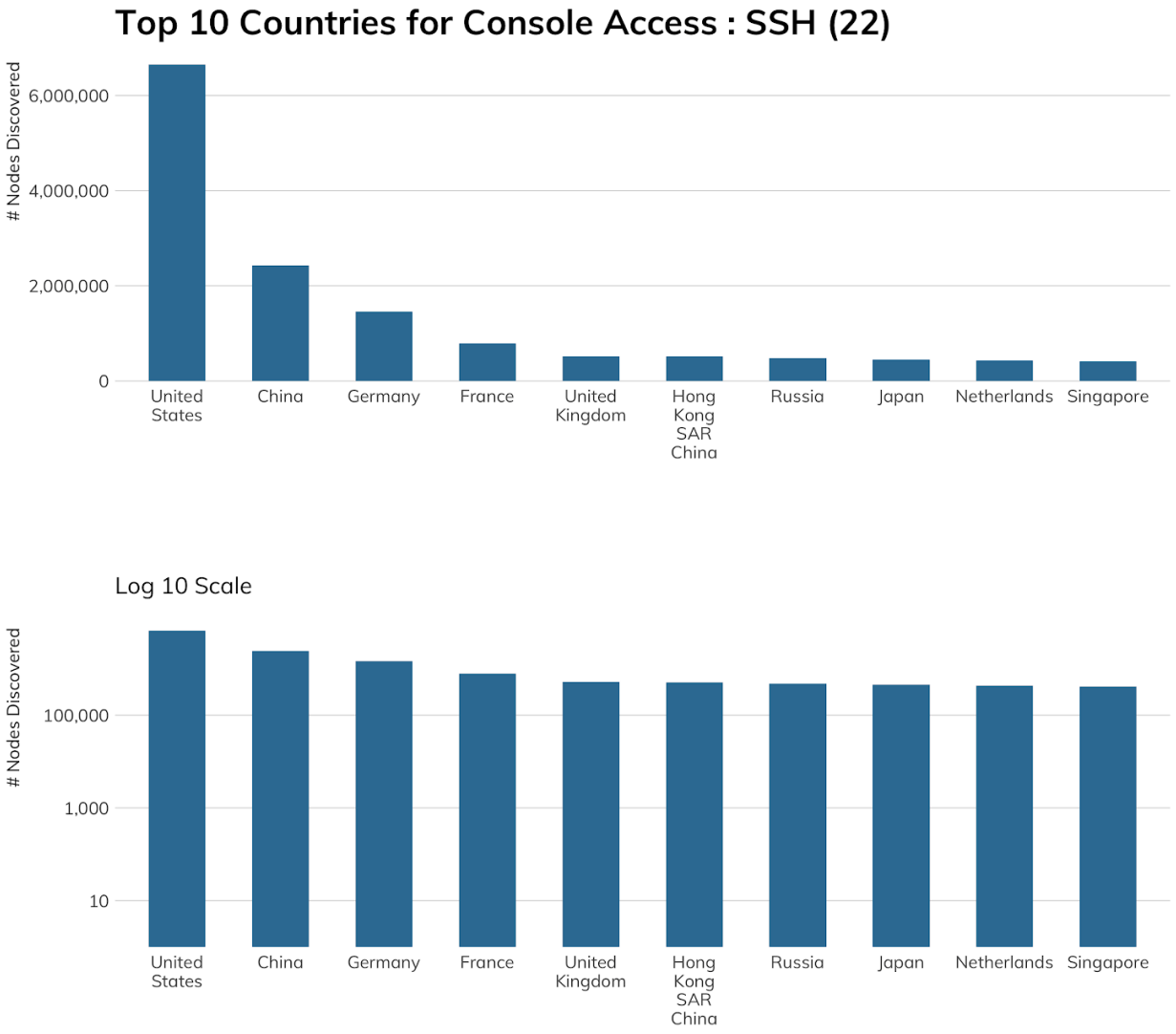
It’s got “secure” right in its name!

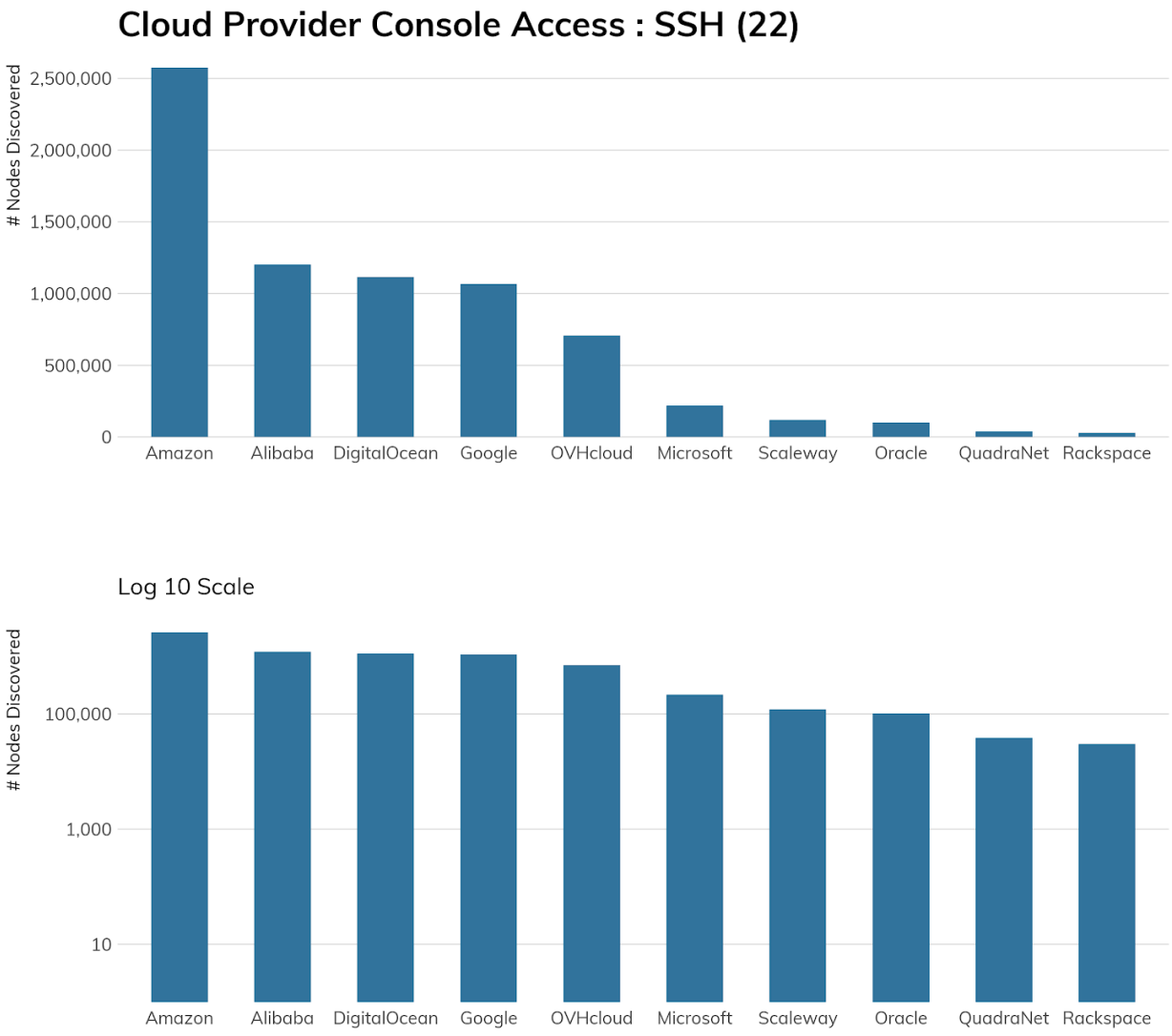
Discovery Details

Secure Shell, commonly abbreviated to SSH, was designed and deployed in 1995 as a means to defend against passive eavesdropping on authentication that had grown common against other cleartext protocols such as Telnet, rlogin, and FTP. While it is usually thought of as simply a cryptographically secure drop-in replacement for Telnet, the SSH suite of applications can be used to secure or replace virtually any protocol, either through native applications like SCP for file transfers, or through SSH tunneling.

One of the bright spots of this report analysis is the fact that SSH deployment has now outpaced Telnet exposure at a rate of six to one—it seems the world has gotten the message

that, while direct console access from the internet might not be the wisest move in the world, about 85% of those exposed shells are secure shells, which takes whole classes of eavesdropping, spoofing, and in-transit data manipulation attacks off the table. Good job, internet, and especially the American network operators around the country—the United States exposes a ratio of 28:1 SSH servers (6.6 million) than Telnet servers (a mere 232,000). Compare this to the 3:1 ratio in China, which is 2.4 million SSH to 734,161. Given that SSH provides both console access and the capability of securing other protocols, whereas Telnet is used almost exclusively for console access, that United States ratio is pretty outstanding.





Exposure Information

Being more complex and making more explicit security guarantees, SSH is not without its own vulnerabilities. Occasionally, traditional stack-based buffer overflows surface as with other network applications written in memory-unsafe languages. In addition, new vulnerabilities in implementations tend to present themselves in non-default combinations of configuration options, or are surfaced in SSH through vulnerabilities in the cryptographic libraries used by SSH to ensure secure communications. Most commonly, though, vulnerabilities in SSH are often associated with unchangeable, vendor-supplied usernames, passwords, and private keys that ship with IoT devices that (correctly) have moved away from Telnet. This is all to say that Secure Shell is not magically secure just due to its use of cryptography—password reuse is weirdly common in SSH-heavy environments, so protecting passwords and private keys is an important part of maintaining a truly secure SSH-based infrastructure.

As mentioned above, administrators and device manufacturers alike are strongly encouraged to adopt the open, free standards of SSH over their cleartext counterparts whenever possible. IoT, OT, and ICS equipment, in particular, is often cited as not having enough local resources to deal with the cryptographic overhead of running secure services, but if that is actually the case, these devices should never be exposed to an internet-connected network in the first place. As mentioned above, it is also not enough to simply move insecure practices such as default, reused passwords from a cleartext protocol to a “secure” protocol—the security offered by cryptography is only as good as the key material in use.

Of the SSH services discovered on the internet, the below table accounts for well over 99.9% of all offerings (only those fingerprintable services with at least 1,000 or more shown here.)

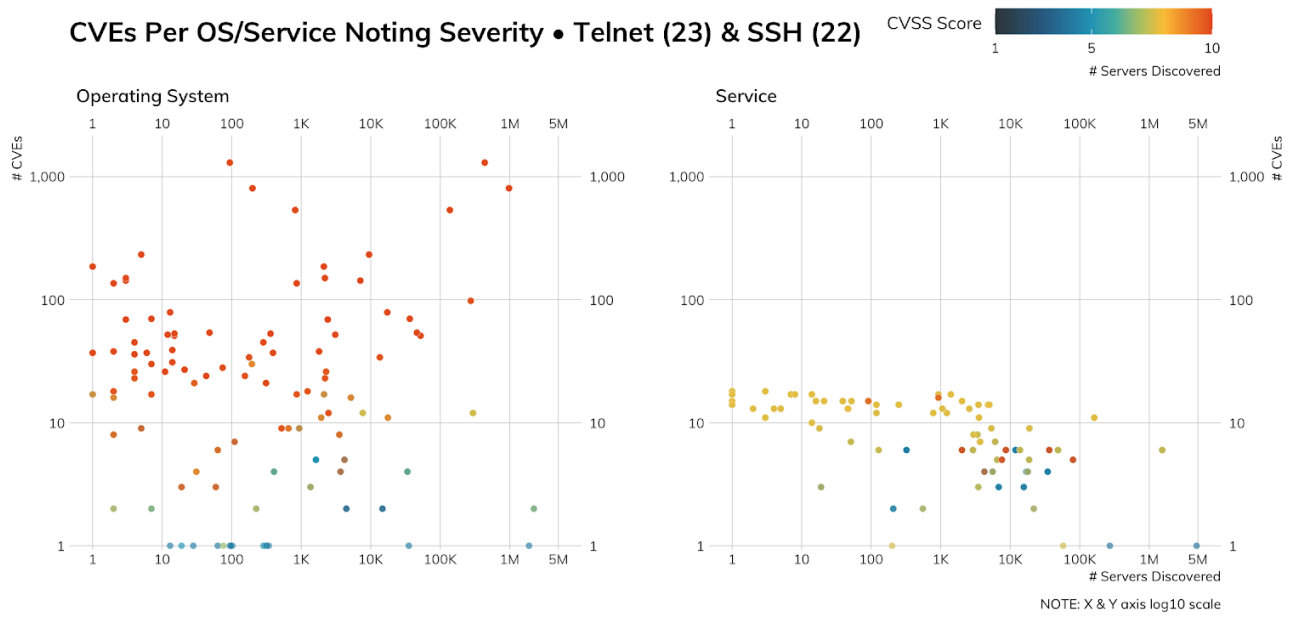
Attacker’s View

SSH provides console access in a better way than Telnet, but it is still just a piece of software with many features, including one that drops you to a command prompt after a successful login, so attackers perform credential stuffing (which can include using stolen certificates, too, for SSH) and vulnerability exploits against the service itself. As such, we’d be just repeating much of the same content as we did in Telnet (and we value your time too much to do that).

What we can do is focus more on two things: vulnerabilities associated with exposed SSH services and how much information exposed SSH services give to attackers.

The most prevalent version of OpenSSH is version 7.5. It turns four years old in December and has nine CVEs. Every single version in the top 20 has between two and 32 CVEs, meaning there’s a distinct lack of patch management happening across millions of systems exposed to the cold, hard internet.

You may be saying, “So what?” (a question we try to ask ourselves regularly when we opine about exposure on the internet). Another exposure view may help answer said question:



The above figure packs quite a bit of information into it, so let's dig in as we hit our second point (how much information exposed SSH services[9] give to attackers).

The left panel says "Operating Systems," which means we can figure out which operating system is in use just from the data we get back from our Sonar SSH connections (enumerated below). Each dot represents one operating system version, the position on the X axis represents how many servers we found, and the position on the Y axis represents the number of CVEs that have been assigned to it. The color represents the highest severity. There is quite a bit of CVSS 8+ on that chart, which means there are quite a few high-priority vulnerabilities on those systems (at the operating system level).

The right panel is the same, just for SSH versions (one of which we enumerated above). The count difference is due to both recog coverage[10] and the fact that a single operating system version can run different versions of SSH, so the aggregation up to operating system category will be higher. There are fewer major vulnerabilities within the SSH services but there are more above 7 than below, which means they're also pretty high-priority.

Adversaries can use this information to plan which attack path they plan on taking and map vulnerabilities to exploits to add to their arsenal. Overall, we'd say attackers could have quite a bit of fun with this particular SSH corpus.

Table 10

Our Advice

IT and IT security teams should be actively replacing cleartext terminal and file transfer protocols with SSH alternatives whenever and wherever they are discovered. If such replacements are impossible, those devices should be removed from the public internet.

Cloud providers should provide SSH as an easy default for any sort of console or file transfer access, and provide ample documentation on how customers can wrap other protocols in SSH tunnels with specific examples and secure-by-default configurations.

Government cybersecurity agencies should actively promote the use of SSH over unencrypted alternatives, especially when it comes to IoT. These organizations can also help encourage industrial use of SSH in areas where Telnet and FTP are still common. In addition, cybersecurity experts can and should encourage good key management and discourage vendors from shipping SSH-enabled devices with long-lived, default passwords and keys.

Once the internet patriarchs had console access, they needed way to get files to and from their remote systems, and file sharing constitutes the majority of modern internet traffic, whether it be encoded streams of movies to our devices, viewing far too many cat pictures, or interacting with websites, which download an average of about 70 files each time you visit an uncached page. [11]

Requests for web resources are not the only type of file sharing on the modern internet. For our report, we looked at three other services under this broad category of “File Sharing”:

- File Transfer Protocol (FTP) (TCP/21) and “Secure” FTP (TCP/990)
- Server Message Block (SMB) (TCP/445)
- Rsync (TCP/873)

Each of these services has its own, unique safety issues when it comes to exposing it on the internet, and we’ll start this breakdown with both types of FTP.

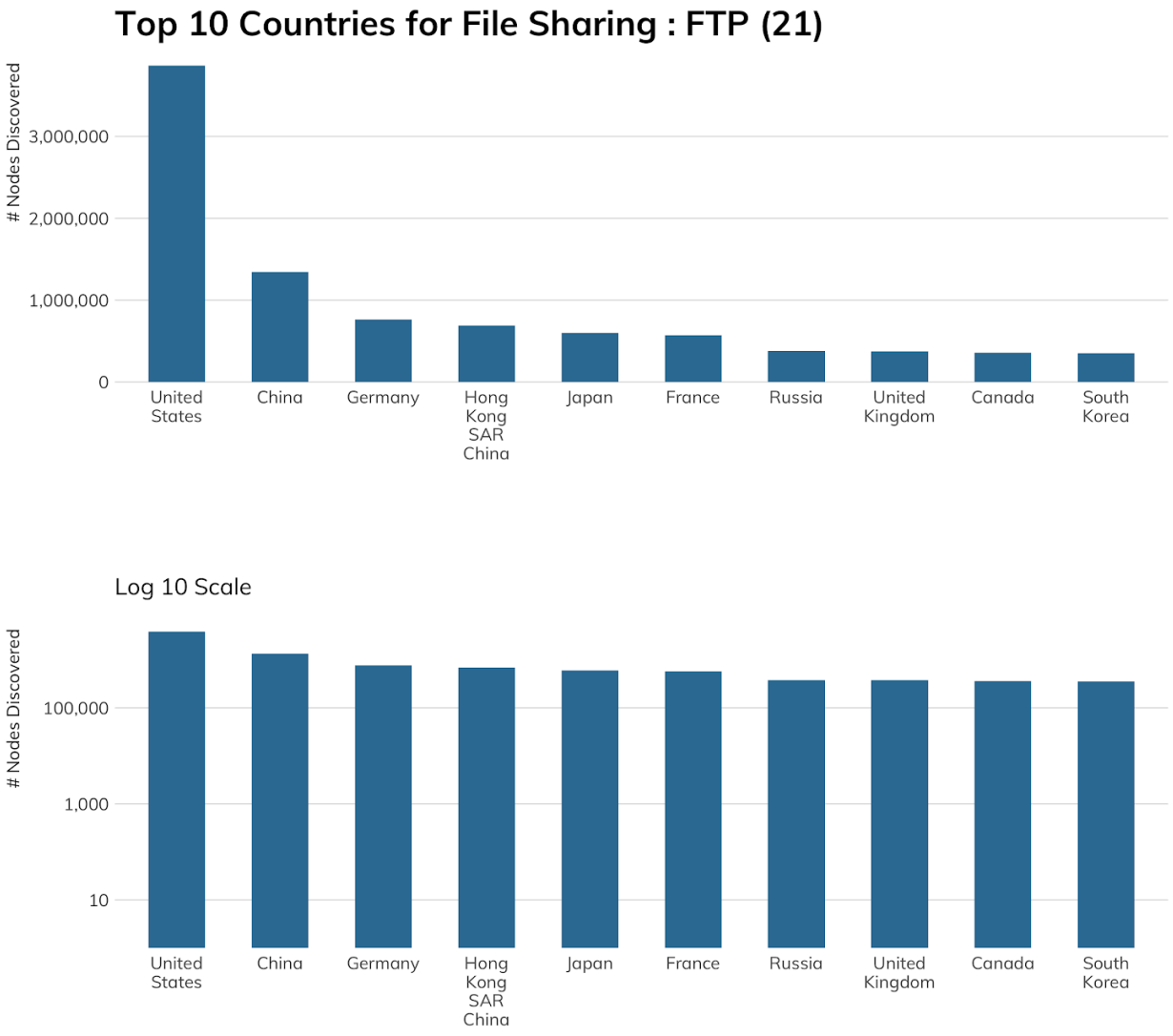
FTP (TCP/21)

A confusing data channel negotiation sequence and totally cleartext—what could go wrong?

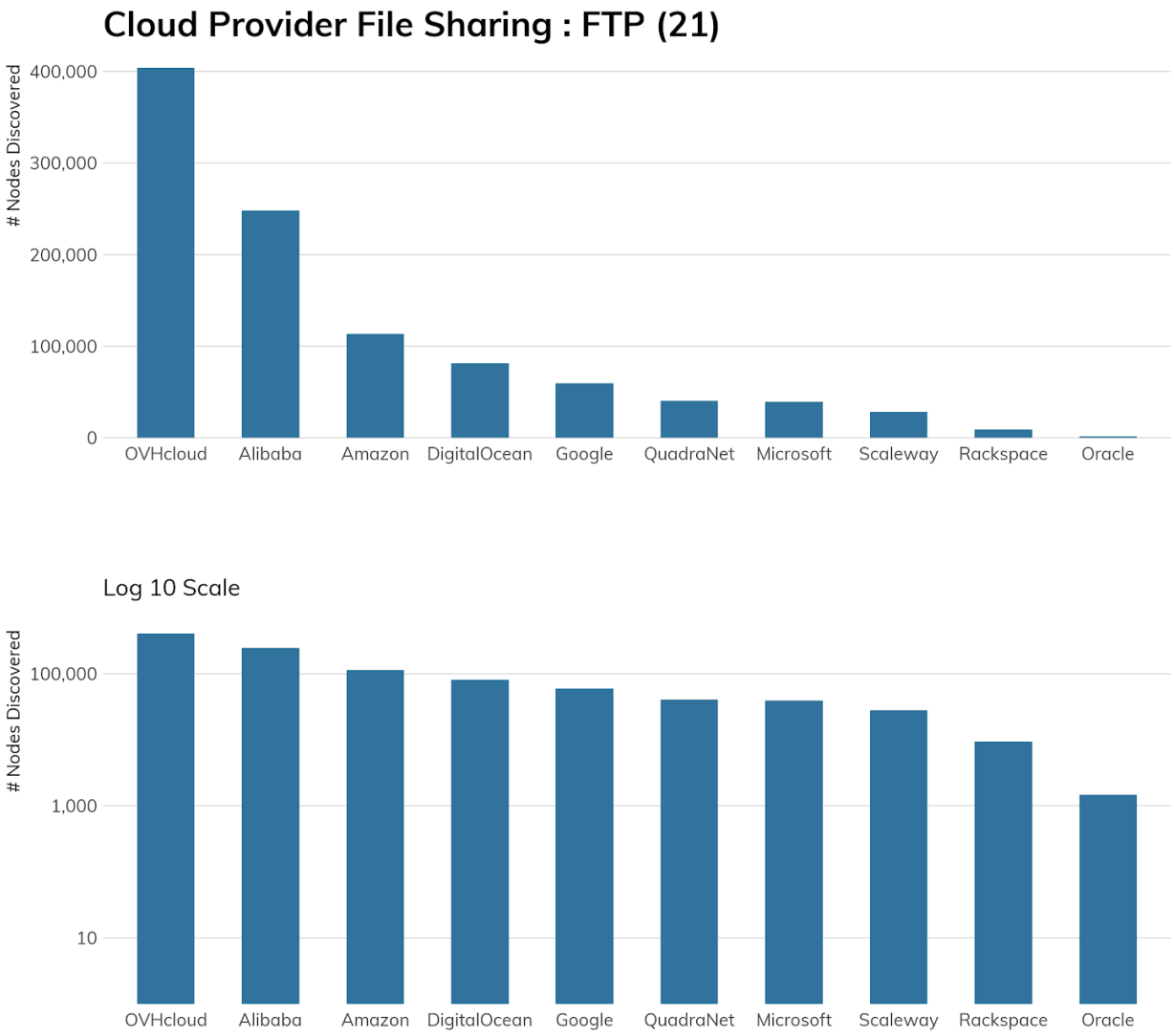
Discovery Details

FTP (the TCP/21 variant) dates back to 1972 and was intended “to satisfy the diverse needs of users of maxi-HOSTs, mini-HOSTs, TIPs, and the Datacomputer, with a simple,

elegant, and easily implemented protocol design."[12] It is a cleartext, short command-driven protocol that has truly outlived its utility, especially now that the two main browser developers are dropping or have dropped support for it.[13]



Project Sonar found just under 13 million active FTP sites (12,991,544), which is down from over 20 million from our previous National Exposure Index study. The United States leads the pack when it comes to exposing FTP to the internet with ~3.9 million hosts awaiting your `OPEN`.



Somewhat surprisingly, there is a large FTP presence in cloud environments, with Alibaba, Amazon, OVH, and Azure hosting around 13.5% of discovered nodes.

When we looked into this a bit more, we discovered that Alibaba generously provides detailed documentation on how to “Build an FTP site” on both Linux[14] and Windows,[15] and that Amazon has a brand-new FTP service offering.[16] Similarly, both OVH[17] and Microsoft[18] have documentation on FTP, and OVH goes so far as to *actively encourage* customers to use it to manage files for their web content.

Furthermore, all the providers have base images that either ship with FTP available or easily installable (via the aforementioned helpful documentation!), so the extent of FTP exposure in cloud environments should really come as no surprise.

Exposure Information

In 2020, there is a single acceptable use of FTP, which is providing an anonymous FTP[19] download service of more or less public data. In this application, and only this application, cleartext credentials won't be exposed (but data in transit will be). Any other use should be considered insecure and unsafe, and should be avoided. Even this use case is edging into unacceptable territory, since FTP doesn't provide any native checksumming or other error checking on file transfers, making man-in-the-middle alterations of any data in transit fairly trivial.

That said, there are millions of FTP servers on the internet. We're able to fingerprint several,[20] and Table 11 illustrates with fairly high fidelity the state of the FTP universe, authenticated and anonymous alike.

Yes, you read that last line of the table correctly: 166 *printers* are exposing their FTP service to the internet. Perhaps more disconcerting are 3,401 Ubiquiti Unified Security Gateway (firewalls) with FTP exposed.

Fourteen of the FTP server types make it somewhat easy to fingerprint version numbers, and vsFTPd, ProFTPD, Bftpd, and Filezilla account for 95% of that corpus (which is ~2.1 million FTP servers, i.e., ~17% of exposed systems). We've put the top five versions of each below along with some commentary, as including the entire version summary table for each would take a few reams of e-paper.

Table 12

Version 3.0.3 is current and appears safe from an RCE perspective, but 3.0.2 has one moderate remote exploit CVE.[21] Version 2.2.2 dates back to 2011 and has a DoS CVE.[22] Version 2.0.5 harkens back to 2008 with an edgy, remote (authenticated) CVE.[23]

Table 13

Three most prevalent versions of ProFTPD—1.3.5/a/b—just happen to have a super-dangerous CVE[24] that allows remote attackers to read and write arbitrary files. 1.3.4.a has a nasty DoS CVE[25], followed by 1.3.5e, with a low-severity CVE[26] that gives authenticated attackers the ability to reconfigure directory structures under certain configurations.

Table 14

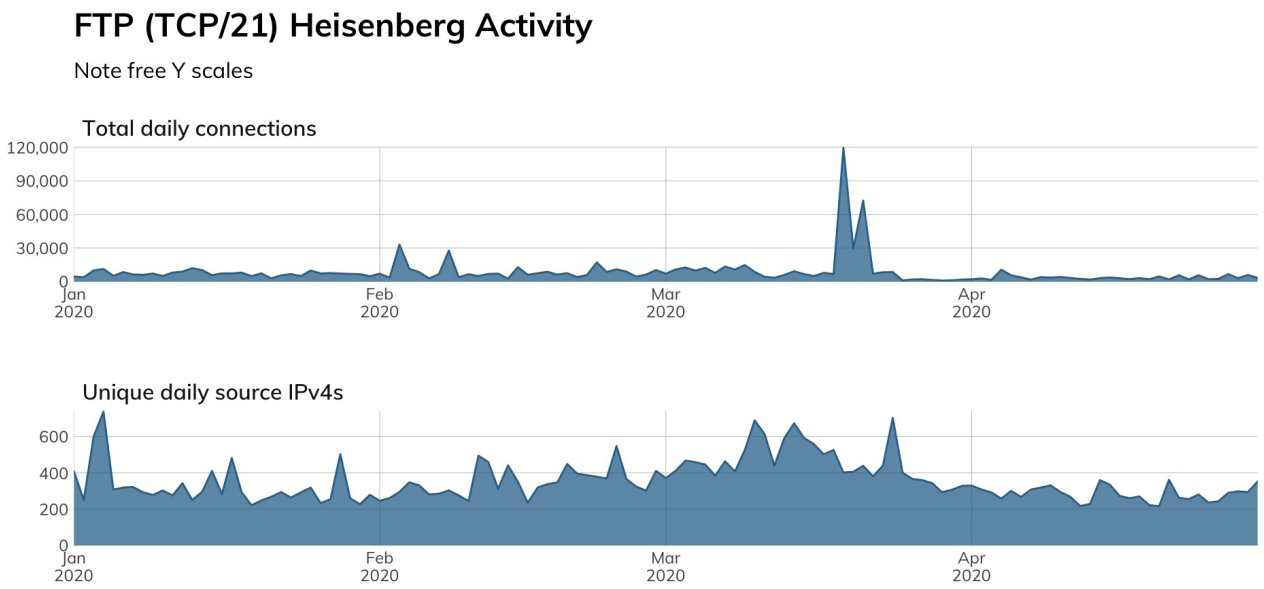
Version 2.2 (which turns 11 years old in November) has a handy DoS CVE,[27] with the remaining ones having a memory corruption CVE[28] exploitable by logged-in users.

Table 15

Shockingly, the top 5 FileZilla versions are CVE-free, but are incredibly old.

Attacker's View

Project Heisenberg does not have an FTP honeypot, but we do capture all connection attempts on TCP/21, and there are often quite a few:



This activity includes attackers looking for an exploitable service on a node they can add to their C2 inventory, or both attackers and researchers searching for documents they can plunder. In either scenario, you're much better off not giving them one more system to conquer.

Our Advice

IT and IT security teams should avoid using FTP (TCP/21) both internally and externally. This cleartext protocol is inherently unsafe, and there are far safer and just better modern alternatives. For system-to-system file transfers, the "S" (FTPS/SFTP) versions of FTP or (preferably) SCP are all safer and often faster replacements that have widespread client and server support. For more interactive use, there are many commercially available or open source file sharing solutions that work over HTTPS. If you truly need to FTP firmware updates to

infrastructure components, you should at the very least ensure those endpoints are not internet-facing and consider putting them behind an internal VPN gateway to keep prying eyes from seeing those credentials.

FTP should also be flagged during vulnerability sweeps and in public attack surface monitoring. During procurement processes, vendors should be asked whether their solutions support, include, or require FTP and use that information to help pick safer products and services.

Cloud providers should discourage use of FTP by only providing base images that do not come with FTP installed or enabled by default, rather than providing friendly and helpful documentation on how to set up a woefully insecure service. Service documentation and other instructional materials should actively discourage the use of FTP, and providers should consider performing regular service scans with associated notifications to customers when they find FTP exposed. If possible, the FTP protocol itself and/or TCP/21 should be blocked by default and definitely not be part of a standard SaaS offering.

Government cybersecurity agencies should provide guidance on the dangers of FTP with suggestions for alternative services. As noted in the **Exposure** section, there are unauthenticated RCE vulnerabilities in a sizable population of internet-facing FTP systems, making them choice targets for attackers and an inherent weakness in public infrastructure of each country with exposed FTP services.

FTP/S (TCP/990)

Discovery Details

FTP/S (the TCP/990 variant) dates back to 1997 and was intended to resolve the outstanding security issues with the FTP protocol[29] described in the previous section. It is, in essence, a somewhat more secure version of FTP. Unfortunately, it only makes logins safer, as file transfers are still performed in cleartext and without any sort of protection from eavesdropping or modification in transit. One other benefit is that FTP/S servers can assert their identities with cryptographic certainty—much like how HTTPS servers can—but this assertion relies on clients actually checking the certificate details and doing the right thing when certificate identification fails.

Given that FTP/S is almost exactly like its insecure cousin but safer to use, you'd think there would be widespread adoption of it. Alas, you would be wrong. Project Sonar barely found half a million nodes on the public internet compared to millions of plain ol' FTP servers. A possible big reason for this is the need to generate, install, and periodically update TLS certificates, a process for FTP/S that is not as simple as it is these days for HTTPS.[30]

There are high concentrations of FTP/S in the U.S. (mostly due to heavy prevalence in Azure, which we'll cover in a bit), but also in Poland due to the home.pl hosting provider, which goes out of its way[31] to get you to use it over vanilla FTP—a laudable documentation effort we would love to see other cloud providers strive toward.

Unfortunately, home.pl didn't expose sufficient nodes to make it into the top 10 cloud providers we used across all studies in this report, but they're doing the right thing, as is Microsoft. Along with the aforementioned FTP documentation, they also have dedicated FTP/S setup and use instructions,[32] and it's fairly trivial to enable FTP/S in IIS, as the table in the **Exposure** section hints at.

For all this effort, though, users are likely using SCP instead of FTP/S, since SCP does, in fact, encrypt the contents of files in transit as well as login credentials.

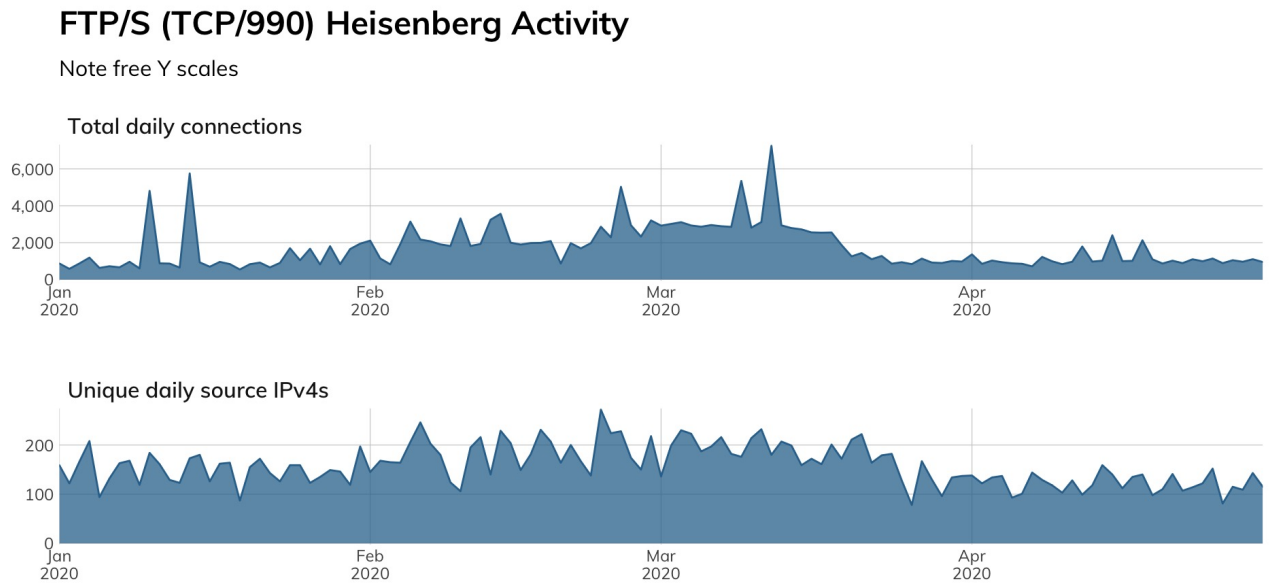
Exposure Information

Recog was only able to fingerprint 84,607 (18%) of the available FTP/S services, but IIS "wins" this time, as there are explicit configuration parameters available for FTP/S and there is a GUI for requesting and installing certificates.

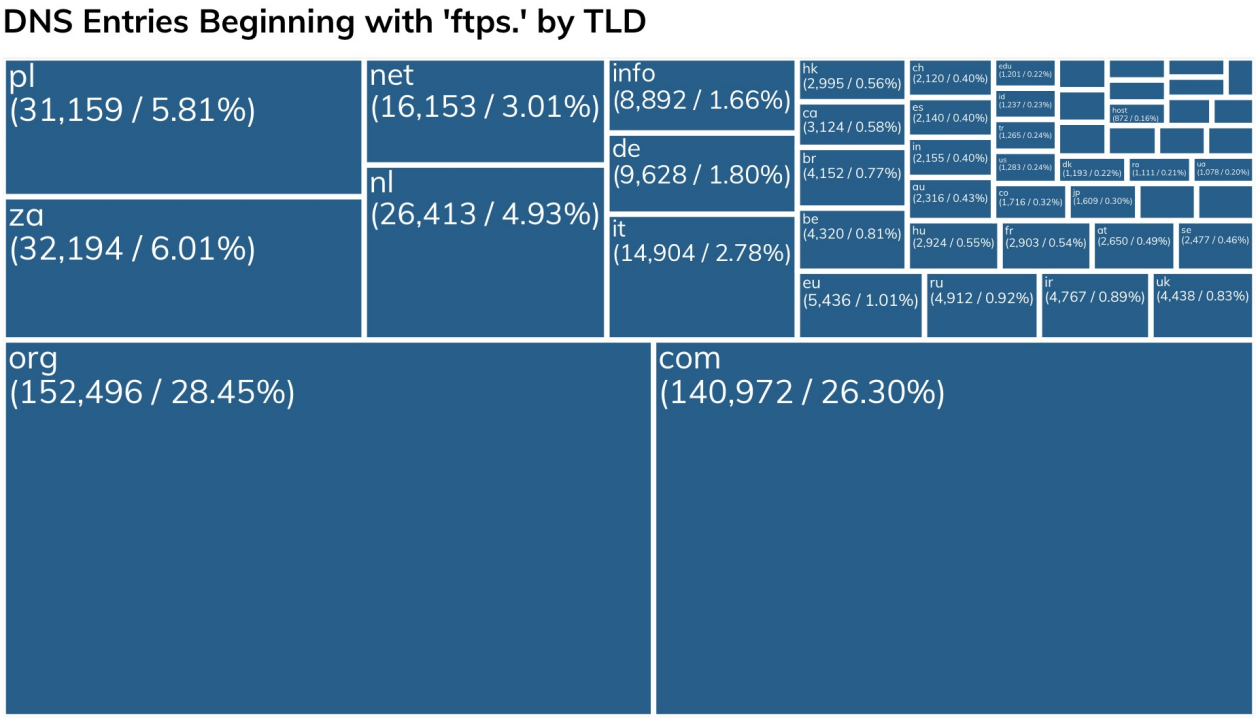
Since many FTP/S servers are both FTP and FTP/S servers, the same vulnerabilities apply. We'll be giving IIS a full workout in the **Web Servers** section later on in the report.

Attacker's View

FTP/S is not used much, and that's generally a sign it's not going to get much love from either attackers or researchers, a posit that our Project Heisenberg activity reinforces (like FTP, we do not have a high-interaction honeypot for FTP/S, so raw total and unique time series counts are used as a proxy for attacker and researcher activity):



Besides exposing FTP/S on the default port, attackers will also have no trouble finding these systems by DNS entries, given just how many entries there are for ftps.….tld in public DNS:



Our Advice

IT and IT security teams should follow the advice given in the FTP section, since most of it still applies. If you are *always* sending pre-encrypted files or the content you are uploading/downloading is not sensitive (and, you're not worried about person-in-the-middle snooping or alteration), you may be able to get away with using FTP/S a bit longer.

Cloud providers should follow home.pl's example and be explicit about what FTP/S can and cannot do for file transfer safety. While there may be some systems that are reliant on FTP/S, cloud providers should be actively promoting total solutions, like SCP and rsync, over SSH.

Government cybersecurity agencies should provide guidance on what FTP/S can and cannot do for file transfer safety and provide suggestions for alternative services. As noted in the **FTP Exposure** section (which applies also to FTP/S), there are unauthenticated RCE vulnerabilities in this small-ish population of internet-facing FTP/S systems, making them available targets for attackers and an inherent weakness in public infrastructure of each country with exposed FTP/S services.

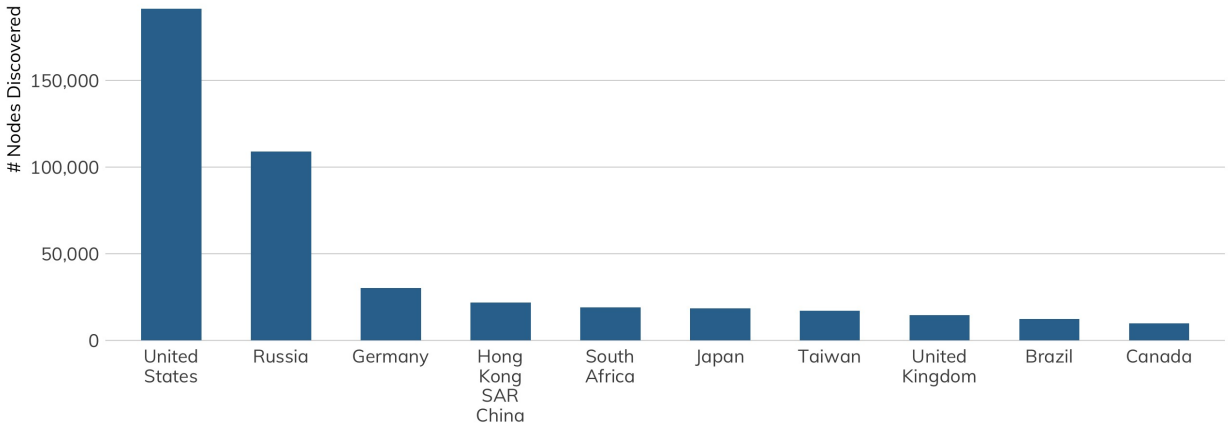
SMB (TCP/445)

Choosy worms choose SMB.

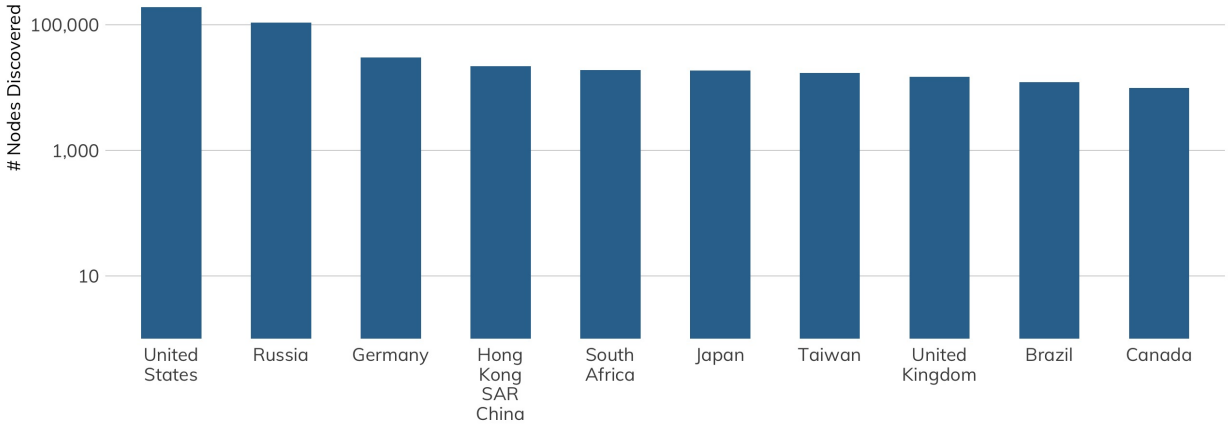
Discovery Details

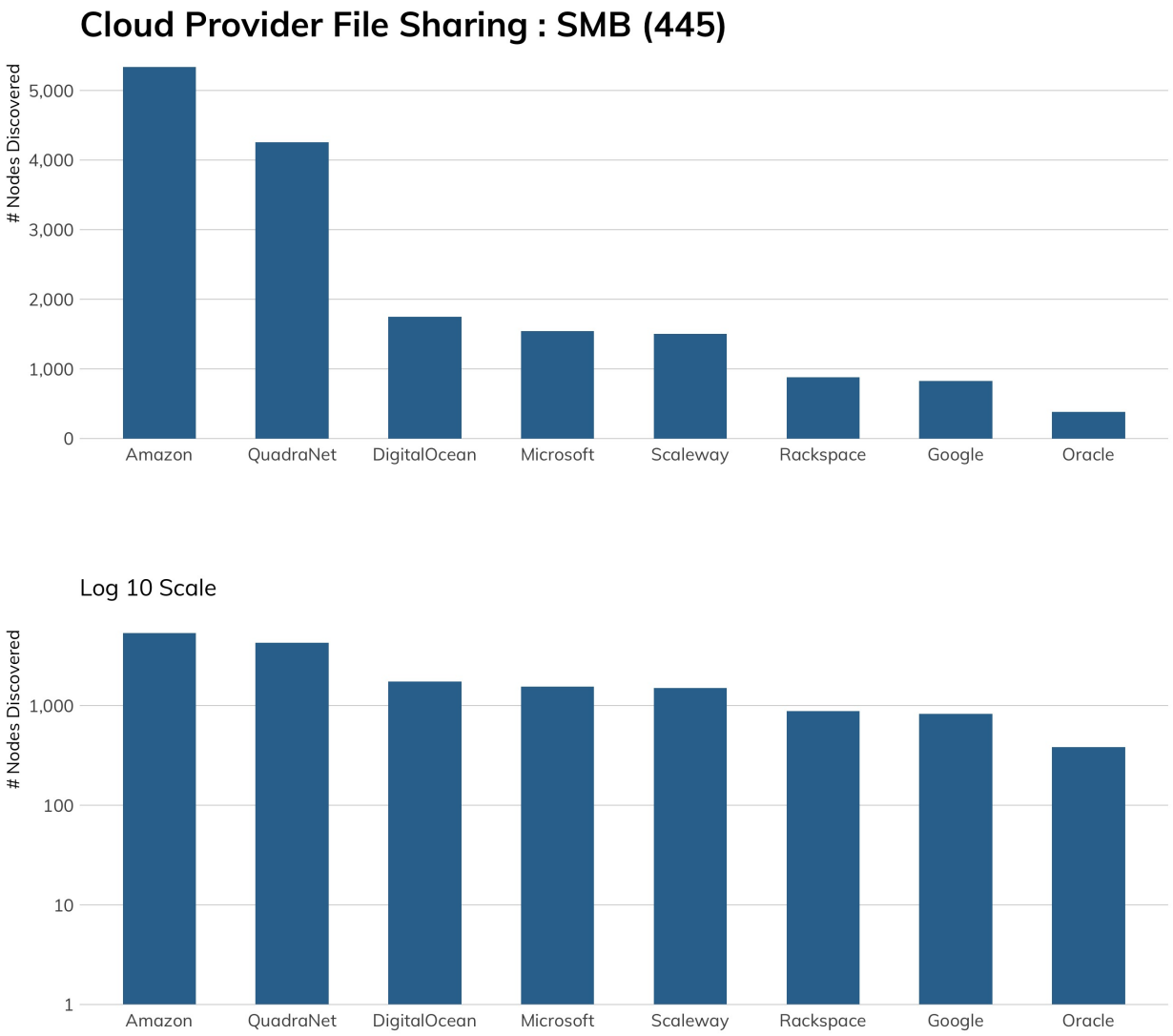
SMB is a continued source of heartache and headaches for network operators the world over. Originally designed to operate on local area network protocols like NetBEUI and IPX/SPX, SMBv1 was ported to the regular TCP/IP network that the rest of the internet runs on. Since then, SMBv2 and SMBv3 have been released. While SMB is primarily associated with Windows-based computers for authentication, file sharing, print services, and process control, SMB is also maintained for non-Windows operating systems in implementations such as Samba and Netsmb. As a binary protocol with negotiable encryption capabilities, it is a complex protocol. This complexity, along with its initial proprietary nature and deep couplings with the operating system kernel, makes it an ideal field for discovering security vulnerabilities that can enable remote code execution (RCE). On top of this, the global popularity of Windows as a desktop operating system ensures it remains a popular target for bug hunters and exploiters alike.

Top 10 Countries for File Sharing : SMB (445)



Log 10 Scale





Exposure Information

Many of the most famous vulnerabilities, exploits, and in-the-wild worms have leveraged SMB in some way. WannaCry and NotPetya are two of the most recent events that centered on SMB, both for exploitation and for transmission. Prior SMB-based attacks include the Nachi and Blaster worms (2003–2005), and future SMB-based attacks will likely include SMBGhost.[34] In addition to bugs, intended features of SMB—notably, automatic hash-passing—make it an ideal mechanism to steal password hashes from unsuspecting victims, and SMB shares (network-exposed directories of files) continue to be accidentally exposed to the internet via server mismanagement and too-easy-to-use network-attached storage (NAS) devices.

As expected, the preponderance of SMB services available on the internet are Windows-based, but the table below shows there is also a sizable minority of non-Windows SMB available.

Table 17

As you can see, these non-Windows nodes are typically some type of NAS system used in otherwise largely Windows environments, and are responsible for maintaining nearline backup systems. While these devices are unlikely to be vulnerable to exactly the same exploits that dog Windows systems, the mere fact that these **backups are exposed to the internet** means that, eventually, these network operators are Going To Have A Bad Time if and when they get hit by the next wave of ransomware attacks.

Unattended Installs

Of the Windows machines exposed to the internet, we can learn a little about their provenance from the Workgroup strings that we're able to see from Sonar scanning. The list below indicates that the vast majority of these machines are using the default WORKGROUP workgroup, with others being automatically generated as part of a standard, unattended installation. In a magical world where SMB is both rare and safe to expose to the internet, we would expect those machines to be manually configured and routinely patched.

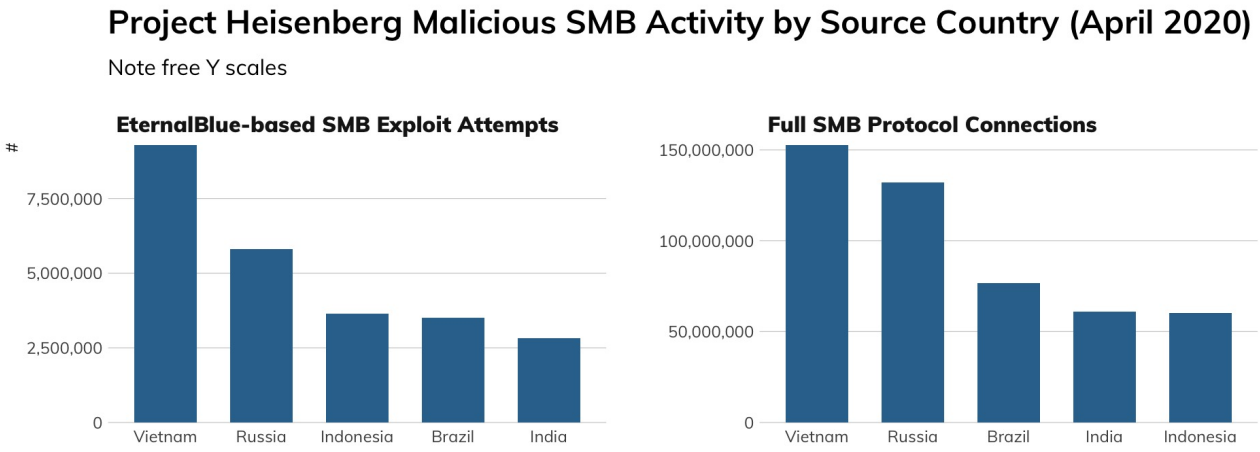
This is not the case, though—these Windows operating systems were very likely installed and configured automatically, with no special care given to their exposed-to-the-internet status, so the exposure is almost certainly accidental and not serving some special, critical business function. Additionally, these aftermarket-default WORKGROUPS are also giving away hints about which specific Windows- or Samba-based build is being used in production environments, and can give attackers ideas about targeting those systems.

Attacker’s View

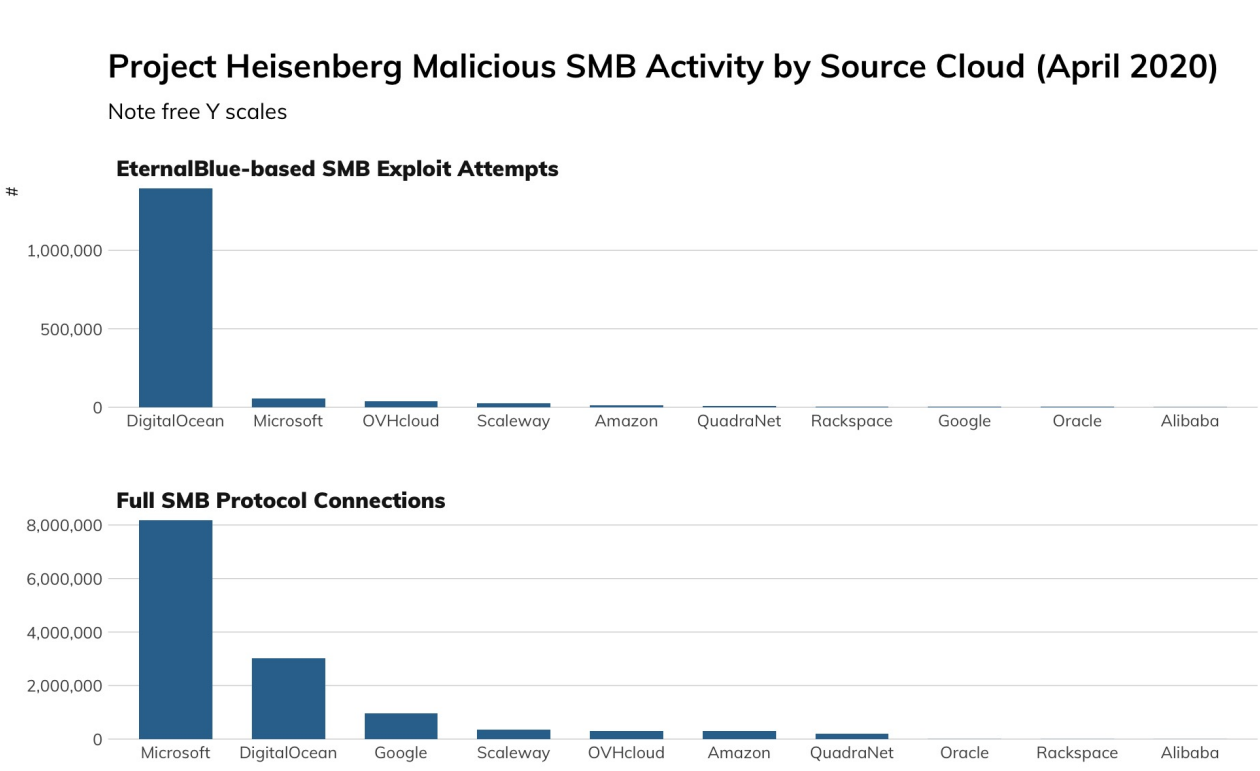
Regardless of the version and configuration of cryptographic and other security controls, SMB is inappropriate for today's internet. It is too complex to secure reliably, and critical vulnerabilities that are attractive to criminal exploitation continue to surface in the protocol. With that said, SMB continues to be a critical networking layer in office environments of any size, and since it’s native to TCP/IP, network misconfigurations can inadvertently expose SMB-based resources directly to the internet. Every organization should be continually testing its network ingress and egress filters for SMB traffic—not only to prevent outsiders from sending SMB traffic to your accidentally exposed resources, but to prevent internal users from accidentally leaking SMB authentication traffic out into the world.

Approximately 640,000 unique IP addresses visited our high-interaction SMB honeypots over the measured period, but rather than think of this as a horde of SMB criminals, we should recall

that the vast majority of those connections are from machines on the internet that were, themselves, compromised. After all, that's how worms work. Very few of these connections were likely sourced from an attacker's personally owned (rather than pwned) machine. With this in mind, our honeypot traffic gives us a pretty good idea of which countries are, today, most exposed to the next SMB-based mega-worm like WannaCry: Vietnam, Russia, Indonesia, Brazil, and India are all at the top of this list.



Among the cloud providers, things are more stark. EternalBlue, the exploit underpinning WannaCry, was responsible for about 1.5 million connections to our honeypots from Digital Ocean, while Microsoft Azure was the source of about 8 million (non-EternalBlue) connections (of which, about 15%, or 1.2 million or so, were accidental connections due to a misconfiguration at Azure). We're not yet sure why this wild discrepancy in attack traffic versus accidental traffic exists between Digital Ocean and Azure, but we suspect that Microsoft is much more aggressive about making sure the default offerings at Azure are patched against MS17-010, while Digital Ocean appears to be more hands-off about patch enforcement, leaving routine maintenance to its user base.



Our Advice

IT and IT security teams should prohibit SMB access to, or from, their organization over anything but VPN-connected networks, and regularly scan their known, externally facing IP address space for misconfigured SMB servers.

Cloud providers should prohibit SMB access to cloud resources, and at the very least, routinely scrutinize SMB access to outside resources. Given that approximately 15% of our inbound honeypot connections over SMB from Microsoft Azure are actually misconfigurations, rather than attacks or research probes, Azure should be especially aware of this common flaw and make it difficult to impossible to accidentally expose SMB at the levels that are evident today.

Government cybersecurity agencies should be acutely aware of their own national exposure to SMB, and institute routine scanning and notification programs to shut down SMB access wherever it surfaces. This is especially true for those countries that are at the top of our honeypot source list.

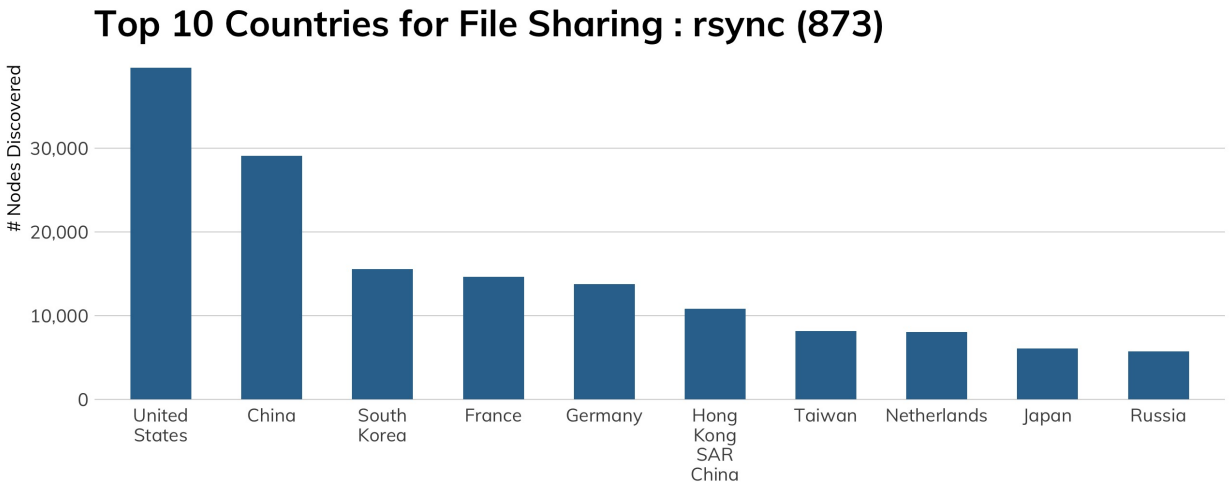
rsync (873)

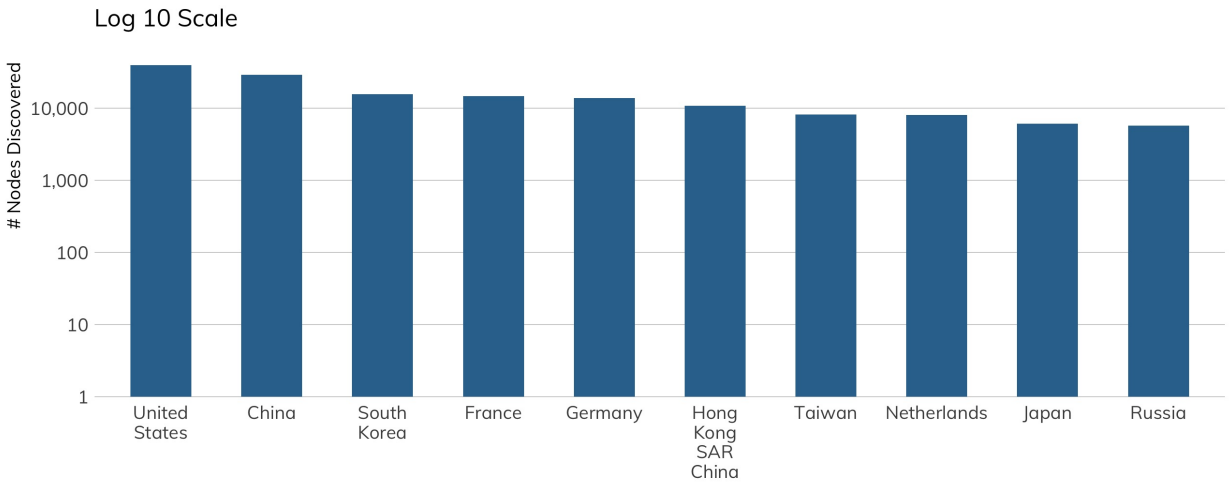
Almost an accident of early internet engineering.

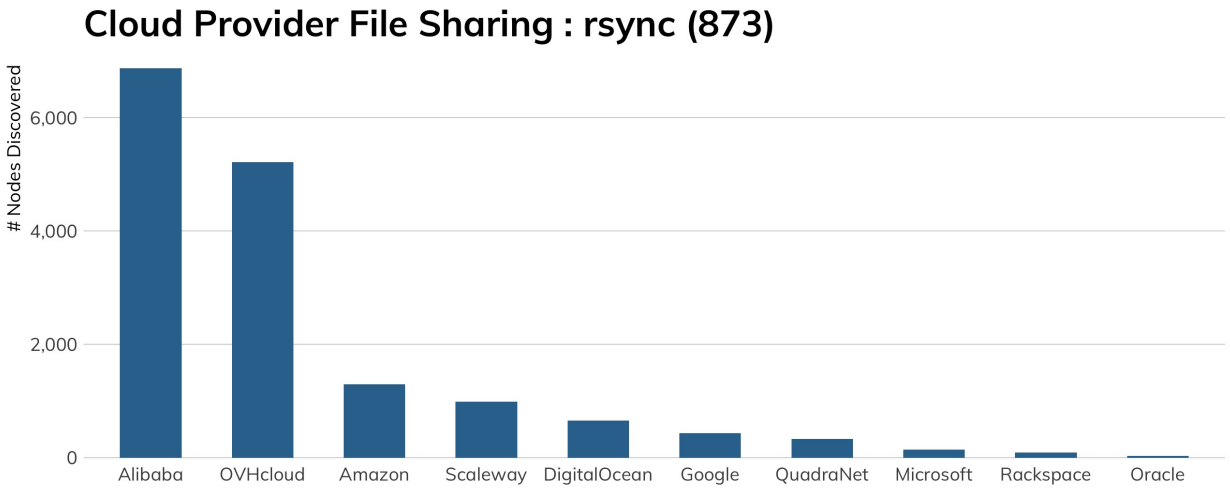
Discovery Details

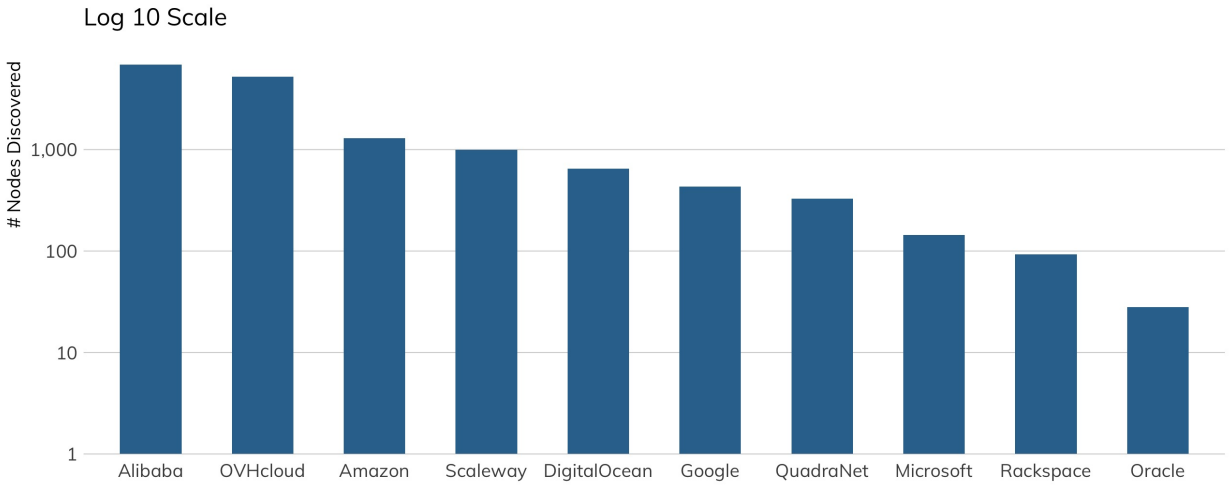
The rsync service is almost old enough to rent a car in the U.S., given that it turned 24 this year. Unlike many network protocols, rsync has no IETF RFC for the protocol itself, but when uniform resource identifiers (URI, but you can think of them as URLs and we won't judge you) became a thing, this venerable protocol achieved at least partial RFC status for its URI scheme.[35] It does have documentation[36] and is widely used on the modern internet for keeping software and operating system mirrors populated, populating filesystems, performing backups, and—as we'll see later—exposing a non-insubstantial number of home network-attached storage (NAS).

Rapid7's Jon Hart and Shan Sikdar took an in-depth look at rsync exposure back in 2018,[37] and only a tiny bit has changed since then, so we'll focus more on the changes than reinvent the wheel.









If you're wondering why Alibaba of China and OVH of France both have a more significant rsync presence than the other providers, look no further than their documentation[38] and services[39], which most folks new to "the cloud" will rely on to get jump-started. Unfortunately, both clouds default to plain ol' rsync but *do* at least mention that you can run it more securely through an SSH tunnel.

Exposure Information

Over half (~57%) of the exposed rsync systems are running with a 10-year-old protocol (version 30), which is an indicator the operating systems are potentially over 10 years old and riddled with other vulnerabilities. While protocol version 31 is most current, this can be a bit misleading. Version 31 was released in 2013. Outside of versions 30 and 31 of the protocol, there is a steep drop-off to the truly ancient versions of the protocol.

We need to deviate a bit from some of the previous sections, since over 20% of rsync exposure lies in residential ISP space (top 25 providers listed in Table 20). Most of these are NAS devices with brand names you've likely seen in online stores or big-box electronic retailers. We'll see why this is important in the next section.

Table 20

The core rsync service has no encryption, so all operations—including authentication—operate over cleartext. This makes it just as bad as Telnet and FTP. The protocol has been extended to support using non-cleartext credentials, but all file transfers still happen in cleartext, so any prying eyes in the right network position can still grab your data.

One of the real dangers of exposing rsync, though, is that you're letting attackers know you have files for the taking and have some operating system that may be worth taking over.

Attacker's View

Since rsync is a big, neon sign saying "I've got files!" and since a large portion of exposed rsync is on residential ISP networks, attackers can correlate other services running on those IP addresses to get an idea of the type of system that's being exposed.

Why are these residential rsync systems sitting on the internet? Vendors such as QNAP need to make these NAS devices easy to use, so they create services such as "myQNAPcloud," which lets consumers create a "nameyouwant.myqnapcloud.com" to access their devices over the internet (meaning they have to punch at least one hole in their home router to do so). It even has handy mobile apps that let them watch saved videos and listen to saved music. Our Sonar FDNS study found over 125,000 myqnapcloud.com entries, and it is super easy for attackers to collect that data as well. That means over 125,000 QNAP NAS device users just gave attackers their home (IP) addresses and placed a quaint "Welcome" mat out for them.

Unfortunately, QNAP (and a cadre of other NAS vendors) have a terrible track record when it comes to vulnerabilities, including seven unauthenticated, remote code execution vulnerabilities, the most recent of which is a doozy.[41] While these vulnerabilities are not exposed *over* rsync, the presence of rsync is, as previously noted, a leading indicator that a NAS device is very likely present.

Attackers use these devices as launch points for other malicious activities on the internet (such as DDoS attacks) and can lock up all the files on these devices for consumer-grade ransomware attacks.

As you can see, exposing one innocent service such as rsync can ultimately get you into a world of trouble.

Our Heisenberg honeypot fleet does not have a high-interaction honeypot for rsync, but we do watch for all connection attempts on TCP/873 and see regular inventory scans (the blue spikes in the figure below) by researchers and attackers looking for new rsync endpoints to conquer.

Our Advice

IT and IT security teams should *never* use vanilla rsync and should always opt to wrap it in a tasty chocolate secure shell (i.e., only use it when tunnelling over certificate-based, authenticated SSH sessions). There is no other safe way to use rsync for confidential information. None.

Cloud providers should know better than to offer rsync services over anything but certificate-based, authenticated SSH. Existing insecure services should be phased out as soon as possible. The most obvious path toward this brave new world of encryption everywhere is to offer excellent, well-maintained, easy-to-find documentation, with examples, on how exactly to "get started" with rsync over SSH, and on the flip side, little to no documentation on how to run rsync naked (users who are determined to do the wrong thing can fight through man pages like we used to in the old days).

Government cybersecurity agencies should regularly and strongly encourage consumers, cloud providers, and organizations to only use certificate-based, authenticated SSH-tunnelled rsync services and work with cloud providers and NAS vendors to eradicate the scourge of rsync exposure from those platforms.

Email, in its currently recognizable form, emerged as the SMTP standard for transferring messages between networks and computers and was enshrined in RFC 788 in November 1981. Email had been in use for at least a decade earlier, but protocols and applications differed wildly among systems and networks. It took SMTP to unify and universalize email, and it continues to evolve today.

On the client side, we've seen shifting habits over time. The POP (Post Office Protocol) and IMAP (Internet Message Access Protocol) families of client-to-server protocols dominated in the 1990s. When the world moved to the World Wide Web, there was a brief decline in their use when people turned to webmail, rather than stand-alone mail user agents (MUAs). This trend reversed in the late 2000s with the rise of Apple and Android mobile devices, which tend to use IMAPv4 to interact with email servers.

SMTP (25/465/587)

The “Simple” in SMTP is intended to be ironic.

Discovery Details

While SMTP is traditionally cleartext with an optional secure protocol negotiation called STARTTLS, we're seeing more SSL-wrapped SMTP, also known as SMTPS, in the world today. The following charts and tables illustrate the distribution of SMTP over port 25, SMTP on port 587 (which is intended for SMTP-to-SMTP relaying of messages), and SMTPS on port 465.

Encrypted vs Unencrypted Mail Services by Country (Top 10)

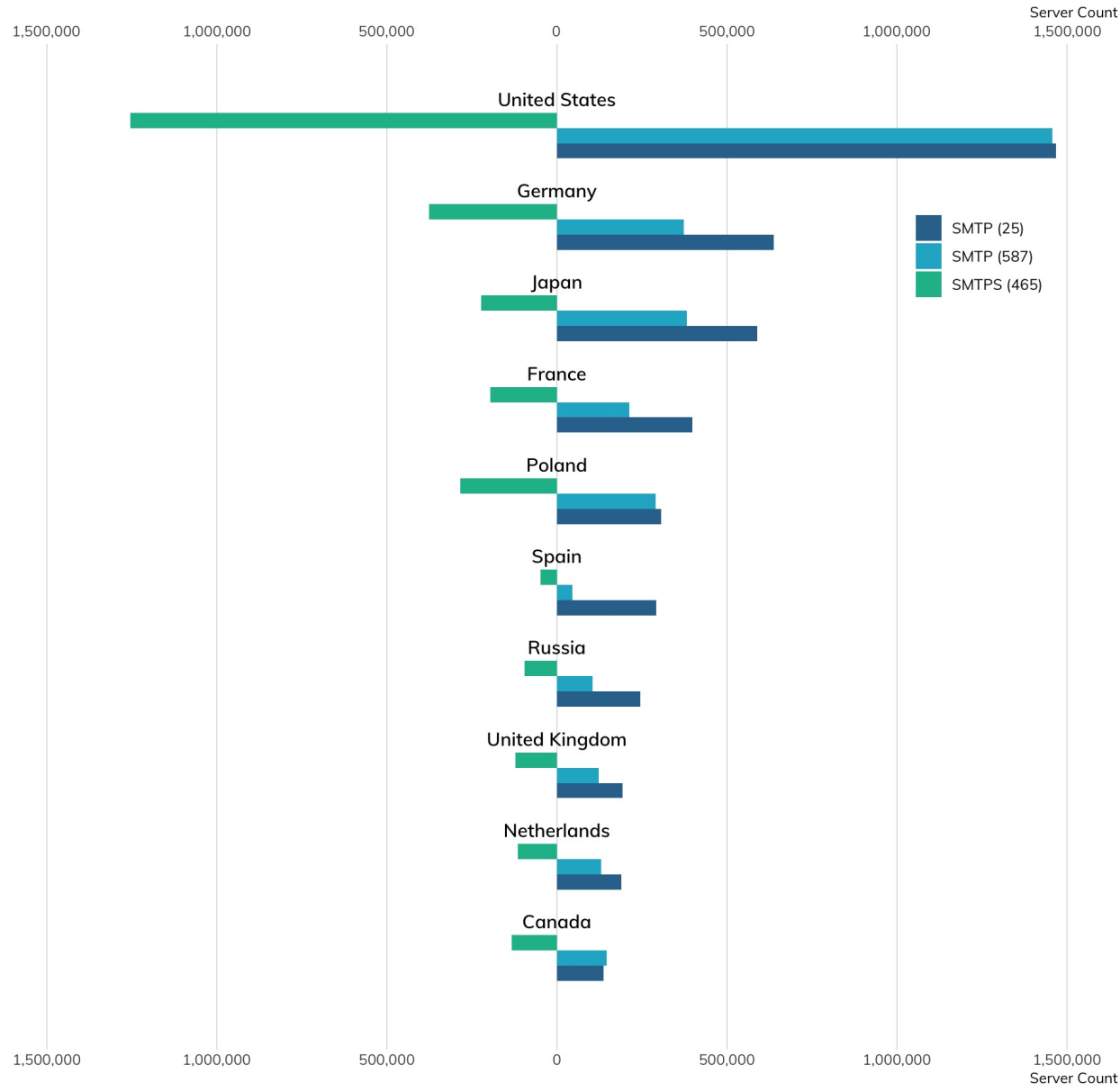


Table 21

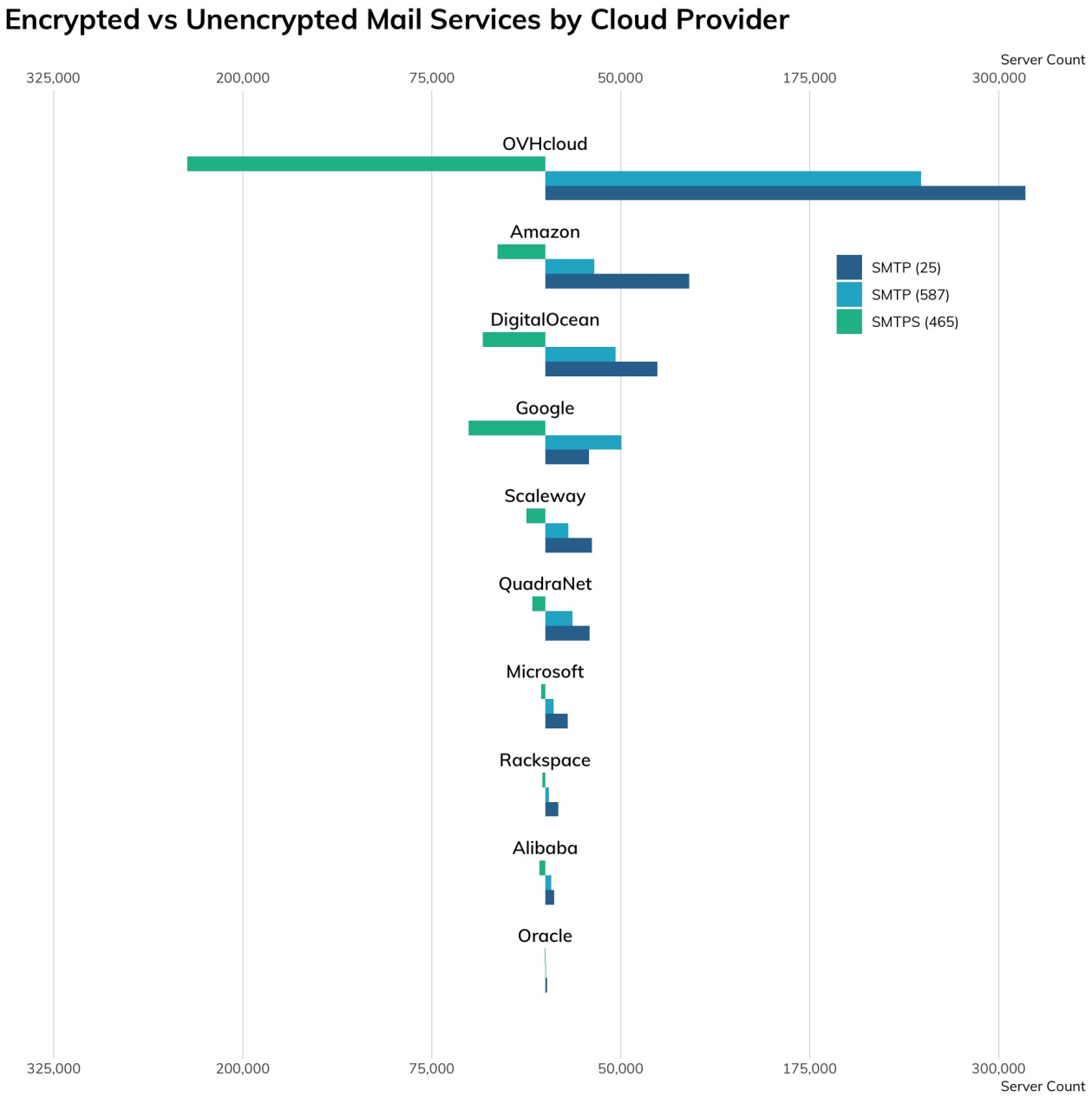


Table 22

As far as top-level domains are concerned, we see that the vast majority of SMTP lives in dot-com land—we counted over 100 million MX records in dot-com registrations, with a sharp drop-off in dot-de, dot-net, and dot-org, with about 10 million MX records in each.

Exposure Information

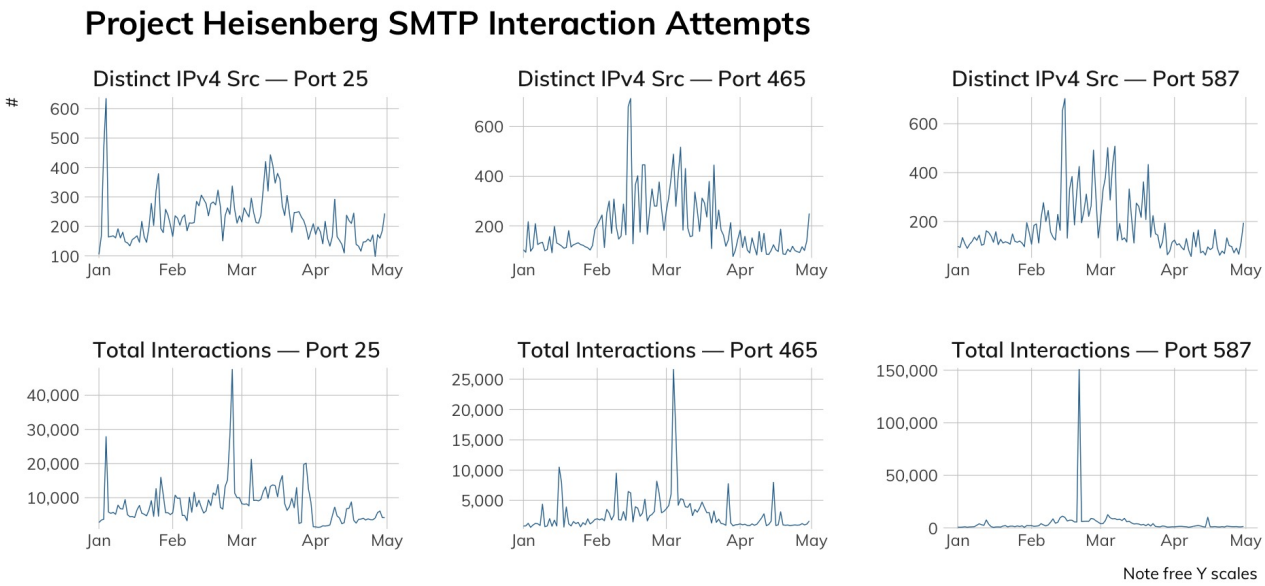
There are dozens of SMTP servers to choose from, each with their own idiosyncratic methods of configuration, spam filtering, and security. The top SMTP server we're able to fingerprint is Postfix, with over a million and a half installs, followed by Exim, Exchange, and trusty Sendmail. In Table 23 is the complete list of every SMTP server we positively identified—mail

administrators will recognize the vestiges of old, little-used mail servers, such as the venerable Lotus Domino and ZMailer. If these are your mail servers, think long and hard about why you're still running these as opposed to simply farming this thankless chore out to a dedicated mail service provider.

Finally, let's take a quick look at the Exim mail server. Like most other popular software on the internet, we can find all sorts of versions. Unlike other popular software, Exim versioning moves pretty quickly—the current version of Exim at the time of scanning was v 4.93, and has already incremented to 4.94 by the time of publication. However, the popularity of the latest version (4.93) versus next-to-latest (4.92.x) is in the 100,000 range, and given the intense scrutiny afforded to Exim by national intelligence agencies, this delta can be pretty troubling. It's so troubling that the American National Security Agency issued an advisory urging Exim administrators to patch and upgrade as soon as possible to avoid exploitation by the "Sandworm team."[42] Specifically, the vulnerability exploited was CVE-2019-10149, and affects versions 4.87 through 4.91—as of the time of our scans, we found approximately 87,500 such servers exposed to the internet. While this is about a fifth of all Exim servers out there, exposed vulnerabilities in mail servers tend to shoot to the top of any list of "must patch now" vulns.

Attacker's View

Given the high value attackers tend to assign to SMTP vulnerabilities, it's no surprise that we see fairly consistent scanning action among threat actors in our SMTP honeypots.



Our Advice

IT and IT security teams should seriously consider converting over to an established email provider such as Microsoft's Office 365 or Google's G Suite. Running your own email remains one of the more truly painful network administration tasks, since outages, patch management, and redundant backups can be tricky even in the best of times, to say nothing of the constant drain of resources in the fight against spam and phishing. Established providers in this space have a proven track record of handling both spam and phishing, as well as achieving remarkable uptimes.

Cloud providers should provide rock-solid documentation on how to set up SMTP services for their customers, starting with SSL-wrapped SMTP as a default configuration. This is one case where we wouldn't be opposed to providers such as Microsoft and Google inserting a little adver-docu-tizing[43] pushing customers over to their hosted mail solutions.

Government cybersecurity agencies should recognize that everyone is challenged by running merely serviceable email infrastructure, and very few organizations are truly excellent at it at any reasonable scale. As far as content-based attacks are concerned, these experts should continue pressing for minimum technical defenses, such as DMARC, and user education in recognizing and avoiding phishing scams.

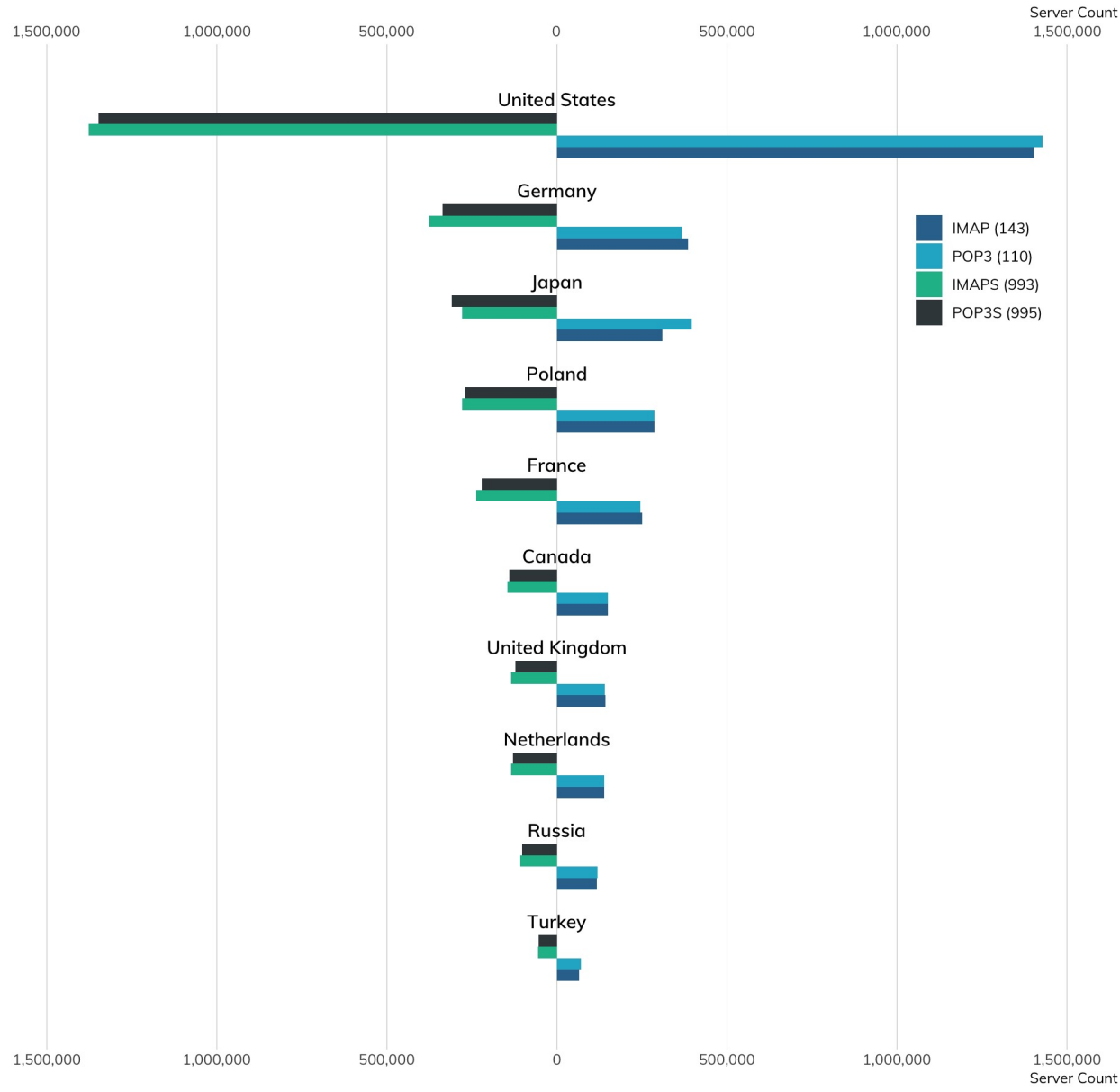
IMAP (143/993) and POP (110/995)

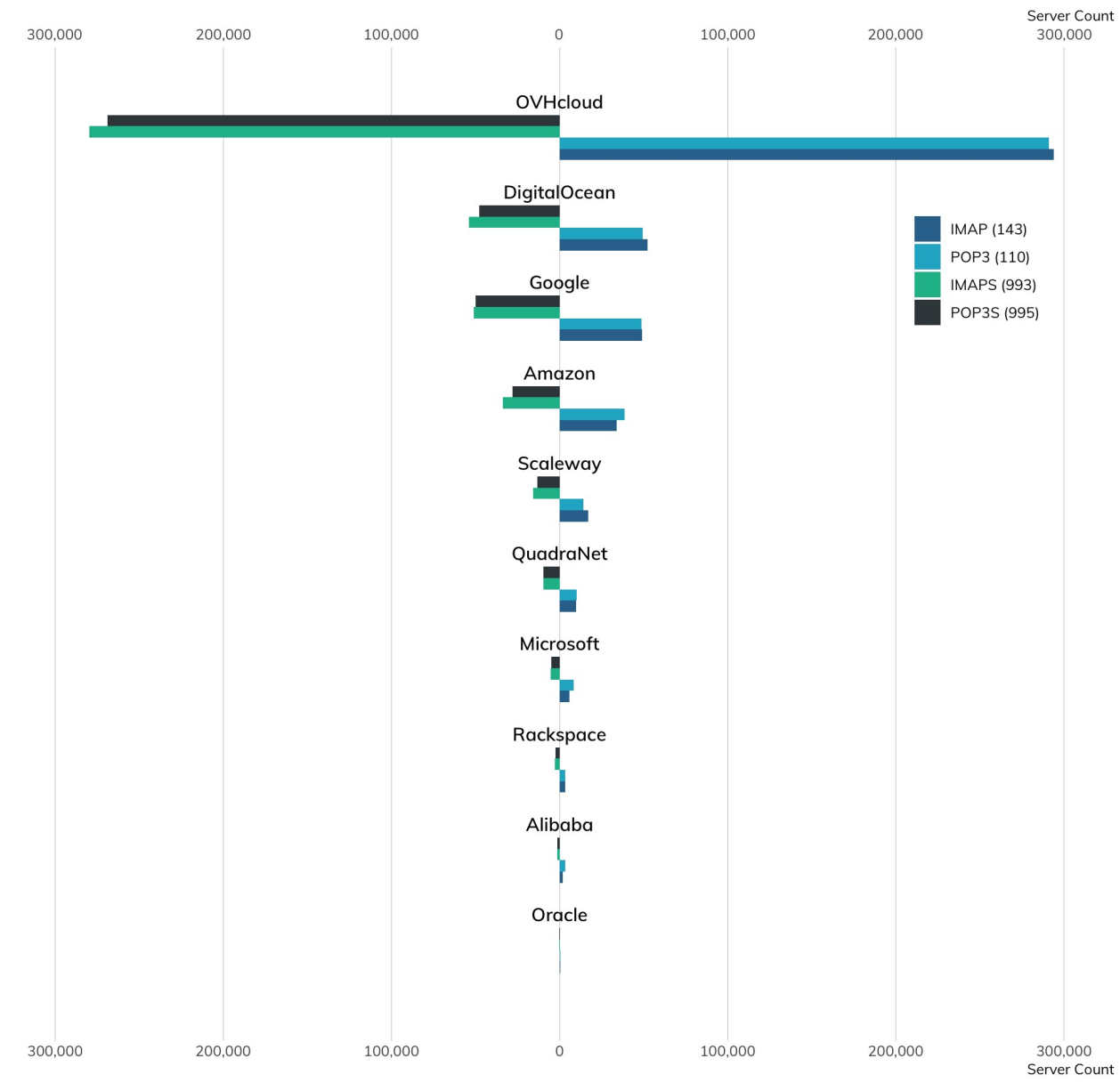
Hey, only 55% of email is technically considered spam

Discovery Details

SMTP handles mail inbound to organizations, while POP and IMAP handle the action of individual users collecting that mail to read and reply to. As with SMTP, we've broken down the prevalence of cleartext versus encrypted versions of these services in the charts and tables below, both overall and by country and cloud.

Encrypted vs Unencrypted Mail Access Services by Country (Top 10)





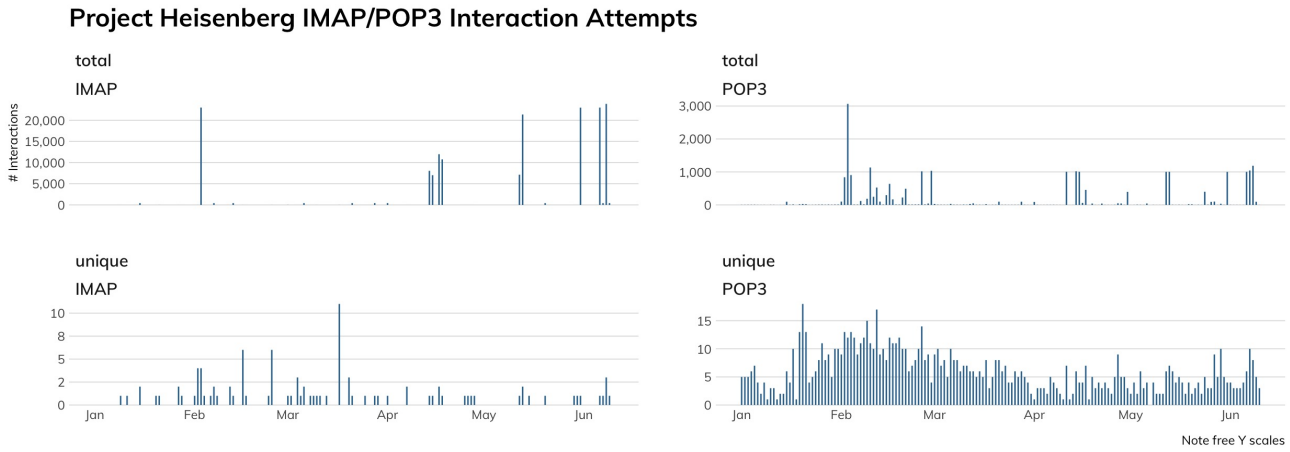
While there are fewer choices in the end-user-accessible mail protocols, we were able to fingerprint IMAP servers by vendor:

Table 28

With that said, we have very little telemetry on a per-version basis. IMAP and POP don't tend to reveal this information in a pre-authenticated way, and we don't have any good tricks in Recog to suss out versions with any sort of accuracy.

Attacker’s View

The charts below graph our honeypot connections to IMAP and POP, and while the spikes may, at first, seem like outliers, in fact, IMAP and POP probing seems to always be pretty spikey. For reasons unknown, these scans have a “high seasonality” in statistical parlance—they tend to come in bursts, rather than the usual constancy that we see in other protocol scans.

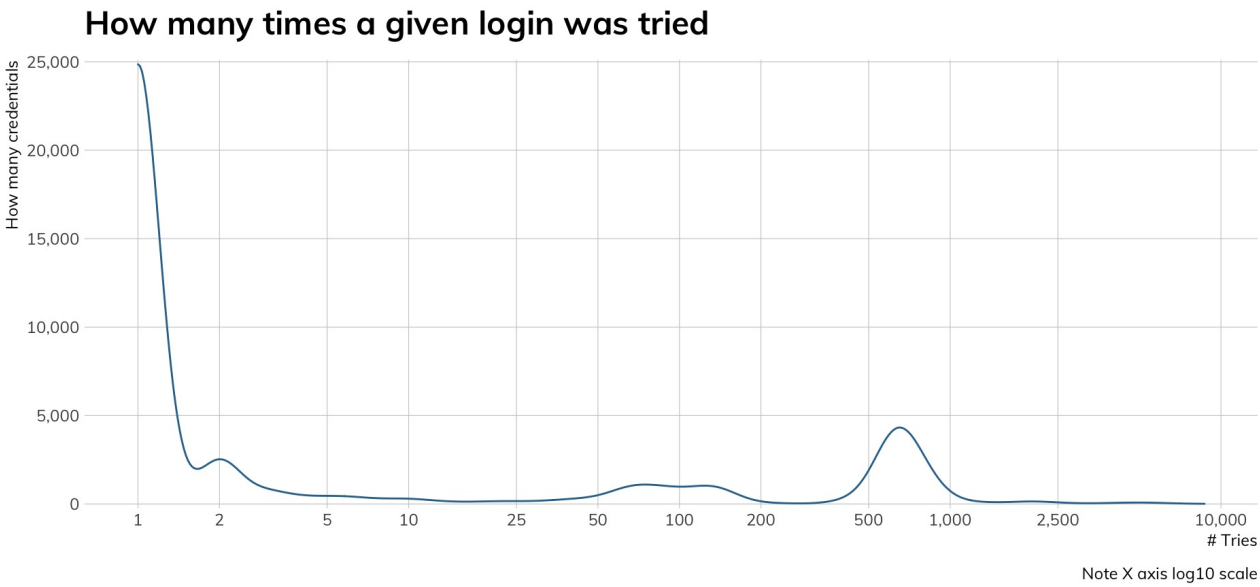


Over the course of the measured period, we saw about 7,500 unique usernames being tested for IMAP and POP. The top 20 usernames tested are:

1. admin
2. test
3. webmaster
4. backup
5. info
6. marketing
7. contact
8. support
9. office
10. sales
11. postmaster
12. mail
13. abuse
14. service
15. spam
16. master
17. helpdesk
18. mailing

19. newsletter
20. recruit

Each of these were tried between 4,000 and 8,000 times. Interestingly, on most runs, each user name is tried precisely once, but sometimes, attackers try one username with 500 to 1,000 passwords in a given attempt.



Our Advice

IT and IT security teams should routinely review the costs involved in running their own on-premises mail infrastructure, in terms of not just money, but time and expertise. If at all possible, they should see about moving off to a professionally maintained email provider, like Outlook 365 or Google G Suite (which offer TLS-backed client mail services by default), and reap the benefits of uptime assurance and spam-scrubbing being Someone Else's Problem.

Cloud providers should, similarly, steer people away from maintaining their own email infrastructure, and gently encourage customers to investigate the sane and stable alternatives. At the very least, cloud provider documentation should clearly explain the differences between POP and IMAP and why you might not need one or the other, then guide customers toward TLS-wrapped client mail services.

Government cybersecurity agencies should advocate for strong encryption alternatives to the cleartext IMAP and POP protocols, and educate the public on the fact that POP and IMAP are

often convenient backdoors to password testing, since they are rarely secured with multi-factor authentication.

We're going to say it: For many tasks, a console terminal session just doesn't cut it. While we're command-line fans as much as anyone, graphical user interfaces (GUIs) to remote desktop/server environments definitely have their place in modern IT ecosystems. Whether it be providing remote access to a traditional (i.e., non-browser-based) desktop applications, bastioning access to intranet resources, or enabling remote use of full desktop interactive sessions, these solutions meet real needs and help organizations Get Stuff Done™.

Recognizing this reality, we've taken a look at two wildly popular remote access services:

- Virtual Network Computing (VNC, TCP/5900)
- Microsoft Remote Desktop (RDP, TCP/3389)

Sorry, X-Window. You had your day. It's time to enjoy your retirement in astronomy labs and Hollywood hacking scenes.

VNC (5900 & 5901)

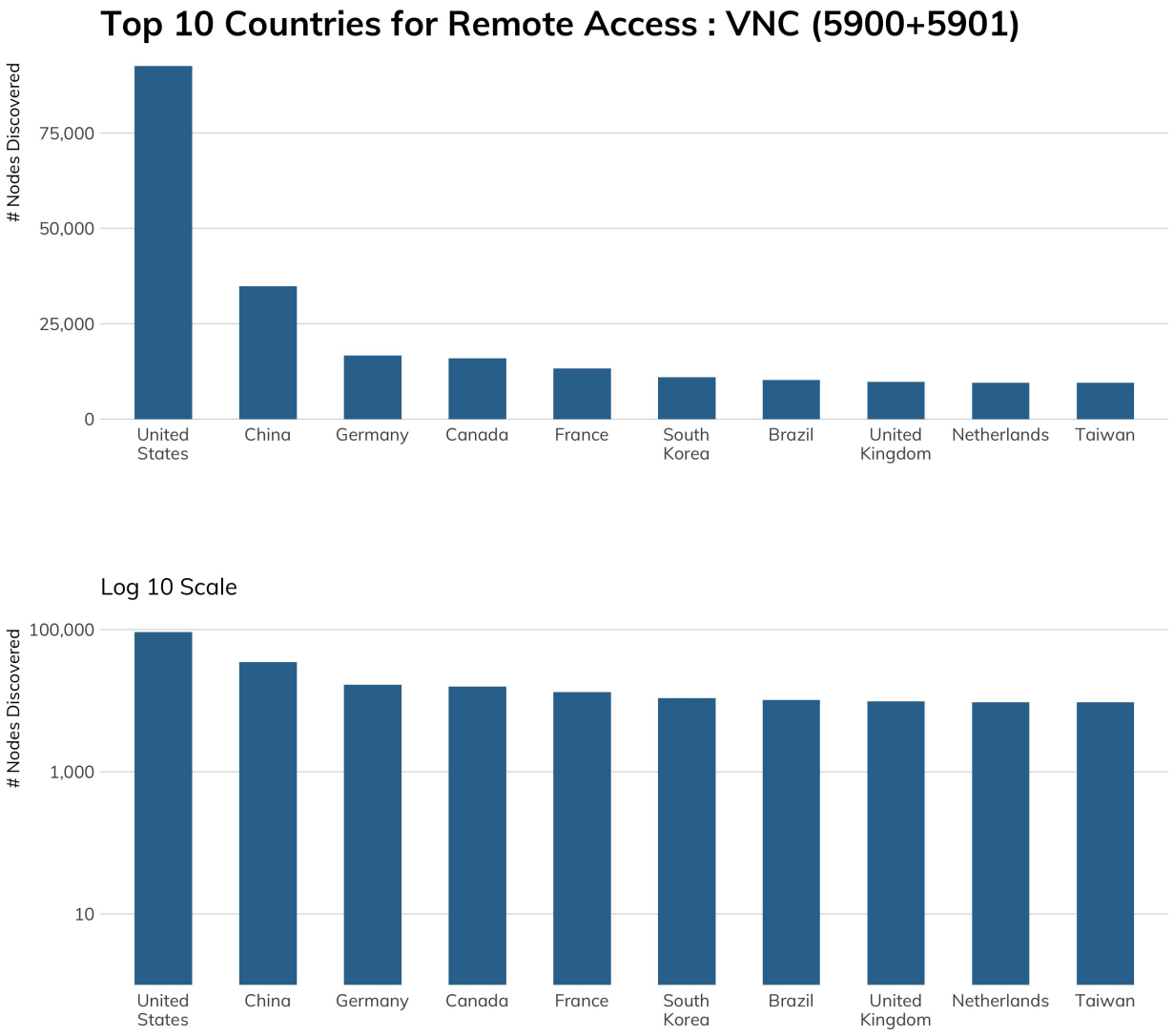
It's really RFB, but weirdly, nobody calls it that.

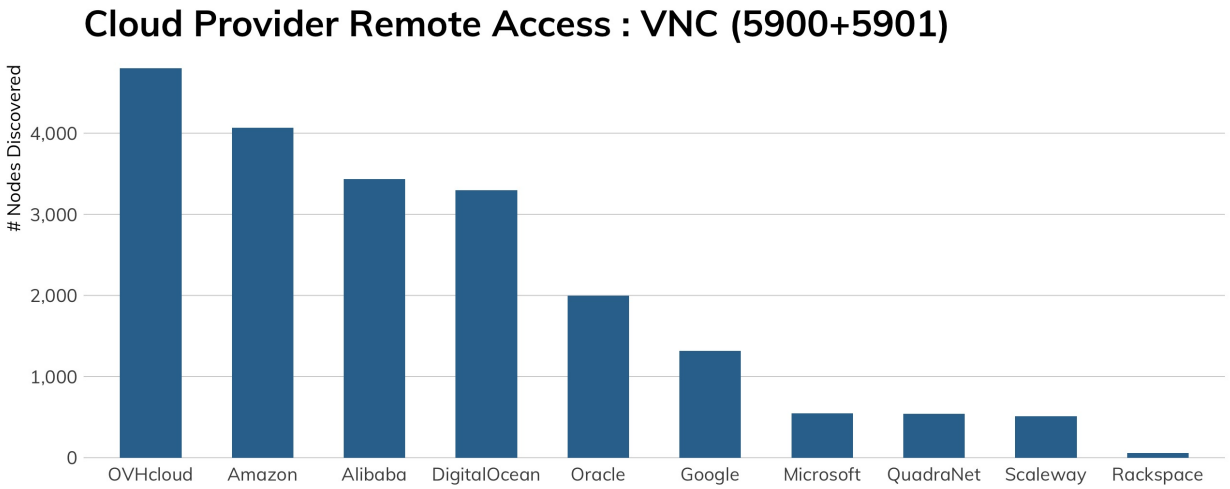
Discovery Details

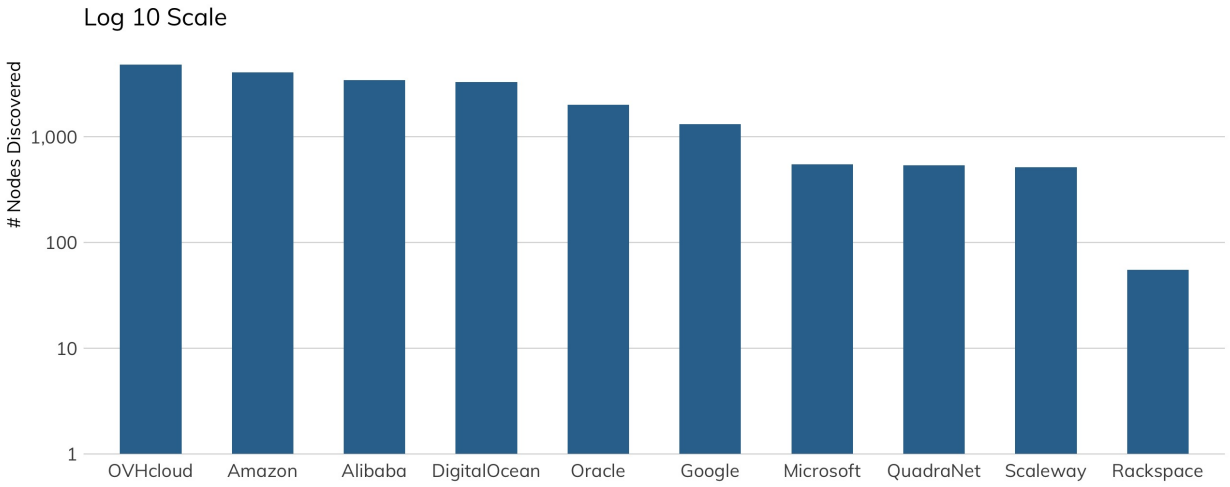
Project Sonar found 347,940 VNC servers on the default port of 5900 and popular alternate port of 5901. If you've seen larger numbers from other sources (namely, the most excellent Kaspersky summary paper[46]), those are very likely accurate, since VNC lives on many, many ports out in the wild west of the internet. For whatever reason, VNC implementations don't adhere too closely to their IANA-assigned port number as a design decision. That means you can use our numbers as an exposure baseline, but you may want to refrain from generalizing distributions by country or network *too much,* though our numbers indicate at least ~60% of VNC users tend to use the most common ports for the service.

OVH is in the lead, here, due to its "helpful" support page on "How to Access a Public Cloud Instance via VNC,"[47] which includes the step of setting a password for the root user, since that's what you log in as over VNC there. This is in violation of so many regulatory standards, it is sufficient to make our research team cry.

Amazon, too, goes out of its way to help you increase your attack surface,[48] along with Alibaba[49] and DigitalOcean.[50] This likely explains their “top four” status in the cloud exposure view.







Exposure Information

As was the case in a few other sections of this report, we need to take a moment to talk about business and residential ISP exposure for VNC, since it makes up a great deal of the total exposure. The following ISPs/providers garnered 30% of total discovered VNC nodes.

Stop us if you've heard this before, but these nodes are primarily businesses that want quick, remote access to *something* on their internal network or residential users who want to access internal home systems remotely.

Attempting to guess credentials in any way would violate the CFAA[51] here in the U.S., and our researchers do *not* look good in orange, so we cannot say how many of these systems require logins, nor can we take screenshots of them since the resultant images may contain sensitive information and, therefore, be in violation of many regional privacy regulations.

Fifty-two percent of the discovered servers are running RFB version 3.8 (the most recently published RFB standard), followed by 19% running some flavor of macOS, making this protocol one of the few where Apple-manufactured computers are likely to show up in significant numbers. We even found a few dozen *multi-function devices*[52] exposing their admin interface over VNC.

Attacker's View

There have been many bugs in all flavors of VNC over the years,[53] but the sort-of-good news is that most require an authenticated session to be effective. Having said that, there are over *8 billion credentials* floating around on the "dark web" just waiting to be tried.

Just as with other services, we do not advertise VNC services in our Heisenberg honeypot network, so the above chart represents either opportunistic scanning or organization misconfigurations (i.e., trying to hit a once-owned VNC server in AWS, Google, or Rackspace that no longer sits at the IP address they thought it did).

The connection attempt counts are non-trivial numbers, so perhaps think twice before exposing VNC to the internet, and make sure you use patched VNC server code with ne'er-before-used, strong credentials backed by multi-factor authentication.[54]

Our Advice

IT and IT security teams should consider putting all VNC access behind a VPN connection or SSH tunnel and never expose VNC directly to the internet. Too many things can go wrong in the setup, care, and feeding of hosts with VNC access to make it safe for public internet use.

Cloud providers should stop encouraging the use of bare-naked VNC and come up with more secure and cost-effective solutions for their customers, such as VNC-over-SSH. Sure, there are some VNC client/server setups that offer enhanced security, but most of the tutorials and setup guides provide instructions that only really increase the attack surface of customers.

Government cybersecurity agencies should release regular guidance on how to safely use VNC, monitor VNC vulnerability disclosures and provide timely alerts, and encourage internet service providers to at least block VNC access on the default port (5900).

Remote Desktop — RDP (3389)

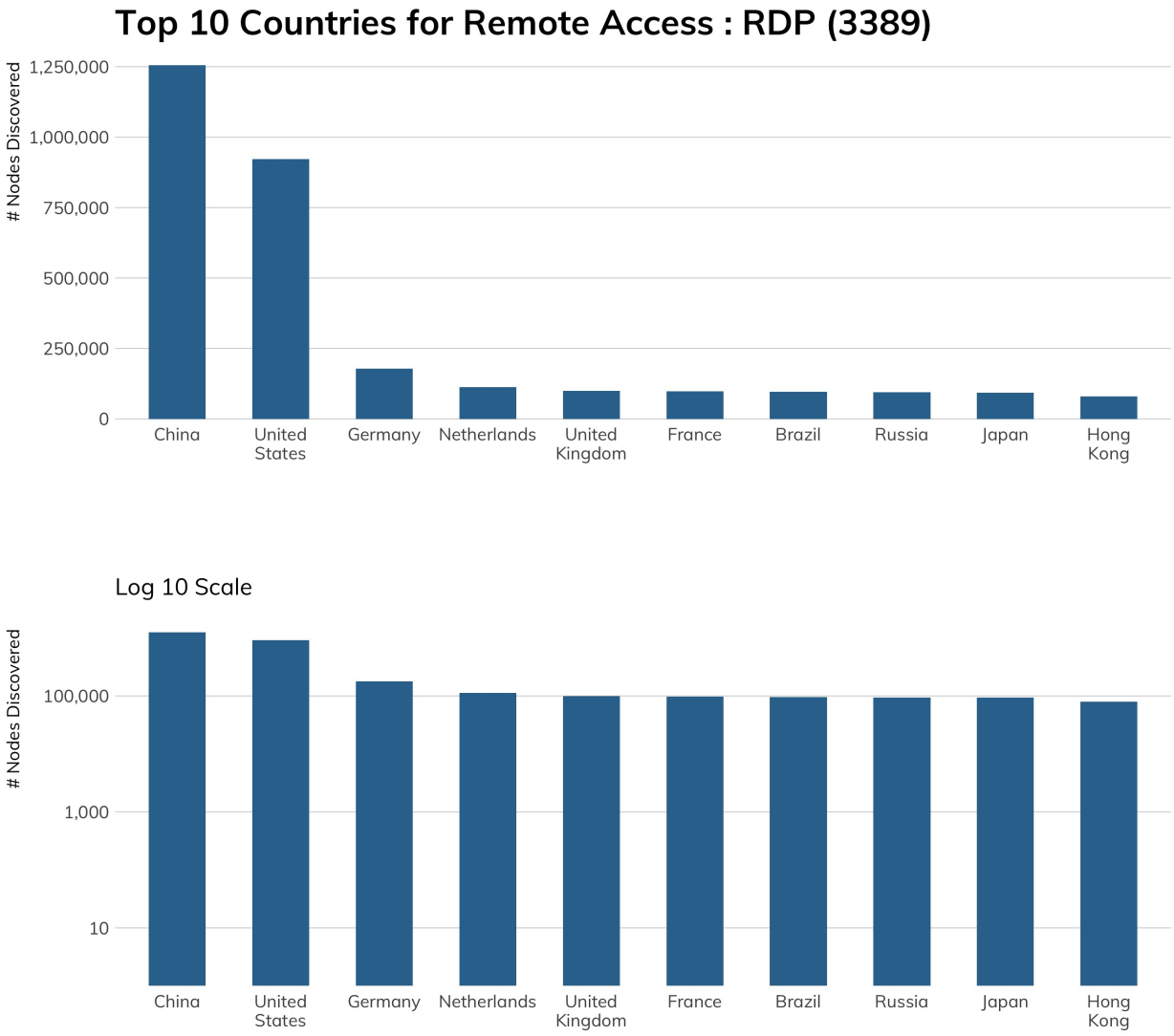
It's like VNC, but more Microsofty.

The introduction of WindowsXP (2001) begat many inventions, including widespread malware distribution, punishing data loss due to Blue Screens of Death, and a newfound ability to easily gain a keyboard-video-mouse (KVM) connection to other WindowsXP systems via a Remote Desktop Connection (back then it was called "Terminal Services," but we suppose Microsoft decided that sounded too grim). It has been in every Windows version since XP and continues to be the primary way most users access Windows systems remotely. Most cybersecurity folks use the shortcut "RDP" (Remote Desktop Protocol) when speaking about Microsoft Remote Desktop.

Discovery Details

Project Sonar found just under 4 million active RDP nodes, with over a quarter of them in China-homed networks.

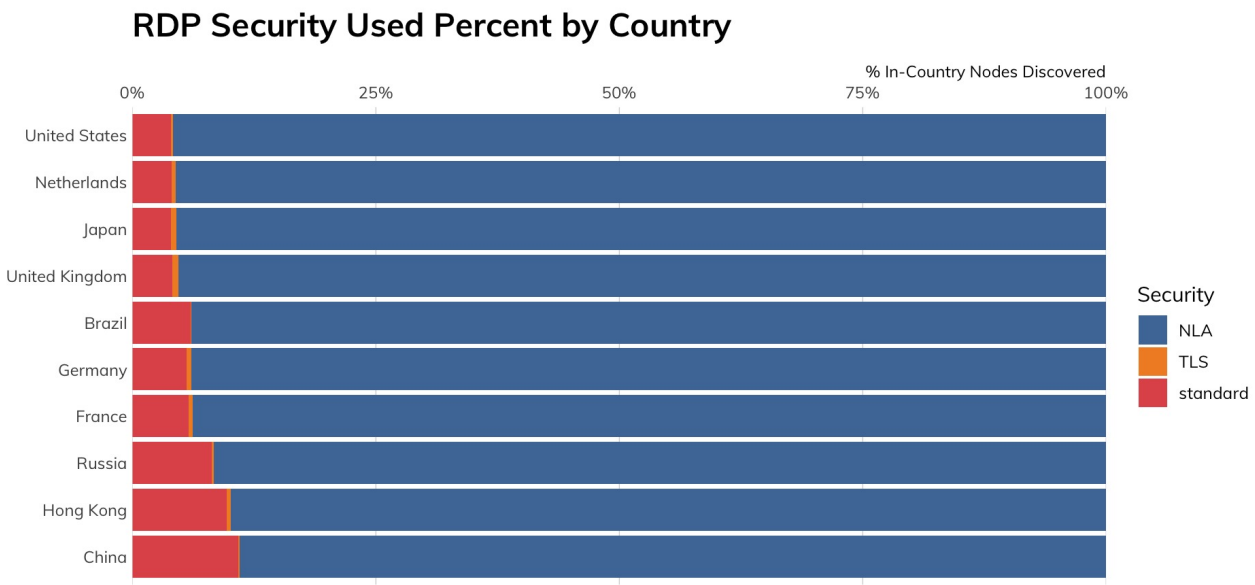
Exposure Information

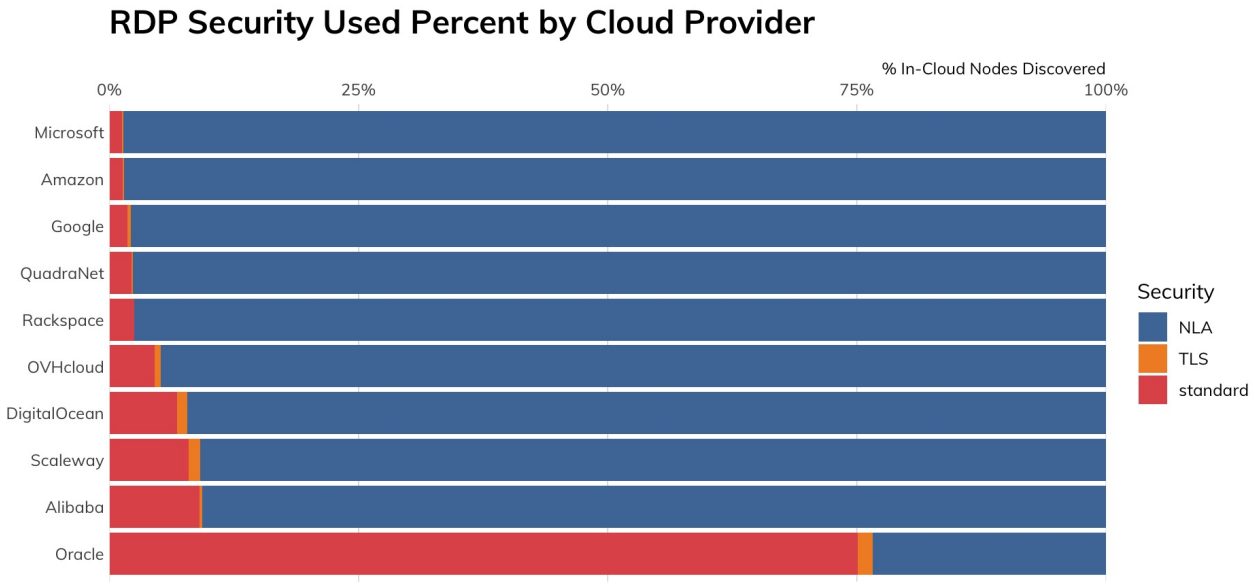


Unsurprisingly, we also find about a quarter of RDP nodes within our in-scope cloud provider networks—after all, one does need to connect to these remote systems to use them—though we would have expected Microsoft Azure to take the top spot (given, y'know, "Windows"). So, imagine our surprise when we found Azure-hosted RDP came in third. It's a strong third place, though, behind the twin cloud giants Amazon and Alibaba.

RDP is *everywhere*, with a total population an order of magnitude more than VNC (about 4 million and change versus less than 400,000 for VNC). Apart from the aforementioned "cloud" use case, small businesses use it off of business-class ISP connections; enterprises use it for remote access and administration; retail operations often have RDP enabled on point-of-sale systems for routine management; and, home users, too, find it easy enough to use to take the time to poke holes in their firewalls to enable remote access to home PCs. This ubiquity, along with pervasive misconfigurations and vulnerabilities, make it an ideal target for attackers, which we'll cover in the next section.

There are *many*[56] ways to configure RDP sessions, and Project Sonar is able to identify three states: “standard” or “legacy,” “network level authentication” (NLA), and transport layer security (TLS). “Standard” is a train wreck waiting to happen, and both NLA and TLS do a fairly good job securing your sessions (provided you’re patching regularly and using safe configurations with multi-factor authentication). And, from the looks of it, most residential and business-class users seem to have received that memo given the high percentage of nodes in the top 10 countries and the fact that clouds are using more secure connection options:





However, all clouds are not created nearly as equal, with Oracle cloud users limping in dead last (again, by percent of nodes in that cloud) when it comes to use of safer security modes. This is

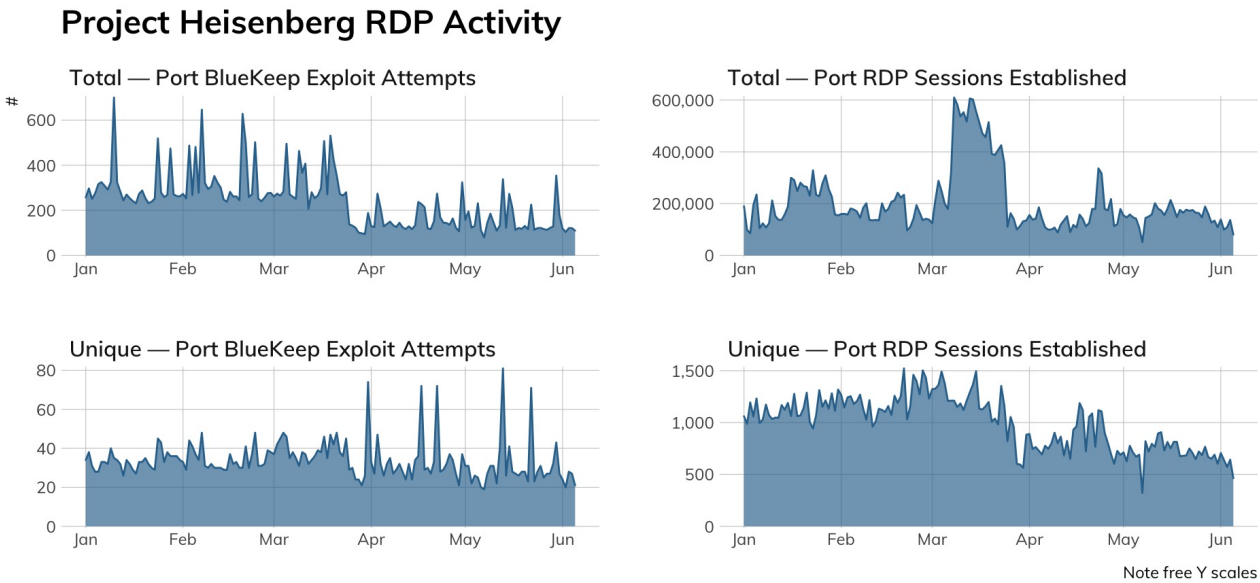
likely due to the default settings when enabling RDP.

Attacker's View

This is one protocol we do have robust honeypot coverage for, and we'll focus on two areas: RDP session establishment and attempts to exploit RDP via BlueKeep.

Despite predictions for cyber-doom and gloom, BlueKeep—the unauthenticated remote code execution flaw in RDP, disclosed by Microsoft in the spring of 2019—pretty much faded into the background, possibly due to folks *actually* patching and switching to NLA security (as we see in the country and cloud top views and in the fact that 92% of surveyed RDP nodes were using NLA). We posit that the early, and consistent, measured warnings at all levels helped keep BlueKeep from being the nightmare it could have been.

That does not mean there is no evidence of BlueKeep activity; however, we tend to see orders of magnitude more credential stuffing attempts than we do use of BlueKeep:



There are four takeaways from the above figure:

- The BlueKeep volume is super low.
- The number of unique source hosts with malicious intent is low.
- RDP activity is a constant.
- Attackers thought they might get lucky right around the time the U.S. entered lockdown because of the 2020 pandemic.

Between January 1, 2020, and April 30, 2020, Heisenberg caught just over 8,500 distinct source IP addresses trying to use either BlueKeep-based exploits or perform credential stuffing against our honeypot network. Eleven percent (11%) of these unique IPv4 addresses originate from DigitalOcean, but 16% of all malicious RDP traffic comes from Hostkey, a hosting provider in the Netherlands. While those two network sources do stand out, 1,310 distinct autonomous systems (ASes) are hosting sources with malicious intent, so there are plenty of places to wag one's chastisement finger at.

Our Advice

IT and IT security teams should strongly consider putting RDP behind a VPN if they are heck-bent on using RDP for GUI remote access. If you need to place RDP directly on the internet, consider doing so on an as-needed basis and ensuring all systems with RDP exposed are patched and have RDP configured as strongly as possible, backed by multi-factor authentication.

Cloud providers should have amazingly secure defaults and regularly warn users if they deploy RDP with weaker settings. Giving users the ability to *easily* enable RDP for only a certain period of time would go a long way toward helping thwart credential stuffing attacks. To help make the internet a bit safer, providers should also monitor for malicious traffic and work with regional CERTs to shut down malicious nodes as soon as possible.

Government cybersecurity agencies should continue to educate their constituents on the dangers of RDP and how to ensure safe use of RDP. Given that there are fairly well-known sources with evil intent, centers should work with regional CERTs and hosting/cloud providers and ISPs to make it easier to stop bad traffic before it compromises remote hosts.

Citrix ADC/NetScaler (TCP/Various)

It's like VNC, but like if Plan9 ever escaped Bell Labs and got super popular.

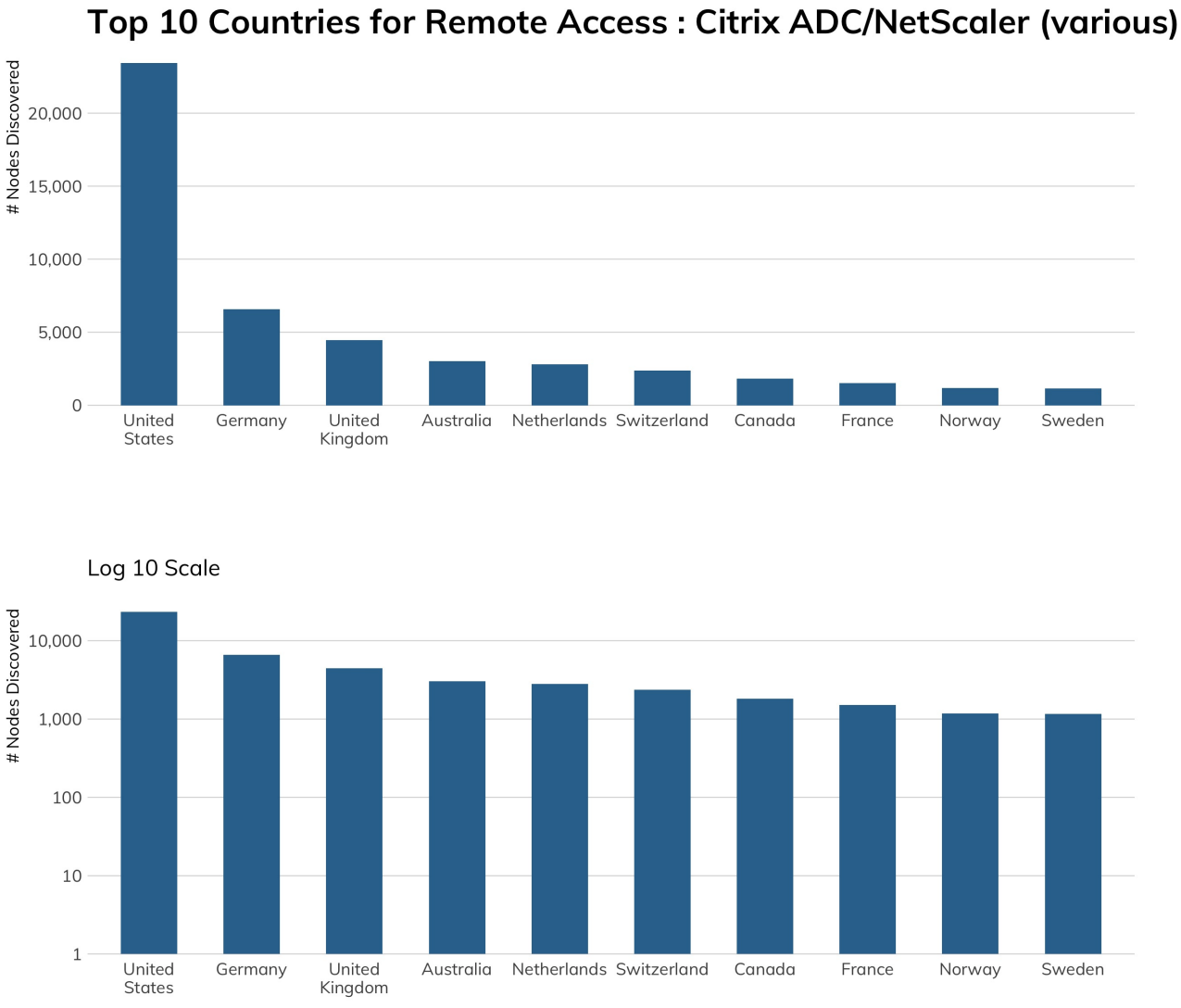
Citrix was founded in 1989 and has a diverse array of remote access solutions over the years. Modern Citrix ADC (application delivery controller) and NetScaler solutions use the Microsoft Remote Desktop Services infrastructure to deliver virtual applications and desktops to remote users. Organizations have the ability to configure stronger access controls than with vanilla Remote Desktop, and there are other optimizations within the Citrix application delivery process that also make it faster and consume less bandwidth than raw Remote Desktop sessions.

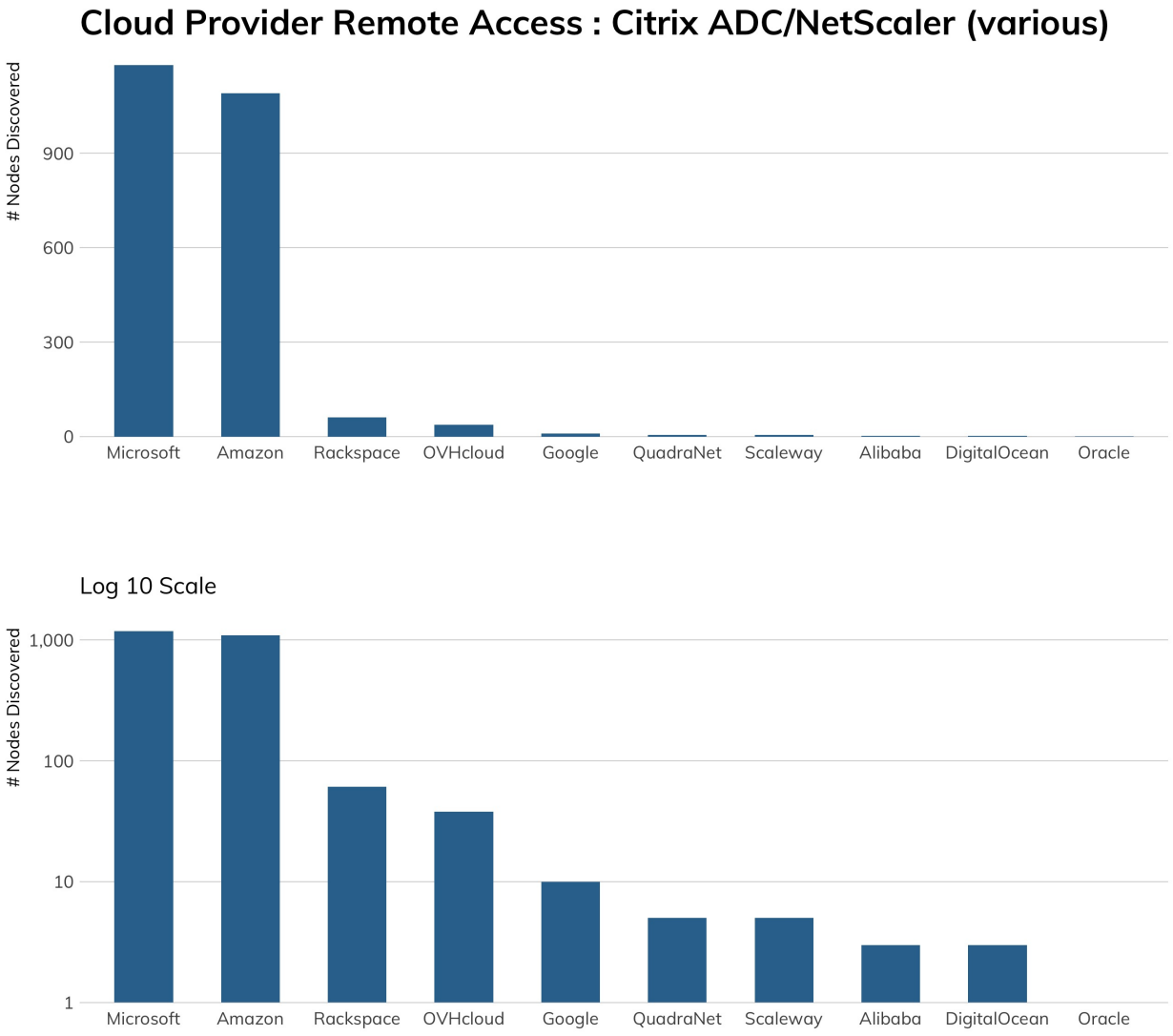
Discovery Details

Identifying Citrix systems on the internet turns out to be pretty easy, since their HTTP and NTP servers kind of go out of their way[58] to proudly let you know they are, indeed, Citrix systems. This makes it easy for Rapid7 Labs researchers to track the spread of Citrix systems on the internet, and in March 2020, we also developed a method to fingerprint the server version based on the version fingerprint of the Citrix client that is offered for download (again, Citrix going out of its way to help folks identify their systems).

The Labs team spent time on this effort because attackers keep compromising systems that haven't patched a gnarly remote code execution vulnerability,[59] and we have many in-flight projects set up to model patch adoption rates of various technologies.

Unlike many other top 10 country lists in this report, China failed to even beat out Sweden in their internet-facing exposure of Citrix systems.

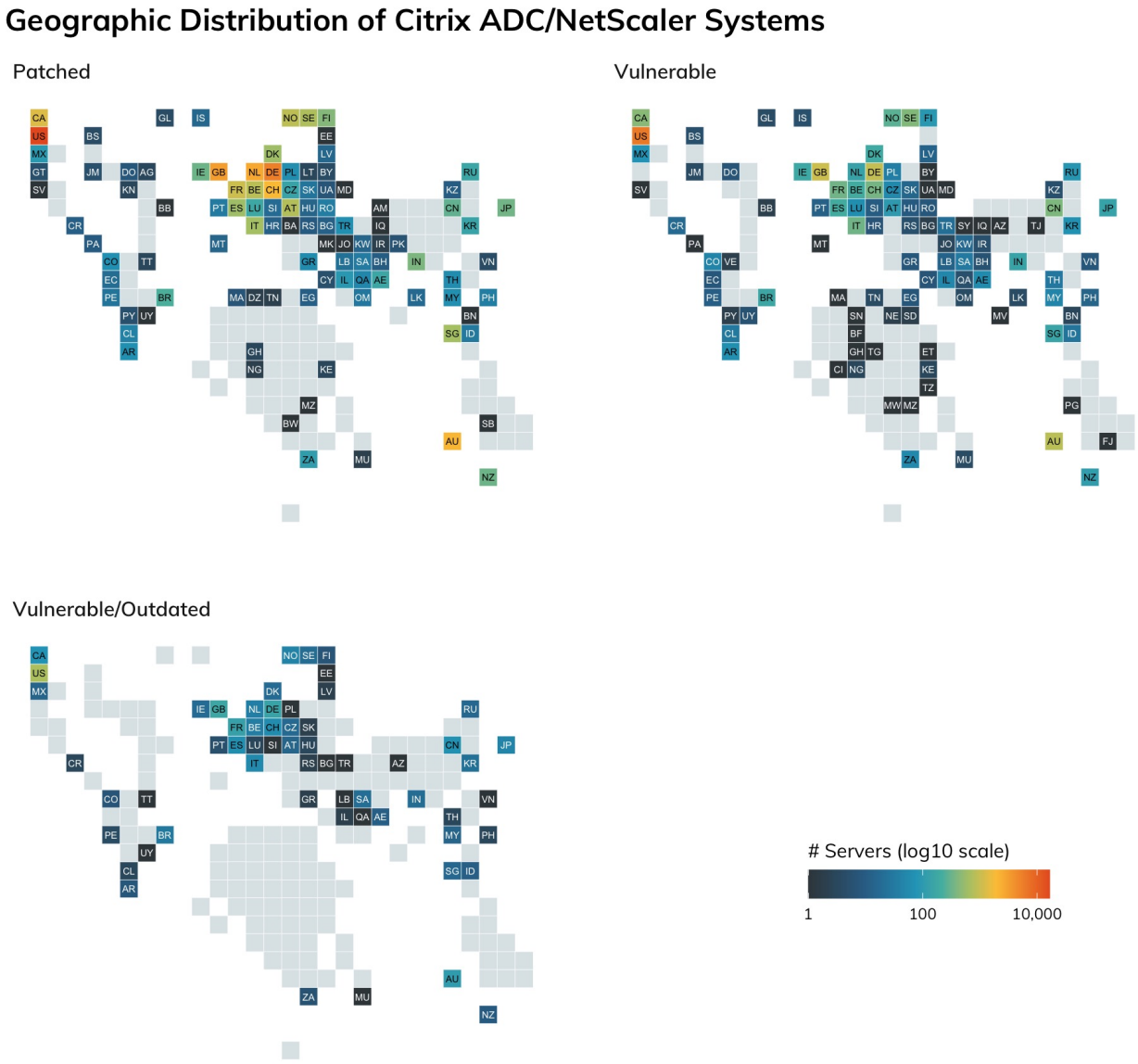




The lack of Citrix in cloud environments makes sense, since this technology is usually associated with virtual desktop infrastructure (VDI), which is almost exclusively found in enterprise/business environments.

Exposure Information

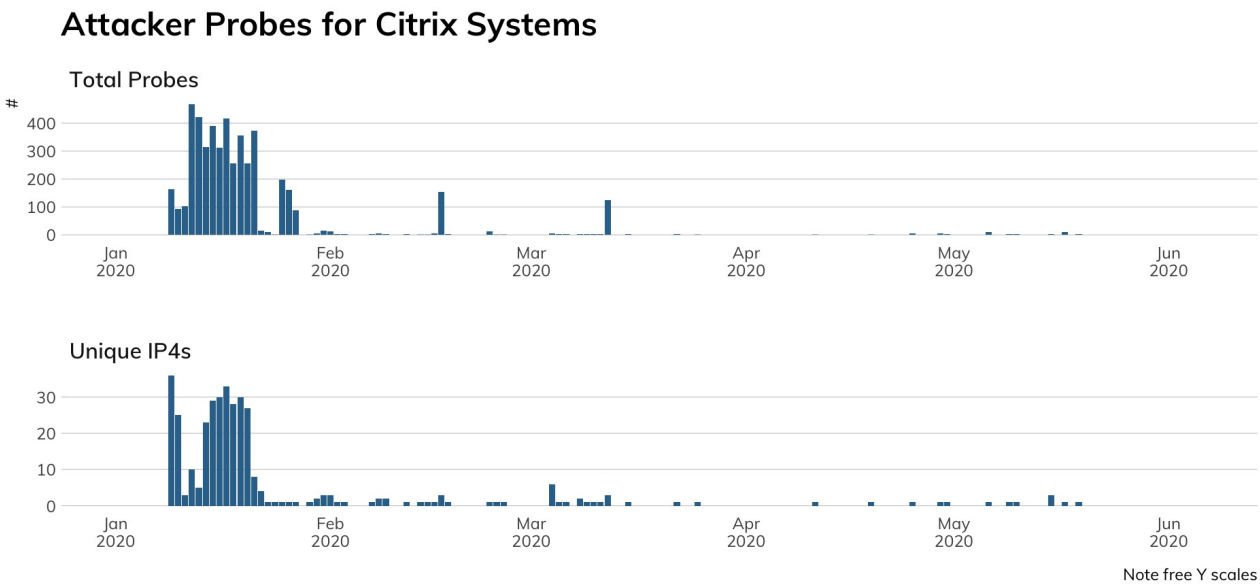
With an actively exploited vulnerability in play and regular government warnings[60] about the situation, you'd likely guess that internet-facing Citrix servers were fully patched or had mitigations in place. And, you'd be wrong (*again*), but this time, the situation isn't as grim as you might expect.



Our version fingerprinting technique showed that 73% of internet-facing Citrix systems have patches or mitigations in place, with the remaining 27% either being vulnerable or woefully outdated (thus having other issues to worry about). It has taken **five months** to get to a patch rate of 73%.

Attacker's View

The vast majority of our Heisenberg honeypot nodes are in cloud environments, and—as we've just seen—clouds are not where Citrix tends to live (at least on public internet cloud segments). Back in January, we caught attackers and researchers looking for exploitable systems quite regularly, but that activity has waned (though it hasn't stopped).



Our honeypots do not emulate Citrix, so the lack of activity is more likely due to our nodes being ignored after each attacker inventory scan. Attackers may also be reusing initial inventory lists or have already established a foothold on the thousands of systems that took forever to be patched.

Our Advice

IT and IT security teams should relentlessly monitor vendor bulletins and CVE reports and patch Citrix environments as soon as possible. With attackers increasingly targeting remote access technologies over the past 18 months, it would also be a good idea to have enhanced monitoring with more detailed logging set up on these systems.

Cloud providers likely can just keep doing what they're doing with regard to Citrix since it does not seem to be widely used, despite Citrix-provided solutions[61] for these environments.

Government cybersecurity agencies should keep up the great work they've been doing calling attention to threat actor activity and the severity of vulnerabilities in remote access technologies like Citrix.

Computers have played a major role in cataloging information ever since their creation, but they really took off in the 1960s due, in large part, to the success of IBM's SABRE[62] system and how it transformed how we all book travel. In the '70s, E.F. Codd published a foundational document describing what we know today as relational database management systems (RDBMS), which are, in essence, normalized tables of information connected by keys. Ever

since then, we've seen all sorts of database types emerge, but for the purposes of this report, we're focusing on two common types: RDBMS and simple, but powerful, key-value storage systems.

The relational database is an amazing technology, and will be with us for a long time to come. That said, there is **no good reason to expose them to the internet**. Ever. We have an entire class of vulnerability—SQL injection—that ultimately describes the **accidental** exposure of database functionality that would otherwise be safely tucked behind a web application. There is no case in which a database should have an open connection addressable by anyone on the planet, regardless of any authentication scheme needed to actually access it. Doing so is asking for trouble.

Exposing databases to the internet is fundamentally ill-advised and promises both heartache and headache. It's bad. The rest of this section should be read as an analysis of just how bad, on a scale delineated by multiples of "very."

MySQL (3306)

My SQL, your SQL, we all SQL for SQL.

We could write an entire paper on the fragmented history of MySQL. It started off as an open source, unified codebase and—since being acquired by Oracle—has variants such as MariaDB,[63] Percona,[64] Google Cloud SQL,[65] and a few others. They all "speak" MySQL, but versioning works a bit differently for each of them. When we do slice and dice by vendor, we'll be focusing on the official Oracle MySQL variant and MariaDB, since they make up 98.8% of discovered nodes.

Discovery Details

Poland barely passes Germany to fall into third position due to hosting provider Home.pl (thanks to Home.pl's aforementioned affinity for, well, less-than-great default configurations as detected with our FTP studies). The United States accounts for 34% of all exposed MySQL, with China being a distant 15%.

Alibaba has both images with Oracle MySQL and MariaDB, but also has its own MySQL-flavored offering in its AsparaDB managed service.[66] Amazon has a similar situation,[67] and OVH also has targeted MySQL offerings.[68] It's strange to see them be in the top 3 of cloud

exposure, as each provider does a pretty good job of offering secure defaults for the images and services they provide and have good documentation on securing MySQL. This means folks either go out of their way to make MySQL appear on the internet or *really* mess up the configuration.

Because we chose to focus on cloud providers and not “hosting” providers or co-location companies for the majority of this report, we need to add some color to this section, since co-location company Unified Layer[69] accounts for 145,967 exposed instances (beating OVH) and hosting provider GoDaddy[70] accounts for 101,775 exposed instances (beating every other provider).

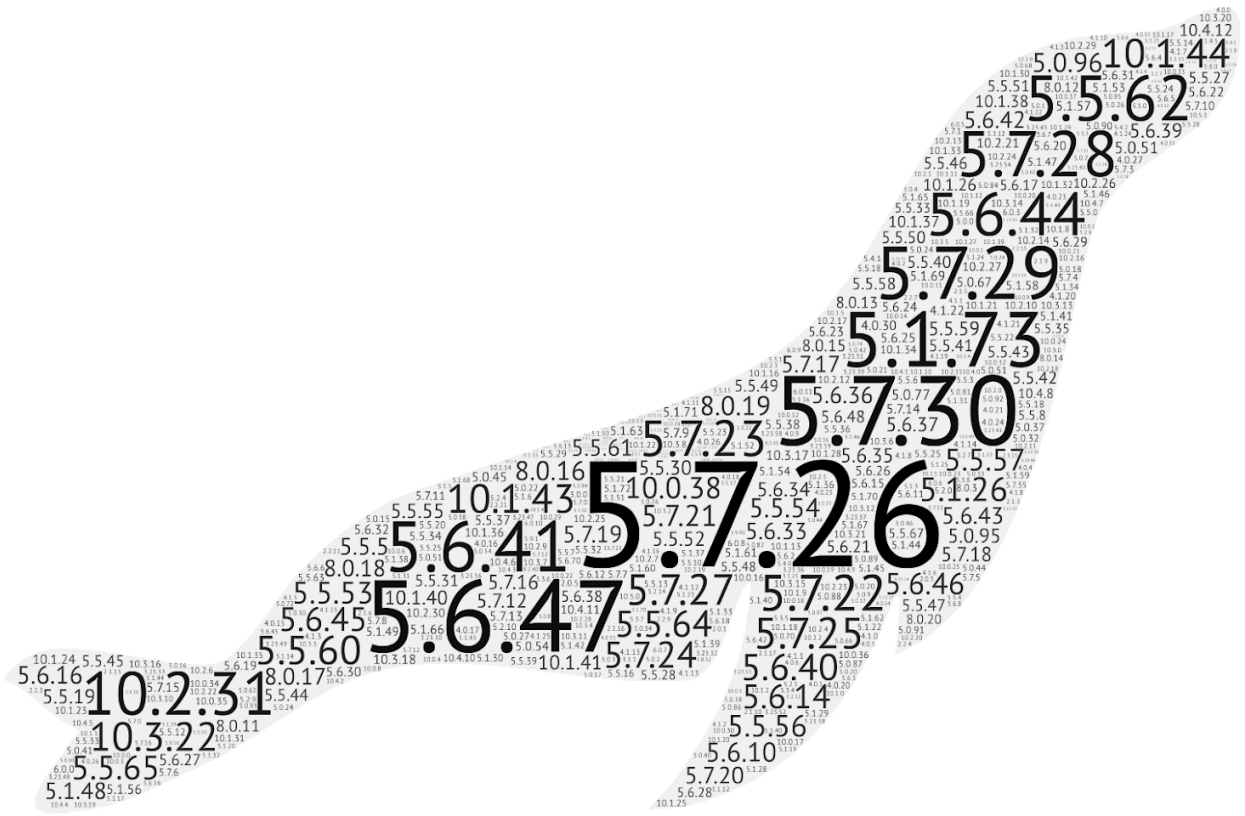
There are **17,876** autonomous systems exposing MySQL instances, with a median exposure of three servers and a mean of 156 servers, so there’s plenty of exposure finger-pointing to go around.

Exposure Information

There are 1,006 vendor+version combinations in the corpus, and that’s if we aggregate vendors into Oracle, MariaDB, Google, Percona, and “Other” buckets. With over 2 million instances on the internet, we may have a sufficient surveyed corpus for it to be safe to say that nobody manages MySQL well on their own. Yes, we’re looking right at *you*, now. Go ahead, type `mysqld --version` at your laptop’s command prompt or a server you regularly interact with (you know you’re running one somewhere). One of our authors—that crazy guy with the shield—did it and even *he* is two patch points behind the latest MariaDB release.[71]

We get it. Patch management is hard. But not patching a local laptop instance only exposed to `localhost` and not patching a MySQL instance *directly connected to the internet* are two vastly different situations.

Still, you likely want to know what the version distribution looks like. We had to get a bit creative for this one (given the huge spread), so we’ve made a word cloud superimposed on the MariaDB logo, because seals are awesome:



Version 5.7.26 was released on April 25, 2019 (Oracle version, which all the other ones mostly flow from).

Version 5.7.26 has 13 moderate vulnerabilities, while 5.7.30 was released on April 27, 2020, so it is relatively current as of this report writing. Oracle maintains official branches for 8.0.x (which is really 5.8.x), 5.7.x, and 5.6.x due to fairly major technical differences between each of those versions. To keep things confusing, MariaDB jumped from 5.6.x to 10.0.x, with the most prominent 10.x release in the corpus as 10.2.31, which was released in January 2020 and has been superseded by 10.2.32 (released in May). MariaDB itself maintains version 10.0.x through 10.4.x.

If you had trouble following that paragraph, you now have a more perfect understanding of how hard database patch management is, since it's all a twisty maze of similar-but-different multi-decimal numbers. So, *stop putting MySQL on the internet!*

Attacker's View

Heisenberg has no MySQL honeypots, and the nature of MySQL connection attempts make it difficult to tell spurious connections from directed attacks or deliberate (albeit, misconfigured)

attempts to legitimately communicate with something someone thought they owned. This means any charts we could have shown here would just result in more questions than answers.

Suffice it to say, Heisenberg generally sees 10,000–30,000 TCP connection attempts daily on TCP/3306 from a median of approximately 250 distinct source IPv4s. A handful of these (daily) are from other researchers scanning for MySQL, and between 5% and 15% are misconfigured clients, as our honeypot nodes are mostly in cloud IP space.

We *can* let you know that back in 2019, attackers launched ransomware campaigns[72] against internet-facing MySQL servers and that there are billions of credentials out there for malicious actors to try against the 2+ million MySQL servers we found, so you really should think twice about putting MySQL on a public server.

Our Advice

IT and IT security teams should never host MySQL on a public IP address and should strongly consider picking one flavor of MySQL vendor+version and make it standard across your entire enterprise (and keep it patched). MySQL often comes bundled with “appliances,” and you should work with your procurement team to ensure the vendor communicates which version is bundled with their solution and also that they provide timely updates when new MySQL releases are announced.

Cloud providers should continue to offer secure, managed MySQL-compatible offerings to help mitigate the threats associated with customers hosting their own MySQL infrastructure. Vendor-managed disk images with MySQL distributions on them should be updated immediately when there are new releases and vendors should communicate with customers to inform them they need to update their legacy versions.

Government cybersecurity agencies should provide meaningful guidance on how to host MySQL securely and provide timely notifications when new attacker campaigns are discovered. Furthemore, an effort should be made to work with cloud providers, hosting providers, and ISPs to prevent MySQL from being connected to the public internet.

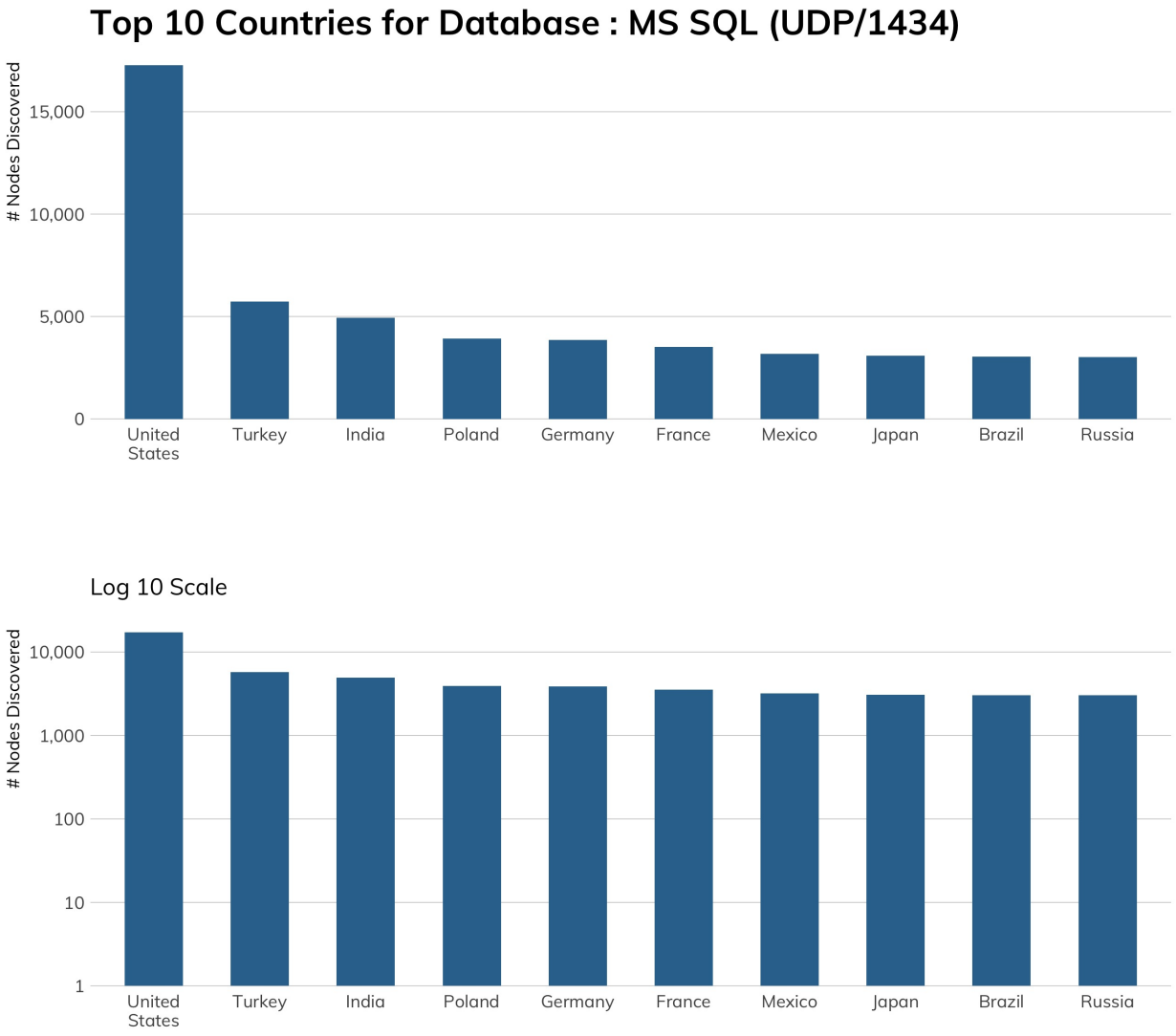
Microsoft SQL Server (MS SQL) (UDP/1434)

*SELECT TOP 1 * FROM quippy_subtitles;*

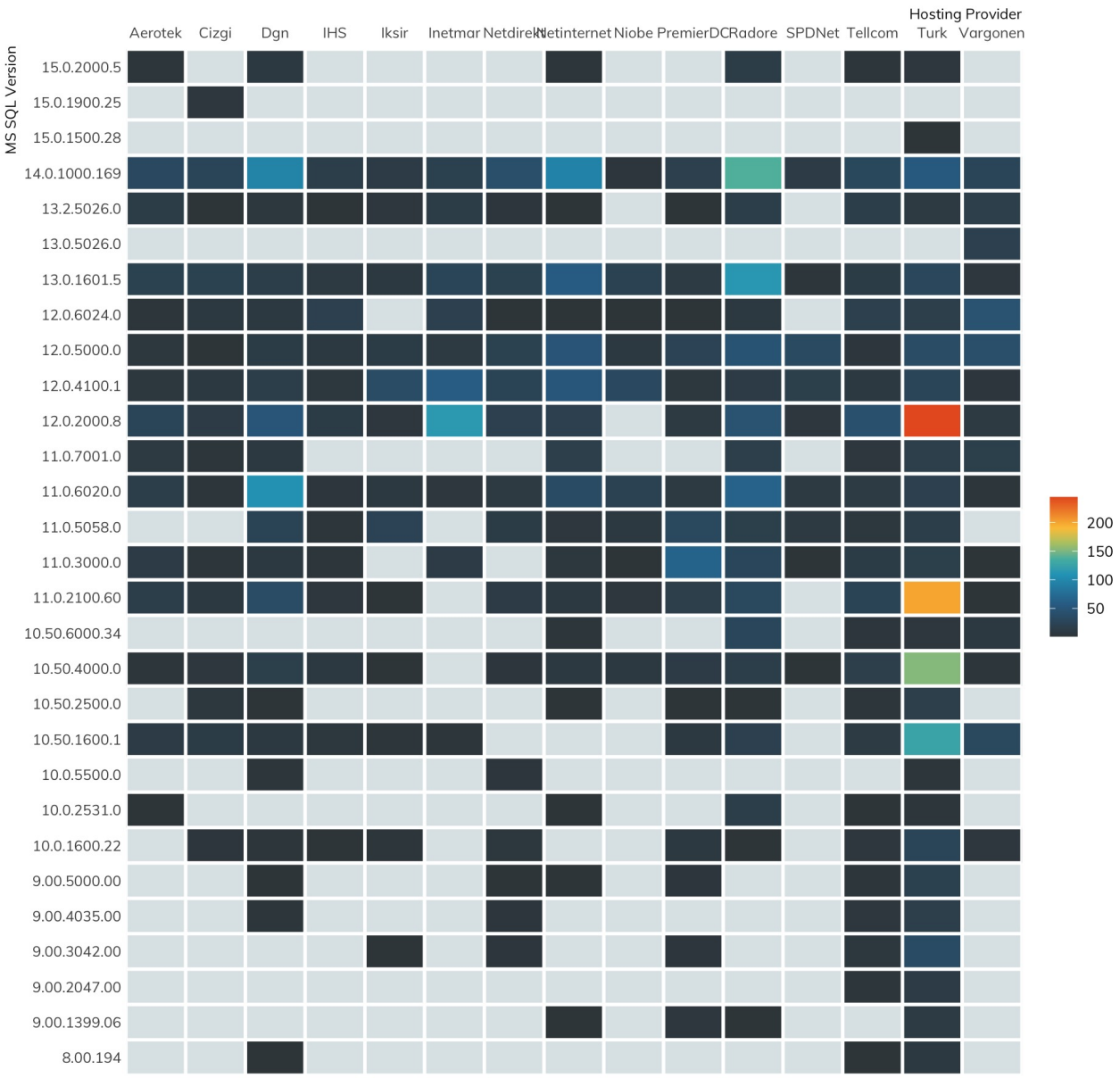
Microsoft SQL Server 1.0—a 16-bit server for the **OS/2** operating system—was first released in 1989 (so, it's older than many of you who are reading this report!). The first version on a Microsoft Windows operating system (NT) was SQL Server 4.2, released in 1993. Presently, Microsoft supports five different major versions of SQL Server: 2012, 2014, 2016, 2017, and 2019, along with its Azure cloud database offering. If you're in a large enterprise, it's almost guaranteed that you have MS SQL Server running *somewhere* supporting some business process/task.

Discovery Details

Despite the fact that one should *never* expose *any* database service to the internet (unless it is via some deliberate API offering), Project Sonar found nearly 99,000 of them more than willing to give us many details about their services (without any authentication). This is a *tiny* number compared to what you saw in the section on MySQL—two orders of magnitude tiny—which is likely due to the fact that MS SQL Server costs money and MySQL is, well, *free*.



It's unusual for Turkey to break into these top 10 lists, let alone take the No. 2 spot away from larger countries, so we dug into the results a bit, and it turns out customers of hosting providers in Turkey *love* to expose Microsoft SQL Server to the internet! Fifteen providers account for 75% of Turkey's exposure, with an impressive variety of SQL Server versions on display. Similarly, 20 hosting companies in India account for 75% of SQL Server exposure there (35 total SQL Server versions). Poland is worth a mention, since it exposes nearly as much SQL Server as Germany. Orange Polska—a large ISP and hosting provider—accounts for 50% of the SQL Server exposure in Poland, with one, large sales, accounting, and HR management SaaS firm accounting for 35% of exposure from Orange's network.



We suspect there's a very popular database management course in some Istanbul university or training facility that's encouraging people to expose Microsoft SQL server. If you know of it, please let us know so we can convince them to stop it!

You'd *think* Microsoft would have come in first when it came to *Microsoft* SQL Server exposure in our in-scope cloud providers, but if we step back a bit and remember that Microsoft Azure has an Azure Database Service[73] offering, it's easier to see why it may not be exposing as many SQL Server instances to the internet, since it's likely very difficult to talk to the Azure Database Service from the outside.

OVH makes a big deal[74] out of providing cost-effective, ready-made images for SQL Server, as does Amazon,[75] but OVH seems to cost less than the other two when it comes to self-hosting MS SQL, which is what likely gave it the top spot.

Exposure Information

We found 54 unique versions of Microsoft SQL Server across those systems, 20 of which make up 95% of the exposure. Red cells indicate that the SQL Server main version is no longer supported. “Vintage” signifies the release date for the listed version.

We must note that none of the supported versions are at the latest patch release. Granted, anyone who is recklessly exposing MS SQL to the internet likely does not have the greatest cyber-hygiene by default, so it is somewhat optimistic to expect to see these folks keeping up with patches.

Attacker’s View

Attackers regularly set their sights on SQL Server, but they’ve gone above and beyond since October 2019 (in reality, going back to 2018, but they really stepped things up in 2019) with a relentless credential stuffing and SQL execution campaign.[76]

Our high-interaction MS SQL honeypots in Project Heisenberg were literally overwhelmed with this activity starting shortly after outlets started reporting on a possible backdoor[77] in some less-than-legitimate distributions of MS SQL Server. Each day saw over 63 *million* credential access attempts, followed by various SQL command sequences (when the honeypots allowed the attackers to log in). Rapid7 Labs has yet to correlate the drop in February to any known, public actions, but this campaign is far from over (though it may have changed hands since Guardicore broke the story).

If this doesn’t convince you to be *extra careful* about not hanging MS SQL Servers directly off the internet, we’re not sure what else would.

Our Advice

IT and IT security teams should never, ever, ever, ever, *ever* put MS SQL Server instances directly on the internet **and** should do a *much* better job than the folks responsible for these ~32,000 have when it comes to patch management. Those responsible for managing access to

internal SQL Server instances should track credential breaches and force password resets for any accounts that match usernames in those in the credential dumps. We know we shouldn't have to say this, but given both the attacker campaign and the sorry state of SQL Server on the internet, you should never leave default accounts enabled and never use default credentials.

Cloud providers should continue to offer secure, private Microsoft SQL Server managed services and ensure their provider's managed images are always at the latest patch level. Customers who use out-of-date and especially out-of-support images should receive regular communications regarding the hazards associated with lack of attention to this matter.

Government cybersecurity agencies should track campaigns against MS SQL Server and provide timely notifications to individuals, organizations, and their constituents with sufficient detail to help them detect possible attacks. Extra focus should be made on providing education and awareness regarding the need to keep MS SQL Server patched and ensure it does not sit directly on the public internet.

Redis (6379)

Even non-relational databases shouldn't be on the internet!

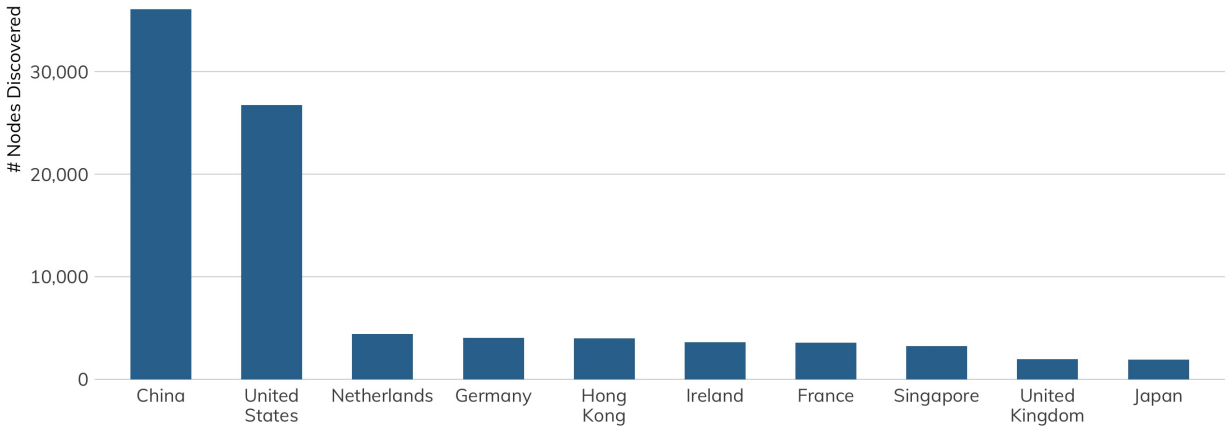
Redis fundamentally reshaped or, at least popularized, the idea of having data that you need always resident in-memory and on-disk, with the sole purpose of the on-disk version to be that of rebuilding the in-memory version and for use in synchronization in high-availability configurations. It is immensely popular in the category of "NoSQL"[80] databases and is used in production at Twitter, GitHub, and many other large-scale environments.

Discovery Details

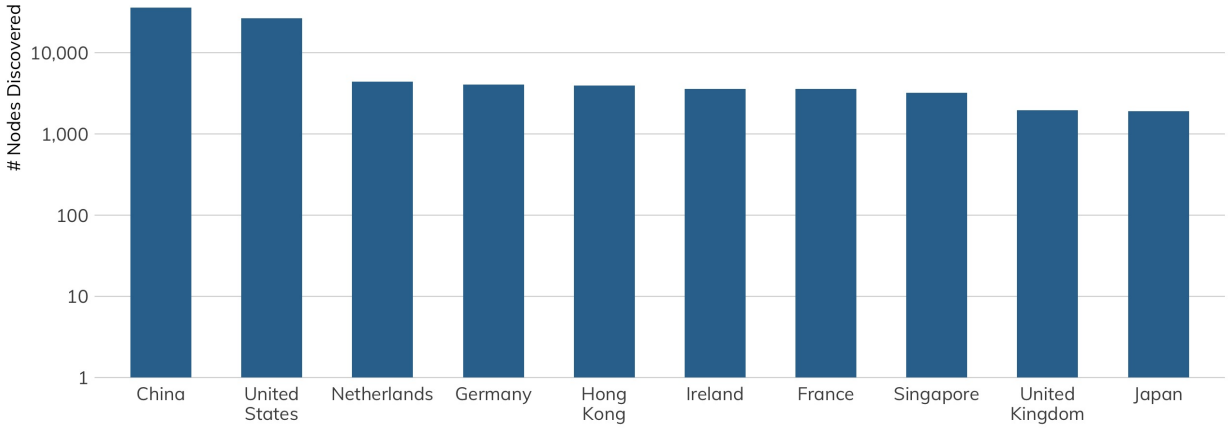
Project Sonar discovered 102,801 Redis instances on the public internet. That's an astonishing figure given that Redis, like all other databases, should never be exposed to the internet.

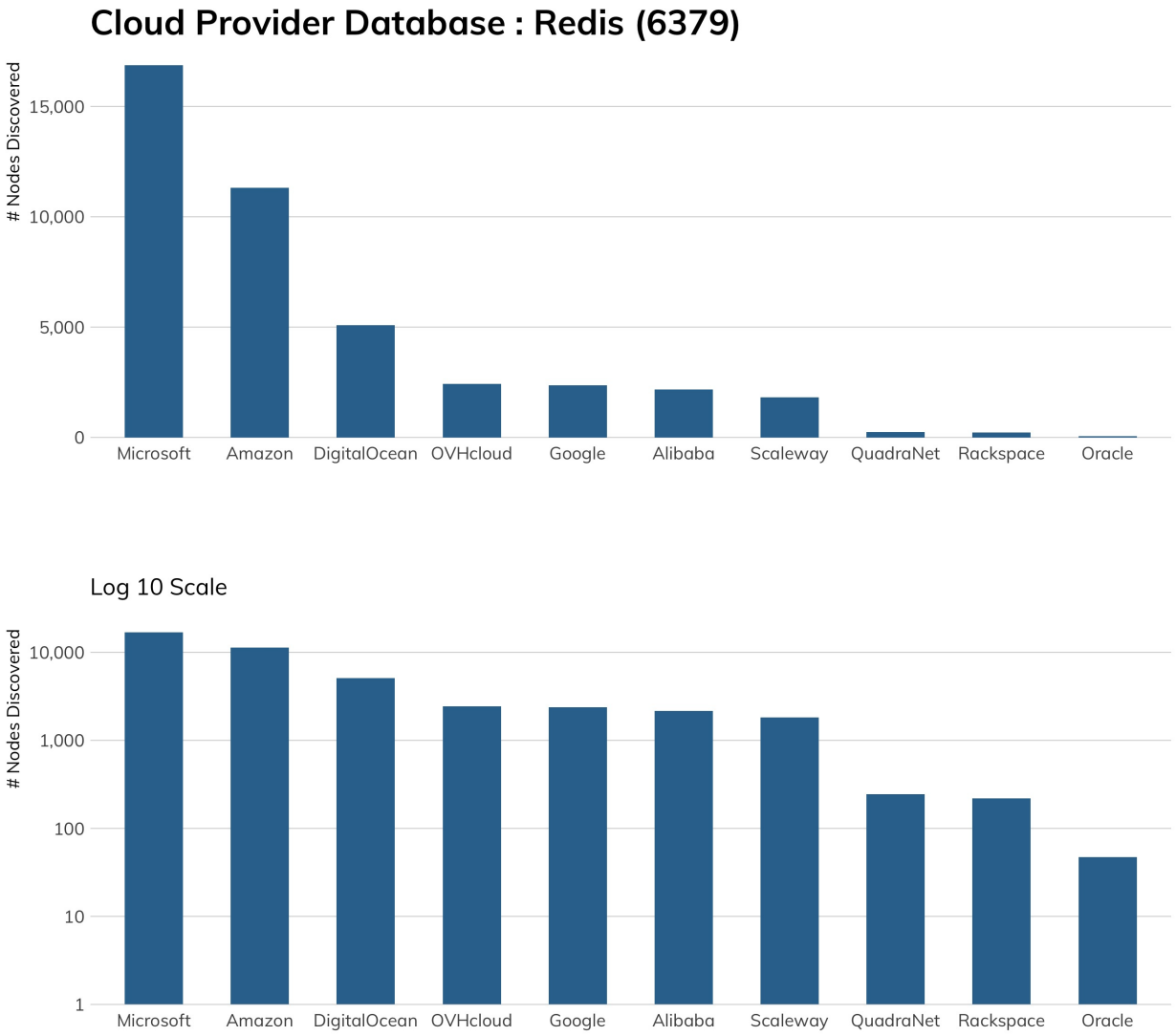
China comes in at No. 1, helped largely by the presence of TencentDB,[81] which fully emulates the Redis protocol. Over 15,000 of these Redis-compatible nodes in China come from the Tencent autonomous system.

Top 10 Countries for Database : Redis (6379)



Log 10 Scale





Finding Redis in cloud environments is no real surprise, since that's where a great deal of public web applications reside today. Microsoft tops the list, as it relies heavily on Redis for its Azure Cache offering,[82] and Amazon's second-place finish is a result of direct AWS support for Redis[83] and their Redis-compatible ElastiCache service.[84]

But, if you're looking at the cloud counts and thinking they seem *low*, keep reading, since we'll speak to this in the **Exposure Information** section.

Exposure Information

As noted in the TLDR, Redis has not had too many CVEs (one in particular[85] *was* pretty bad), but the chief weakness of Redis is that—by default—it binds to all network interfaces.[86] That means unless you go out of your way to secure a Redis instance or Redis cluster, anyone with access to the same network segment will be able to talk to it (we'll come back to this statement

in a bit). The developer of Redis, antirez, demonstrated that with just a little knowledge of how SSH works, it takes just about five seconds to gain remote access to an unsecured Redis instance.[87]

Redis can be run on alternate ports, configured to require authentication, and wrapped in TLS tunnels to help secure the service, but it really has no business being on the internet. Since there are over 100,000 of them on the internet (on the default port), let's see what state they're in.

Given how easy it is to compromise a remote Redis instance, it is reassuring to see around 75% of these servers requiring authentication. We have no idea how strong those credentials are, since testing credentials without permission is verboten. It was also encouraging to see 9% of systems running in "protected" mode. Since version 3.2.0, when Redis is executed with the default configuration (binding all the interfaces) and without any password in order to access it, it enters a special mode called "protected mode" and will only reply to queries from the loopback interface and reply with an error message to queries from all other interfaces.

We *were* able to extract version and other information (via an INFO request) from around 13% of Redis instances and managed to count 112 different version strings. Thirty of them cover 90% of the discovered versions:

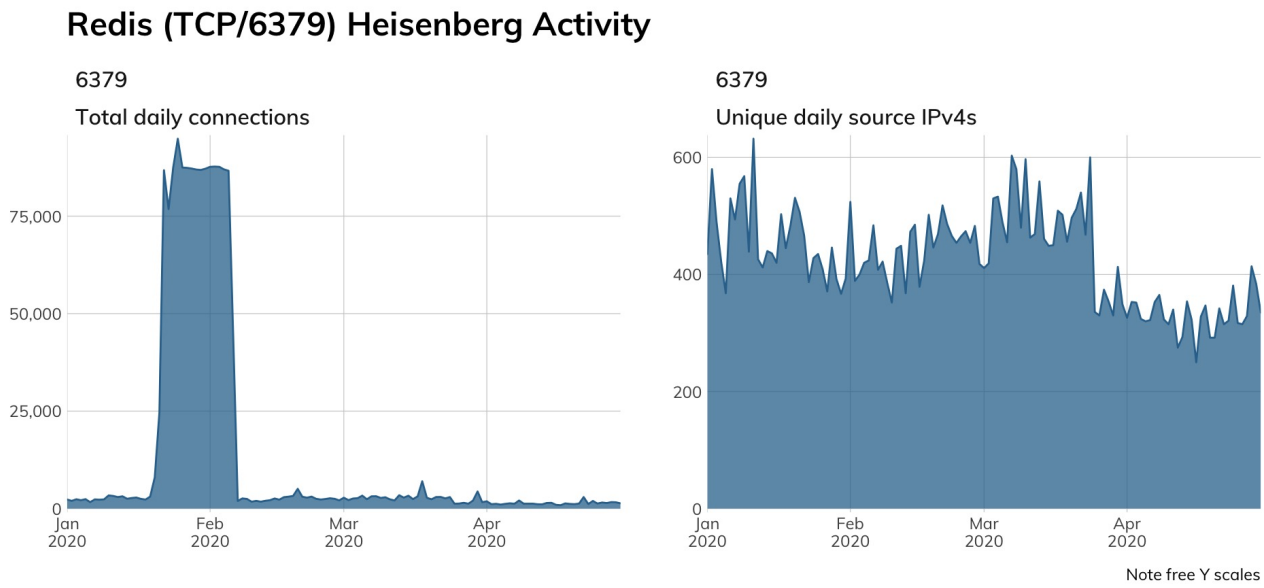
Part of the reason version 3.0.504 is so prolific is due to improperly secured TencentDB instances (that appears to be one of the versions it reports, though there are others). That version was also the last version released by Microsoft before Azure Managed Cache was shuttered.[88]

One final thing to note here is that the most current version from these April 2020 studies was 5.0.8, and indeed, that is the third most common version of Redis. At the time of publication, though, Redis is already on version 6.0.5. Redis version numbers increment fast.

Attacker's View

Given how easy it is to compromise an unsecured Redis instance, it stands to reason attackers do all sorts of terrible things to them,[89] such as gaining a new host to perform other malicious actions from, holding them for ransom, or installing a cryptocurrency miner.

Project Heisenberg does not emulate Redis, but we can use attempted TCP connections to port 6379 as an initial, imperfect proxy to see whether attackers are, indeed, scouring the internet for exposed Redis nodes.



Since we saw that highly abnormal spike, and since Redis is one of those “you can shove as many commands into a single packet as you can,” type of protocols, our interest was piqued sufficiently to dig into the packet captures for that time period.

It turns out that we certainly did receive many Redis requests during that period, specifically in our Rackspace honeypot nodes, and that it appears someone misconfigured their Redis cluster replication configuration to include our Heisenberg IPs (at that time) and managed to send us ~5GB of web server log count data, making an average of 1,700 Redis **PUBLISH** requests an hour.

This is the perfect time for us to remind you to not only secure your Redis instances but also maintain draconian control over your cluster configurations, lest ye inadvertently leak 5GB of your own data to unsuspecting security researchers.

Our Advice

IT and IT security teams should have well-established playbooks and automation for running Redis in production. They should also ensure developers configure Redis properly during application development to further ensure they aren’t exposing unsecured instances for attackers to take advantage of. No Redis instance should be internet-facing.

Cloud providers should ensure their managed Redis services are secured in their own defaults and make it super hard to run them insecurely. After all, Redis was explicitly designed to favor openness and usability over security. If machine images with Redis pre-installed are provided, those images should be updated with each Redis release, and users of those images should be notified they need to upgrade their deployed nodes.

Government cybersecurity agencies should provide guidance on running Redis safely and monitor for malicious actors attempting to gain a foothold on Redis systems.

memcached (UDP/11211)

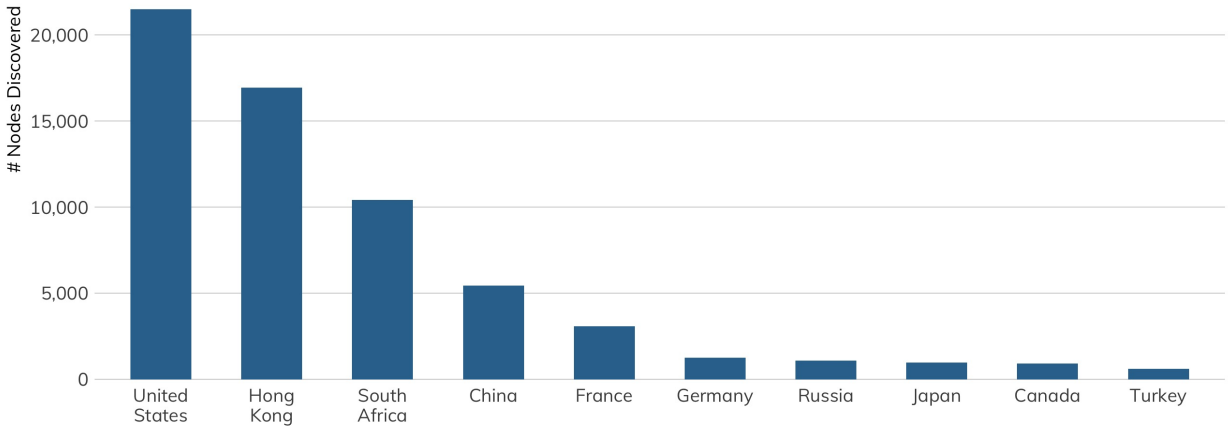
It's an easy-to-use DDoS Howitzer AND a NoSQL database![90]

Memcached is an in-memory key-value store for small chunks of arbitrary data (i.e., strings, binary objects) from results of database calls, API calls, or web page rendering. Its simple design has made it wildly popular, as it promotes quick deployment and ease of development.

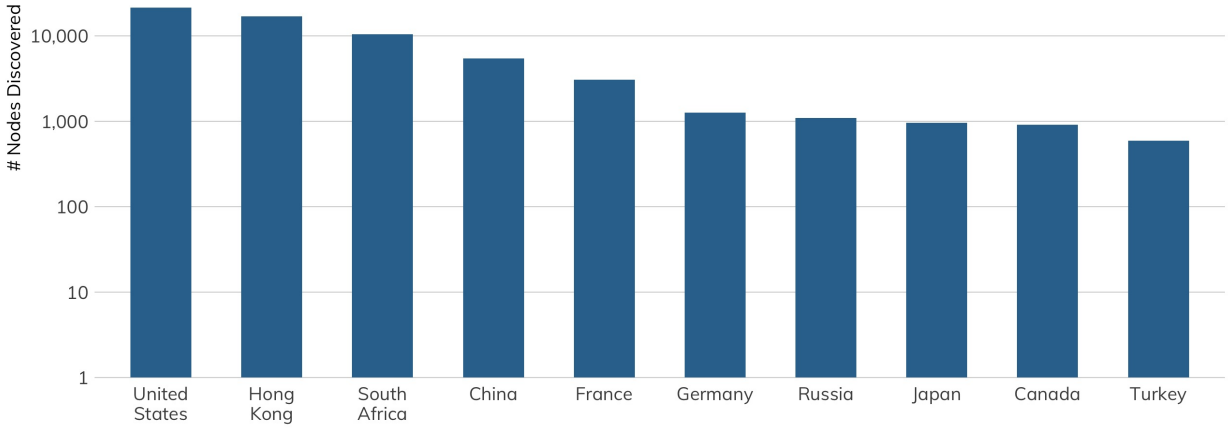
Discovery Details

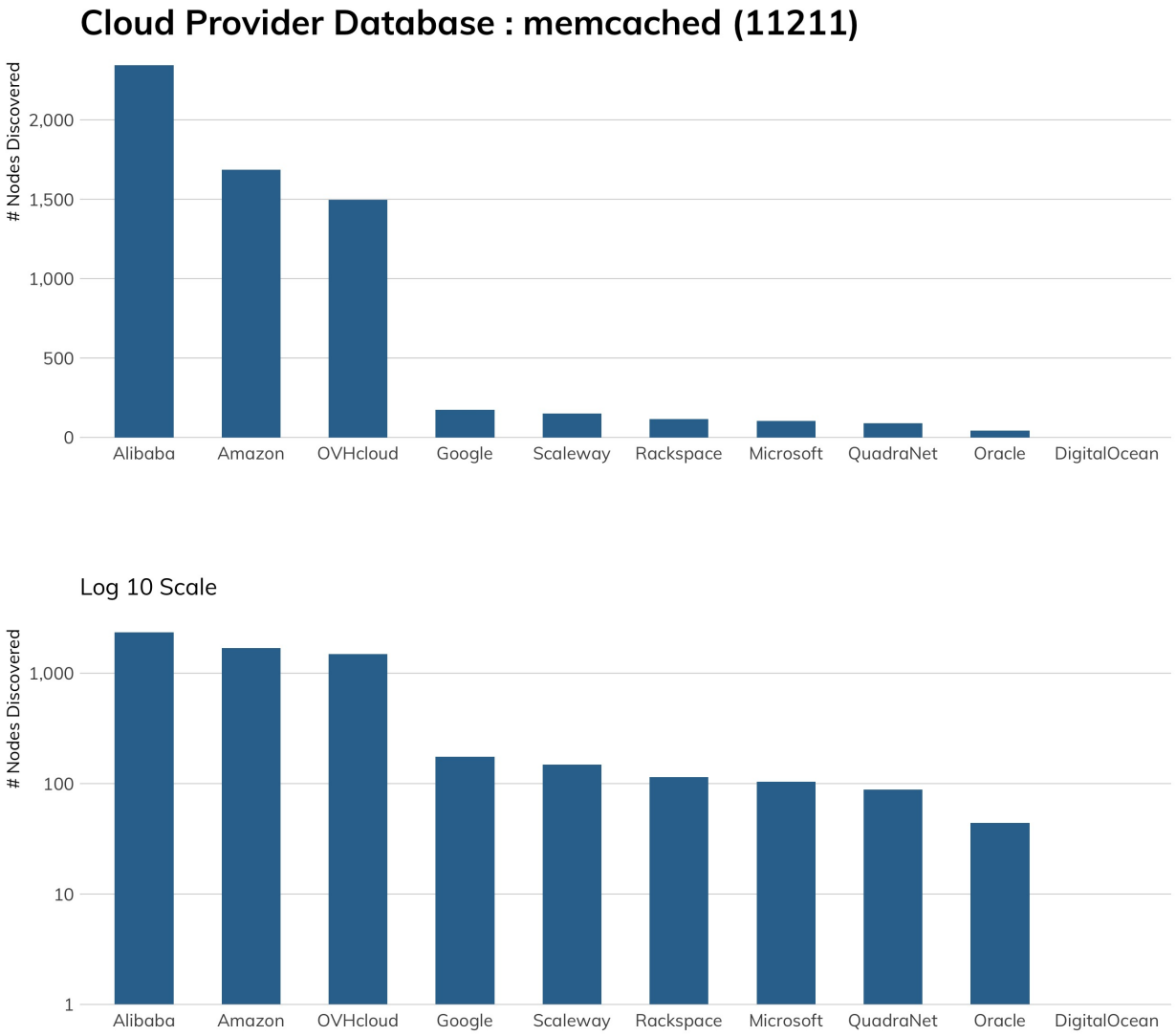
Project Sonar found 68,337 exposed memcached hosts, and we did a double-take when we saw that South Africa is in third place, since we don’t often find it in any other top 10 lists of exposure. Most (97%) of these SA nodes are in two autonomous systems: Icidc Network (87%) and Internet Keeper Global (10%), and a majority of hosts in each autonomous system appear to have similar exposure counts in both nginx and SSH.

Top 10 Countries for Database : memcached (11211)



Log 10 Scale





As noted in our section on Redis, Amazon has a cloud “cache” service offering that can also be configured to use memcached directly or emulate the exquisitely diminutive memcached protocol, and Alibaba has a managed memcached service offering,[91] so it is no surprise finding some in those environments, but it is somewhat disconcerting that these instances are exposed to the internet. What’s more surprising is that OVH *goes out of its way* to help you secure memcached[92] and, yet, ~1,500 folks apparently did not get that rather crucial memo.

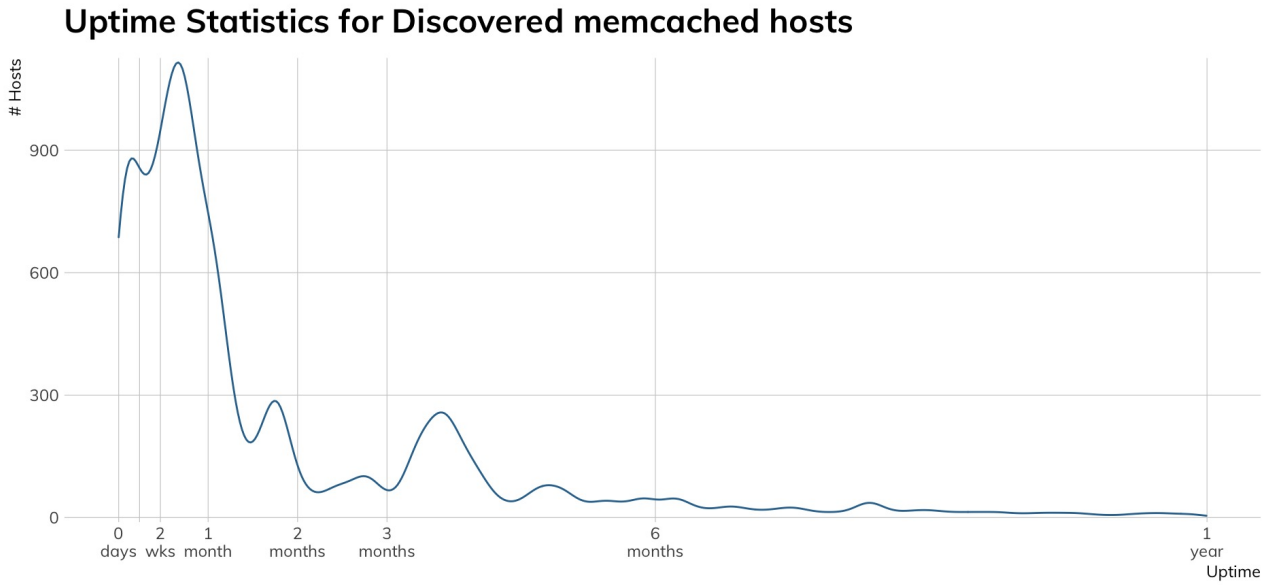
Exposure Information

Why all the fuss about memcached? Well, way back in 2018,[93] an “Insufficient Control of Network Message Volume” vulnerability[94] in the UDP portion of memcached was used to launch the largest distributed denial-of-service amplification attack *ever,* and disrupted many major internet services. Cloudflare summed up the flaw quite well in its blog[95] at that time:

"There are absolutely zero checks, and the data WILL be delivered to the client, with blazing speed! Furthermore, the request can be tiny and the response huge (up to 1MB)."

Rapid7 Labs keeps an eye on memcached in the hopes we won't find any, and, as seen in the previous section, said hopes are constantly dashed. Of the 68,337 nodes discovered, 68,044 return a valid version number (so ~300 folks think they're being awfully clever). If the valid version numbers are correct, 47% of them (~32,000) are below the fixed release version of 1.5.6, but the "fix" in 1.5.6 and above was to disable UDP by default. Our study uses UDP and sends the **stats** query command and receives a full reply, so all these hosts can potentially be used in other amplification attacks.

Oh, and since we get a full response from that **stats** inquiry, it is good to see that most of these servers restart at least once a month:[96]



Attacker's View

While we do not have a memcached honeypot, we can see connection attempts on UDP 11211 and peek at the packet captures to see if memcached commands (most often, the **stats** one we use, though we occasionally see what look like misconfigured clients that are connecting to what they think are their memcached servers). The daily memcached command connections we see are mostly from Shadowserver,[97] which is (correctly) self-described as a "nonprofit security organization working altruistically behind the scenes to make the internet more secure for everyone," despite their scary name. They also scan the internet every day to try to get a

handle on exposure and help organizations (for free) get a picture of their attack surface. We heart Shadowserver.

During the first third of 2020, we saw spikes of near 80,000 data-less UDP connections on 11211 across a handful of days, but none of this appears truly malicious in nature.

Our Advice

IT and IT security teams should never expose memcached to the internet, and should ensure via playbooks and automation that development, test, and production memcached environments are rigidly controlled.

Cloud providers should continue to offer secure service alternatives to self-hosted memcached, ensure their provider-maintained machine images are kept patched with a default configuration of memcached only available on non-internet-exposed interfaces, and—frankly—not allow memcached to be exposed to the internet on host in their sphere of network control.

Government cybersecurity agencies should provide regular reminders about the dangers of memcached, offer guidance on how to run memcached safely, monitor for malicious use of memcached, and strongly encourage ISPs and cloud providers to block connections to memcached's default port.

etcd (TCP/2379)

Gleaming the Kube(rnetes)

The etcd key-value service is part of the Kubernetes[98] ecosystem and is designed to hold system/service configuration and state information. The Kubernetes API Server uses etcd's watch API to monitor the cluster and roll out critical configuration changes or simply restore any divergences of the state of the cluster back to what was declared by the deployer. It exposes a JSON API over the HTTP protocol.

Discovery Details

Project Sonar found 2,560 etcd nodes exposed to the internet. The counts by country (top 10) and provider are below:

We're including etcd for completeness (since we've mentioned in the previous sections on Redis and memcached), but the sample size is way too small to dig into, since we have no data on which ones are honeypots and which ones are real.

Just like the other two key-value databases, etcd should never be exposed to the internet. Unlike the previous two services, etcd tends to be purpose-driven for Kubernetes orchestration environments, which is another great reason *not* to expose it to the internet directly.

You would not get far on a roadtrip across most any country without roads, bridges, stoplights, maps, and the ability to request and receive directions. The same is true when it comes to the internet. There are quite a number of services we tend to treat as "invisible" that enable us to get from site A to site B or use the internet-enabled apps we've come to rely on. We've chosen a few key ones to see how they're used, how safe they are, and what characteristics each one exhibits.

Domain Name System (DNS) (UDP/53)

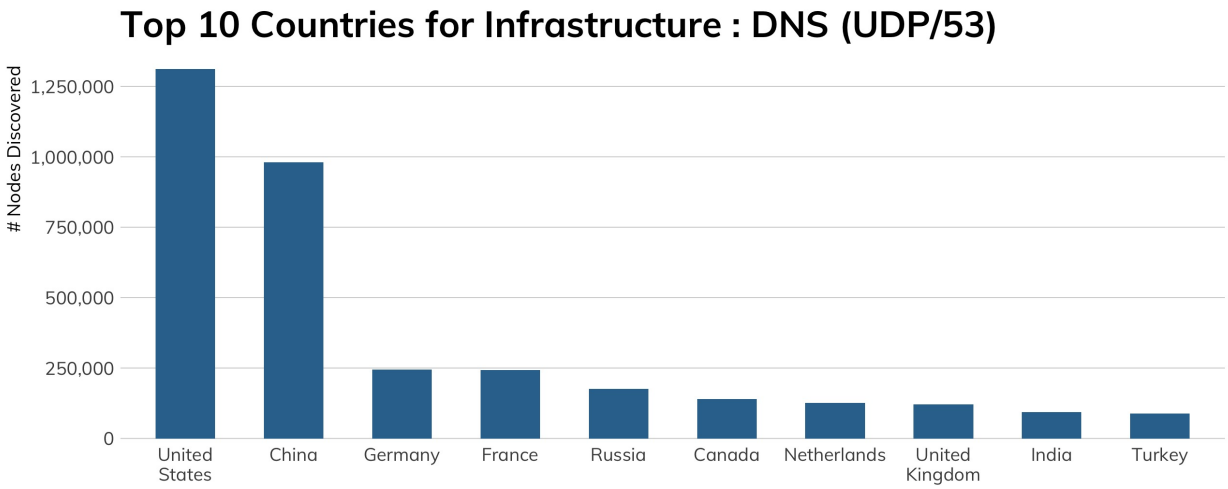
"The Achilles Heel of the Internet" - Sir Tim Berners-Lee

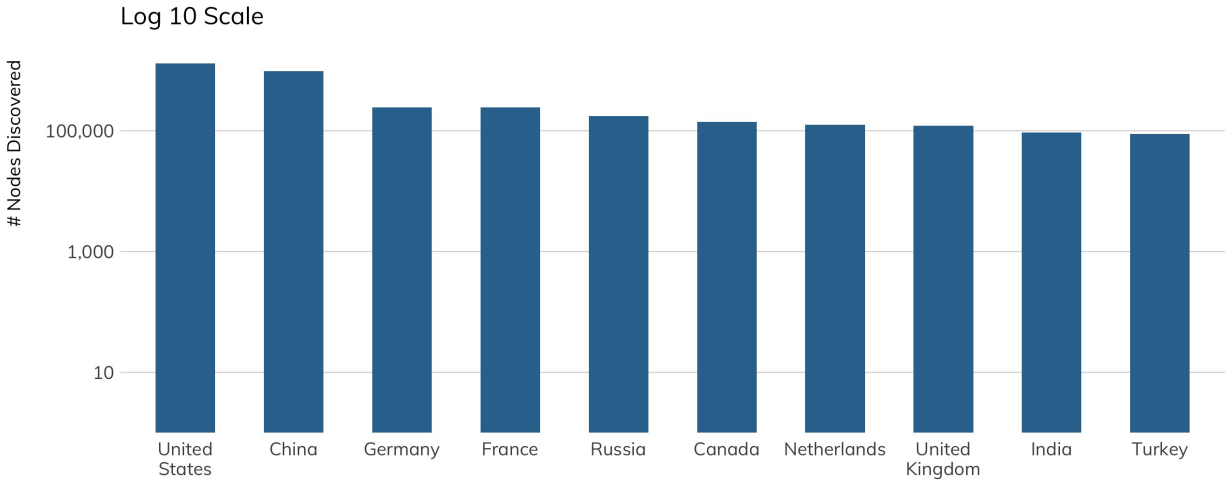
Nobody wants to memorize IP addresses in order to get to network resources, nor does anyone want to maintain a giant standalone list of hostname to IP address mappings. However, nobody also wants to wait forever to get a response to the request for the IP address of, say, example.com. Thus was the atmosphere that begat what we posit is the most ubiquitous user-facing but also most user-overlooked service on the internet: the Domain Name System (DNS).

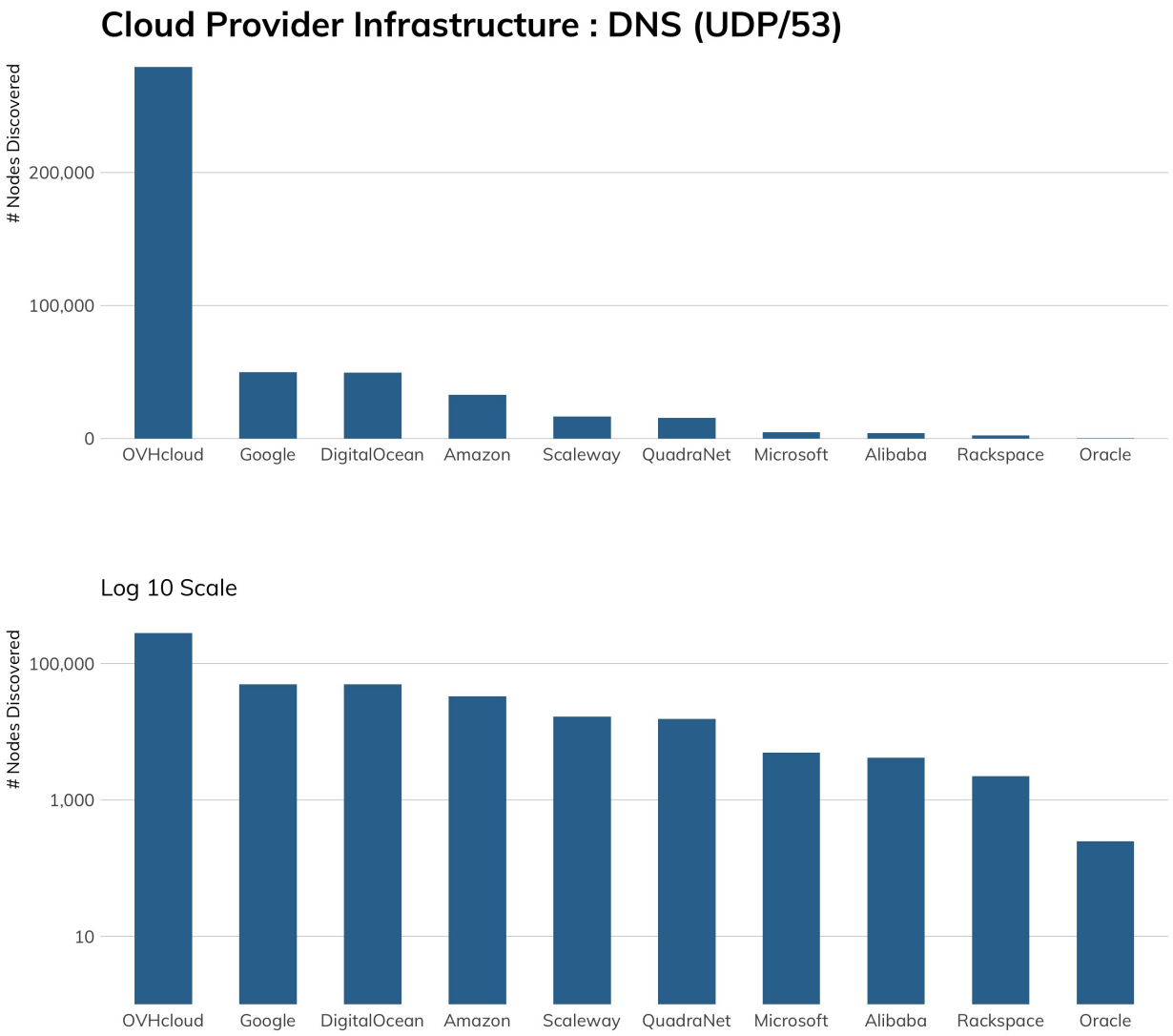
Discovery Details

Project Sonar discovered nearly 5 million DNS servers via UDP requests on port 53. This is a far fewer number than the total sum of, say, web servers, but it is a non-trivial number of systems and the reasons for that make sense. ISPs provide DNS services to home and small-business users, organizations host their own DNS to maintain control of their brand namespace, and vendors provide customized DNS services in either an outsourcing capacity or to provide enhanced services such as malware and other types of content filtering. Finally, large technology companies such as Google, Cloudflare, IBM (via Quad9), and others also provide centralized DNS services for various good (?) reasons. This is all to say, outside of the giant centralized DNS providers, the global DNS footprint tends to track very closely with the

allocated country IPv4 space; the more IP allocations a given country has, the more DNS servers are there to keep track of them all.







Conversely, it really doesn't make much sense to waste precious (and costly) cloud resources by hosting DNS in there. However, it seems OVH users have plenty of cycles (and money) to burn. Yep, those aren't just OVH's DNS servers. We come to that conclusion based on the diversity of DNS vendor software and the version spread. Now, OVH does have the largest data center on the planet[100] and is not just a cloud services provider, so it's pretty reasonable to see that it can and should be in the top spot.

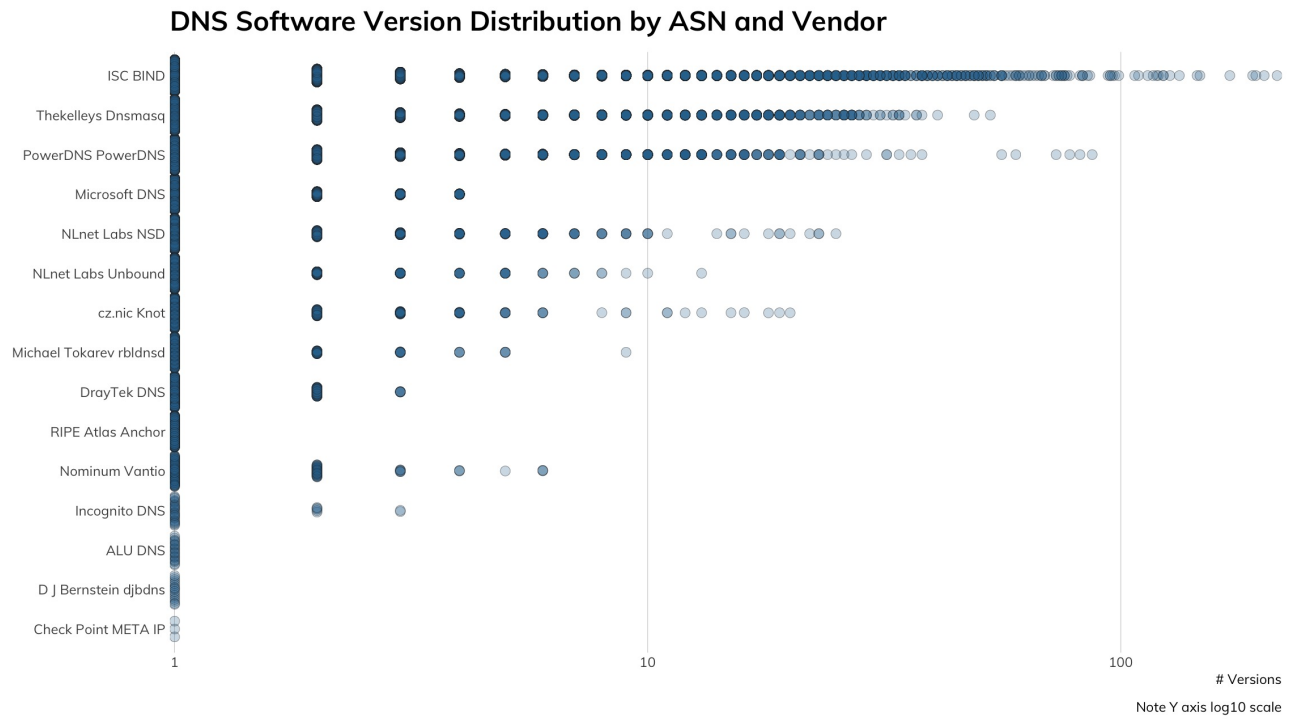
Given that most small orgs use their ISPs' external DNS (directly or via recursive DNS) and that the vast majority of home users still use their ISP DNS, you can imagine that autonomous system DNS server distribution has a very long tail.

Exposure Information

DNS has had … *challenges* … over the years. It is a binary protocol that receives quite a bit of attention paid to it by both researchers and attackers. Because of this, and the nature of the

UDP service, it is possible to craft a binary DNS request that ends up being around 60 bytes that asks for a DNS response, which ends up potentially being near 4,000 bytes (~7:1 amplification), making it great for use in low-to-mid-level amplification DDoS attacks.[101] It is also possible to compromise a DNS server via specially crafted binary messages, though that task gets more difficult with each passing year.

BIND (now ISC BIND) was the first DNS server and is still the most prevalent one (of those we had Recog fingerprints for), which is likely why it has 119 CVEs (most all of them DoS-related). The picture really isn't this clean, though. Within ISC BIND alone, we found 550 *distinct* version strings (most legit, too). We can look at version diversity by vendor across all autonomous systems with DNS servers to see just how crazy the situation really is:



If this were a social media service instead of a serious research paper, now's about the time we'd post a "Do You Even…?" meme gif with the word "Patch" in it. So, not only do we forget about DNS when we're using it, we also seem to forget about it when we run it, too. Denial-of-service flaws are found every year in these servers, but when DNS is running, it's running, and you likely need it to keep running.

Attacker's View

We're not in the DDoS protection services racket market, nor do we have DDoS probes sitting in key locations to be able to detect when DDoS attacks are happening. We see both TCP- and

UDP-based DNS traffic in Heisenberg, but they're mostly inventory scans or misconfigurations.

This is not to say attackers care not about DNS anymore. Every DDoS mitigation vendor makes a point of reminding us about this a few times a year in their service reports, and Verizon noted a serious uptick in DoS in general in 2019[102] (in which DNS played a part). And, there are always new, crafty attack vectors being researched and developed.[103]

But, attackers do more with DNS than just DoS. Organizations must register public, top-level domain names and set up various types of records for them so we can all buy things without leaving home.[104] This exposes two potential avenues of for attack: first at the registrar level, which is why it is **vital** that you protect your domain registration account with multi-factor authentication (preferably app-based for this versus just SMS) and then do the same for your external DNS provider (if you're using an external DNS provider). In May 2020, the Internet Systems Consortium hosted a webinar[105] on this very topic that should help provide more background information, and SpamHaus estimates[106] that GoDaddy has around 100 newly hijacked domains daily.

They who control DNS control who you are on the internet.

Our Advice

IT and IT security teams should safeguard registrar and external DNS provider accounts with multi-factor authentication, keep internal and external DNS systems fully patched, relentlessly monitor DNS for signs of abuse and configuration changes, and consider treating DNS like a first-class application in your environment as opposed to the plumbing that sits hidden behind drywall.

Cloud providers that offer DNS registration and hosting services should mandate multi-factor authentication be used and have processes in place to detect potential malicious activity (i.e., takeover attempts). All machine images with DNS services installed by default should be updated immediately after new DNS server versions are released and then notify all existing users about the need to upgrade.

Government cybersecurity agencies should provide timely notifications regarding DNS attacks of all kinds and have resources available that document how to securely maintain DNS infrastructure.

DNS-over-TLS (DoT) (TCP/853)

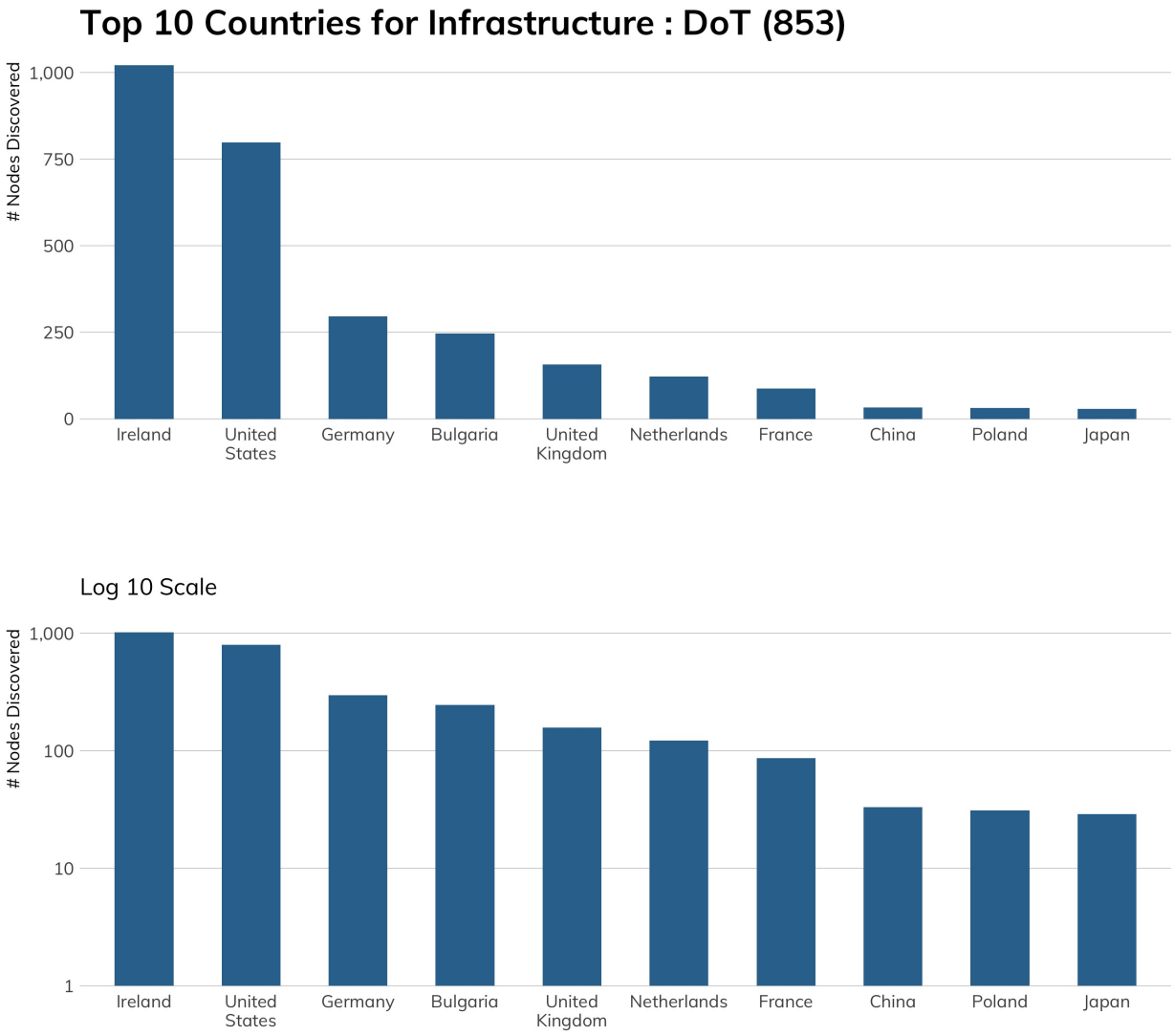
Encrypting DNS is great! Unless it's baddies doing the encrypting.

At face value, DNS over TLS (henceforth referred to as DoT) aims to be the confidentiality solution for a legacy cleartext protocol that has managed to resist numerous other confidentiality (and integrity) fixup attempts. It is one of a handful of modern efforts to help make DNS less susceptible to eavesdropping and person-in-the-middle attacks.

Discovery Details

We chose to examine DoT because web browsers have become the new operating system of the internet, and DoT and cousins all allow browsers (or any app, really) to bypass your home, ISP, or organization's choices of DNS resolution method and resolution provider. Since it's presented over TLS, it can also be a great way for attackers to continue to use DNS as a command-and-control channel as well as an exfiltration channel.

We chose to examine DoT versus DoH because, well, it is far easier to enumerate DoT endpoints than it is DoH endpoints.[111] It's getting easier to enumerate DoH since there seems to be some agreement on the standard way to query it, so that will likely make it to a future report, but for now, let's take a look at what DoT Project Sonar found:



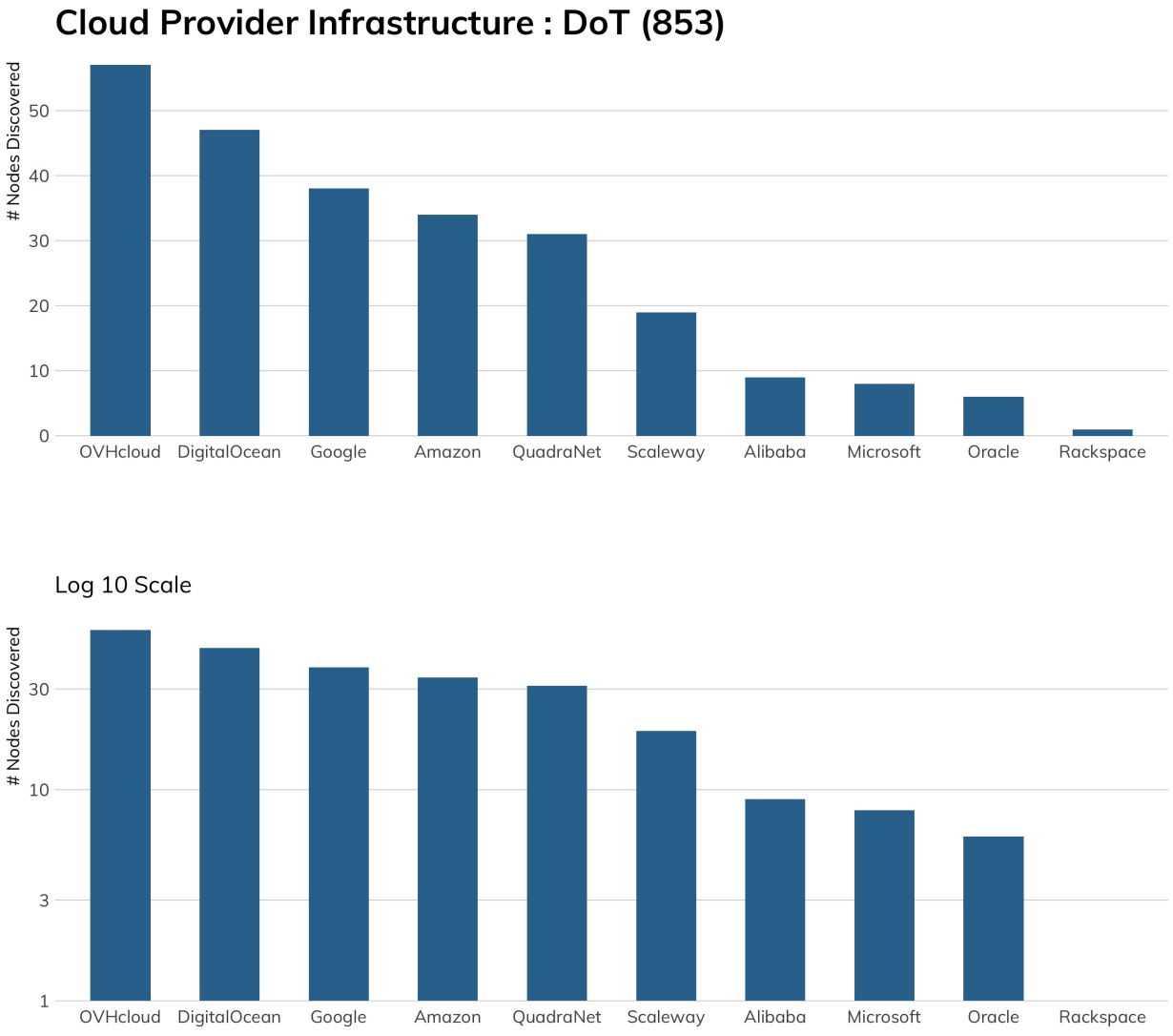
Yes, you read that chart correctly! Ireland is No. 1 in terms of the number of nodes running a DoT service, and it's all thanks to a chap named Daniel Cid[112], who co-runs CleanBrowsing[113], which is a "DNS-based content filtering service that offers a safe way to browse the web without surprises." Daniel has his name on AS205157,[114] which is *allocated* to Ireland, but the CleanBrowsing service itself is run out of California.[115] In fact, CleanBrowsing comprises almost 50% of the DoT corpus (1,612 nodes), with 563 nodes attributed to the United States and a tiny number of servers attributed to a dozen or so other country network spaces.

Both the U.S. and Germany have a cornucopia of server types and autonomous systems presenting DoT services (none really stand out besides CleanBrowsing).

Since Bulgaria rarely makes it into top 10 exposure lists, we took a look at what was there and it's a ton (relatively, anyway: 242) of DoT servers in Fiber Optics Bulgaria OOD,[116] which is a

kind of “meta” service provider for ISPs. Given the relative scarcity of IPv4 addresses, setting aside 242 of them just for DoT is a pretty major investment.

Even though the numbers are small, Japan’s presence is interesting, as it’s nearly all due to a single ISP: Internet Initiative Japan Inc.[117]



In case you have been left unawares, Google is a big player[118] in the DoT space, but it tends to concentrate DNS exposure to a tiny handful of IP addresses (i.e., that bar is not Google-proper). When we filter out CleanBrowsing (yep, they’re *everywhere*), we’re left with the major exposure in Google being … a couple dozen servers running an instance of Pi-hole[119] (dnsmasq-pi-hole-2.80, to be precise). Cut/paste that finding for OV and DigitalOcean and yep, that same Pi-hole setup is tops in those two clouds as well.

You don't need to get all fancy and run a Pi-hole setup to host your own DoT server. Just fire up an nginx[120] instance, create a basic configuration,[121] set up your own DNS[122] behind it, and now, you too can stop your ISP from snooping your DNS queries.

Exposure Information

Here is where we'd normally talk about versions and CVEs, etc., but the DoT situation is complicated by a few things. First, we have big players in this space using proprietary solutions, so version fingerprints such as "CleanBrowsing v1.6a" are not very useful information. Second, should we focus on the version of the web server or of the back-end DNS server (or, both)? The latter might not be useful, since you can configure an nginx DoT setup to proxy to a third party, and that's what will get picked up in the response. Lastly, even if we focus on the second-tier "big guns," such as PowerDNS,[123] we end up with a situation like this:

Table 36

```
PowerDNS
PowerDNS Authoritative Server 4.0.9 (built Jul 31 2019 00:17:38 by buildbot@4842127c3c
PowerDNS Authoritative Server 4.1.1
PowerDNS Authoritative Server 4.1.10 (built Jun 20 2019 23:00:19 by root@e0846b1bda55)
PowerDNS Authoritative Server 4.1.11
PowerDNS Authoritative Server 4.1.3 (built May 24 2018 12:54:13 by root@cebf8df1a8ce)
PowerDNS Authoritative Server 4.1.4 (built Aug 29 2018 14:08:38 by root@7d1469b19fba)
PowerDNS Authoritative Server 4.1.6
PowerDNS Authoritative Server 4.1.6 (built Feb  6 2019 14:44:29 by root@FreeBSD-Core-d
PowerDNS Authoritative Server 4.2.0 (built Jan  5 2020 00:41:27 by nobody@pkg.ssnet.ca
PowerDNS Authoritative Server 4.2.0 (built Jan 27 2020 11:34:42 by root@113amd64-quart
PowerDNS Authoritative Server 4.2.1
PowerDNS Authoritative Server 4.2.1 (built Nov 29 2019 11:55:12 by root@2ccf7cb0fe12)
PowerDNS Authoritative Server 4.2.1 (built Nov 29 2019 13:34:28 by root@f06a46cf444e)
PowerDNS Authoritative Server 4.3.0-beta2 (built Feb 19 2020 01:48:39 by root@eacdc4c9
PowerDNS Authoritative Server 4.3.0-rc2 (built Mar 18 2020 20:04:33 by root@freebsd_12
PowerDNS Authoritative Server 4.4.0-alpha0.189.master.gdbcbb6820 (built Mar 27 2020 18
PowerDNS Recursor
PowerDNS Recursor 3.7.3 (jenkins@autotest.powerdns.com built 20151009082750 suresh@)
```

Giving you that glimpse does help to show it's utter chaos even in PowerDNS-land, but DNS and chaos seem to go hand in hand.

Attacker's View

There are no DoT honeypots in Heisenberg, but DoT is just a TLS wrapper over a traditional DNS binary-format query. When we looked for that in the TCP/853 full packet captures, we saw us (!) and a couple other researchers. Not very exciting, but with the goal of DoT being privacy, we really shouldn't see random DoT requests.

Attackers are more likely to stand up their own DoT servers or reconfigure other DoT servers to use their DNS back-ends and then use those as covert channels once they gain a foothold after a successful phishing attack. This is a big reason we enumerate/catalog DoT, and we're starting to see more DoT in residential ISP space and traditional hosting provider IP space. It looks like more folks are experimenting with DoT with each monthly study.

Our Advice

IT and IT security teams should block TCP/853, lock down DoT and DoH browser settings as much as possible so there is no way to bypass organizational IT policies, and monitor for all attempts to use DoT or DoH services internally (or externally). In other words, unless you're the ones setting them up, disallowing rogue, internal DoT is the safest course.

Cloud providers should consider offering managed DoT solutions and provide patched, secure disk images for folks who want to stand up their own. (This is one of the few cases where organizational advice and cloud advice are quite nearly opposite.)

Government cybersecurity agencies should monitor for malicious use of DoT and provide timely updates to the public. These centers should also be a source of unbiased, expert information on DoT, DoH, DoQ (et al).

NTP (123)

In the immortal words of The Smiths, "How soon is now?"

The internet could not function the way it does without NTP. You'd think with that much power NTP would be all BPOC[125] and act all smug and superior. Yet, it does it's thing—keeping all computers that use it in sync, time-wise—with little fanfare, except when it's being used in denial-of-service attacks. It has been around since around 1985, and while it is not the only network-based time synchronization protocol, it is The Standard.

NTP servers operate in a hierarchy with up to 15 levels dubbed *stratum*. There are authoritative, highly available NTP servers we all use every day (most of the time provided by operating system vendors and running on obviously named hosts such as time.apple.com and time.windows.com).

Virtually *anything* can be an NTP server, from a router, to your phone, to a RaspberryPi, so dedicated appliances that key off of GPS signals as a time-source. Now, just because something *can* be a time server does not mean it *should* be a time server.

Discovery Details

Project Sonar found 1,638,577 NTP servers on the public internet, so one might say we have quite a bit of time on our hands.[126] Our editors say otherwise, so let's see what time looks like across countries and clouds.

The United States has many IPv4 blocks, many computers, and many major ISPs and IT companies that like to control things. It also has a decent number of businesses that run NTP for no good reason. All of this helps push it to the top spot. Russia *finally* shows up in second place, for similar reasons, though two of Russia's major ISPs account for just over 40% of Russia's exposure. China—with its vast IPv4 space and population—comes in at No. 3, which means businesses and ISPs have figured out needlessly exposing NTP can cause more problems than it's worth.

Rapid7 Labs was glad to see cloud environments (both the runners and the customers) seem to take the dangers of running NTP seriously as well, with most having almost no exposure.

Exposure Information

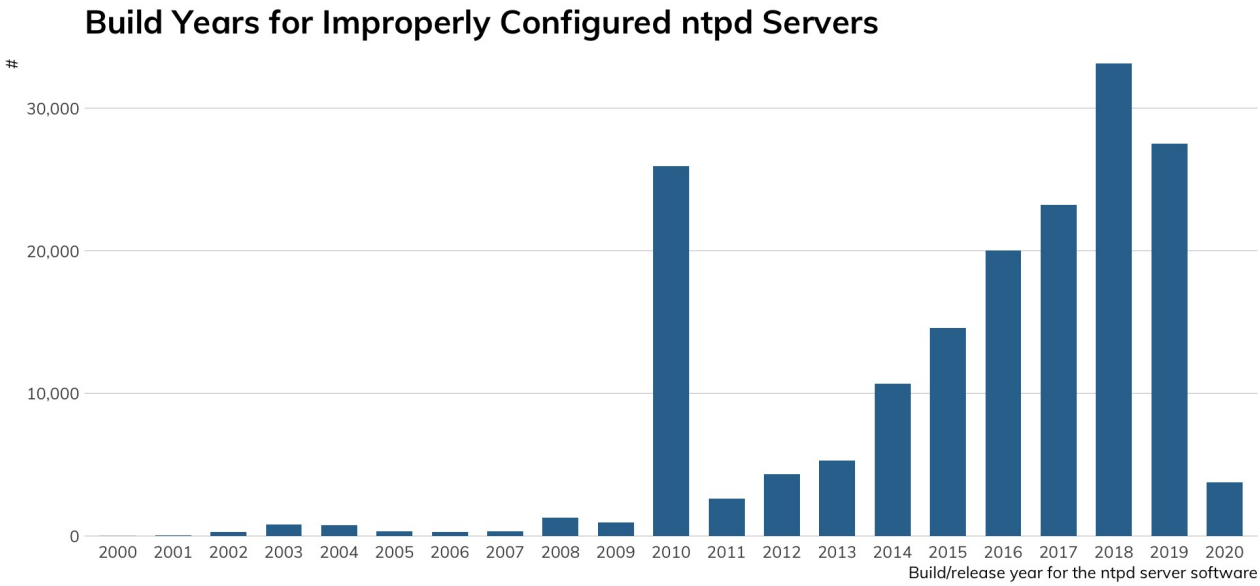
Now, you know NTP has to be a *bit* dangerous if the main support site for the protocol itself has a big, bad warning about the dangers of NTP right at the top of its page.[127] The biggest danger comes from using it for amplification DDoS (it is a UDP-based protocol). While it is still used today, there are way better services, such as memcached, to use for such things.

NTP servers are just bits of software that have vulnerabilities like all other software. When you put anything on the internet, bad folks are going to try to gain control over it. If an organization needs—for some odd reason—to run its own NTP server, there's no reason it has to be on the

public internet. And, if there is some weird reason it does, there's no reason it has to be configured to respond to requests from all subnets.

Why are we picking nits? Well, it's one more thing you're not going to patch. Then, there's the problem of all the information you might be giving to attackers about your network setup. In our NTP corpus, 255,602 (15.5%) reveal the private IP address scheme on the internal network interface.

Over 1.5 million NTP servers give hints about the operating system and version they run. In total, 180,410 (11%) give us precise NTP version and build information, with all but roughly 4,000 giving us the precise release date:



There's an [un]healthy mix of remote code execution, information leakage, local service DoS, and amplification DDoS spread throughout that mix of NTP devices.

Hopefully we've managed to at least start to convince you otherwise if you were thinking, "Well, it's *just* an NTP server" at the start of this section.

Attacker's View

The **Exposure Information** section provided a great deal of information on the potential (and measured) weaknesses in NTP systems. Attackers will judge your potential as a victim (and cyber-insurers will likely up your premiums) from how your attack surface is configured. NTP

can reveal all the cracks in your configuration and patch management processes, and even provide a means of entry.

And, attackers still use NTP in amplification attacks, so that NTP server you didn't realize you had or really thought you needed will likely be used in attacks on other sites.

Our Advice

IT and IT security teams should use NTP behind the firewall and keep it patched. If you do need to run NTP externally, only let it talk to specific hosts/networks.

Cloud providers should keep up the great work by only exposing as much NTP as they need to and offering guidance to customers for how to run NTP securely (off the internet).

Government cybersecurity agencies should provide timely notifications when new vulnerabilities in NTP surface or there are known, active NTP DoS campaigns. Educational materials should be made available on dangers of exposing NTP to the internet and on how to securely configure various NTP services.

Last year (2019) marked the 30th anniversary of the World Wide Web. Tens of thousands of bloggers and news outlets and vendors told you about it and then opined incessantly about what the web is and where it's going, so we'll save you from such exposition. "Web servers" is a very loaded term, since at their core, they are pieces of software that receive Hypertext Transfer Protocol (HTTP) protocol requests and deliver responses. They front-end most everything today, from components in traditional server environments that deliver ranty blog posts and cat pictures to conduits for API requests that fit on a microchip inside your refrigerators, network routers, doorbells, automobiles, or drones. Because they are fairly diminutive and ridiculously easy to interact with, they are *everywhere*.

In this section, we focus on encrypted (HTTP 443) and encrypted (HTTP 80) **internet-facing** web servers. We *kind of* already covered some of this a bit back in the Remote Access section when we talked about Citrix servers, since the initial connection (from Citrix clients) and response comes from a web server.[128] Unlike the dwarves of Moria, we will not delve *too* deeply in this section, since there's only so much you can say about exposure from lightweight HTTP probes.[129]

HTTP (TCP/80) & HTTPS (TCP/443)

One protocol to bring them all, and in the darkness, bind them.

We're going to talk about both HTTP and HTTPS combined (for the most part) as we identify what we found, some core areas of exposure, and opportunities for attackers. It'll be a bit different than all the previous sections, but that's just part of the quirky nature of HTTP in general.

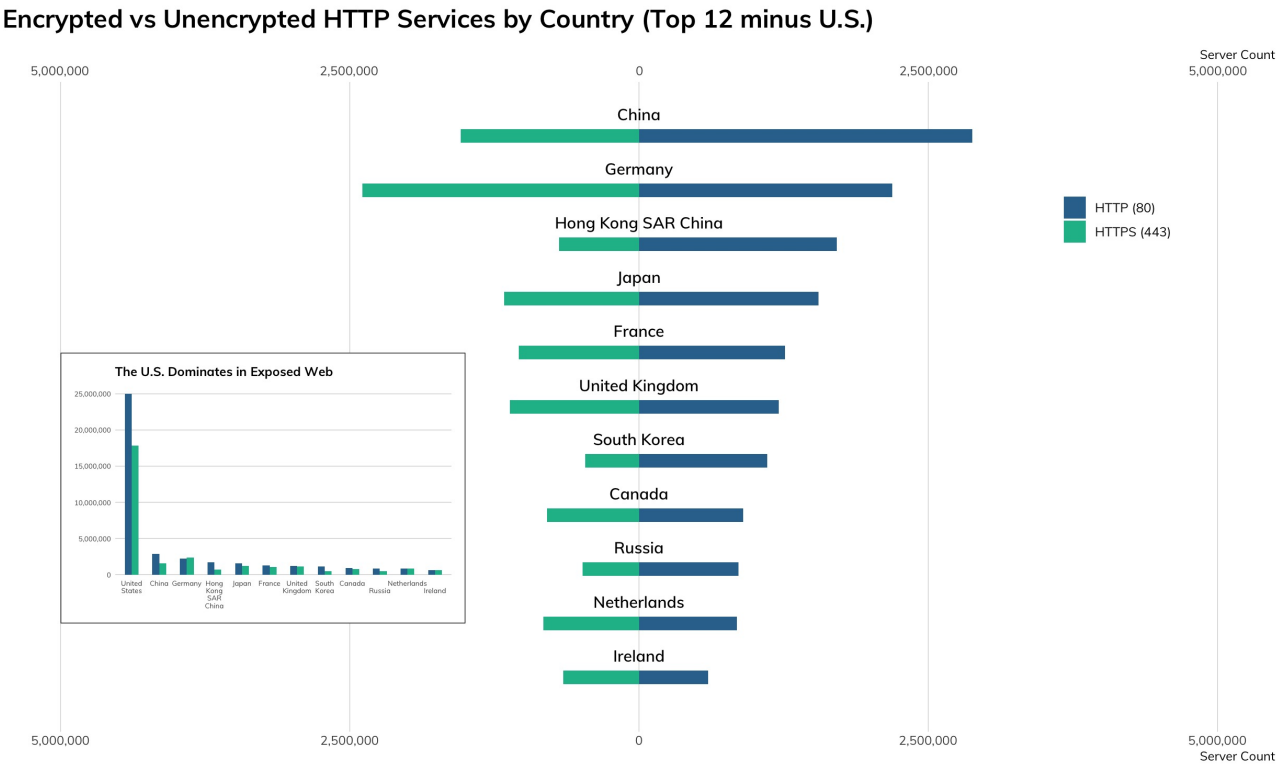
Discovery Details

Way back in our Email section, we compared encrypted and unencrypted services. We'll do the same here, but will be presenting a "top 12" for countries since that is the set combination between HTTP and HTTPS.

There are 30% more devices on the internet running plaintext HTTP versus encrypted HTTPS web services. The U.S. dwarfs all other countries in terms of discovered web service, very likely due to the presence of so many cloud services, hosting providers, and routers, switches, etc. in IPv4 space allocated to the U.S.

Germany and Ireland each expose 9% more HTTPS nodes than HTTP, and both the Netherlands and U.K. are quickly closing their encryption disparity as well.

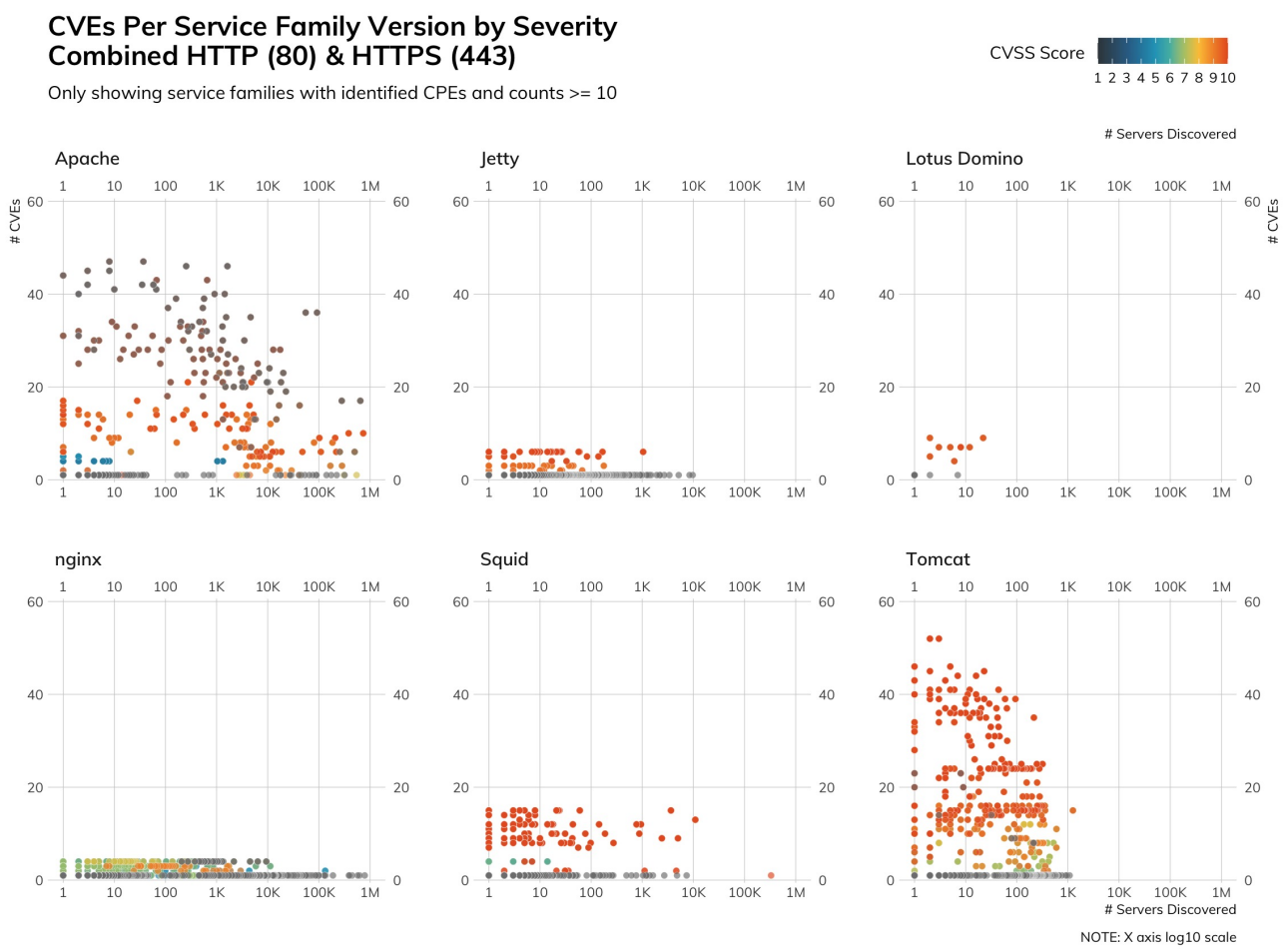
We'll skip cloud counts since, well, everyone knows cloud servers are full of web servers and we're not sure what good it will do letting you know that Amazon had ~640K Elastic Load Balancers (version 2.0!) running on the day our studies kicked off.



Exposure Information

To understand exposure, we need to see what is running on these web servers. That's not as easy as you might think with just lightweight scans. For example, here are the top 20 HTTP servers by vendor/family and port:

Remember, we're just counting what comes back on a `GET` request to those two ports on each active IP address, and the counts come from Recog signatures (which are great, but far from comprehensive). For some servers, we can get down to the discrete version level, which lets us build a Common Platform Enumeration[132] identifier. That identifier lets us see how many CVEs a given instance type has associated with it. We used this capability to compare each version of each service family against the number of CVEs it has. While we do not have complete coverage across the above list, we do have some of the heavy(ier) hitters:



We limited the view to a service family having at least having 10 or more systems exposed and used color to encode the CVSS v2 scores.

The most prevalent CVE-enumerated vulnerabilities are listed in the table below. While it's technically possible that these CVEs have been mitigated through some other software control, patching them out entirely is really the best and easiest way to avoid uncomfortable conversations with your vulnerability manager.

And, the top 30 most prevalent are:

Table 39

While we expect to see traditional web servers, there are other devices connected to the internet that expose web services or administrative interfaces (which we've partially enumerated below):

For instance, we found nearly a million Cisco ASA firewalls. That fact is not necessarily "bad," since they can be configured to provide remote access services (like VPN). Having 123

instances on port 80 is, however, not the best idea.

Unlike Cisco, most MikroTik routers seem to be exposed sans encryption, and over 75% of them are exposing the device's admin interface.[133] What could possibly go wrong?

Upward of 50,000 Synology network-attached storage devices show up as well, and the **File Sharing** section talked at length about the sorry state of exposure in these types of devices. They're on the internet to enable owners to play local media remotely and access other files remotely.

There are printers, and light bulbs; DVRs and home router admin interfaces; oh, and a few thousand *entire building control systems.*[134] In short, you can find pretty much anything with a web interface hanging out on the internet.

Attacker's View

There are so many layers in modern HTTP[S] services that attackers likely are often paralyzed by not knowing which ones to go after first. Attacking HTTP services on embedded systems is generally one of the safest paths to take, since they're generally not monitored by the owner nor the network operator and can be used with almost guaranteed anonymity.

Formal web services—think Apache Struts, WebLogic, and the like—are also desirable targets, since they're usually associated with enterprise deployments and, thus, have more potential for financial gain or access to confidential records. HTTP interfaces to firewalls and remote access systems (as we saw back in the **Remote Access** section) have been a major focus for many attacker groups for the past 18–24 months since once compromised, they can drop an adversary right into the heart of the internal network where they can (usually) quickly establish a foothold and secondary access method.

You're also more likely to see (at least for now) more initial probes on HTTP (80), as noted by both the unique source IPv4 and total interaction views (above). It's hard to say "watch 80 closely, and especially 80→443 moves by clients," since most services are still offered on both ports and good sites are configured to automatically redirect clients to HTTPS. Still, if you see clients focus more on 80, you may want to flag those for potential further investigation. And, definitely be more careful with your systems that only talk HTTP (80).

Our Advice

IT and IT security teams should build awesome platforms and services and put them on the internet over HTTPS! Innovation drives change and progress—plus, the internet has likely done more good than harm since the first HTTP request was made. Do keep all this patched and ensure secure configuration and coding practices are part of the development and deployment lifecycles. Do **not** put administrative interfaces to *anything* on the internet if at all possible and ensure you know what services your network devices and "Internet of Things" devices are exposing. Finally, disable `Server:` banners on everything and examine other HTTP headers for what else they might leak and sanitize what you can. Attackers on the lookout for, say, nginx will often move on if they see Apache in the Server header. You'd be surprised just how effective this one change can be.

Cloud providers should continue to offer secure, scalable web technologies. At the same time, if pre-built disk images with common application stacks are offered, keep them patched and ensure you have the ability to inform users when things go out-of-date.

Government cybersecurity agencies should keep reminding us not to put digital detritus with embedded web servers on the internet and monitor for campaigns that are targeting these invisible services. When there are major issues with core technologies such as Microsoft IIS, Apache HTTP, or nginx, processes should be in place to notify the public and work with ISPs, hosting, and cloud providers to try to contain any possible widespread damage. There should be active programs in place to ensure no critical telecommunications infrastructure has dangerous ports or services exposed, especially router administrative interfaces over HTTP/HTTPS.

First, and foremost, the sky most certainly is not falling. Sure, each and every service we've reported on is extremely messy, and we humans are doing a terrible job at maintaining service currency and configuration safety. But, we're fairly certain you and your family are reading, watching, and posting to Facebook, Twitter, Instagram, TikTok, et. al. every day just fine, and that you've performed a number of online banking and online purchasing transactions within the last four weeks with nary a care in the world about their security, and that you've used countless SaaS applications with or without your direct knowledge. All of that was done despite deep levels of inherent vulnerability that lie across the entirety of the internet.

Sure, bad things happen, like 23,000 denial-of-service attacks per day (16 every minute),[135] Magecart javascript injections[136] on sites with exposed vulnerabilities, and EternalBlue-based exploits still work and are still prevalent[137] all these years after WannaCry. While these

disruptions no doubt inflict varying levels of pain on the victims (both consumers/visitors and operators/businesses), the internet—for the most part—moves ever on and on for the vast majority of users.

We are not suggesting that the status quo is okay. Far from it. But, we do feel more like the environmental scientists who continue to warn about the real and present danger of global warming or health professionals who implore us to wear masks and social distance. We're pounding the same drum, just to a different beat. We are crawling, not racing, toward systematic disaster with the way we're managing the global internet. Things aren't great, but not disastrously bad, and relatively small changes in how we design, develop, and deploy services will still have a great impact on the stability, safey, and security of the internet as a whole.

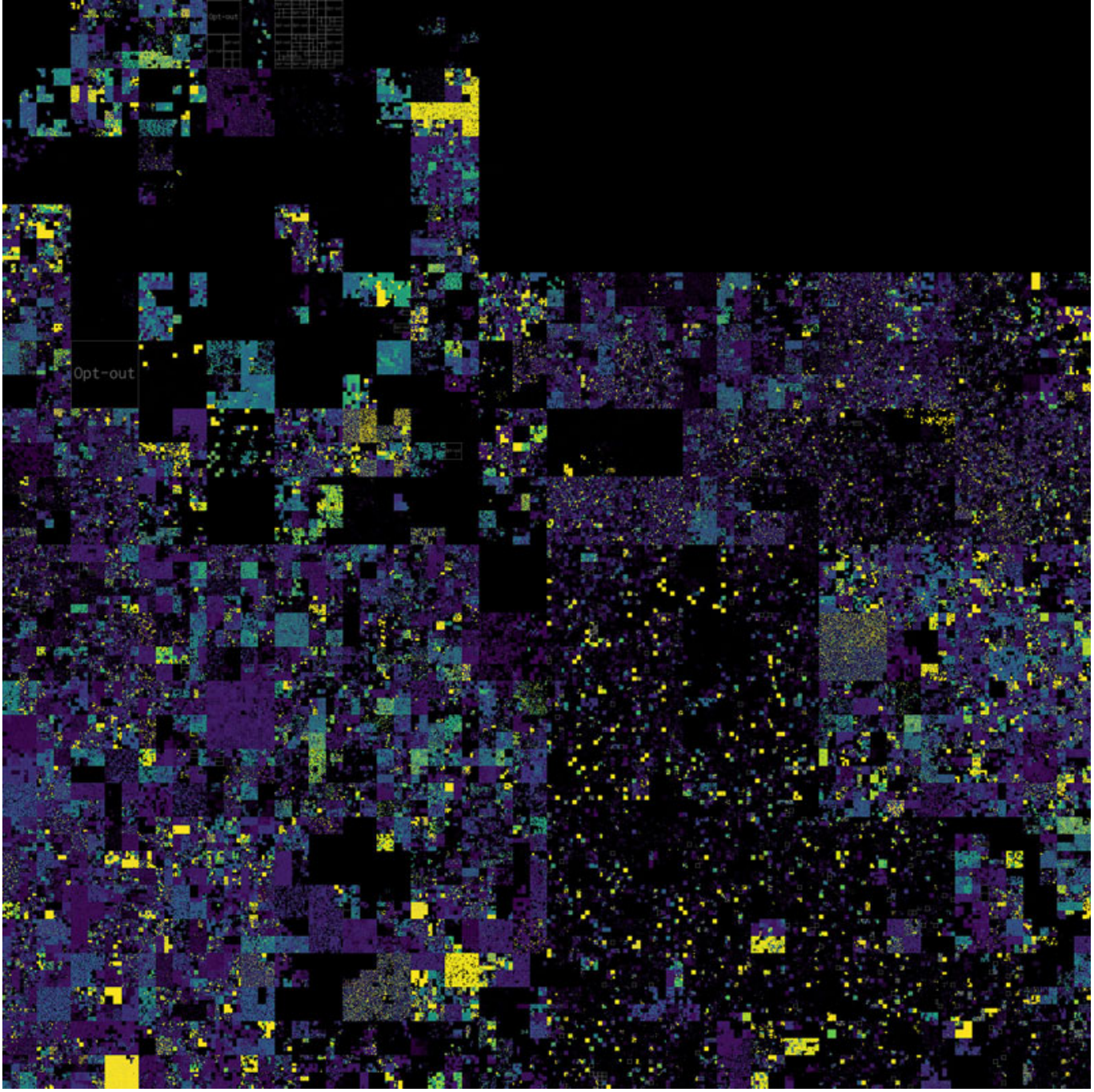
If you put anything on the internet, it should be:

- **Exposed deliberately.** You meant to expose it versus accidentally exposed it.
- **Configured competently.** The configuration is secure and designed to perform only the necessary tasks.
- **Patched regularly.** You need to make a real effort to keep with current version levels and especially when there are critical vulnerabilities identified. You also need to go out of your way to ensure you can pull patched versions even when your default package repositories stop updating.
- **Monitored mindfully.** You have just increased the attack surface of both yourself/your organization and the internet as a whole; as such, you are inherently responsible for doing your part in the defense of those entities.
- **Assumed attacked.** Researchers aren't the only ones looking for services, and attackers do not have the ethical and legal restrictions we do, so they'll look harder and attack at will. You cannot assume that your services will only receive benign interactions using the methods you outlined in your benign use-cases.

We're not saying you should not innovate or experiment. By all means, invent and release brand-new protocols and services **with a hostile internet in mind**. Such services that expect attack and failure, and compensate for such events, will thrive on the future internet. In the end, any new service you deploy may fall under the control of attackers who are smart, resourceful, and relentless, so engineer accordingly.

The internet has grown chaotically and explosively since its inception back in the 1960s.[138] The 1,000th node was connected to the internet when it was 18 years old, in 1987. In 2020, the internet is over 50 years old, and we are able to discover 330,087,843 active nodes that are offering some sort of service on the internet—web servers, file sharing sites, streaming video services, terminal consoles, and anything else capable of sending network data. If every computer server were a citizen of The Cybercratic Machines' Republic of the Internet, that country would be the fourth most populated country in the world.

At its most fundamental level, the internet runs on the Internet Protocol, or IP, which is the basic addressing and routing system to reach machines hosted in different networks. It stands to reason, then, in order to be on "the internet," a node or resource needs to be reachable by this lingua franca of public addressing via an "IP address." Below is a representation of the entire population of every four-byte IPv4 address, with each pixel representing one block of 255 individual addresses, and colored dark to light depending on how many of those nodes respond to scanning. All together, there are 2^32 possible addresses, or about 4.2 billion.



As one can see, not every address in the 2^32 bit space is actually open for connections. For starters, the large black rectangles represent reserved and non-public addresses. You might see these addresses, such as 10.1.1.1 and 192.168.0.1, on your local network, but they are not directly reachable over the internet under normal circumstances. Other dark areas are either unpopulated, not responsive to probes, or on our blocklist of those networks that have opted out of scanning (more on that below). These unscannable areas aside, the internet is pretty densely packed in some of its more populous regions, and it's changing more rapidly than ever.

Measurement Strategy and Tools

Collecting telemetry to grasp the makeup of the internet is no easy task. As mentioned above, the total possible IPv4 internet consists of 4.2 billion addresses, 3.7 billion of which are actually

assignable on the public internet.[139] Over the past two decades, as computing power has increased alongside advancements in networking, obtaining an accurate picture of the internet went from an impossible task to one that could be completed in under five minutes.[140] As it would turn out, coupled with new technology that embraces dynamism, such as cloud computing, incredibly fast scanning is not only necessary, but insanely important to obtain a picture that is as accurate as possible of a vast, churning internet. In other words, don't count on one IPv4 address with its set of services to look the same the next time you look. In fact, prior internal Rapid7 research (associated with our Heisenberg honeypots) has shown that IPv4 address leases could change hands and character within seconds.

While we're discussing IP address space, let's take a minute to appreciate IPv6, the newer[141] and expanding addressing and routing scheme for the internet. Unlike IPv4's 2^32 space, IPv6 has space for 2^128 addresses. Just for fun, that's 340,282,366,920,938,463,463,374,607,431,768,211,456 IPv6 addresses, compared to IPv4's much more fathomable 4 billion. Let that set in (actually, you can't, but don't feel too bad about it).

However, IPv6 address adoption by clients still stands at around 31% as of the time of writing of this report, according to Google[142] (and that is counting physical systems that are addressable by both IPv4 and IPv6), and nearly all IPv6 services today are also reachable over the IPv4 network. So, while keeping an eye on IPv6 is definitely important, it's still reasonable to assume that the IPv4-addressable internet is "pretty much the whole internet," and we don't believe there is a totally separate and invisible-to-Sonar IPv6 server-side internet out there.

Now that we've discussed the challenges of active scanning, let's talk a little about passive internet telemetry collection. Passive collection typically involves sitting a computer on the internet and ... listening. Yep, really that's it. In fact, pretty much any device you have sitting on the internet is serving this purpose, right now, attracting all sorts of nonsense, such as proxy requests for scraping nike.com (yes, seriously), and brute force attempts against SSH on TCP/22. Check your web server logs if you don't believe us. One specialized form of passive collection, a honeypot, involves placing a device deliberately on the internet to entice someone or something to attack or probe it, and trap deeper behavior you wouldn't otherwise normally be able to track. As it turns out, this behavior gives you a wealth of information, including building an understanding of post-exploitation behavior, trapping malware samples for later analysis, and understanding common non-malicious errors made in computing infrastructure, such as when

someone forgets to remove that dynamic cloud IP they no longer own from receiving sensitive application messages.

Lastly, once you obtain your telemetry through active and passive means, you need a way to enrich it to understand more about the overall complexion, or general aspect and character, of the internet. Being able to determine that an IP address is hosting a specific OS, like Windows ME (let's hope not), is critically important to understand the types of devices and extrapolate risk based on past weather.[143]

That thing called Sonar

Rapid7 has run Project Sonar, an active internet scanning project, since 2013. Project Sonar was created to measure IPv4 address space, as well as Forward (FDNS) and Reverse (RDNS) DNS records.[144] IP scanning activity today focuses entirely on IPv4-based services, though support for IPv6 is considered for future work, as adoption continues to increase.

Project Sonar scans, otherwise known to the Rapid7 Labs research team as "Studies," are run on regular intervals. These intervals differ, however, depending on the type of study. For example, Rapid7 runs HTTP scans every two weeks across a variety of ports, while we run FDNS and RDNS weekly. This amounts in a staggering amount of data, easily in the petabyte range when considering complete history. One of our largest studies, HTTP/80, routinely comes in around 50–60GB as a compressed single gzipped file, each row of the uncompressed results containing a unique response from an interaction from a server in IPv4 address space.

The amount of IPv4 addresses monitored by Project Sonar are further reduced through a blocklist Rapid7 implements and honors when inbound email requests ask us to refrain from scanning certain IP address ranges. This blocklist represents around 20 million–30 million IPv4 addresses, a small, but notable set.

Project Sonar scans are run from our data centers in San Diego, California, and London. Running scans from multiple locations allows us to collect telemetry from different "perspectives" of the internet, while also balancing scan load across multiple hosts for more efficient collection. Rapid7 Labs does not routinely analyze the difference in perspectives of the internet from these data center locations today, though this is an area of future research.

Now may also be a great time to mention that a large amount of Project Sonar data is available free of charge for academic, research, or practitioner (use for your own security, corporate, or

otherwise) use cases at the Open Data website.[145]

Heisenberg, too

Project Heisenberg is a net of honeypots Rapid7 has deployed across the globe to gain regional threat intelligence and understand internet weather. As mentioned earlier, this honeynet listens on the internet for unsolicited attacks against each honeypot in the network. These attacks are aggregated and sent to Rapid7 for enrichment and analysis. Rapid7 has 120 honeypots deployed across five continents.

One critical component of the Heisenberg honeynet is remaining unidentified as a honeypot, as to not tip off attackers to its capabilities. For this reason, some of the methods it deploys will remain secret and excluded from this report. However, suffice to say, Heisenberg routinely traps credentials used in brute force attacks, exploitation attempts, proxy traffic, and unintentional information leakage due to configuration errors from other adjacent infrastructure.

And then there's Recog

The last piece of the picture is Project Recog. Recog is an open source fingerprint database[146] that can be used to turn identifying protocol details (such as banners, version strings, and other technical hints) collected during internet scans into normalized operating system, software, hardware, and service fingerprints. Recog spun out of Rapid7's Nexpose (now known as InsightVM), from its earlier days. Recog is used by Rapid7's InsightVM and Metasploit products, and remains to this day one of several important information sources for identifying and classifying the makeup of the internet.

Recog supports a variety of protocols and protocol responses. For example, for HTTP replies, we can turn headers, cookies, and even HTTP body title tags into some type of fingerprint. More concretely, if we see a Server header come back with a value of `Microsoft/IIS 6.0`, we can not only deduce IIS 6.0 is running[147] on the host running that HTTP service, but also that the host is running Windows 2003, an obsolete operating system. This knowledge allows us to estimate or, in some cases, directly identify risks related to global infrastructure we observe.

Despite what you may believe, see, or have tried yourself, measuring the internet is difficult. But, what do we—Rapid7 Labs—mean by "measuring the internet?" As it relates to the components of this report, we define "measure" as "enumerate all the internet IPv4 endpoints at

a given point in time—usually between 30 and 240 minutes, depending on the day, time, and what we're looking for—that respond positively to a request using protocol X." First, we perform a probe against all of IPV4 space to see if there's something listening at all on a given port using ZMap,[148] then perform a protocol-level request against all nodes that responded to the initial probe.

While one might be inclined to believe this is a full/complete inventory, it isn't. It's really a sample from a superpopulaton of nodes connected to the internet. Why isn't it complete? Well, just like in ecology, when those scientists attempt to count all the squirrels in a given forest region, they do this by looking at them in defined block areas. They most certainly aren't counting every squirrel with whatever devices they're using, but they are getting a statistically usable sample count that they can work with.

Similarly, the internet is, to some degree, as dynamic as a natural forest. Portions of the internet go down regularly,[149] the cloud enables organizations to rapidly provision and deprovision resources, many of which will be observable from the internet, and many ISPs also continue to use short-lifespan IPv4 DHCP assignments. Plus, for some dangerous services, such as SMB, ISPs often work with national authorities to block access to those resources. Finally, Rapid7 Labs honors opt-out requests (as seen in our massive IPv4 heatmap), so we miss whatever is in those blocks.

We note this to caution against making 100% definitive statements based on any internet scan data, including ours. This is why you see rounding and estimates used throughout the report. The Labs team work diligently to ensure the efficacy of our Sonar studies, but has little control if a given country (or weather event) decides to take down the internet somewhere as scans are being performed.

Unless otherwise noted, all the measurements were taken from April 2020 Sonar studies in an effort to perform measurements based on the "new (for the moment) normal" after the 2020 SARS-CoV-2 global pandemic.

Why We Chose the Services We Chose

Readers familiar with our previous work on the National Exposure Index (NEI)[150] will note a significant difference between those research efforts and this one, most prominent of which are the use of full protocol scans. The NEI papers of yore used plain SYN scans, which, while

useful, don't paint as accurate a picture of specific protocol exposure. Folks can, and do, stick all kinds of network services on every imaginable, usable TCP and UDP port. By switching to showing results from full protocol scans, we paint a far more accurate picture of those specific services, and can—where we have Recog fingerprints and where said services expose enough information to use those fingerprints—provide a rich set of detail that we can use to approximate the relevance/severity of the exposure of those services.

The NEI series relied on the statistical service encounter frequencies from nmap.[151] For NICER, the research team wanted to help create a usable atlas/reference of core, critical, and/or colloquially common services to both help paint a picture of exposure and and highlight areas they felt were important in 2020. This is by no means an exhaustive list of what Sonar studies on a regular basis, nor is it the totality of the most important services. However, we believe it strikes the right balance of importance and usability (in other words, the report is already fairly comprehensive and also fairly long, and making it longer would not have necessarily made it better).

Why We Focused on Countries

Like it or not, country-level autonomous system/CIDR assignments are still the most accurate types of IPv4 entity resolution we commonly have. It is also far from perfect, with major industrial nations, such as the United States and United Kingdom, having a higher IPv4-to-country attribution accuracy (~85–92+%) than those with less modern infrastructure.[152] We use MaxMind[153]for all country/region-level entity resolution, with said mapping performed at the time of the Sonar study.

As noted in various sections, we believe civilian cybersecurity government authorities can and should help make the internet safer. Thus, having an understanding of what exposure looks like in their jurisdictions should help representatives craft better guidance and set up more complete education, monitoring, and alerting programs to help with this safety mission.

Why We Focused on Clouds

An ever-increasing percentage of mission-critical business workloads are moving to cloud environments for all the reasons every bit of advertising you receive about the benefits of "the cloud" have told you a million times already, so we won't bore or annoy you by re-enumerating them. With providers such as Amazon and Microsoft each having a customer base exposing

services across over a million IPv4s (each), the research team felt it was important to dig into what portion of services in our survey came from cloud environments. We chose the cloud providers we did based on IPv4 allocated capacity, discovered usage, and which ones were more prevalent in the context of real business use (versus hobbyist use).

Where possible, we used official lists provided by providers and augmented these with known mappings from provisioned autonomous systems. While we have strived for completeness, IPv4 entity mapping even in cloud space is less than a precise science.

How We Ranked

Ranking is always a challenge, since at some point humans incorporate their biases in terms of what an individual or group believes should be more important than something else. An example would be "*exposing SMB is far more dangerous and irresponsible than exposing DNS*" (which is a truth embedded in the NICER rankings). Not all exposure is created equal, and we imposed similar expert biases or "opinions" in previous NEI work. A key difference for NICER rankings is that we also have data on likely real vulnerabilities, both in terms of count and severity, for each service for each entity being ranked.

We gave an "Attacker's View" for each service in the report to provide threat context for defenders. In this appendix, we've collected relevant MITRE ATT&CK[154] tactic/technique mappings relevant to each service to enhance this context and provide a path toward identifying attacker actions against these services by implementing relevant detections associated with these ATT&CK techniques (provided the detection technologies within your organization support ATT&CK mappings).

Services in this report and tactic phases are grouped within each technique, and an abbreviated description is included. Readers are encouraged to review the detailed descriptions and associated mitigation strategies of services relevant to your organizations on the main MITRE ATT&CK website.

Table 41