Third Time's Not a Charm: Exploiting SNMPv3 for Router Fingerprinting

Taha Albakour
TU Berlin

Oliver Gasser
Max Planck Institute for Informatics

Robert Beverly
Naval Postgraduate School

Georgios Smaragdakis
TU Delft

ABSTRACT

In this paper, we show that adoption of the SNMPv3 network management protocol standard offers a unique—but likely unintended—opportunity for remotely fingerprinting network infrastructure in the wild. Specifically, by sending unsolicited and unauthenticated SNMPv3 requests, we obtain detailed information about the configuration and status of network devices including vendor, uptime, and the number of restarts. More importantly, the reply contains a persistent and strong identifier that allows for lightweight Internet-scale alias resolution and dual-stack association. By launching active Internet-wide SNMPv3 scan campaigns, we show that our technique can fingerprint more than 4.6 million devices of which around 350k are network routers. Not only is our technique lightweight and accurate, it is complementary to existing alias resolution, dual-stack inference, and device fingerprinting approaches. Our analysis not only provides fresh insights into the router deployment strategies of network operators worldwide, but also highlights potential vulnerabilities of SNMPv3 as currently deployed.

CCS CONCEPTS

• **Networks** → *Network protocols*; **Network management**.

KEYWORDS

Simple Network Management Protocol (SNMP), Device Fingerprinting, Alias Resolution.

ACM Reference Format:
Taha Albakour, Oliver Gasser, Robert Beverly, and Georgios Smaragdakis. 2021. Third Time's Not a Charm: Exploiting SNMPv3 for Router Fingerprinting. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3487552.3487848

1 INTRODUCTION

Remote management functionalities are fundamental to efficient network operation. To address this need, the Simple Network Management Protocol (SNMP) was introduced in the 1980s and has since served as the de facto protocol for fault notification, diagnostics, configuration management, and statistics gathering in IP networks [8]. As a core IP management protocol that is widely implemented, it is unsurprising that SNMP has been both exploited and leveraged as an attack vector—indeed, there are over 400 SNMP-related CVEs [39]. The protocol itself has historically been insecure, with the first standardized versions (SNMPv1 and SNMPv2) including only basic authentication via unencrypted "community strings." Security conscious operators were therefore forced to restrict SNMP access to internal networks.

The current SNMPv3 standard, introduced in 2002, is implemented on virtually all modern network equipment [27]. The primary focus of SNMPv3 is to provide a secure version of the protocol by including mechanisms for robust authentication, integrity, and privacy. Of direct relevance to our work is the so-called SNMP "engine ID." During synchronization with a client, the SNMPv3 agent exchanges its engine ID as a unique identifier. As noted in the RFC: the "snmpEngineID is the unique and unambiguous identifier of an SNMP engine. Since there is a one-to-one association between SNMP engines and SNMP entities, it also uniquely and unambiguously identifies the SNMP entity" [27].

As the engine ID is integral to the protocol's key localization mechanism, the SNMPv3 agent returns this strong device identifier even in response to *unsolicited* and *unauthenticated* requests. Moreover, real-world implementations commonly use one of the device's MAC addresses when forming the engine ID. This behavior offers unique—but likely unintended—opportunities for remotely fingerprinting network devices in the wild. We leverage the engine ID to not only identify device vendors, but also to perform IP alias resolution for both IPv4 and IPv6, and thus, dual-stack identification. Our method introduces a new avenue to characterize network routers and other infrastructure that is typically impervious to traditional methods such as TCP/IP stack fingerprinting, due to closed security postures and a lack of responding services. And, as compared to previously explored router identifiers such as low-entropy IP IDs (16 bits), engine IDs formed from MAC addresses can provide a strong, persistent, and accurate identifier.

In addition to the engine ID, critical information about the configuration and operation of devices running SNMPv3 can be obtained via unauthenticated requests. For example, we show that it is possible to retrieve the uptime of a device as well as the number of reboots. The combination of these two parameters can also be used as a unique device identifier.

To the best of our knowledge, previous studies and scans that considered SNMP [17, 31] focused on SNMPv2 or SNMPv2c and were thus unable to establish communication with the majority of devices in the wild. In contrast, our Internet-wide scans find more than 12 million IPs returning a unique identifier to unsolicited and

IMC '21, November 2–4, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9129-0/21/11.
https://doi.org/10.1145/3487552.3487848

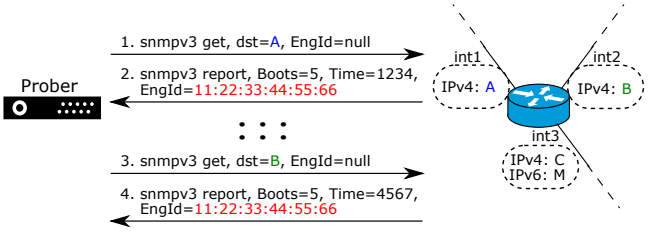

Figure 1: Unauthenticated, unsolicited SNMPv3 probes to router IPs return the SNMP engine uptime, reboot count, and engine ID. The engine ID is a unique, persistent identifier, typically formed from a MAC address on the device. Here, IPv4 interfaces A and B return the same engine ID, facilitating efficient alias resolution and vendor fingerprinting. Similarly, our technique can identify dual-stack aliases, e.g., IPv6 address M is an alias of A, B and C.

unauthenticated SNMPv3 requests. Our controlled lab experiments, show that some vendors enable SNMPv3 by default once SNMPv2 is enabled, suggesting that network operators may be unaware that their devices are responding to our queries. Among the 12M unique SNMPv3 responses, we discover and characterize approximately 350k routers. Our contributions thus include:

- A general and lightweight technique to fingerprint the vendor of devices running SNMPv3, including routers.
- A new accurate and efficient large-scale alias resolution method, including the first that can reliably identify dual-stack IPv4/IPv6 router aliases. We show that our alias resolution complements existing techniques.
- Unique insights into router deployment strategies by network operators worldwide, including market share estimates and network device homogeneity.
- Uncovering previously unrecognized concerns in potential vulnerabilities and misuse of SNMPv3 as deployed in the wild.
- Sharing our datasets and analysis scripts with interested researchers and providing updated graphs at **https://snmpv3.io**.

2 BACKGROUND

The Simple Network Management Protocol (SNMP) is an Internet standard to remotely manage, configure, and monitor network devices. While a wide variety of equipment implements SNMP, some of the most common devices running SNMP include routers, switches, and servers. Over several decades, SNMP has continued to mature and evolve.

2.1 SNMP Evolution

SNMPv1: Introduced in the late 1980s, and now deprecated, SNMPv1 became the de facto network management protocol [8, 9]. In SNMPv1, requests to a device are authenticated using a clear-text community string. Thus, in addition to weaknesses arising from common and default communities or brute forcing, an eavesdropper able to view messages can easily obtain the community. Security conscious operators were therefore forced to restrict SNMP access to internal or out-of-band management networks.

SNMPv2: The second generation of SNMP addressed some of the shortcomings of SNMPv1, especially performance and security [10, 11]. However, the security mechanisms of SNMPv2 were largely not adopted due to their complexity. Instead, the community string-based approach of SNMPv1 was reused to create SNMPv2c [12, 13], which was widely deployed. As with SNMPv1, the SNMPv2 standards are now in historical or obsolete status within the IETF.
SNMPv3: Our work concerns the current SNMP version [27]. SNMPv3 was designed to be secure by including strong user-based authentication, integrity, replay protection, and encryption. We detail aspects of SNMPv3 relevant to our research next.

2.2 SNMPv3 Unsolicited Request

As with other SNMP versions, SNMPv3 uses UDP. Because an explicit goal of SNMPv3 was to be stateless, it does not employ a challenge-response authentication mechanism [53]. Instead, messages include a digest produced via a keyed HMAC (either HMAC-MD5-96 or HMAC-SHA1-96). This message authentication code serves both to authenticate the message and provide integrity protection. In order to resist brute force attacks and ensure that each agent stores a different key, SNMPv3 employs *key localization*. The localized key is stretched and derived from the user key and the device's *engine ID*.

In order to communicate with an agent, the client must know this engine ID so that it too may derive the appropriate key to facilitate a different shared symmetric key per device. Thus, any client can initiate an unsolicited request, with a missing engine ID, in order to retrieve the authoritative agent's own engine ID (steps 1 and 2 of Figure 1). Because an authenticated message must, by definition, know the engine ID in order to create the HMAC, this synchronization message is not authenticated. Hence, *any client can send this request, even if it does not know a valid username or password.* Note that without valid credentials, we are still unable to retrieve any of the device's configuration or statistics. However, we trivially obtain the engine ID and other important meta-data including the "engine time" and "engine boots".

Compared to previous versions of the protocol, SNMPv3's authentication design makes it substantially easier to send unsolicited requests, as there is no need to guess or capture community strings. Instead, as shown in the dissected request packet of Figure 2, the synchronization request includes an empty engine ID and no password or username. Notably, the SNMPv3 response includes, in addition to the engine ID, values for variables related to configuration and operation of the device. For example, as shown in Figure 3, there is information about the SNMPv3 implementation (Conformance), the format of the engine ID (Engine ID Format), the vendor (Engine Enterprise ID), the number of device boots (engine boots), and the uptime of the SNMP agent since last restart (engine time).

2.3 SNMPv3 Unique Identifiers

Engine ID as a Unique Identifier: As described in RFC 3411 [27], different information can be used to form the engine ID: (i) an IPv4 or IPv6 address, (ii) a MAC address, (iii) ASCII text, (iv) byte values, or vendor-specific formats.

Although the engine ID can be created in various ways, its value is critical to the derived key from the key localization process [5].

```
Simple Network Management Protocol
    msgVersion: snmpv3 (3)
    msgGlobalData
    msgAuthoritativeEngineID: <MISSING>
    msgAuthoritativeEngineBoots: 0
    msgAuthoritativeEngineTime: 0
    msgUserName:
    msgAuthenticationParameters: <MISSING>
    msgPrivacyParameters: <MISSING>
    msgData: plaintext (0)
```

Figure 2: SNMPv3 unsolicited synchronization request.

```
Simple Network Management Protocol
    msgVersion: snmpv3 (3)
    msgGlobalData
    msgAuthoritativeEngineID: 800007c703748ef831db80
        1... .... = Engine ID Conformance: RFC3411 (SNMPv3)
        Engine Enterprise ID: Brocade Communication Systems, Inc.
        Engine ID Format: MAC address (3)
        Engine ID Data: BrocadeC_31:db:80 (74:8e:f8:31:db:80)
    msgAuthoritativeEngineBoots: 148
    msgAuthoritativeEngineTime: 10043812
    msgUserName:
    msgAuthenticationParameters: <MISSING>
    msgPrivacyParameters: <MISSING>
    msgData: plaintext (0)
```

Figure 3: SNMPv3 synchronization response containing engine ID.

If the engine ID changes, then the stored, localized keys, must be re-generated. Because this re-keying is cumbersome, the RFC recommends that the engine ID "persist across re-initializations" [27].

Two of the major vendors report in their documentation on the need to reconfigure SNMP users if the engine ID changes [15, 29]. Hence, in many popular implementations, the engine ID is not an IP address (which may change), but instead one of the device's IEEE hardware MAC addresses.

Thus, the engine ID can be considered a unique identifier of a device: once it is generated, it is not recommended to change. While we do not expect collisions when devices use MAC addresses for engine ID, we acknowledge that they may use non-device unique strings, be empty, or be ill-formatted. We address these issues via a thorough filtering process in Section 4.

(Last reboot time, Engine boots) Tuple as a Unique Identifier: As shown in Figure 3, the synchronization request also returns the SNMP agent's engine time and engine boots. The engine time is the number of seconds since the last time the authoritative SNMP engine has been "booted". By subtracting the scan time we obtain the last reboot time, i.e., the time and date at which the SNMP engine was last rebooted. The engine boots value on the other hand is the number of times the authoritative SNMP engine has restarted. These values are included as part of the synchronization in order to provide timeliness mechanisms, i.e., to prevent replay attacks and to detect duplicate messages [53].

It is unlikely that two devices have exactly the same last reboot time and engine boots. The only case that this occurs is if these devices take an identical amount of time to boot, and were restarted at exactly the same point in time, e.g., if they are co-located when a power outage occurs. Although the combination of (last reboot time, engine boots) is a slightly weaker unique identifier, it can be used in combination with the engine ID to differentiate, e.g., devices with the same engine ID due to misconfiguration. We find that more than 97% of all IPs provide (last reboot time, engine boots) tuples belonging to a unique engine ID (cf. Appendix B).

3 METHODOLOGY

Conceptually, our methodology is straightforward. As depicted in Figure 1, we send unsolicited UDP SNMPv3 synchronization packets to IPv4 or IPv6 addresses without username or password credentials (e.g., to IP *A* in step 1). Note that we send well-formed, compliant SNMPv3 packets. If the router with the target IP address is running SNMPv3, it will return the engine ID, uptime, and boots meta-data as part of the normal protocol synchronization (step 2). The meta-data can frequently be used to identify the SNMPv3 implementation and device vendor, for instance when it is formed from a MAC address (in this example, the engine ID is `11:22:33:44:55:66`).

This single packet exchange process repeats for all target IPv4 and IPv6 addresses in the scan campaign. If a probe to an IP address returns the same unique identifier, then we have inferred an alias. For example, in steps 3 and 4 of Figure 1, IP *B* returns the same MAC address as received when *A* was queried. Finally, by probing both IPv4 and IPv6 addresses, we can discover dual-stack aliases, and thus infer that IPv4 addresses *A* and *B* are also aliases of IPv6 address *C*.

3.1 Using SNMPv3 Unique Identifiers

When both the engine ID and the (engine time, engine boots) tuple are present in the response, we have two strong unique identifiers that can be used to identify a single device. Further, we find that in practice, we must employ a series of filters and tests to ensure consistent and reliable inferences. We describe these filtering operations in Section 4.2.

By collecting this information from SNMPv3 responses, we can perform a number of measurement tasks that otherwise would have required extensive active probing, massive data analysis, complex techniques, and would lack "ground truth" information. Further, our technique is complementary to existing approaches, which often cannot obtain a usable response from a target, such that it increases overall inference coverage.

SNMPv3-based Alias Resolution: Router interface IPs that are associated with the same unique identifier are mapped to the same router.

SNMPv3-based Dual-stack Inference: Router interface IPs of different protocols (IPv4 and IPv6) that are associated with the same unique identifier are mapped to the same router.

SNMPv3-based Vendor Fingerprinting: The information contained in the engine ID can be used to infer the vendor of the router. Our confidence in the inference of device vendor is highest when the engine ID is generated using a MAC address, as the upper three bytes encode the IEEE Organizational Unique Identifier (OUI), i.e., the company that registered a particular block of MAC addresses. Nevertheless, useful information about the vendor can be retrieved if the engine ID is generated differently. The Enterprise ID, that encodes the manufacturer of the device, is always present in the SNMPv3 response if RFC 3411 is followed, and can provide more confidence or be used as an alternative when the engine ID does not unveil the vendor.

SNMPv3-based Uptime: In addition, the engine time value allows us to determine the uptime of routers. This can be valuable information in order to determine, e.g., the patch status of devices, network reliability, and outage statistics.

3.2 SNMPv3 Active Scanning Campaigns

As there are no Internet-wide SNMPv3 scan results available, we perform our own active scanning campaigns. We use ZMap [20] to initiate SNMPv3 unsolicited synchronization requests by sending an SNMPv3 payload to UDP port 161. During the scans, we capture all SNMPv3 synchronization replies. We perform our IPv4 probing from a single server in a well-connected European data center and send at a rate of 5 kpps. For IPv6 we probe from a server located in a research network at a rate of 20 kpps.

3.3 Ethical Considerations

In designing our active scanning, we endeavored to minimize any potential ethical implications or harm from our study. First, the packets we send are not only well-formed and conforming to the SNMPv3 protocol, they are "normal" packets that any SNMP agent would expect in the course of its operation. Our randomized probing spreads the load and each IP receives at most one SNMP packet. Thus, we have no reason to believe that our packets would impair SNMP agents. Our probes are connectionless UDP packets, which are generally more innocuous as compared to TCP packets, and greatly reduce the potential for unintended issues related to maintaining state, e.g., by firewalls or middleboxes. Second, we coordinated with our local network administrators to ensure that our scanning did not harm the local or upstream network.

Next, we follow active scanning best practices [18, 20, 44] and ensure that our prober's IP address has a meaningful DNS PTR record, and run a web server serving that hostname. The web server describes the purpose of our scanning, as well as contact and opt-out information. To date, we have not received any complaints or opt-out requests. This is in striking contrast to other active scan campaigns performed in the past, e.g., TLS scanning and Web port scanning, which received complaints from the target IP owners and system administrators of the data centers hosting the scanning infrastructure. Finally, our work uncovers potentially sensitive security, robustness, and business information about network providers. We therefore aggregate and anonymize our results so as to not identify any individual network. We publish regularly updated graphs of aggregated results at **https://snmpv3.io**.

3.4 Limitations

While our technique provides a new method for fingerprinting and alias resolution, as well as affording novel insights into operational network deployments, we do note several limitations of our method and study.

First, in this work, we limit our study to network routers in order to provide meaningful comparisons with existing topology datasets and alias resolution methods. However, in our Internet-wide IPv4 and IPv6 scans we obtain a large number of responses from SNMP agents that cannot be matched to known router alias sets. These responses may represent routers missing in CAIDA or RIPE topologies, or may be servers, Customer Premise Equipment (CPE), or other devices, e.g., IoT. In future work, we plan to investigate these responses and devices in more detail.

Second, a deployed router may not be configured to enable SNMPv3, or may block outside queries by IP-level or other access control mechanisms. In such instances, our technique naturally cannot provide any vendor or alias inferences. However, because it uses a distinct and previously unused identifier, our method is complementary to existing approaches.

Third, while we seek to measure core network routers, our data can capture edge and periphery devices, especially in IPv6 where residential devices are a routed hop [46]. In such cases, the IP address of these devices can change on time-scales shorter than our scans [21, 47], causing different engine IDs to be returned from the same IP address. While we filter these instances of inconsistency from our data, we plan to investigate this effect in more detail in future work.

4 SCANNING FOR SNMPV3 DEVICES

SNMP has been a popular protocol for decades and, thus, has been the subject of many measurement studies [3, 17, 31, 35]. However, this prior measurement work targets SNMPv2. In addition, Censys [14] and Shadowserver [50] report regular Internet-wide scanning results, but solely for SNMPv2. Moreover, compared to our SNMPv3 measurements both services report substantially lower responsive IPv4 addresses, with 1.6M for Censys and 1.2M for Shadowserver, respectively.

4.1 Active Scan Targets

We perform active scans for SNMPv3 in IPv4 as well as IPv6. Table 1 shows an overview of our measurements. In addition to our own measurements, we utilize multiple third-party datasets containing known router IP addresses and aliases as shown in Table 2.

4.1.1 IPv4 & IPv6 SNMPv3 Scans. We launch two Internet-wide SNMPv3 campaigns for the IPv4 protocol in April 2021. By employing two measurements instead of a single one, we can filter out ephemeral addresses (cf. Section 4.4). We target all ~2.9 Billion routable IPv4 addresses. We receive valid SNMPv3 responses from about 31M IPv4 addresses. For each IPv4 address, we send one packet with size 88 bytes. For the large majority of responses, we receive one packet[1] with an average size of 130 bytes.

For IPv6, we target ~364M addresses in non-aliased IPv6 prefixes [21] of the IPv6 Hitlist Service [22]. We run two consecutive scans on April 13 and 14, 2021. In contrast to IPv4 we receive only about 180k SNMPv3 responses from these IPv6 scans. For each IPv6 address, we send one packet with size 108 bytes and receive a response of on average 150 bytes.

4.1.2 Router Interface Tagging. To annotate IP addresses belonging to router interfaces, we use publicly available datasets, namely CAIDA's ITDK [6] and intermediate hop IPs extracted from RIPE Atlas traceroute measurements [41] for IPv4 and IPv6. The ITDK is a curated dataset that also includes IPv4 router-level topologies, as inferred by MIDAR [32] and IPv6 router-level topologies inferred

[1]We receive multiple packets for about 0.6% of the responding IPv4 addresses, see Section 8 for a discussion.

Measurement	Date	#IPs	#Engine IDs	#IPs w/ valid engine ID	#IPs w/ valid engine ID & engine time
IPv4 scan 1	Apr. 16–20, 2021	31.8M	18.8M	} 27.0M	} 12.5M
IPv4 scan 2	Apr. 22–27, 2021	31.5M	18.6M		
IPv6 scan 1	Apr. 13, 2021	182k	68k	} 152k	} 140k
IPv6 scan 2	Apr. 14, 2021	180k	67k		

Table 1: Overview of our SNMPv3 measurement campaigns: Number of unique SNMPv3-responsive IPs, number of unique engine IDs, number of IPs with valid (i.e., consistent and non-filtered) engine ID, engine ID and engine time values, respectively.

Router dataset	Date	IPv4 addrs. (SNMPv3)	IPv6 addrs. (SNMPv3)
ITDK	Mar. 2021	2.9M (447k)	533k (36k)
RIPE Atlas	Apr. 2021	560k (85k)	260k (36k)
IPv6 Hitlist	2020–2021	n/a	63.7M (54k)
Union		3.1M (461k)	65M (78k)

Table 2: Overview of router datasets: Number of unique router IP addresses and coverage in our SNMPv3 measurements for IPv4 and IPv6, respectively.

via the Speedtrap technique [36]. For IPv6 only we use router addresses obtained from the IPv6 Hitlist Service traceroutes [22]. In Table 2 we show an overview of these router datasets as well as the number of matches to responses in our more general Internet-wide SNMPv3 scans described previously.

In IPv4 the ITDK dataset contains 2.9M router IPs with 447k of those responsive to our SNMPv3 measurements. RIPE Atlas adds a few thousand additionally tagged router IPs, bringing our total known SNMPv3 responsive router IPs to 461k.

In IPv6 we find that ITDK Speedtrap and RIPE Atlas traceroutes each cover about 36k SNMPv3 addresses. The vast corpus of IPv6 router addresses from the IPv6 Hitlist Service contains many Customer Premise Equipment (CPE) device addresses which are frequently changing [21], thus leading to this large size. With this dataset we obtain the highest SNMPv3 overlap with more than 54k IPv6 addresses. The union of all IPv6 router addresses leads to an overlap of more than 78k SNMPv3 addresses, i.e., more than half of all addresses with either valid engine ID or engine time (see below).

4.2 Engine ID

To identify an SNMP agent, the protocol uses the engine ID as defined by RFC 3411 [27]. In our measurements we collect 18.8M different engine IDs for IPv4 and 68k for IPv6. Many SNMP devices have more than one IP address assigned to them (e.g., because they are routers). As such we see the same engine ID for different IP addresses. Figure 4 shows the distribution of the number of IP addresses per engine ID for IPv4 and IPv6. In IPv4 more than 80% of engine IDs are seen on a single IP address. The same is true for more than half of all engine IDs collected with our IPv6 measurements. We find that the distribution is heavy-tailed, with the vast majority of all engine IDs being seen on 10 or fewer IPs, with some outliers with a single engine ID for more than 1000 IPs (not shown).

The engine ID can be in different formats, e.g., a device's MAC address, IP address, a text string, a byte string, or a vendor-defined format. In Figure 5 we show the distribution of different engine ID formats. Almost 60% of engine IDs for IPv4 and IPv6 are MAC-based. The use of MAC addresses provides a strong unique identifier and the high share of this engine ID format is therefore crucial for the uniqueness of engine IDs. In IPv4, opaque byte strings ("Octets"), non-SNMPv3-conforming, and Net-SNMP-specific engine ID formats contribute 10-20% each. The non-SNMPv3-conforming engine IDs do not contain any format information but rather just the byte values (e.g., `0x0300e0acf1325a88`). Similarly, the SNMPv3-conforming Octets format also contains raw bytes values (e.g., `0x3910910680002970`). Net-SNMP on the other hand, is a popular software-based SNMP implementation [42], which uses an enterprise-specific format (e.g., `0x0f010e3732bed25e00000000`). For IPv6 we also find a relatively high share of non-SNMPv3-conforming engine IDs, and a rather low number of Octets and Net-SNMP formats. Interestingly, we find more than 15% of engine IDs collected in our IPv6 measurements contain IPv4 addresses. These might reveal IPv4-IPv6 dual-stack deployments, which we investigate in detail in Section 5.

To better understand the randomness of Octets and Non-SNMPv3-conforming, which is crucial to their ability to serve as fingerprinting identifiers, we analyze their Hamming weight distribution. Figure 6 shows the relative Hamming weight distribution for both formats. The Hamming weight can be used as an indicator of randomness. Thus, the expectation of a randomly generated number would have half of its bits set to '1' and the other half set to '0'. The number of ones, i.e., the Hamming weight, for a large set of randomly generated numbers is therefore distributed according to the normal distribution $\mathcal{N}$ with a mean around half the length of the bit string. To meaningfully compare Hamming weights of

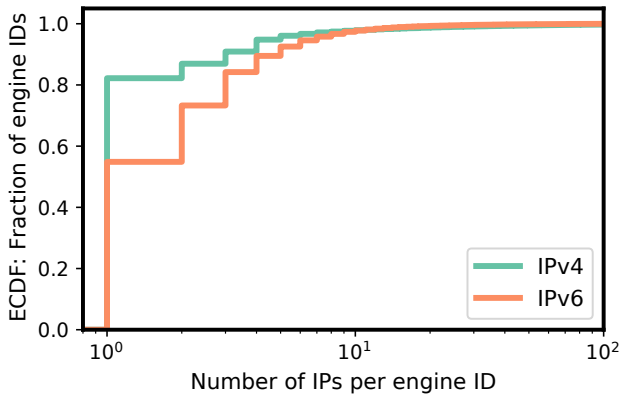

Figure 4: Number of occurrences per engine ID.

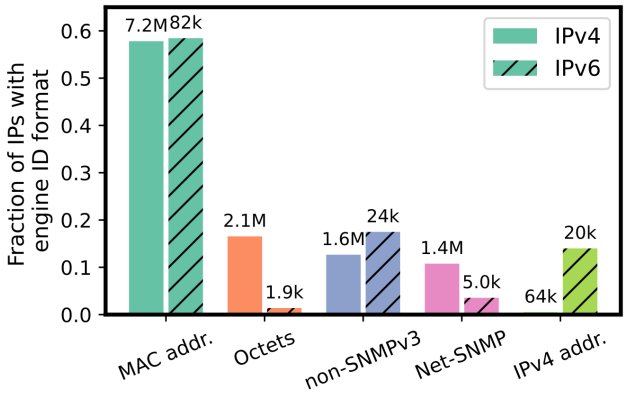

Figure 5: Distribution of different engine ID formats for IPv4 and IPv6 scans.

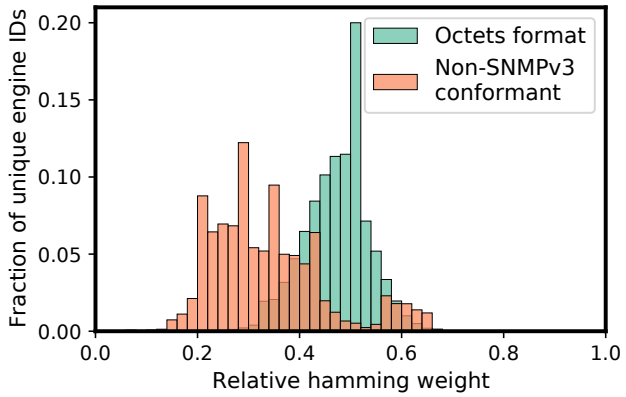

Figure 6: Relative Hamming weight distribution of Octets engine IDs and non-SNMPv3-conforming engine IDs.

variable-length bit strings, we choose to display the relative Hamming weight, i.e., the fraction of bits set to '1'. As can be seen in Figure 6, the relative Hamming weight of the Octets format is centered around the mean of 0.5, indicating a mostly random source behind the generation of these engine IDs. Non-SNMPv3-conforming engine IDs on the other hand seem to be distributed on not completely random input, as the relative Hamming weight distribution has a positive skew, i.e., more engine IDs with this format have fewer than expected bits set to '1'. For this reason, we apply a series of filters, which increase the confidence in the uniqueness of SNMP engine IDs. Note that randomly generated engine IDs can still be persistent (i.e., they remain the same for every query) and in fact we enforce engine ID consistency in our filtering pipeline.

4.3 Engine Time and Engine Boots

In addition to the engine ID we also use the engine time and engine boots SNMPv3 response fields for alias resolution. By subtracting the engine time from the exact packet receive time, we can derive the *last reboot time* for each responsive IP. The tuple of (last reboot time, engine boots) serves as an additional strong identifier (cf. Appendix B) in our alias resolution technique (cf. Section 5), while also being useful for fingerprinting purposes (cf. Section 6).

Figure 7 shows the distribution of the last reboot time of the top three engine IDs for IPv4 as well as IPv6. If those would belong to the same device, then we would expect to see them centered around the same time, i.e., a single device with a unique engine ID should have the same last reboot time value. We see, however, that five of the six most popular engine IDs have last reboot time values spanning multiple years. One prominent reason for this engine ID reuse are software bugs in routers, as is highlighted by our #1 IPv4 engine ID. We find more than 181k instances of IPs which all share the same engine ID `0x800000090300000000000000`. This artifact can be traced back to a bug which was acknowledged by the vendor [16], resulting in a constant MAC-based engine ID. These examples underline the importance of using the tuple of (last reboot time, engine boots) as a second identifier in combination with the engine ID.

4.4 Filtering Responses

We perform multiple filtering steps for all SNMPv3 responses:

Missing engine IDs. First, we remove responses with a missing engine ID. This is mostly due to non-SNMP-compliant responses. With this filter we remove about 5k IPv4 and 15 IPv6 responses.

Inconsistent engine IDs. In this step we merge the first and second scans for IPv4 and IPv6, respectively. Due to inconsistent answering behavior, likely due to devices changing IP addresses in the interim time between our scans, we have an overlap of 30.2M IPv4 addresses out of the 31.8M and 31.5M for the first and second scan respectively. We remove an additional 1.4M responses which show inconsistent engine ID values for the scans. In IPv6, 172k out of the 182k and 180k responsive IPs are overlapping, and we remove 557 instances of inconsistent engine ID values.

Too short engine IDs. As we rely on the uniqueness of engine IDs, we filter responses with overly short engine IDs. We use a threshold of four bytes, in order to keep IPv4-based engine IDs in the data set. We ensure that IPv4 engine IDs provide enough uniqueness in the following steps. In this step we remove about 5% for each protocol, i.e., 1.5M for IPv4 and 10k for IPv6.

Promiscuous engine IDs. We leverage the enterprise ID, which is part of the engine ID field and contains vendor information, to check for promiscuous engine ID values. Specifically, we check whether the same engine ID value is present across multiple vendors. If this is the case we label the engine ID as promiscuous and as a result we remove 96k IPv4 and 555 IPv6 responses.

Unroutable IPv4 engine IDs. In this filtering step we check whether IPv4-address-based engine IDs actually contain routable IPv4 addresses. Non-routable addresses (e.g., reserved, multicast, private addresses) are not guaranteed to be unique, and we therefore remove 68k IPv4 responses and and 7.8k IPv6 responses.

Unregistered MAC engine IDs. For all MAC-based engine IDs we map the contained MAC addresses to get the associated OUI [30]. We remove 113k and 1.4k MACs without matching OUIs for IP4 and IPv6, respectively.

Zero engine time or engine boots. We remove 834k IPv4 and 9.4k IPv6 entries with zero or empty engine time or engine boots values.

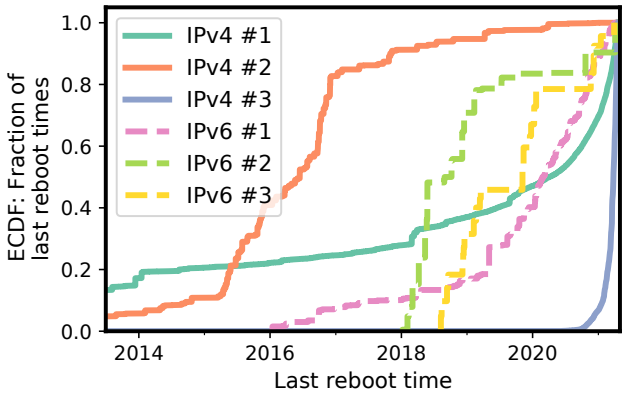

Figure 7: Distribution of the last reboot time for the top 3 engine IDs for IPv4 and IPv6.

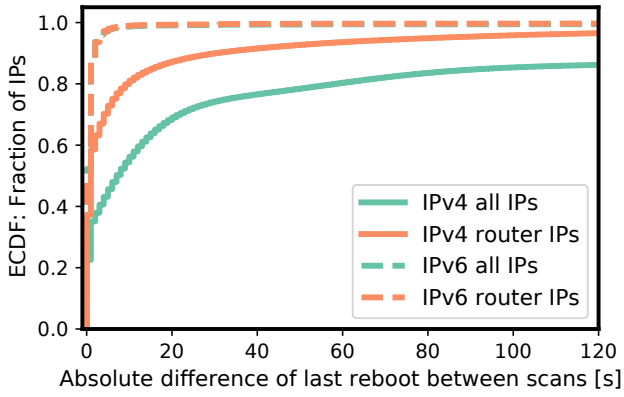

Figure 8: Distribution of the last reboot time difference between both scans for all IPv4, IPv6, and router IPv4, IPv6 addresses.

Engine time in the future. Next, we compare the engine time value with the packet receive time. As we rely on the engine time value reflecting a real time value, we remove entries where the engine time is in the future. In this step we remove 23k and 18 IPs for IPv4 and IPv6, respectively.

Inconsistent engine boots. In this step we compare the engine boots values for both scans. If they differ (e.g., because of a reboot) we can not rely on the reset engine time value and we therefore remove 3.8M IPv4 entries and 802 IPv6 entries.

Inconsistent last reboot time. We compare the derived last reboot time value from both scans to check for consistency. As timekeeping is prone to clock skew [33, 40, 49, 57] and suddenly running clock synchronization daemons, we first analyze the difference of the last reboot time as shown in Figure 8. As can be seen, the last reboot time values in IPv6 overlap very consistently, while they are more spread out in IPv4. We selectively also depict the distribution for router IP addresses (cf. Section 4.1.2), which shows more consistent last reboot time values. We choose a threshold of 10 seconds between scans, at the "knee" of the IPv4 router IPs distribution. With this last filter we remove 9.8M IPv4 addresses and 1.7k IPv6 addresses.

After this rigorous filtering pipeline we continue our analysis with the remaining 12.5M IPv4 addresses and 140k IPv6 addresses. Although this is a significant decrease from the initial set of SNMPv3-responsive IPs—especially for the more than 30M initial IPv4 responses—we err on the side of precision by applying this conservative filtering approach.

5 ALIAS RESOLUTION

Against the filtered IPv4 and IPv6 dataset, we run an alias resolution algorithm—first for IPv4 and IPv6 separately, and then on the combined set—to identify IP addresses belonging to the same SNMP device. We try variations of our technique (cf. Appendix A) and choose an approach which mimics similar thresholds as our filtering pipeline described in the previous section. We group all IP addresses together if they contain the same engine ID, the same engine boots, and a very similar last reboot time for both scans. In our filtering pipeline we select a last reboot time threshold of 10 seconds. To account for the fact that groups of IP addresses might deviate 10 seconds each, we map the last reboot time time into 20 second bins.

5.1 IPv4 and IPv6

By grouping based on these six fields (engine ID, engine boots, and last reboot time, for both scans respectively) we create alias sets. This results in 4.7M alias sets for IPv4, of which 824k are non-singletons (i.e., they contain more than one address). As a result, more than 8.7M of the 12.5M (70%) input IPs are grouped into non-singleton alias sets. Each alias set contains on average 10.6 IP addresses.

For IPv6, we use the same technique and end up with 59k alias sets of which 26k are non-singleton sets. These non-singleton IPv6 alias sets contain more than 106k of the initial 140k IPv6 addresses, a coverage of more than 75%. Due to the lower number of responsive IPv6 addresses in our measurements, the average number of 4.2 addresses per IPv6 non-singleton alias set is smaller than in IPv4.

Finally, we also resolve dual-stack aliases (i.e., devices with IPv4 and IPv6 addresses) by applying the same alias resolution technique on the joined IPv4 and IPv6 alias sets. After this final alias resolution step we have 4.6M IPv4-only alias sets (i.e., alias sets containing only IPv4 addresses), 27.7k IPv6-only alias sets, and 31.2k dual-stack alias sets. Of those 796k, 11.3k, and 31.2k are non-singleton alias sets. These non-singleton alias sets contain 7.4M IPv4-only addresses (9.3 addresses per set), 49.5k IPv6-only addresses (4.4 per set), and 1.4M dual-stacked addresses (45.4 addresses). We find that especially the high number of average addresses for dual-stack alias sets is an impressive confirmation that our technique is able to reliably identify enterprise routers with many interfaces.

In Figure 9 we show the distribution of IP addresses per alias set for IPv4, IPv6, and router IPs as identified by at least one router being part of a well-known router dataset. In contrast to overall IPv4 and IPv6 alias sets, we find that router alias sets contain many more addresses. This is an indicator that SNMPv3 is widely used on routers with many IP addresses and interfaces.

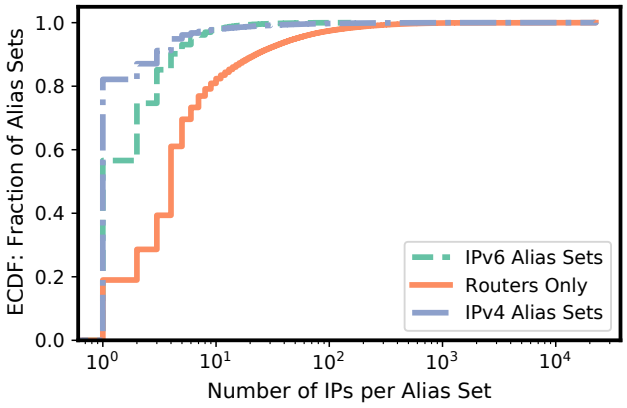

Figure 9: Distribution of the number of IP addresses per alias set for IPv4, IPv6, and router addresses.

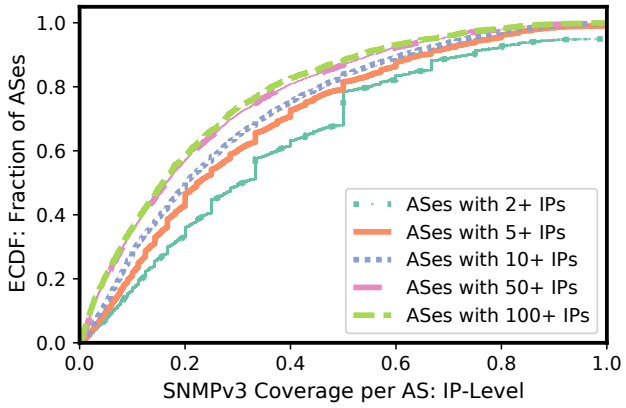

Figure 10: Coverage of responsive SNMPv3 router IPv4 addresses per AS.

5.2 Comparison with Router Names

Next we compare our identified alias sets with the CAIDA Router Names dataset. This dataset is built using the technique from Luckie et al. [37] by getting the reverse DNS name for IP addresses and then using regular expressions to group routers together. Specifically, we obtain the per-domain suffix regular expressions ("regexes") created by CAIDA and derived from their most recent March 2021 ITDK topology. These regexes extract the hostname from a complete PTR record to identify the router; multiple interfaces with a common hostname are then assumed to be aliases of the same router. Conservatively, we only use regexes where their algorithm produced a positive predictive value of 0.8 or higher.

We apply each suffix's regex to the IPv4 and IPv6 interface PTR records available in the March 2021 CAIDA ITDK. Note that not all interface IP addresses have PTR records and we necessarily exclude these. We then coalesce those names into routers that have both IPv4 and IPv6 interfaces with PTR records where the hostnames match (the interface names need not match). In total, using these regexes, we obtain 12.4k dual-stack non-singleton alias sets containing 63.8k IP addresses, i.e., 5.2 addresses per alias set. This dataset is overall significantly smaller compared to our 838k non-singleton alias sets. Even when looking at dual-stack non-singleton alias sets we identify more than 2.5x (31.2k). We compare the content of the alias sets identified by both approaches and find only 9 exactly matching alias sets. When looking for partial matches we identify 5.9k partially overlapping alias sets, i.e., at least one IP address from CAIDA's Router Names dataset is in one of the SNMPv3 alias sets. This finding shows that the SNMPv3 approach partially covers about half of all Router Names alias sets. In summary, we conclude that both alias sets are complementary, as they contain only partially overlapping addresses. This is likely due to the different used techniques, e.g., some routers may not have useful rDNS entries, while others might not respond to SNMPv3. Overall, however, the SNMPv3 technique is able to identify significantly more alias sets.

5.3 Comparison with ITDK

Next we compare our alias set results with CAIDA's ITDK, namely the March 2021 MIDAR dataset [32] for IPv4 and the Speedtrap dataset [36] for IPv6. Those datasets leverage the presence of a monotonically increasing IP ID value to identify aliases. MIDAR identifies 8.4M IPv4 alias sets of which the overwhelming majority are singletons. There are 94k non-singleton sets containing about 363k IP addresses, i.e., 3.9 IPs per alias set. Speedtrap groups IPv6 addresses into 525k alias sets—again, the majority are singletons, with only 5.3k alias sets with more than one address containing 13.6k addresses. Our identified sets find 222k and 4.3k perfectly overlapping alias sets in MIDAR and Speedtrap, respectively. More than 95% of those overlaps are singleton sets. Finally, we identify partial overlaps for 1.1M MIDAR and 533k Speedtrap alias sets, with the vast majority being singletons. To summarize, MIDAR and Speedtrap also provide complementary views of aliased routes, likely due to different support of the used techniques. Overall, we find almost a magnitude more non-singleton alias sets compared to both sets.

5.4 SNMPv3 Coverage

To assess how many IPv4 addresses per AS we can de-alias, we define *coverage* as the ratio of responsive SNMPv3 router IPv4 addresses compared to the total number of IPv4 addresses per AS within the IPv4 union router dataset containing 3.1M addresses (cf. Table 2). Overall, we find that 16% of the IPv4 router addresses respond to SNMPv3 probes. In Figure 10 we plot the empirical CDF of SNMPv3 coverage for ASes with thresholds of at least 2, 5, 10, 50, and 100 IPs in our dataset. The number of ASes for each threshold are 11.8k, 9.1k, 7.9k, 2.9k, and 1.8k, respectively. The main observation in Figure 10 is that SNMPv3 coverage is slightly better for ASes with fewer IPs than those with higher ones. The coverage also deviates substantially across different networks. Regardless of the threshold, the coverage is less than 10% for about a quarter of the networks, and is more than 80% for top 10% of the covered networks. Recall, MIDAR contains around 94k non-singleton sets that consist of 362k router IPv4 addresses. 28.4% of those addresses

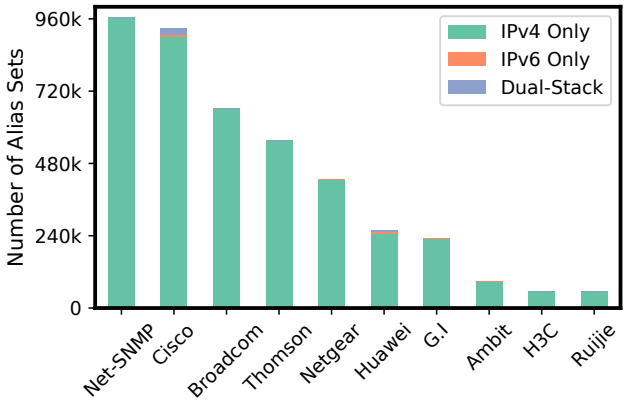

Figure 11: Vendor popularity.

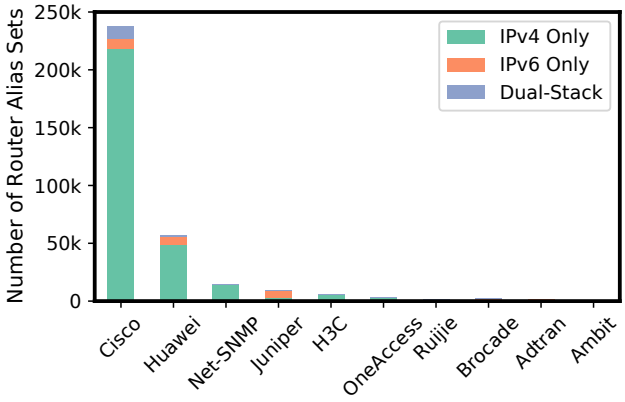

Figure 12: Router vendor popularity.

also respond to SNMPv3. Further, more than 461k router addresses respond to SNMPv3 with only 22% overlapping with MIDAR's non-singleton sets. Overall, when combining both techniques one can potentially increase the coverage of de-aliased router IPv4 addresses from 11.7% (MIDAR only) or 14.8% (SNMPv3 only) up to 23%.

6 ROUTER FINGERPRINTING

Our technique offers the unique opportunity to identify the vendor of millions of devices, including routers. These fresh insights allow us to estimate the market share of infrastructure vendors at an unprecedented scale with our lightweight and accurate method.

6.1 Baseline Results

In Figure 11 we report the popularity of major network equipment vendors as unveiled with our method. In total, we are able to de-alias 4,617,690 devices (aka alias sets). As illustrated in the figure, the majority of these devices use exclusively IPv4. The largest fraction of the devices are UNIX-based (Net-SNMP). Major network equipment vendors are also in the top 10 list, including Cisco (more than 900k devices), Broadcom (580k devices), Huawei (220k devices), and H3C (5k devices). In the top 10 vendors there are also home and office network equipment vendors such as Thomson (580k devices) and Netgear (420k devices). Thus, our technique provides a view into many popular edge devices. Unfortunately, attackers will also have this view when they send unauthorized and unsolicited requests and can exploit known vulnerabilities. The count of devices belonging to a given vendor drops drastically past the top 10 vendors. Indeed, the top 10 vendors are responsible for more than 80% of the devices we identify with our technique.

Next, we study the vendor popularity for the routers we identified with our method. To compile the set of routers, we consider all the alias sets as before, but we also require that the IPs in these alias sets are present in the most recent ITDK and RIPE Atlas datasets, from March 2021 and April 2021, respectively. In total, we identify 346,951 routers. The large majority are IPv4-only (307,404), while there were 24,641 IPv6-only routers and 14,906 dual-stack routers. These numbers show that the fraction of IPv6 only and dual-stack routers is significantly higher for routers as compared to the overall set of devices. This is also visible in Figure 12, where we show the popularity of router vendors by protocol. In this figure it is also clear that when we consider routers there is higher vendor consolidation compared to the overall devices. Indeed, the four major vendors, namely Cisco, Huawei, Juniper, and H3C, are responsible for more than 95% of routers we identified with our method. Cisco is the most popular with around 240k routers followed by Huawei with 52k routers.

6.2 Validation

6.2.1 Lab testing. To better understand the default behavior and configuration of Cisco devices running SNMPv3, we setup Cisco routers running IOS 15.2(4)S7 (released in 2015) and IOS XR 6.0.1 in a controlled environment. We use Net-SNMP to issue queries from a Linux machine directly connected to the router and passively monitor the network traffic via a packet capture. By default, the Cisco router does not run SNMP and answers neither v2 nor v3 queries. We enable SNMP via a single line of configuration `snmp-server community pass123 RO` which sets the SNMP read community string to a private value. We then confirm that the router answers SNMPv2 queries using this private community string by querying for the `sysDescr` MIB value and receiving the response.

Next, we issue an SNMPv3 query, again for the `sysDescr` MIB object, using the username `noAuthUser` and the security level `noAuthNoPriv`, i.e., the same unauthenticated query we issue in our Internet-wide measurements. While the query is rejected with a "unknown user name" error as expected, for both versions of Cisco IOS, the response packet includes a Cisco OUI MAC address within the engine ID field. The router has multiple interfaces, each with different MAC and assigned IP addresses. Regardless of the IP address queried, the router returns the same MAC address in the engine ID response. This MAC address corresponds to the "first" interface as reported by the routers via the command-line management interface. This MAC in the engine ID was not the numerically smallest MAC address among all the interfaces, which contradicts the guidance in the SNMPv3 behavior specification [27].

Of special note is that the Cisco router responds to these SNMPv3 queries without any specific v3 configuration. Simply configuring

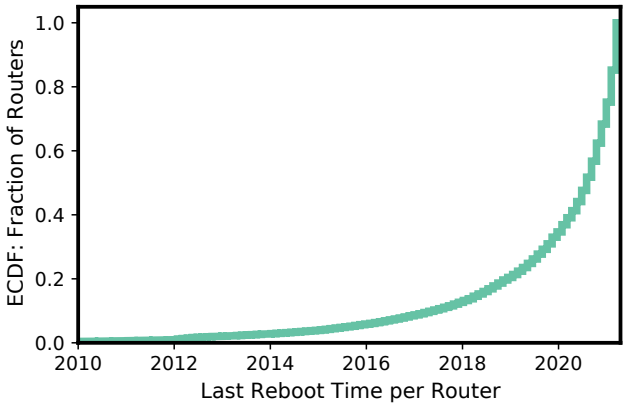

Figure 13: Distribution of time since the last reboot.

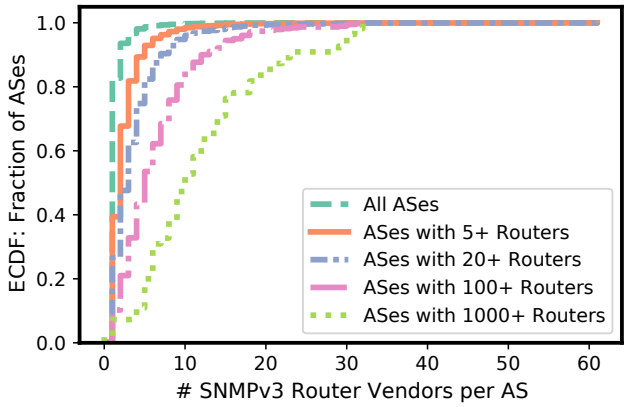

Figure 14: Number of router vendors per AS.

a read-only community string, which is only pertinent to SNMPv2, seemingly enables SNMPv3 on the router. Thus, operators may inadvertently be enabling SNMPv3 on their devices in the process of setting up SNMPv2. We notified Cisco of this behavior and our report was assigned a case number by their product security incident response team. However, as of this paper publication date, we did not receive any details of further actions or remediation.

We replicate the same experiment with Juniper Junos (version 17.3, 2017 release). We notice that the behavior is similar to Cisco, i.e., operation of SNMPv2 enables SNMPv3. However, Juniper requires to explicitly enable services on a given interface, which may result in less visible Juniper routers with our SNMPv3 scan.

6.2.2 Operators Survey. To validate our results, we contacted network operators. In our request for comment, we shared with them the alias sets, i.e., the set of IPs for each router and the router vendor as we identified it with our technique. Six of them replied to our request. The network operators confirmed that we correctly de-alias the router and identified the router vendor in all responses. We also notice that identified "net-snmp" and "unknown vendor" devices, typically correspond to network appliances and programmable network devices, respectively. However, some of the operators pointed out potential limitations of our method. Indeed, we were unable to uncover some router interfaces with our SNMPv3 scans, as those ACL interfaces drop packets sent to well-known ports.

6.2.3 Comparison with Nmap. Nmap [43] is a popular tool for operating system and device fingerprinting. It runs up to 16 TCP, UDP, and ICMP tests in order to generate a signature for a target, and then attempts to match it against its database. Nmap requires at least one open and one closed TCP port to achieve accurate results. If Nmap is unable to run all the required tests, or find an exact match to the resulting signature, it attempts to provide a best-guess of the target OS. Unsurprisingly, Nmap's approach generates a significant amount of traffic and is not suitable for a large-scale measurement. As such, we decide to test it only on a small subset of all SNMPv3 responsive routers. We randomly pick a single IPv4 address from each router. We target 26.4k router IPs in total and compare the resulting fingerprints from Nmap (version 7.60) with the one obtained via SNMPv3.

For 22.2k routers, Nmap was unable to report any results. This is likely due to the fact that none of those routers are running any common TCP service (ftp, ssh, telnet, etc.) which is required for Nmap to work properly. Note that by default, Nmap will attempt to find an open TCP port by scanning only the top 10 services in its database. We acknowledge that Nmap may fingerprint those routers when using more aggressive options (such as full TCP port scan) but we opted not to do so due to the excessive load it would generate. Further, the Nmap fingerprint did not agree with our SNMPv3 fingerprint for 1.3k routers. In all of those instances, Nmap attempted to guess the operating system rather than providing an exact match from its database indicating that it was unable to complete some of its tests. Finally, 2.9k Nmap results match the fingerprint obtained with SNMPv3; Nmap was able to provide additional information such as OS version for the majority of those routers. Recall that our method is not using any statistical inference as our vendor identification is based on either the MAC address or Enterprise ID—that are typically unique per vendor [1]—both of which can be obtained from the SNMPv3 engine ID data. We acknowledge that Nmap's thorough tests and large database can fingerprint devices beyond the vendor level. Nevertheless, contrary to Nmap the SNMPv3 technique allows for Internet-scale fingerprinting by sending only a single probe packet per target address.

6.3 Time Since Last Reboot

Accurate router fingerprinting allows us to answer important questions about the status of routers in the wild. In Figure 13 we plot the CDF for the time since the last reboot for around 346k routers we identified with our method. Less than 25% of them had their last reboot more than a year before our first scanning campaign (ca. April 2021). More than 50% of the routers had a reboot since the beginning of the year of our measurement (2021), and around 20% during the last month. These results show that, potentially, a large fraction of routers did not recently install security updates, for which a reboot is normally necessary. We are currently launching more campaigns and we will continue monitoring the last reboot time to provide more insights in the future.

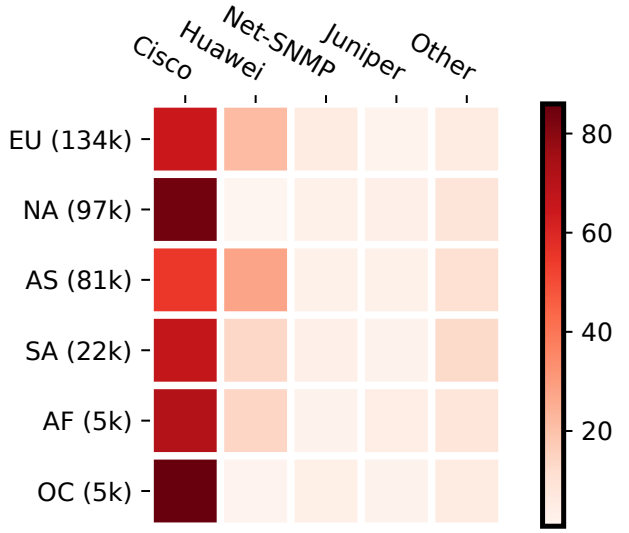

Figure 15: Router vendor popularity per continent with the total number of routers per region in parenthesis.

6.4 Router Deployment Insights

6.4.1 Distribution of Routers per AS. We consider router deployments in 22,787 networks. In 4,059 networks we identified at least 5 routers, in 1,557 networks more than 20 routers, in 381 networks more than 100 routers, and in 55 more than 1,000 routers using our SNMPv3 scans. Our analysis shows that the distribution of routers per AS does not vary significantly across different regions, see Appendix C.

6.4.2 Number of Vendors per AS. In Figure 14 we report how many different router vendors we can identify in a single AS. In 40% of the networks with more than 5 routers all of the routers are from the a single vendor. In less than 10% of the networks the number of router vendors exceeds five. When we are focusing on larger networks, with more than 100 or 1,000 routers, then we observe that there are more routers vendors present. This is to be expected as these networks run complex network operations and they can host specialized routers and network equipment from different vendors. In Section 6.4.4 we do a case study of the 10 largest networks by number of routers and we show that although the number of vendors may be high, the majority of the routers are from a couple of vendors. This is also true in most of the networks with 5 routers or more in our study.

6.4.3 Regional Vendor Popularity. We then focus on the market share of different router vendors at different regions. In Figure 15 we present a heat map for the popularity of each vendor in all the ASes of a region (continent). Cisco is the dominant vendor across all regions. The second most popular is Huawei with about 27% market share in Asia, around 22% in Europe, and close to 14% in South America and Africa. However, we could not find any Huawei router in North America and less than 1% in Oceania. The contributions of other router vendors is very low across regions. We conclude that although the number of vendors per AS may be high only a relatively small number of routers contribute to this diversity, as there is a strong consolidation driven by two major router manufacturers, Cisco and Huawei.

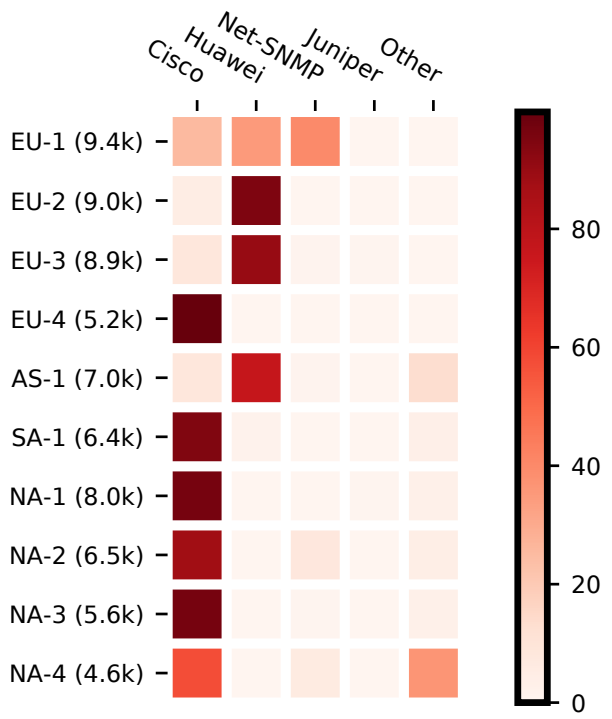

Figure 16: Router vendor popularity for top 10 networks by number of routers with number of routers in parenthesis.

6.4.4 Vendor Popularity in Large Networks. Finally, we perform a case study for the top 10 networks by number of routers. These are networks with at least 4.5k routers. They are distributed in different continents, four in Europe, four in North America, and one in Asia and South America. In Figure 15, we report the popularity of each vendor and in parenthesis the number of routers per AS. In 6 out of 10 of these networks Cisco is dominant. However, in the network from Asia and in two of the networks in Europe, Huawei is dominant. Typically, the large networks have only one dominant vendor, however, one of the networks in the top 10 list has deployed both Cisco and Huawei routers as well as UNIX-based router solutions. Again, although we see multiple router vendors present in all top 10 networks, the large majority of the routers (typically, more than 95%) belongs to only one or two vendors. Thus, large networks can be potentially vulnerable to vulnerabilities of a single major vendor as their deployment is quite uniform.

6.5 Vendor Dominance

We also study the homogeneity of router vendors in each network. In order to analyze this homogeneity we introduce a new metric, *vendor dominance*, i.e., the fraction of routers belonging to the most dominant router vendor in that AS. Thus, the closer the vendor dominance is to 1, the higher higher is the share of routers belonging

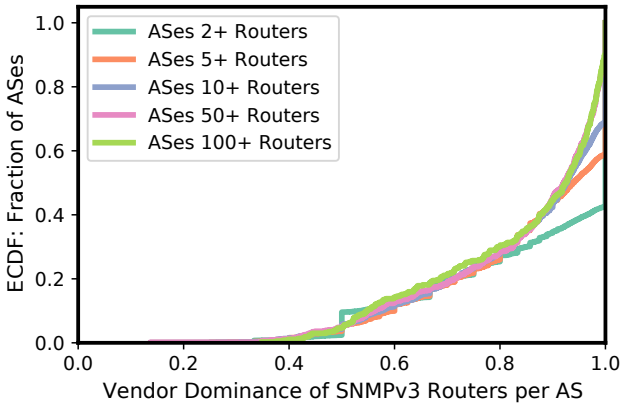

Figure 17: Vendor dominance for routers found with SNMPv3 measurements per AS.

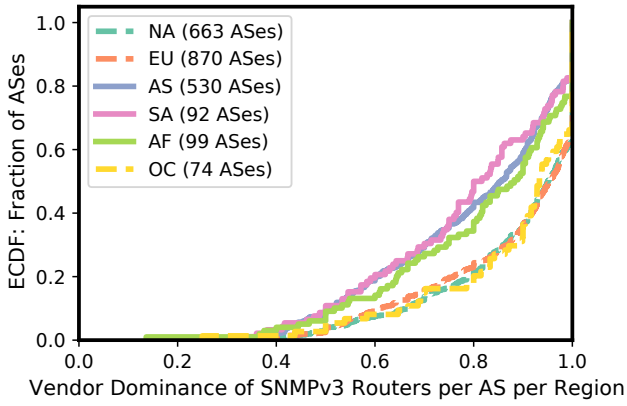

Figure 18: Vendor dominance for routers found with SNMPv3 measurements per region for ASes with at least 10 routers.

to only a single vendor in this network. This is a critical property of a network, as vulnerabilities of the dominant vendor may be exploited and affect a large fraction of the deployed routers. In Figure 17 we show the distribution of the vendor dominance across ASes. We notice that the the values of vendor dominance are high throughout many networks: more than 80% of the networks have a vendor dominance of 0.7 or more. This shows that, typically, there is a single very popular vendor per network, that can well be different from network to network. Next, we turn our attention to the regional characteristics of networks with at least 10 routers as measured with SNMPv3, see Figure 18. We notice that there are two groups of regions: (i) South America, Asia, Africa, and (ii) Oceania, North America, Europe. The vendor dominance of networks in the first group is typically lower than in those of the second group.

7 RELATED WORK

Prior work has developed passive and active techniques that leverage identifiers and implementation-specific differences to fingerprint and de-alias devices at various granularities. This section details these existing methods.

7.1 Router Vendor Fingerprinting

Nmap: Nmap is an open-source network scanning and reconnaissance tool that can perform operating system fingerprinting [43]. It sends a series ICMP echo requests, UDP packets, and TCP probes with different field values, flags, and options to fingerprint the remote system. By examining the responses, e.g., length, options, window size, sequence numbers, IP ID and TTL values, checksum, and flags, Nmap finds the best matching implementation in its database of operating system fingerprints. The latest version of Nmap (7.91) contains 5,679 fingerprints; of these, approximately 160 and 22 correspond to Cisco and Juniper routers respectively. While Nmap is a powerful tool for TCP/IP fingerprinting, it requires the remote host to listen and respond on an open TCP port. Because routers in the wild are secured and typically do not respond to unsolicited TCP probes, Nmap is generally ineffective for router fingerprinting.

Scanning: A second popular technique for remotely inferring operating system and vendor information is "banner grabbing," whereby publicly available services leak information. For instance, the Cisco SSH server implementation returns identifying information in its response string. Internet-wide scanning and banner grabbing are performed regularly [14, 19, 20, 31]. Recent work has sought to find banners and augment with active and analysis of banners and augmentation with active measurements [28] to perform large-scale network vendor inference. Unfortunately, as with Nmap's reliance on TCP, banner analysis requires a publicly responsive service that returns discriminating information. Routers are frequently tightly secured and unresponsive to banner queries.

TTL: Due to the relatively closed nature of routers, Vanaubel et al. developed a fingerprinting technique based solely on TTL responses [55]. They send TCP, UDP, and ICMP probes to and toward the target, and show that the tuple of inferred initial TTL (iTTL) values from the responses can differentiate between some well known platforms, including Cisco and Juniper. Unfortunately, the universe of possible iTTL values is small, and can lead to a large number of incorrect inferences, e.g., Huawei has the same iTTL signature as Cisco.

7.2 Router Alias Resolution

Identifiers: Techniques for discovering router aliases were first developed by exploiting an implementation behavior where ICMP port unreachable responses were generated with a source address of the interface toward the destination, regardless of which interface IP was originally probed [24]. Other techniques have included pre-specified timestamps [51] and graph analysis [25]. More recently, Marder proposed using path length estimation [38] while Vermeulen et al. leverage ICMP rate limiting [56]. However, to date, state-of-the-art systems use IP ID based alias resolution.

IP ID: The IP identifier (IP ID) field is used for fragmentation and reassembly. Since IP ID is a mandatory field in IPv4, it is possible to elicit an IP ID value from a router via a simple ICMP echo. This value is typically set in one of three ways: zero (in conjunction with the "don't fragment" bit), randomly, or sequentially for each new packet. While this information can convey some information about

the router vendor, IP IDs can be used for alias resolution—the process of determining if multiple IP addresses correspond to different interfaces on the same router (i.e., are aliases). In many implementations, the IP ID counter is shared among different interfaces on the router [52]. Current state-of-the-art alias resolution techniques sample the IP ID value of candidate IPs over time and perform a monotonic bounds test on IP ID sequences to infer aliases [32]. Similar techniques have been developed for IPv6 [36]. As the IP ID is only 16 bits long, it can increment and wrap faster than it can be sampled if the router has a large traffic rate. As a result, IP ID techniques suffer from false positive and false negative errors [37]. Further, finding aliases based on IP ID at Internet-scale remains challenging, and requires intensive probing and computation. In contrast, our technique elicits a strong, unique, and persistent identifier that can discover aliases much more efficiently.

7.3 Dual-stack Inference

Comparatively less work has made progress on finding dual-stack aliases, especially for routers. Scheitle et al. use TCP timestamp skew to identify IPv4/IPv6 siblings, but, as routers are generally unresponsive to TCP probes, their work largely centers on servers [49]. Berger et al. developed a method to reveal dual-stack addresses of name servers [2]. And while Czyz et al. examined IPv6 security posture as compared to IPv4 [17], they too had no method to resolve dual-stack router aliases. To date, the only viable technique for dual-stack router alias resolution has been through DNS inference, in particular using Luckie's method for extract router hostnames [37] which we compare against.

8 DISCUSSION

Follow Best Current Security Practices When Running SNMPv3. To our surprise, millions of devices, including hundreds of thousands of routers, respond to unauthorized and unsolicited SNMPv3 requests. Recall that all of our requests were launched by a server in a single data center, implying that these devices either were not behind a firewall or the firewall was not properly configured. If the network administrators had applied best current security practices, e.g., access control lists or segregated out-of-band management, we would expect significantly fewer responses. This would have mitigated the potential vulnerabilities we identified at large. Similarly surprising, as mentioned in Section 3, we did not receive any complaints (although we provided contact information) when we scanned routers, in striking contrast to scans performed in the past for servers. This can be attributed to the fact that SNMPv3 scans are stealthy as they send only a single UDP packet, but it also suggests that routers and other connected devices that run SNMPv3 are not well monitored by network administrators for scanning activity. Operators may want to more rigorously monitor such external management queries to their infrastructure.

Implicitly Enabling SNMPv3. In addition to the open network access policies previously identified, a likely explanation for the large number of responsive SNMPv3 devices is that some vendors and implementations automatically enable SNMPv3 when SNMPv2 is enabled (see Section 6.2). Instead, we recommend that implementations require explicit configuration to enable SNMPv3 to ensure that operators are consciously running SNMPv3.

Potential Vulnerabilities of Current SNMPv3 Implementation. There are more than 400 vulnerabilities related to SNMP [39]. While many of these vulnerabilities are related to specific implementations, our observations on SNMPv3's behavior exposes a more fundamental fingerprinting weakness. Whereas an unauthorized and unsolicited request in SNMPv2 does not elicit a response, SNMPv3, by design, does respond. Our study shows that potentially millions of devices will respond to such SNMPv3 requests. Moreover, as SNMPv3 is UDP-based, it is trivial to spoof the source of these requests, akin to other spoofed-source attacks [4, 23, 45, 48]. More concerningly, in some of our measurements, a single request resulted in multiple (identical) responses. For example, in the the first IPv4 scan, more than 182k IPv4 addresses responded with more than one request, 48 of which returned more than 1,000 responses within two to twenty hours. One of them, sent back 48,500,523 response packets within two hours. Although the exact cause of this behavior is not known, similar behavior has been reported for other handshaking protocols [34]. In the second IPv4 scan, we also saw different addresses return a large number of responses. Thus, the harm by exploiting the authentication of SNMPv3 can be potentially a serious threat.

Towards SNMPv4: Rethinking SNMPv3. Although SNMPv3 offers a stronger security model than its predecessors, our study uncovered some shortcomings in the protocol that enable fingerprinting. A fundamental design tension exists between maintaining the stateless operation of SNMP and the security mechanisms. While proposals to utilize standards such as TLS with SNMP have been drafted to protect datagrams [26], such approaches may present other concerns, including the ability to perform certificate-based fingerprinting. A more immediate solution is to not use MAC addresses as the engine ID. While the engine ID needs to be persistent, re-purposing MAC addresses has a long history in network security of enabling fingerprinting and other attacks. Further, researchers have shown that obtaining the persistent engine ID permits brute force SNMPv3 password recovery attacks [54]. We thus encourage protocol designers in the future to consider the weaknesses of a persistent engine ID, as well as discourage the use of MAC address-based engine ID.

9 CONCLUSION

In this paper, we show that the adoption of a secure network management protocol, SNMPv3, surprisingly increases device fingerprinting capabilities. By design, devices that run SNMPv3 respond to unauthorized and unsolicited authentication requests with a device unique identifier and other critical information about the status and configuration about the device. We show that SNMPv3 allows for light and accurate alias resolution, dual-stack association, and fingerprinting with only a single request per device. With our technique we were able to de-alias and fingerprint more than 4.6 million devices, including around 350k routers. Our analysis provides fresh insights on the router vendor market share, router deployment strategies of network operators around the world, as well as statistics such as the router uptime and distribution of vendors at different regions. We hope that our technique can be used for answering other network analytics questions in the future, e.g., inferring NAT and load balancers in the wild.

Acknowledgments

We thank Cas D'Angelo for providing invaluable operational network validation. We would also like to thank our shepherd Mattijs Jonker and the anonymous reviewers for their valuable feedback. This work was funded in part by the European Research Council Starting Grant ResolutioNet (ERC-StG-679158), BMBF BIFOLD 01IS18025A and 01IS18037A, and the U.S. National Science Foundation (CNS-1855614). Views and conclusions are those of the authors and should not be interpreted as representing the official policies or position of the ERC, BMBF, U.S. government or the NSF.

REFERENCES

[1] Internet Assigned Numbers Authority. 2021. PRIVATE ENTERPRISE NUMBERS. https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers.
[2] A. Berger, N. Weaver, R. Beverly, and L. Campbell. 2013. Internet nameserver IPv4 and IPv6 address relationships. In *ACM IMC*.
[3] R. Beverly. 2002. RTG: A Scalable SNMP Statistics Architecture for Service Providers. In *USENIX LISA*.
[4] R. Beverly, A. Berger, Y. Hyun, and k. claffy. 2009. Understanding the Efficacy of Deployed Internet Source Address Validation Filtering. In *ACM IMC*.
[5] U. Blumenthal and B. Wijnen. 2002. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). IETF RFC 3414.
[6] CAIDA. 2021. Macroscopic Internet Topology Data Kit (ITDK). https://www.caida.org/catalog/datasets/internet-topology-data-kit/.
[7] CAIDA. 2021. The CAIDA AS Ranking. https://asrank.caida.org/.
[8] J. Case, M. Fedor, M. Schoffstall, and J. Davin. 1988. A Simple Network Management Protocol. IETF RFC 1067.
[9] J. Case, M. Fedor, M. Schoffstall, and J. Davin. 1990. A Simple Network Management Protocol (SNMP). IETF RFC 1157.
[10] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. 1993. Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework. IETF RFC 1452.
[11] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. 1993. Introduction to version 2 of the Internet-standard Network Management Framework. IETF RFC 1441.
[12] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. 1996. Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework. IETF RFC 1908.
[13] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. 1996. Introduction to Community-based SNMPv2. IETF RFC 1901.
[14] Censys. 2021. Censys Scanning and Data Collection. https://censys.io/.
[15] Cisco. 2021. Catalyst 2960 and 2960-S Software Configuration Guide, 12.2(55)SE. https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_55_se/configuration/guide/scg_2960/swsnmp.html.
[16] Cisco. 2021. Cisco Bug: CSCts87275 - Cat4k with sup7e : same snmp engineID on different cat4k switches. https://quickview.cloudapps.cisco.com/quickview/bug/CSCts87275.
[17] J. Czyz, M. Luckie, M. Allman, and M. Bailey. 2016. Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy. In *NDSS*.
[18] David Dittrich et al. 2012. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. *US DHS* (2012).
[19] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *ACM CCS*.
[20] Z. Durumeric, E. Wustrow, and J. A. Halderman. 2013. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *USENIX Security Symposium*.
[21] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle. 2018. Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists. In *ACM IMC*.
[22] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle. 2021. IPv6 Hitlist Service. https://ipv6hitlist.github.io/.
[23] Oliver Gasser, Quirin Scheitle, Benedikt Rudolph, Carl Denis, Nadja Schricker, and Georg Carle. 2017. The Amplification Threat Posed by Publicly Reachable BACnet Devices. *Journal of Cyber Security and Mobility* 6 (Nov 2017). Issue 1. https://doi.org/10.13052/jcsm2245-1439.614
[24] R. Govindan and H. Tangmunarunkit. 2000. Heuristics for Internet Map Discovery. In *IEEE INFOCOM*.
[25] M. H. Gunes and K. Sarac. 2006. Analytical IP alias resolution. In *2006 IEEE ICC*.
[26] W. Hardaker. 2011. Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP). RFC 6353 (Internet Standard). https://doi.org/10.17487/RFC6353
[27] D. Harrington, R. Presuhn, and B. Wijnen. 2002. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. IETF RFC 3411.
[28] J. Holland, R. Teixeira, P. Schmitt, K. Borgolte, J. Rexford, N. Feamster, and J. Mayer. 2020. Classifying Network Vendors at Internet Scale. arXiv, https://arxiv.org/abs/2006.13086.
[29] Huawei. 2021. S2750, S5700, S6700 V200R003(C00, C02, and C10) Configuration Guide - Network Management and Monitoring. https://support.huawei.com/enterprise/en/doc/EDOC1000027472?section=j005.
[30] IEEE. 2021. List of MAC OUIs. http://standards-oui.ieee.org/oui/oui.txt.
[31] L. Izhikevich, R. Teixeira, and Z. Durumeric. 2021. LZR: Identifying Unexpected Internet Services. In *USENIX Security Symposium*.
[32] K. Keys, Y. Hyun, M. Luckie, and k claffy. 2013. Internet-Scale IPv4 Alias Resolution with MIDAR. *IEEE/ACM Trans. Networking* 21, 2 (2013).
[33] T. Kohno, A. Broido, and KC Claffy. 2005. Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable and Secure Computing* 2, 2 (2005).
[34] M. Kührer, T. Hupperich, C. Rossow, and T. Holz. 2014. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks. In *USENIX WOOT*.
[35] M. Luckie and R. Beverly. 2017. The Impact of Router Outages on the AS-level Internet. In *ACM SIGCOMM*.
[36] M. Luckie, R. Beverly, and W. Brinkmeyer. 2013. Speedtrap: Internet-Scale IPv6 Alias Resolution. In *ACM IMC*.
[37] M. Luckie, B. Huffaker, and k. claffy. 2019. Learning Regexes to Extract Router Names from Hostnames. In *ACM IMC*.
[38] A. Marder. 2020. APPLE: Alias Pruning by Path Length Estimation. In *PAM*.
[39] MITRE. 2021. Common Vulnerabilities and Exposures, SNMP. https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=SNMP.
[40] S. J. Murdoch. 2006. Hot or not: Revealing hidden services by their clock skew. In *ACM CCS*.
[41] RIPE NCC. 2021. RIPE Atlas. https://atlas.ripe.net/.
[42] Net-SNMP Project. 2021. Net-SNMP. http://www.net-snmp.org/.
[43] Nmap. 2021. Nmap: the Network Mapper - Free Security Scanner. https://nmap.org/.
[44] Craig Partridge and Mark Allman. 2016. Ethical Considerations in Network Measurement Papers. *Commun. ACM* (2016).
[45] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse.. In *NDSS*.
[46] E. C. Rye and R. Beverly. 2020. Discovering the IPv6 Network Periphery. In *PAM*.
[47] E. C. Rye, R. Beverly, and kc claffy. 2021. Follow the Scent: Defeating IPv6 Prefix Rotation Privacy. In *ACM IMC*.
[48] Matthew Sargent, John Kristoff, Vern Paxson, and Mark Allman. [n.d.]. On the Potential Abuse of IGMP. *ACM SIGCOMM CCR'17* ([n. d.]).
[49] Q. Scheitle, O. Gasser, M. Rouhi, and G. Carle. 2017. Large-Scale Classification of IPv6-IPv4 Siblings with Variable Clock Skew. In *TMA*.
[50] Shadowserver. 2021. Open SNMP Scanning Project. https://scan.shadowserver.org/snmp/.
[51] J. Sherry, E. Katz-Bassett, M. Pimenova, H. Madhyastha, T. Anderson, and A. Krishnamurthy. 2010. Resolving IP aliases with prespecified timestamps. In *ACM IMC*.
[52] N. Spring, R. Mahajan, and D. Wetherall. 2002. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*.
[53] W. Stallings. 1998. SNMPv3: A security enhancement for SNMP. *IEEE Communications Surveys* 1, 1 (1998), 2–17.
[54] S. Thomas. 2021. Brute forcing SNMPv3 authentication. https://applied-risk.com/resources/brute-forcing-snmpv3-authentication.
[55] Y. Vanaubel, J.-J. Pansiot, P. Merindol, and B. Donnet. 2013. Network Fingerprinting: TTL-Based Router Signatures. In *ACM IMC*.
[56] K. Vermeulen, B. Ljuma, V. Addanki, M. Gouel, O. Fourmaux, T. Friedman, and R. Rejaie. 2020. Alias Resolution Based on ICMP Rate Limiting. In *PAM*.
[57] S. Zander and S. J. Murdoch. 2008. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *USENIX Security Symposium*.

A COMPARISON OF DIFFERENT ALIAS RESOLUTION APPROACHES

Table 3 shows the results for different alias resolution variations. "First" and "both" revers to using fields from the first scan only vs. using fields from both scans for the matching process. The techniques differ only on the matching threshold for the last reboot time. All other fields are matched exactly. "Exact" denotes an exact matching of last reboot time, "Round" means that the last digit is rounded, "Divide by 20" means that the last digit is divided by 20 and cut off (i.e., put into bins of 20 seconds), "Divide by 20+round" denotes division by 20 and rounding of the resulting floating point number.

	Alias sets	Non-singleton alias sets	IPs in non-singleton alias sets	IPs per non-singleton alias set
Exact first	5.3M	903k	8.2M	9.1
Exact both	5.9M	892k	7.5M	8.4
Round first	4.6M	826k	8.7M	10.6
Round both	4.7M	835k	8.7M	10.4
Divide by 20 first	4.6M	820k	8.8M	10.7
Divide by 20 both	4.6M	824k	8.7M	10.6
Divide by 20+round first	4.6M	820k	8.8M	10.7
Divide by 20+round both	4.6M	824k	8.7M	10.6

Table 3: Comparison of different alias resolution approaches.

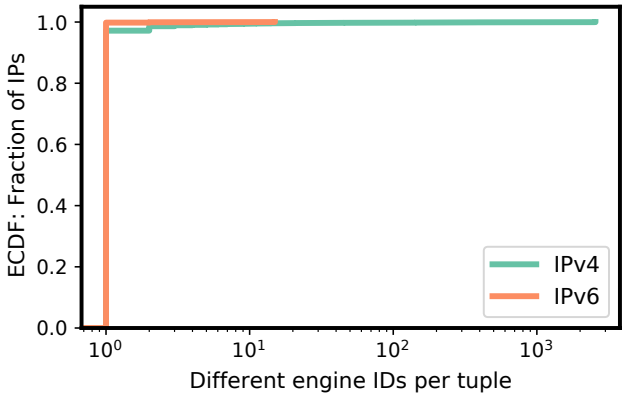

Figure 19: Uniqueness of (last reboot time, engine boots) tuples across IPv4 and IPv6 addresses: For the vast majority of tuples we see a single engine ID. Note that the x axis is log-scaled.

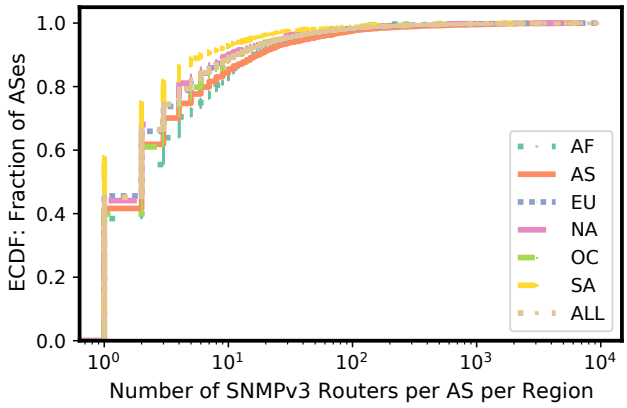

Figure 20: Distribution of number of routers per AS for different regions.

B UNIQUENESS OF LAST REBOOT TIME AND ENGINE TIME

In addition to the engine ID SNMP send engine time and engine boots values in their response. Our alias resolution technique leverages these information as well. By subtracting the engine time from the current time of the scan we calculate a last reboot time. In order to learn more about the uniqueness of last reboot time and engine time tuples, we look for cases where the same last reboot time and engine time values are seen at different devices, i.e., devices with different engine IDs. Figure 19 shows the distribution of the same last reboot time and engine time tuples for different number of engine IDs. We find that for 97.2% and 99.8% of IPv4 and IPv6 addresses have last reboot time and engine time tuples with a single unique engine ID. This shows that this tuple is indeed a valuable addition for our alias resolution technique.

C DISTRIBUTION OF ROUTERS PER AS AND REGION

In Figure 20 we plot the distribution of routers that our technique identified per network (AS) and per region. To map ASes to countries, we utilized CAIDA's AS Ranking [7]. This results in an AS-to-country reaches 99,9% for the ASes with routers our study, i.e., 22,769 out of 22,789. We are aware that a network may span multiple countries within a continent, i.e., we decided to map a network to a continent (region). Again, there are networks that span multiple regions, but in this case, typically, they register different ASes and address space in different regions. If this is not the case then, we assign the region of the headquarter of the network operator to the AS it owns. Our analysis does not show significant distributional differences across continents. However, most of the networks with largest number of routers are in North America (NA) and Europe (EU).