The Curious Case of Port 0

Mark Luchs and Christian Doerr
Delft University of Technology
2628 CD Delft, The Netherlands
Email: m.luchs, c.doerr@tudelft.nl

***Abstract*—In order to direct network traffic towards applications, transport layer protocols such as TCP and UDP add the notion of a port number. A share of these numbers is registered for well-known services such as a web or mail, while some is left to be dynamically assigned by the OS to client connections. A special case is port 0 which is reserved but was never assigned. Traffic from and to port 0 is unusual, because it should not occur in the wild. As port 0 is unassigned, there is no common service listing for connections here. Furthermore, operating systems usually interpret the request to open port 0 as the request to allocate and open any currently unused port. Thus, traffic from and to port 0 should not occur, because no application should listen there and applications cannot send from port 0.**

In practice, we do however see traffic from and to port 0, which indicates that someone makes the effort to bypass the normal operating system network stack to create these unusual packets. As a corner case of network protocols, the aspect of port 0 has basically never been thoroughly investigated. In this paper, we analyze network traffic collected through a /15 network telescope over a period of 3 years to characterize these curious data flows. We find that port 0 traffic seems to be used in the wild by a select few for a variety of purposes, from DDoS attacks to system fingerprinting, and that some of these actors possess a surprisingly sophisticated knowledge of OS behavior.

***Index Terms*—port scanning, back scatter, port 0**

I. Introduction

In order to differentiate between network flows and deliver network traffic to a specific application on the destination host, transport layers such as TCP or UDP include a port number as part of their header information. Port numbers are either registered at IANA and thus standardized for commonly used protocols such as telnet or HTTP, or dynamically allocated by the OS for temporary use by applications. The 16-bit port number in TCP and UDP allows for a total of 65536 ports, out of which port numbers 49152 through 65535 are designated for dynamic assignment by clients, and the remainder intended for services. Operating systems typically classify ports below 1024 as system ports, which for example on Linux may only be opened with elevated user privileges.

A curious case is port number 0, which is reserved by IANA but was never assigned to a specific application [1]. When you open a listening socket on a host and request port 0 from the operating system, this call is typically interpreted as a request to return a single, currently unused port to the application. In other words, opening a socket on port 0 will actually open a random port most likely in the range of 1025 to 49151. Also when a client is making a new connection to a server socket it will require a port number that is stamped onto the outgoing packets in order to allow the operating system to properly deliver any incoming returns. As stated above, system software will usually dynamically allocate one of the ephemeral port numbers in the range of 49152 through 65535 for these temporary connections. For this reason, there should not be any outgoing packet that originate from port 0, packets on the Internet should thus list any port numbers *but* port 0.

In reality, there exists a small but not negligible amount of packets on the Internet that either originate from or are sent to port 0, in some cases even both. Per our analysis above, this should not happen, as port 0 is neither registered to a specific application, and normal usage of the operating system's networking API should not result in a static or dynamic allocation of this number. If one would want to send packets from or to port 0, one strategy would be to bypass the operating system's networking stack, craft packets in a custom application and inject them in the network through a raw access mode such as libpcap. The required effort thus implicitly indicates a deliberate effort and an objective. This hence raises this curious question of the origin of this unexpected traffic, which we will investigate in this paper.

Being a curious corner case of network protocol design, the issue of port 0 has never received much consideration or a systematic analysis over the past 20 years in the trade or academic literature. In result, most of what is known about this type of traffic and its use is either speculation or folklore, with explanations ranging from its use in targeted system reconnaissance and operating system fingerprinting to stealth firewall exploitation and bypassing. This issue is further aggravated by the fact that a systematic study of this extremely low-volume traffic ($< 0.02\%$) requires a network of significant size to observe it and draw sound conclusions. Out of $140 \cdot 10^9$ packets received in the telescope over the period of 3 years, roughly $20 \cdot 10^6$ had port 0 as destination and/or source port.

This paper is the first systematic evaluation of this unusual type of network traffic, in which we investigate the origin of port 0 traffic in detail. Based on a monitoring of two /16 IP address ranges over 3 years, we find that port 0 traffic is used for a variety of goals, ranging from specialized DDoS attacks, scanning and reconnaissance activities to system fingerprinting. Specifically, this paper makes three contributions:

- We investigate the origin of the claim that port 0 is used for OS fingerprinting and trace the origin to a 2003 tool named *the Gobbler*. We show that this proof-of-concept would however not be effective at distinguishing between

ISBN 978-3-903176-16-4

current and old operating systems, and based on data from the telescope prove that it is not a source of port 0 traffic.
- We perform a systematic analysis of OS responses to port 0 packets under all possible header combinations and indeed to find type- and version-specific differences that could be used for fingerprinting. These fingerprints are different from what was originally described in *the Gobbler*, and an analysis of network traffic reveals that these fingerprinting techniques are actually adopted by adversaries in the wild. We further find that these actors utilize the different fingerprinting steps in a way that maximizes entropy, which indicates both detailed knowledge about OS behavior as well as the intention to operate with a minimum traffic footprint and visibility.
- We demonstrate that besides OS fingerprinting, the bulk of port 0 activity is actually linked to distributed denial-of-service attacks or port scanning. Curiously, DDoS practices in port 0 somewhat differ from those targeting specific ports likely due to the hypothesis of an adversary that port 0 leads to a successful firewall evasion.

The remainder of this paper is structured as follows: Section II summarizes the existing body of knowledge and expectations about port 0. Section III briefly outlines our measurement infrastructure. Section IV describes the principle of port 0 fingerprinting as described in the original proposal by [2], and tests the presence of this technique in our longitudinal dataset. Section V then presents alternative explanations for port 0 traffic. Section VI reviews our findings and concludes our investigation of the curious case of port 0.

II. Related Work

As a niche case for transport layer protocols and in general a tiny portion of network traffic, port 0 has received almost no systematic attention in the past. In result, much of today's claims of what this traffic is about is not rooted in data analysis and scientific experiments, but rather hypotheses and sometimes speculation. This is a pity, since there is an interesting story to tell as we will show in the remainder of this paper. In the following, we will describe what is known to date about port 0 and the evolution of these artifacts.

When one queries the Internet for keywords related to port 0 traffic, one discovers a collection of forum questions and blog posts from network administrators searching for the origin of this small, but unusual type of traffic. These forum discussions and support inquiries usually arrive at the conclusion that port 0 traffic should not normally be generated by hosts due to the reasons discussed in the introduction, and can be attributed to malicious fingerprinting activities, reconnaissance and potentially firewall evasion techniques, for example in [3] and [4]. Often the precise motivation for these use cases is rather specific. For example, in case of firewall evasion they came from some firewall configuration GUIs that allowed ports to be selected from 1 to 65535 but left out 0. Findings and claims such as this have led to the notion that port 0 traffic can hence be used to bypass defenses, and gather information which would otherwise be blocked by a network filter.

The origin for the fingerprinting hypothesis can be traced back to a post by Ste Jones to the Nmap developer mailing list [2], where he put forth the idea of using traffic from and to port 0 as a means of operating system identification. Given seven different probing packets, Jones's post lists some differences between eight operating systems used at the time. The idea later evolved into a proof-of-concept tool, the *Gobbler*, whose open source development was stopped a few months later in 2003. In the meanwhile, the design of the Linux operating system, header files, and compilers has advanced, and *the Gobbler* source code could soon no longer be compiled on modern environments. Yet, the mailing list post and the proof-of-concept somehow cemented the idea of port 0 traffic being related to operating system fingerprinting.

In the expanding universe of penetration testing and security tools, a number of applications do have support to attempt detection of the operating system based on subtle variations in the way they react to network traffic. The most common tool nmap [5] for example elicits behavioral differences when sending a train of eight TCP and UDP packets with unusual header values. Windows 7 should for example react to a TCP SYN packet sent to an open port with certain options in a predictable way, returning a SYN+ACK with a TTL between 0x7B and 0x85, the TCP sequence number of the response being a value not equal to 0 and not the sequence number of the SYN packet, and the acknowledgement number being the probe's sequence number incremented by one. If the observed response matches this pattern, Windows 7 remains in the pool of candidate OSes, otherwise Windows 7 is discarded. The process continues for all eight tests, and nmap returns a set of operating system guesses based on its current database of some 5600 patterns. These same principles can to some extent also be exploited passively, by observing features in ongoing connections. This is exemplarily implemented by p0f [6], which stands for passive fingerprinter. P0f also bases its OS guess on header field initialization or progression values (it can also discover network topology setups), but injects no network traffic itself. As header values might be modified in transit by forwarding (such as the IP TOS, TTL, etc.) or actively replaced by firewalls, XProbe2 [7] relaxes the strict matching and introduces fuzzy fingerprint matching. Instead of ruling out candidates in case of a non-match, XProbe2 assigns each OS/fingerprint combination a match based on level of certainty, and thus includes potential but not-conclusive candidates in the list of results. None of the tools however leverage the initial discovery by [2]. They implement OS detection solely on packets sent to open and closed ports, but do not implement a suite of probes based on port 0.

The first and only academic analysis of port 0 is the analysis of Bou-Harb et al. [8] in 2014. As part of an incident response effort, the authors analyzed 30 GB of packet traces collected from a darknet for two separate days in 2013, and singled out TCP traffic originating from port 0. In this data set, they identified 27 hosts sending such packets, of which they linked 28% via malware domain lists to the Sality malware.

The loose ends around the hypothesis of port 0 traffic as a

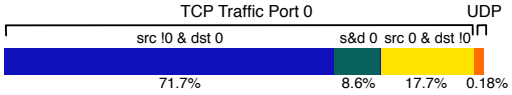


Fig. 1. Percentage of port 0-related traffic by port combination from the telescope study described in [9].

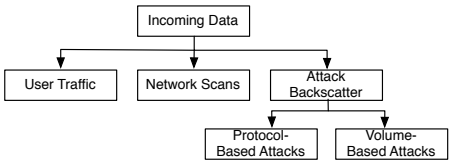


Fig. 2. Traffic on an IP address is a traffic mix of user data, network scans and attack reflections, so-called backscatter.

source of fingerprinting yet the absence of this technique in all common fingerprinting tools, as well as the limited empirical analysis of this traffic, does not provide a definite conclusion what actually is the origin and purpose of this traffic. Using the /15 network telescope described in [9], we collected a longitudinal trace of darknet data spanning a period of 3 years with a total of 11.5 TB of traffic. Aside from a long duration, we extend the analysis to both TCP and UDP, as well as traffic from and to port 0. We see that the vast majority of traffic (71.7%) actually comes from a non-zero port but is directed to port 0. TCP traffic from source port 0 only accounts for 17.7% in this long term trace, and traffic simultaneously from and to port 0 accounts for another 8.6%. UDP traffic from or to port 0 is minor at 0.18% in total as shown in figure 1. Combined we find a total of 33,776 sources responsible for all traffic, of these 23,541 send TCP traffic and 10,235 send UDP traffic. Only 611 sources send both TCP as well as UDP.

The analysis presented in [8] investigates a spike in TCP traffic directed at port 0 over a period of two days. There is no discussion however on the presence of other port 0 traffic however. Our investigation into three years of telescope data however revealed the presence of both TCP as UDP traffic, direct at and originating from port 0. Furthermore, the long term usage of port 0 traffic could be significantly different from that observed during incidents.

In the remainder of this paper, we will thus analyze both TCP and UDP traffic both to and from port 0, and perform a longitudinal analysis over a period of 3 years, to classify the purpose and the dynamics of this usual type of network traffic.

III. Data Collection

As soon as one connects a host to the Internet, also immediately the first packets trickle in, even if the connected host has not initiated any connection itself. Some of the packets are network scans, probing publicly connected networks for attached hosts and open ports. Knowledge of open ports and available services can in later steps be used to launch compromisation attempts or to utilize services with a high amplification factor between request and response sizes. Such amplification factors are used in distributed denial-of-service (DDoS) attacks in order to saturate the connection of the victim with a large volume of unwanted data. In non-amplification attacks, an adversary will attack hosts with the attempt to exhaust some finite resource. Operating systems and application services typically have a fixed limit of the total number of simultaneous connections they are willing or able to handle in parallel, and by making many connection requests this pool is drained by the attacker so that regular users may no longer utilize the service. In order to try evasion, an adversary will normally spoof the network packet's source IP addresses in this protocol-based attack to a random value. This means that the attack cannot be attributed to the perpetrator and is also much more difficult to block. As the response of the victim to the connection request is delivered to the alleged origin, this means that every IP address on the Internet normally receives a small amount of these attack reflections which can be used to quantify and assess attack activities on the Internet.

When monitoring the traffic on a publicly-reachable IP address on the Internet, the incoming data will thus be a mix of these three traffic sources as shown in figure 2. In order to increase the prediction power and statistical reliability of the conclusions, normally a large number of IP addresses are bundled together to increase the overall volume of scan and backscatter traffic [9]. A so-called network telescope observes a large number of otherwise unused IP addresses for exactly this purpose; as the addresses are not connected to clients, the traffic collection will also contain no user data and aside from cleaning the data to only contain malicious traffic, conveniently addressing the concern of user privacy.

The analysis presented in this paper is based on a network telescope of two partially-used address ranges of size /16. The infrastructure has been in use for approximately 3 years, and has captured approximately 11.5 terrabytes of adversarial traffic. From this traffic we isolate for our analysis any TCP or UDP traffic either originating from port 0 or destined to port 0 (or both). As port 0 traffic is a rather obscure case, the total share of these packets is comparatively small, again underlining the need to utilize a large telescope and an extended period of observation to establish reliable results. From the billions of captured packets, only a total of 19,718,047 packets remains on which the analysis in the following sections is based.

Universally, scanning traffic is defined as a source sending more than 3 packets towards a single host. Within the port 0 dataset however we notice there is a significant number of sources with limited traffic, let alone single IPs. Dropping traffic which less than k packets has the following effects on packets and sources that would remain in the dataset:

	≤ 0	≤ 1	≤ 2	≤ 3
Packets (%)	100	99.90	99.86	99.84
Sources (%)	100	42.04	30.39	25.13

As we can see, the influence on the number of total packets is hardly noticeable, however the number of sources will drop drastically. Filtering our data this way would have the

consequence that the total picture of port 0 becomes skewed, additionally such packets could be sent from so-called slow scanners collaborating together in scanning campaigns. One such example will be given further down this research. As such we maintain the full port 0 dataset for our work.

IV. The Gobbler: OS Fingerprinting using port 0

As the probing tool *the Gobbler* is the only reference that explicitly mentions the use of port 0 to fingerprint operating systems, it may be that this tool or derivatives of it are today in use to probe hosts. We have hence analyzed the description provided by [2] and the source code to compile a distinctive fingerprint for the tool itself, which allows us to identify usage of the application in our telescope dataset.

The operational concept behind *the Gobbler* is the same as for other tools, sending specifically crafted packets towards a host. Differences in the observed replies can then be matched against fingerprints of known operating systems. *The Gobbler* is distinctive from other fingerprinting tools in that is only sends TCP SYN and UDP packets destined for and/or from port 0, behavior which is not incorporated by any other tools.

A. Gobbler Fingerprinting Techniques

In the context of his fingerprinting proposal, Jones described seven different probing packets to which operating systems would respond differently. All probes either originated from port 0 or were sent to port 0, and depending on whether a response was returned and the flags it contained, one or more OS could be a potential candidate for the remote host. Specifically, the proposal defined the following test packets:

- P1 Send TCP packet from source port 0 to port 0
- P2 Send TCP packet from source port $\neq 0$ to port 0
- P3 Send TCP packet from source port 0 to open port
- P4 Send TCP packet from source port 0 to closed port
- P5 Send UDP packet from source port 0 to port 0
- P6 Send UDP packet from source port 53 to port 0
- P7 Send UDP packet from source port 0 to closed port

which in the source code were executed sequentially without any further control logic or branches triggered based on earlier responses. According to the documentation, probe 6 is specifically tied to port 53, DNS, as it is most likely to bypass firewall configurations. Why this is not defined specifically for other probes remains unspecified. Since each probe requires either an open or closed port, prior knowledge of the system being probed has to be available. *The Gobbler*'s documentation further included a fingerprint database in a similar format to nmap, which listed a set of reference responses for eight operating systems from that time. Figure 3 shows an example for OpenBSD 3.2/3.3. A value of N indicates no response, an entry of Y that a reply was received. In a return packet some flags could be set, and A and R indicate that the acknowledgement bit and reset bit flags were set in the TCP header. The list appears as a work-in-progress, naming for example the "Linux OS" but without indicating the distribution or kernel version. Other entries such as SunOS 5.6 contain

```
Fingerprint OpenBSD 3.2/3.3
P1(Resp=Y%Flags=AR)
P2(Resp=Y%Flags=AR)
P3(Resp=N)
P4(Resp=Y%Flags=AR)
P5(Resp=N)
P6(Resp=N)
P7(Resp=Y)
```

Fig. 3. Sample fingerprint from the Gobbler for OpenBSD 3.2/3.3

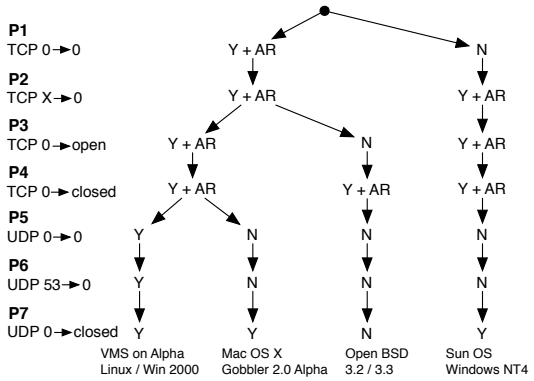


Fig. 4. Gobbler probing procedure and original OS fingerprints.

remarks such as "can someone confirm please", or questions such as "Service Pack?" or "with checkpoint?".

On inspection of the full database in [2], one can see that even for the provided examples, there is actually limited variation in the OS responses which makes the signatures rather unspecified for the task at hand. Furthermore, if we analyze the actual entropy which might be reaped from each probe, we notice that certain probing packets such as a P4 make no difference in response - every OS replies with a frame where the ACK and the RST bits are set. The proposed fingerprints have thus limited prediction power.

This issue is visualized in figure 4, which depicts the responses expected for each of the seven probing methods for all entries listed in the fingerprinting database. As we can see from the graph, the first test of a TCP frame from port 0 to port 0 will split the entire database in two sets, on the one hand SunOS and Windows NT4 and on the other all remaining database entries. From now on, no probe has any further prediction power to differentiate between the two candidates in the first set. Also in the left part of the graph several probes appear unsuited to further break down the candidate pool. A more efficient fingerprinting system should inject probes in order of maximum expected entropy, ideally pruning items in the probing sequence given new knowledge and selecting those where the highest new knowledge gain can be expected. Of more use would have been the static probing order P2, P4, P1, and P5 which would have resulted in more insightful splits at a faster pace. We further need to note that to any adversary P3 and P4 basically appear as one, as apriori a scanner would not have any knowledge whether a port is actually open or closed. In order to be effective, this OS fingerprinting must be preceeded by a port scan for open ports.

TABLE I
RESPONSES TO GOBBLER PROBING TO A SELECTION OF UBUNTU AND WINDOWS OPERATING SYSTEMS.

	Ubuntu 04.10	Ubuntu 10.04	Ubuntu 12.04	Ubuntu 16.04	Windows 7	Windows 95	Windows XP
P1	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20
P2	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20
P3	N —	N —	N —	N —	Y — 20	Y — 20	Y — 20
P4	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20	Y — 20
P5	N —	N —	N —	N —	N —	N —	N —
P6	N —	N —	N —	N —	N —	N —	N —
P7	N —	N —	N —	N —	N —	N —	N —

As *the Gobbler* database was, and remains, incomplete and network stack implementations today could leak more or different information based on this unexpected type of traffic, we re-implemented the now defunct *Gobbler* implementation and tested the strategies on a number of operating systems that have appeared over the past decade. Table I shows the responses to the seven probing packets for a variety of Ubuntu and Windows versions, with an N indicating no response and the value next to Y describing the decimal value of the flag combination returned. The only appearing flag combination 20 or 0b010100 (16+4) stands for ACK+RST, given the bit vector URG ACK PST RST SYN FIN in the TCP header. As we can easily see from the table, the only difference between Windows and Ubuntu Linux is the response to P3, a TCP packet from source port 0 to an open port, while no variations between the different versions of OS families exist.

B. Measuring Gobbler Usage

Aside from the distinctive sequence of unusual packets, also the internal implementation of *the Gobbler* tool contains some behavioral characteristics in the way it generates and sends probe packets. Based on a source code review, the tool will

- send a probe and then wait no longer than 250,000μs for a response. After this period another probe will be sent. This process will repeat itself 20 times before continuing.
- repeat the process for all 7 probing methods, sequentially from probe 1 to probe 7. The only decision logic here is based on whether there is an open or closed port available, which might lead to skipping of test 3 or 4.
- require knowledge of open and closed ports to do its fingerprinting analysis. As such, a port 0 fingerprinting attempt is preceded by a port scan which determines the state of the ports on a host. The tool itself has references to do this via nmap's port scanning functionality.
- send packets, where the TCP sequence number, the IPID, and MAC addresses are filled with random information. While the MAC addresses are not visible beyond a LAN, we can statistically test the header data for randomness.

Given this characteristic fingerprint, we analyzed the incoming network telescope traffic for sequences of the seven probes, and in a second step verifying the operational interarrival time or resend patterns specific to *the Gobbler* implementation. As in a dark IP space there will be no responses, each test packet should repeat itself up to the maximum of 20.

When we plot the distribution the port 0 traffic matched to the semantics of the seven probes as shown in figure 5, we

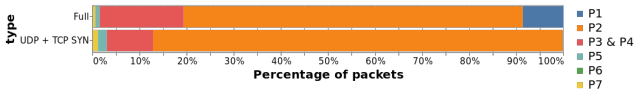


Fig. 5. Distribution of Gobbler probing packets in the telescope data.

can immediately notice that the main source of the incoming port 0 probing traffic cannot be *Gobbler* fingerprinting traffic as the number of packets per category is drastically different while it should be approximately equal. P3 and P4 are listed together in the figure, as we have no insight whether the adversary expected to find the targeted port to be open or closed. Furthermore, we can also conclude that *the Gobbler* is not used at all due to the complete absence of P6 probes from the dataset. Needless to say, no source IP matched the characteristic interarrival time or repeat pattern of *the Gobbler*. It must therefore be another tool that creates these probes.

While we can conclude that *the Gobbler* tool is no (longer) in use to facilitate operating system fingerprinting, we can also notice from figure 5 the astonishing fact that the most commonly used probing methods in descending order P2 - P3/P4 - P1 - P5 matches exactly our analysis from above on the algorithmically most efficient scanning order. As less than a quarter of probes continue from P2 to P3/P4, it would also suggest that some information about the progress is considered into the process. This would suggest that the techniques originally presented in *the Gobbler* paper have been incorporated and potentially further developed by someone performing port 0 based fingerprinting analysis. We will get back to this observation in the next section again.

V. THE MIXED USAGE OF PORT 0

While there does not exist any evidence that *the Gobbler* is (still) being used for fingerprinting using port 0 traffic, packets to and from port 0 indeed do elicit a minor difference between operating systems, and our traffic inspection reveals that these probes see minuscule usage in practice. In this section, we will analyze this data for commonalities and recurrent patterns to identify potential purposes behind this traffic.

When we look at the volume of port 0 packets over time, we see that it deviates from the normal network activity that is observed for regular network traffic, such as the typical diurnal behavior of backbone traffic or the basically continuous port scanning activity as background noise. Figure 6 shows the amount of port 0 traffic as a function of time, on the top of all port 0 packets and at the bottom for TCP SYN+ACK

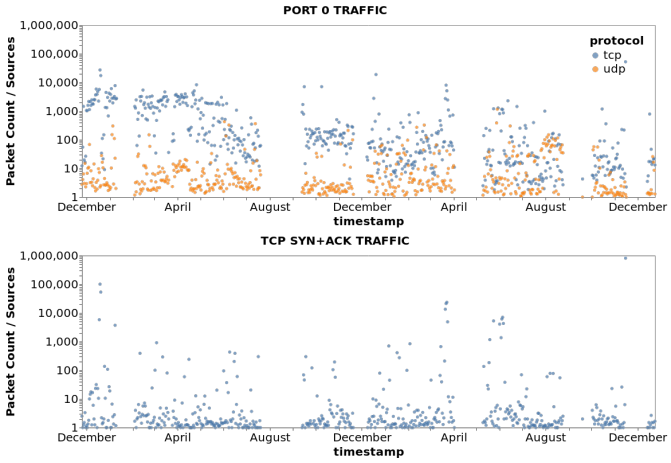


Fig. 6. Incoming port 0 traffic volume averaged by the number of sending sources, on top for all traffic, on the bottom for TCP SYN+ACKs.

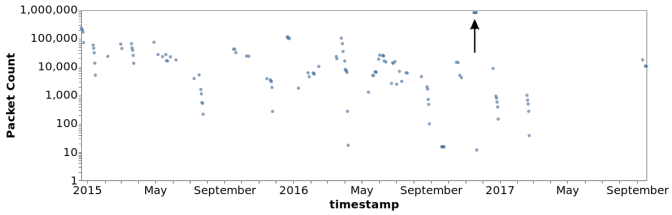


Fig. 7. Instances with a ten fold increase in momentary packet count.

which can be associated with backscatter based on [10]. We immediately see that the traffic volume is neither static nor shows any repeating patterns, at the same time we can identify that much of the activity around port 0 in terms of packets sent appears in bursts at concise moments in time indicating the activities of single, coordinated sources. A momentary burst in backscatter drives also a surge in general port 0 traffic. This is observed in figure 6 by a spike occuring in both figures simultaneously. The reverse is however not true, and peaks in general port 0 traffic can be frequently linked to the presence of high-speed scan campaigns, as discussed in section V-B.

How we experience port 0 traffic thus depends on the moment and interval of observation. If we tuned in during one of the comparatively rare but regularly occurring DDoS attacks or scan campaigns, an entirely different picture would emerge than if our analysis would fall into a period of more background activity. Over a period of 3 years, we have discovered the following different types of behaviors.

A. Denial of Service

As discussed in section III, TCP SYN+ACK traffic is typically interpreted as backscatter traffic. Such backscatter traffic will appear in our sensors when a victim is flooded with a (D)DoS and the source IP used by the packets is spoofed and part of our telescope range.

Based on the comparison of total port 0 traffic to port 0 TCP SYN+ACK traffic in figure 6 we have seen that bursts in overall traffic volume are typically driven by a synchronous increase in SYN+ACK traffic, in other words port 0 surges are usually the result of backscatter from denial of service attacks. If we were to sum and group the volume of received port 0 packets by source IP, we see that when viewed over the entire time frame 24.7% of the entire DDoS traffic volume originates from just a single victim (or source IP) while exactly a third of all packets could be attributed to 10 victims during a handful of short, but heavy DDoS attack instances. The rarity of these events becomes further evident in figure 7 which depicts the data from figure 6, but only displays a data point whenever the traffic volume during a specific day temporarily exceeded the moving average by a factor of 10. Over the entire three year period, we spot only 85 of these instances, but for the three largest events (indicated by an arrow) the volume actually exceeds the normal level by a factor of 1000. While a handful of IP addresses dominate the overall ranking in terms of packet count and would hence skew our interpretation of port 0 traffic to be tightly connected with (D)DoS activity, an analysis from the perspective of unique sources does paint another picture. Viewed over the three year period, only 10.3% of source IP addresses ever caught in the telescopes had actually sent a SYN+ACK packet, and only 8.06% of sources had *only* been seen sending such traffic which corresponds to the typical behavior of a backscatter victim. Thus, while in terms of volume a few DoS instances stand out, it is actually not the main activity behind port 0 traffic in terms of the number of hosts as 90% of all source IPs are engaged in other activities.

While the relative distribution of DoS attack sizes somewhat resembles the skewed distribution of non-port-0 backscatter in [9], port 0-based reflections do also differ significantly from previous attacks observed. Recall from our brief introduction on network telescopes in section III the general working principle of protocol-based DDoS attacks such as TCP SYN floods, where the perpetrator will normally spoof the source IP addresses of the attack packets to random IP addresses to cloak his identity and make mitigation more difficult. Statistically speaking, this random spoofing would result in an approximately equal amount of backscatter traffic across our telescope. Figure 8 displays the actual count of backscatter received for the IP addresses in our two telescope ranges. We notice that while some source IP randomization is going on, this practice seems to be not well developed for port 0-based DoS as most telescope IPs are never hit with backscatter. While massive source randomization is a standard practice in "regular" resource-depletion DDoS attacks, this practice is not adopted to the same extent in these type of DDoS attacks. Thus, the denial-of-service attack actually does appear less distributed from the perspective of the victim. This would make it in principle easier to defend against, but the fact that we actually receive port 0-related backscatter implies that the victims have not configured their firewalls to filter out port 0 traffic in the first place. If this filter would be applied, how randomized and distributed the DDoS attack actually is would no longer matter, as any port 0 DDoS traffic would hit this

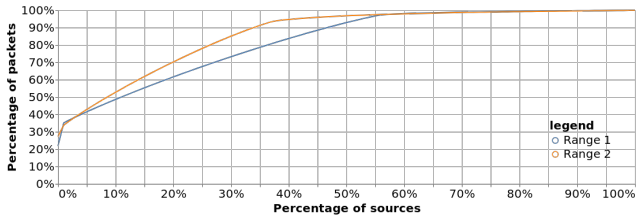


Fig. 8. Backscatter received per IP from sources only sending SYN+ACK.

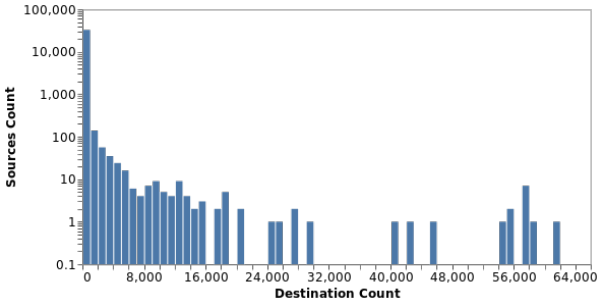
rule. This lack might serve as an explanation why actors might forego source randomization in port 0 DDoS attacks.

B. Scanners

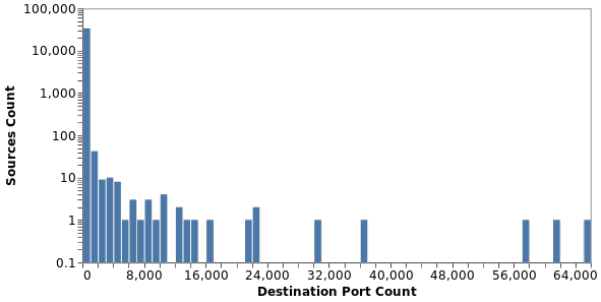
Recall from our discussion above that port 0-related DoS attacks were only experienced by a comparatively small amount of source IPs. This in turn implies that most are actually engaged in other activities, specifically we found that the bulk of port 0 traffic was related to port scanning. TCP SYN traffic, which is typically interpreted as scanning traffic, constitutes 48.8% of all traffic. Port scanners are traditionally classified into three groups. First, vertical scanners that test a large number of ports on a single host, which typically occurs when an adversary is trying to break into a specific host and needs to locate any available open port. Second, horizontal scanners test the same port across a large range of hosts. They specialize in a specific type of service and aim to find any host with that service that they can exploit. And third, block scanners which test a selection of ports across a large number of hosts.

Figure 9 displays the targeting behavior of sources sending traffic towards our IP ranges, in subfigure (a) a histogram of the number of IPs in our telescope that are targeted per source IP, in (b) the total number of ports a particular source IP scans on a given target. From the graphs and a combination of the data points which is not shown here, we actually find all three scanning behaviors are present in the port 0 traffic: (1) there exist a large number of scanners that only target a few IPs but scan many ports, (2) there exist scanners which target one IP across wide ranges and (3) there are those which test thousands of hosts for hundreds of open ports. Note that the number of ports and hosts is displayed logarithmically, which is necessary to the heavy skew in the distribution. What stands out is that most sources only scan a few IPs, and that by overall volume most only specialize on a limited number of ports.

If source IPs target only a few destination IPs, this could either mean that the scanner is not very determined to accomplish a high coverage, or that multiple sources work together while individually sending only a few packets. This would allow such a group to stay under the radar and avoid detection. While we will discuss an example of such collaborative behavior in the next section, we will now focus on the high volume scanners that systematically test our IP ranges at scale. In figure 9(a), we identify 14 such instances which do probe more than 40,000 IP addresses in the telescope. Taken together, these 14 senders account for a whopping 45.31% of all port 0 packets in the entire dataset.



(a) Number of destinations scanned



(b) Number of ports scanned

Fig. 9. Number of destination hosts and ports scanned.

Despite their similar profile and comparable activities, these horizontal and block scanners are actually quite different in how they go about their scan. Table II lists the total number of destination IP addresses targeted, the total number of unique IP IDs in the probing traffic, the total number of destination ports targeted, the total amount of source ports, the total number of unique TCP sequence numbers in the fingerprinting traffic, and the number of days these high profile scanners were observed. While most share one property, for example most are horizontal scanners of which half have a fixed IP ID in common, groups of the same property do not in general align with those groups identical in another way. In other words, the only common criteria is that they scan a lot, but it appears that they all differ in the concrete way they instantiate and operate their scan. A low number of unique IP IDs, source ports, or TCP sequence numbers for most of them does indicate that these probe packets are custom crafted and directly injected into the network by a specific low-level tool, as the operating systems networking stack would normally randomize these values. The tools normally leave a distinctive fingerprint in the packet headers - like the resend pattern and probe interarrival time in case of *the Gobbler* - and we tested the probing traffic with the method described in [11], which can identify the most common port scanning tools nmap, unicorn, masscan and zmap. The port 0 traffic matched however none of these commonly used tools, therefore hinting to a special tool developed for this purpose.

One property which was common across most scanners was the surprising speed in which they combed through our telescope ranges, typically finishing their scan in a matter of a

TABLE II
BEHAVIORAL SIMILARITIES OF THE SCANNERS TARGETING AT LEAST 40.000 IP ADDRESSES.

source IP	# dIPs	# unique IP IDs	# dPorts	# sPorts	# TCP seq	Days active
212.90.62.209	40729	30420	1	1	40729	4
116.66.207.130	45167	24553	1	1	24558	1
46.244.10.94	55059	1	1	1	0	2
162.244.28.23	55595	37521	1	2	55594	1
195.19.11.226	57185	38097	1	1	57184	3
195.168.26.64	57222	38222	1	1	57221	2
122.224.153.110	57554	38437	1	2	57554	1
107.154.64.10	42715	1	10551	1	1	34
119.9.107.156	57448	1	1	27117	90641	4
155.94.254.133	54880	1	1	27622	54880	4
80.82.64.213	57389	1	3	1	127767	2
119.9.107.180	57501	1	1	28123	156500	2
222.39.152.147	57498	32768	1	1	100267	3
123.151.42.61	58921	1	1	10	49	33
113.108.21.16	61942	1	1	20	208	95

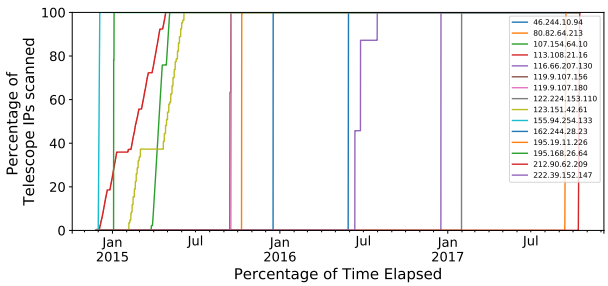


Fig. 10. Progression of the horizontal and block scans over time.

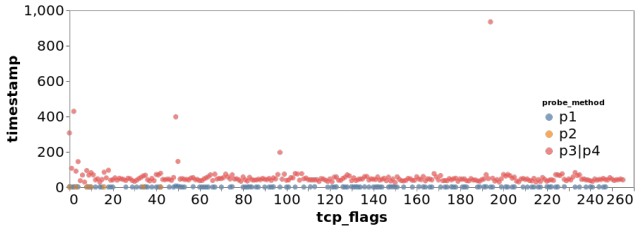


Fig. 11. Distribution of TCP header flag combinations for P1, P2, and P3.

few days. Figure 10 shows the percentage of the IPs a scanner has tested in the telescope as a function of time over the observation period. As we can see in the graph, every scanner only performs one scan which is finished in one sweep. As soon as the scan is completed, these addresses never come back during the entire three year period, but every couple of months another IP address again targets our range.

C. Fingerprinting

Starting point for our investigation of port 0 was the widespread belief that these unusual packets were meant to fingerprint operating systems. Over the course of the analysis in this section we have however seen the bulk of the traffic could be attributed to backscatter from denial-of-service attacks and port scanning activities. Still, we do find that there is a small yet interesting amount of operating system fingerprinting happening on port 0.

While *the Gobbler* had proposed fingerprinting based on TCP and UDP packets from and to port 0, it had based its method on plain packet headers, in other words aside from the port number the packets did not contain any header information which would be illegal or otherwise unusual. In order to test whether operating systems did indeed react differently to corner cases in TCP header values in combination with port 0, we reran our fingerprinting study for a variety of operating systems for all possible header flag combinations. This would hence include flag combinations which are undefined or prohibited, and would thus be well suited to trigger customized and unusual responses from OS implementations. Given the 8-bit flag header of TCP, this resulted to a total of 256 combinations per probing method, thus yielding 1028 signatures per operating system. Given this extended test set, a set of 52 flag combinations stand out as they trigger clear differences between almost all OSes tested. Table III shows a small set of five TCP probes from port 0 to an open port (which the tool named P3) for different TCP flags from the total list of all variations. Our comprehensive fingerprinting not only allows a differentiation between Ubuntu Linux and Windows as the original tool was able to, but also to distinguish between old and more recent versions of them. Our finding shows that fingerprinting using port 0 is indeed possible. Compared to other fingerprinting techniques however the obtained knowledge is best used during initial exploration, especially considering that port 0 is hardly monitored and easily escapes detection.

When we look at the TCP flags in use across all port 0 traffic as shown in figure 11, we find that for the TCP traffic from port 0 to port 0 (in *Gobbler's* terminology P1) and TCP from port $\neq$ 0 to port 0 (P2) no unusual combinations are actually used excessively in practice. This is however different for P3/P4, a TCP packet from port 0 to an open/closed port, for which a handful of TCP flag combinations stand out from the crowd. Interestingly, the flag combinations that are targeted above average in the wild have for the top 20 flags a significant overlap of 18/20 with those which in our experiments elicited a different between operating systems, thus most adversaries

TABLE III
SELECTION OF ENTROPY-RAISING FINGERPRINTING PROBES, HERE FOR DIFFERENT UBUNTU LINUX AND WINDOWS VERSIONS.

Probe	TCP flags	Ubuntu 04.10	Ubuntu 10.04	Ubuntu 12.04	Ubuntu 16.04	Win 7	Win 95	Win XP
P3	3	Y — 18	Y — 18	N —	N —	Y — 20	Y — 18	Y — 18
P3	96	N —	N —	N —	N —	Y — 20	Y — 20	Y — 20
P3	202	Y — 18	Y — 82	Y — 82	Y — 82	Y — 18	Y — 18	Y — 18
P3	203	Y — 18	Y — 82	N —	N —	Y — 20	Y — 18	Y — 18
P3	235	Y — 18	Y — 82	N —	N —	Y — 20	Y — 18	Y — 18

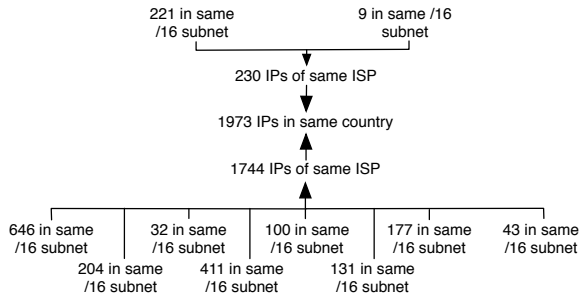


Fig. 12. Network- and geographic relationships of 1973 IP addresses participating in a coordinated fingerprinting activity.

appear not to blindly use TCP flags or fuzz remote hosts but seem to be very selective in what they do. To further support this hypothesis, direct interaction with the adversaries scanning our systems is required, mimicing the reponses of the tested OSes. The low activity pattern for suspicious actors however means that a multi-year capture period is required for sufficient data, this is currently underway and left for future work.

When we select the top 20 flag combinations and group those IPs together which utilize a similar or identical set of methods, sets of clusters emerge that appear to perform directed and very informed OS fingerprinting, accounting for about 6% of all IP addresses involved in port 0 communication.

Especially noteworthy is the largest of these clusters. Compromised of approximately 2000 source IP addresses, the source IPs all send the same sets of probes, target the same destination port and use 17 out of the 20 distinctive fingerprint combinations. The activity of the cluster is comparatively stealthy as it only sends 0.069% of all packets in the dataset, which is distributed across all cluster members. Taken together these hosts however display a remarkable amount of coordination, as out of the 9765 probes with which it targets our telescope, 8908 probe packets are unique and together complement each other within a systematic survey of our systems. Figure 12 shows the location of the IP addresses from the largest fingerprinting cluster with respect to their network addresses. The 1973 source IP addresses logically fall into ten /16 networks, 8 of which belong to network provider A, 2 to a network provider B, both located in the same country.

VI. CONCLUSION

In this paper we have investigated the curious case of port 0 traffic, packets that either originate from or are sent to port 0 which under normal circumstances should never happen. The fact that they do occur means that they are explicitly crafted in specially designed programs to bypass the regular network stack, an effort the program designer will only undertake to accomplish a specific purpose. While to date almost no systematic evaluation of port 0 traffic has been performed, forum and blog posts associate it with OS fingerprinting based on a post that proposed the technique in the early 2000s.

Based on a 3 year trace, we have systematically analyzed the characteristics of port 0 traffic. We could demonstrate that fingerprinting based on the methods proposed some 15 years ago is entirely absent, but that the idea of eliciting corner cases was experimentally shown to be possible. We could empirically show that some use it in the wild efficiently.

The curious case of port 0 is however more nuanced, as we find that port 0 traffic is also and actually predominantly used for purposes other than OS fingerprinting. A large use case in terms of traffic volume are DoS attacks, which occur comparatively rarely but when they do are large in magnitude. Nearly half the entire port 0 traffic is however due to the activities of network scanners, all of which exist and utilize port 0 as a source port to scan networks for connected hosts and open ports. We find the presence of high profile scanners, which systematically probe tens of thousands of IPs in our network telescope and have enough resources to complete the process in a matter of days, only to never return.

Port 0 pertains to a small percentage of network traffic, and protection can be easily gained by dropping all traffic with this port. Nonetheless this anomalous traffic teaches a valuable lesson: It is important to stay aware and vigilant with respect to assumptions that are made. Only because it shouldn't happen does not mean it does not do so in practice.

REFERENCES

[1] IETF, "RFC 1700 - assigned numbers," 1994.
[2] S. Jones, "Port 0 OS fingerprinting," *Nmap mailing list*, 2003.
[3] L. Constantin, "Spike in traffic with TCP source port zero has some researchers worried," in *PCWorld*, 2013.
[4] R. Havelt and W. G. Henrique, "Earth vs. the giant spider: Amazingly true stories of real penetration tests," *DEFCON 19*, 2011.
[5] G. Lyon, "Nmap security scanner." https://nmap.org/, 2018.
[6] M. Zalewski, "p0f passive fingerprinter." http://lcamtuf.coredump.cx/p0f3/, 2016.
[7] O. Arkin and F. Yarochkin, "Xprobe v2.0 a "fuzzy" approach to remote active operating system fingerprinting," tech. rep., The Sys-Security Group, 2002.
[8] E. Bou-Harb, N.-E. Lakhdari, H. Binsalleeh, and M. Debbabi, "Multidimensional investigation of source port 0 probing," *Digital Investigation*, 2014.
[9] N. Blenn, V. Ghiette, and C. Doerr, "Quantifying the spectrum of denial-of-service attacks through internet backscatter," in *ARES*, 2017.
[10] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston, "Internet background radiation revisited," in *SIGCOMM IMC*, pp. 62–74, 2010.
[11] V. Ghiette, N. Blenn, and C. Doerr, "Remote identification of port scan toolchains," in *IFIP NTMS*, 2016.