Remote Identification of Port Scan Toolchains

Vincent Ghiëtte, Norbert Blenn, Christian Doerr
Delft University of Technology
Cybersecurity Group
Delft, The Netherlands
{v.ghiette@student., n.blenn@, c.doerr@}tudelft.nl

Abstract—**Port scans are typically at the begin of a chain of events that will lead to the attack and exploitation of a host over a network. Since building an effective defense relies on information what kind of threat an organization is facing, threat intelligence outlining an actor's modus operandi is a critical ingredient for network security. In this paper, we describe characteristic patterns in port scan packets that can be used to identify the tool chain used by an adversary. In an empirical analysis of scan traffic received by two /16 networks, we find that common open source port scan tools are adopted differently by communities across the globe, and that groups specializing to use a particular tool have also specialized to exploit particular services.**

Keywords—threat intelligence, port scan

I. Introduction

Once one connects a new host to a public IP address, it only takes seconds to a few minutes before the first data arrives: probe traffic that tests the machines for open ports and applications running behind them. Scans are typically the precursor to a subsequent exploitation attempt, aiming to seize control over the host and repurpose it to send SPAM, use it as an amplifier in distributed denial-of-service (DDoS) attacks, or install malware that monitors the user and steals valuable information. In order to effectively defend computer networks, we need to begin threat mitigation already during the adversaries' probing for potential targets.

There are a number of ways this reconnaissance step may be used by the defender. A study of targeted ports across machines reveals which services the remote party aims to exploit and for specific ones also why, an analysis of the sequence of probed hosts reveals where individual subranges were singled out after open source intelligence (OSINT) or whether the adversary runs a trawl scan across large parts of the Internet. In this paper, we will take an unusual angle to port scan analysis, namely to remotely identify which scan tools the scanners are using in their campaigns. Knowledge of this tool may be used for scan mitigation by predicting the next target [1], or by returning false information to the originator.

The results presented in this paper were derived in two ways. First, characteristic patterns in probe traffic suitable to identify scan tools were found by source code inspection and were validated experimentally. Second, the tool identification patterns were then applied across probe traffic obtained from a large network telescope which revealed surprising differences in how these tools are adopted and used. This paper presents the following key findings:

- We demonstrate that it is possible in practice to remotely distinguish between software used by scan originators. This may be used to identify scan campaigns, or identify hosts collaborating in a single campaign.
- We show that tool adoption highly differs around the world. Usage of old programs such as NMap follows the global distribution of IP addresses, while ZMap is dominated by Western institutional usage and Masscan/Unicorn fuels massive scan campaigns in South East Asia.
- We find that even though programs have comparable functionality, there exist statistically significant differences in how they are used. Nearly a third of all ZMap usage is to find open SSH services, while originators relying on Masscan scan hundreds of uncommon non-privileged ports.

The remainder of this paper is structured as follows. Section II will present an overview of the previous literature on Internet port scans. Section III will describe the setup used for our measurements and the data collection process. Section IV will describe the process of port scanning as well as explain how residual artifacts in scan traffic may be used to identify scan tools. Section V will describe the empirical findings of how these tools are used in practice. Section VI will conclude our work.

II. Related Work

For many years, port scan campaigns have been observed on the Internet, and Lee et al. [2] and Yegneswaran et al. [3] provide an empirical analysis of early scan behavior and targets. In an analysis of intrusion attempts between 1994 and 2006 the authors in [3] note the cyclic nature of targets over time with the evolution and adoption of services, for example probes for open FTP servers started to appear around 2001, tests for RPC ports around 2003, and in 2004 adversaries began to systematically test the observed networks for a backdoor port opened by a malware. With the advent of for-hire botnets around 2004, cyber criminal activities began to significantly change to specialization and professionalization of services [4]. Indeed, we see that early results from 10 to 15 years ago such as [2], who report that 91% of all horizontal scans traversed the IP space sequentially, do not hold any longer.

In a longitudinal survey of scan behavior, Allman et al. [5] report a drastically increasing general volume of scans in the

Internet and note the presence of two types of behaviors: (1) "heavy hitters" which send large amounts of probe packets as fast as possible, and (2) the existence of slow scanners who emit only few packets at a slow rate. Given computational limitations in analyzing packets at the time, it was however challenging to identify and single out such slow scanners over longer periods of time.

Due to the challenging nature of detecting such scans and the availability of data, empirical analyses of scan behavior are sparse in the literature. In recent years however, Dainotti et al. [6] presented a traffic analysis from a network telescope observing a distributed and coordinated scan for SIP services. The authors could track the activity back to a botnet, and note that while the individual bots did work together, the overall nature of the campaign was relatively uncoordinated and unstructured, showing significant overlap and redundancy in scanned targets.

Conducting a scan in a distributed manner has the significant advantage for the attacker of reducing the detection surface, as each source will make only a few attempts. Javed et al. [7] found evidence of coordinated SSH attacks in their networks, where adversaries would attempt to brute-force login information from many different source IPs, increasing the difficulty to detect and prevent such low-rate distributed attacks.

While in the past years coordinated campaigns have been observed in network traces, to this date comparatively little is known yet about the mechanics and strategies of these scans. In the following, we will report techniques to identify scanner tools for incoming traffic and show empirical results on the adoption footprint and targeting of these tools.

III. Scanning for Open Ports

When monitoring an IP address, three different types of traffic is going over the link: First, there will be user data in and out of a host, either being requests leaving or responses returning. Second, backscatter traffic will flow into the host, which are misdirected responses of ongoing attacks on the Internet, in which the adversary has randomly spoofed the source IP addresses of the attack traffic to avoid attribution. This leads to the responses mistakenly being delivered to the local IP. Third, the host will receive probes that scan it for open ports and available services.

The idea behind these probes is that operating systems react differently to incoming packets whether or not a port is open. Following RFC793 [8], a TCP packet with the SYN flag sent to an open port is answered by a TCP packet with both the SYN and ACK flags set. If the packet was directed at a closed one, the packet is responded with a packet having the RST flag enabled as shown in figure 1. In case of applications using the UDP transport layer, no universally applicable behavior exists as a reaction to an incoming UDP packet. Whether or not a response is triggered depends on the application and whether the probe's content and format was within the expectation of the application. When testing for the presence of common place applications operating at their default ports (e.g., DNS

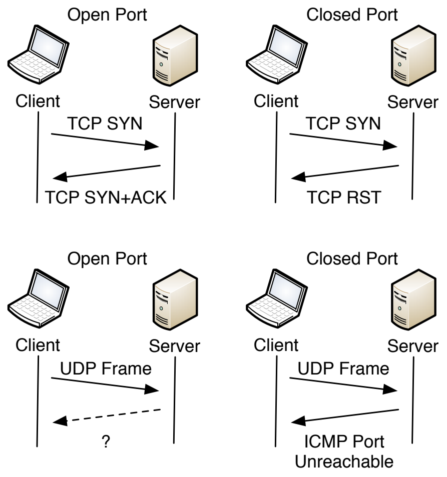


Fig. 1. TCP and UDP ports react differently to received packets, depending on whether they are open or closed.

at port 53, SIP at 5060), the originator of the scan could send packets triggering a response for these defaults. In case of a closed UDP port, the operating system should reply with an ICMP Port Unreachable reply, which might however be filtered out at the host or the network level.

In this study we analyze traffic from a two /16 network telescope, a block of 128,000 unused IP addresses. Since no user data would flow into this range, the remaining traffic must thus be backscatter or scan probes. We can distinguish between the two through an analysis of the packet headers and contents: TCP backscatter that is reflecting off from a host under attack would have the SYN+ACK flag set, incoming TCP scan probes will however only have the SYN flag set to elicit a difference between open and closed ports as described above. Separating backscatter from scan probes in UDP is somewhat more elaborate as it requires parsing and interpreting the packet's payload itself. A DNS query packet for example would thus test for the presence of an open DNS server and thus constitute a scan, while a packet containing a DNS response would not elicit a response from the receiving port and in turn imply backscatter. For our measurements we have implemented a library of packet parsers for the most important applications and protocols.

The findings presented in this work build on the analysis of an 18 month long observation of a /16 network telescope. As the block is otherwise unused, the telescope receives approximately 15 GB of scan probes and backscatter per day. As the 8 TB dataset is too large for a per packet/per source analysis in a single block, snapshots of one month each were taken out of the corpus and the analysis performed on them. The figures in this paper stem from an analysis of the month of April 2015, with general findings validated for other slices.

IV. Identifying Scanners

Over the last 2 decades, a number of tools to conduct port scans, in the following named "scanner", have been developed. Scanners are specialized programs geared to send probe packets towards a predefined set of IP addresses as fast as possible, and can either craft packets on the fly or for higher throughout combine per-packet headers with a static payload.

Conducting a scan through a large number of IP addresses efficiently is not trivial. The obvious approach of opening a TCP connection to a remote port and negotiating the TCP handshake has the major disadvantages that the 1.5 round trips make the connection establishment very slow. Furthermore, as the TCP connection would be handled by the operating system, the maximum connection pool sizes of the OS would limit the number of simultaneous tests a scanner could execute. In result, port scanners using the operating system and a full TCP handshake would be impracticably slow for large scans.

While for many years nmap was the default tool for port scanners, a new generation of tools such as ZMap [9], Masscan (github.com/robertdavidgraham/masscan) or Unicorn (unicornscan.org) have appeared recently that are aimed towards scanning of large IP ranges at high speed. The authors of ZMap state that it is possible using their tool to scan the entire IPv4 space on a 1 Gbps link in under 1 hour [9]. These speedups are created by injecting scan probes as low as possible into the networking subsystem, and avoiding to maintain internal state about which destination IPs and hosts have already been probed.

This however creates a new challenge that scanners now need to reidentify the response traffic and matching it up with a previously issued request from other data, such as traffic generated by other applications running on the same host. Currently available tools usually follow one of two strategies to address this: First, they keep state internally and record which probes have been sent where. This is for example applied by nmap, which has the distinct advantage that it is possible to look up which request packets are still unanswered and resend them to account for potential packet losses. As a scan over a large range will generate a large state table, as a second strategy scanners encode meta information identifying the scan within the packet itself. By hashing an identifier and using it as a TCP sequence number in the original SYN packet, the returned SYN+ACK packet would acknowledge the original sequence number and let the scanner identify the reply and resume the state. This provides the significant advantage that all record keeping is offloaded to the traffic itself and allows the scanner to run at a higher rate and with less resource consumption.

While each of the above four major scanners could all be used to conduct a port sweep over TCP port 80 and generate comparable results[1], there exist subtle differences in how these programs generate their request packets. We will in the following describe these differences from specifications and source code inspections and how they can be used to associate incoming scans with a particular tool.

[1]the main difference would be false negatives in case of state-less scanners that cannot send retries to account for packet losses in the network

A. *Masscan*

Masscan reduces its internal state by implementing a minimal lookup table. For outgoing packets the source and destination IPs and ports are hashed using SipHash and this 64 bit identifier later used for reidentification. The IP specification includes a 16 bit IP identification header, which is used to group together individual packets belonging to the same flow. This number is normally chosen randomly, masscan creates its arbitrary IP ID by performing a computation on the packet meta data: ipID = dstIP $\oplus$ dstPort $\oplus$ tcpSeq

Since all inputs to the computation are known to the receiver, it can check whether they match the IP ID in the IP header. Since a random match would only occur with a probability of $\frac{1}{2^{16}}$, after the local network has received k probes it can establish with very high confidence p $> 1 - (\frac{1}{2^{16}})^k$ that the probes have originated from a scanner implementing a masscan-like procedure.

B. *ZMap*

In order to verify that it has indeed send out the probe to which it now receives a reply, ZMap initializes a random encryption key at startup. The TCP sequence number of each outgoing request is set to be the ciphertext of the IPv4 source and IPv4 destination address of this probe, encrypted using AES under above key. As the TCP SYN+ACK will acknowledge the receipt of the sequence number of the previous packet, ZMap has all information to establish its authorship of a previous request.

While the TCP sequence number provides little handle, the fact that ZMap maintains no internal state means that it does not know which IP addresses it has already contacted as part of the current scan. Instead of generating targets randomly, it must follow an algorithmic approach and ZMap creates the sequence of IP addresses to test in a /0 scan using a generator in a multiplicative group modulo $2^{32} + 15$, the first prime larger than 2^{32}, which guarantees that all IP addresses will be covered once. Given the predictable nature of this generator, it is possible using a number of algorithmic improvements to brute force the seed used by ZMap during a scan within the time frame of 10 minutes on a GPU cluster [1]. This seed will then describe the sequence of IP addresses to be scanned by this host in the exact order. To identify itself to the network monitors, ZMap has also a hardcoded IP ID of 54321 in the source code.

If the incoming probes follow this exploration order, the TCP sequence numbers over multiple packets are distributed randomly (which most operating systems don't do but AES-encrypted ciphertext does), and have a fixed IP ID (54321 or another one after a source code modification) we can with very high likelihood conclude the adversary to be using ZMap or a software implementing all of these components.

C. *Unicorn*

Similar to ZMap, the Unicorn scanner also encodes its re-identification information within the outgoing packets. At startup, the software creates a random 32 bit session key, which

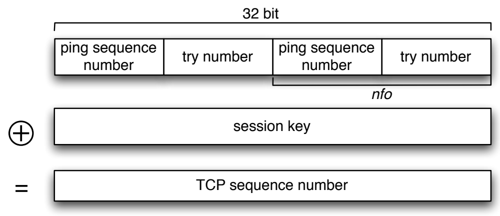


Fig. 2. Generation of nmap's TCP sequence number

is run with a bit-wise exclusive OR against the other meta data of the probe:

$$\text{tcpSeq} = \text{sessionKey} \oplus \text{srcIP} \oplus ((\text{srcPort} << 16) + \text{dstPort})$$

While the sessionKey is not known, this is essentially a stream cipher with a fixed initialization vector. As the "plaintext" as well as the "cipher text" is known, XORing the meta data from the IP and TCP header against the TCP sequence number yields the random session key for the duration of the scan. If the TCP sequence numbers of subsequent packets match this established session key, we can with a high likelihood conclude that the probes are generated by a unicorn-like software.

D. Nmap

Nmap also relies on a random session key to generate its TCP sequence number. nmap is the only one of the discussed tools that maintains a state of sent probes for retransmits, and the sequence number of pings as well as the number of retries forms the basis of the TCP sequence number. As shown in figure 2, the two fields are concatenated twice to arrive at the 32 bit length of the sequence number and obfuscated by a bit-wise XOR against the session key.

As the session key remains fixed, the original concatenated data may be easily reconstructed using subsequent packets, as $\text{tcpSeq}_1 \oplus \text{tcpSeq}_2 = ((\text{nfo}_1||\text{nfo}_1) \oplus \text{sessionKey}) \oplus ((\text{nfo}_2||\text{nfo}_2) \oplus \text{sessionKey}) = (\text{nfo}_1||\text{nfo}_1) \oplus (\text{nfo}_2||\text{nfo}_2)$. If the ping sequence number and try number would be unknown to the receiver, we could only establish these packets as statistical outliers, as the likelihood of encountering a TCP sequence number of two identical 16-bit segments is only $\frac{1}{2^{16}}$, but for an nmap-like scanner this occurs for every probe.

If nmap is executed without specifying the source port to be used, the software selects a random base port between 33000 and (65536 - 256), adds the ping sequence number (if this ping is equal to 0, the try number is added) and sends an probe from the resulting port number. We can thus confirm whether this value is in line with the TCP sequence number, and further increase our confidence that the remote part is using an nmap-like tool.

V. Scanner Usage

After a verification of scanner characteristics from source code inspection in field trials, we implemented modules to match up scan traffic from our network telescope with specific tool usage. Although all the above scan tools may be used to test any TCP/UDP port combination and pose minimal restrictions on their usage, we find that each tool is used in different regions, communities, with differently sized setups and with different goals in mind. Due to space constraints, we will only highlight two of these aspects in this paper, differences in regional adoption and the services targeted by their users.

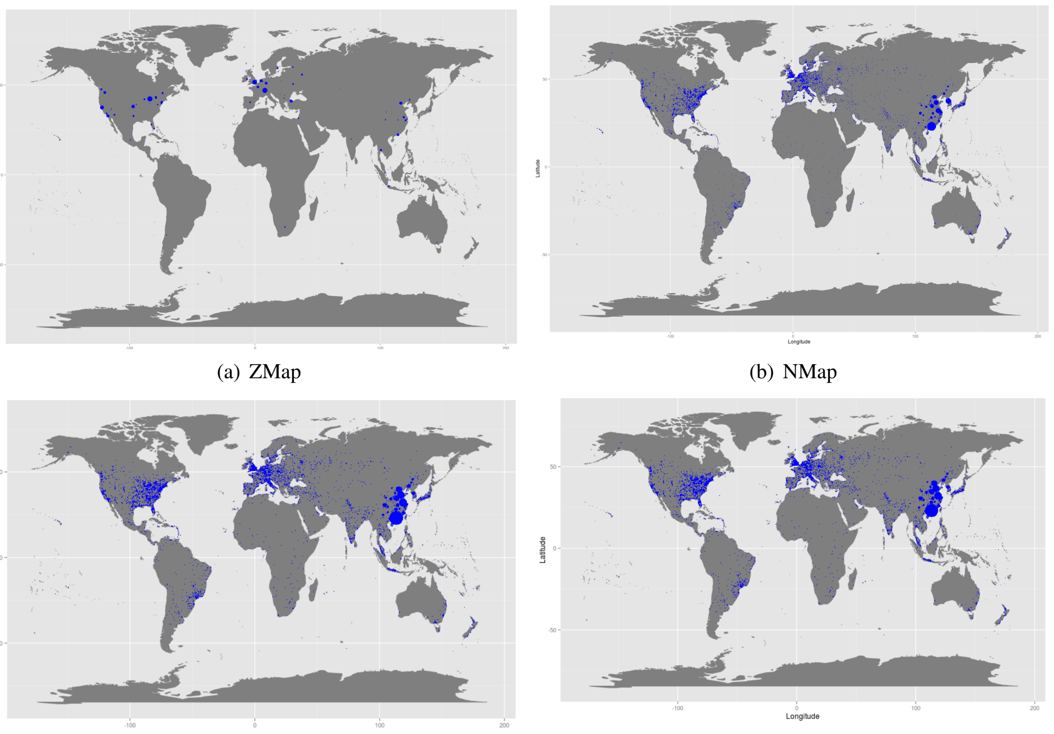
A. Regional and Community Preferences

Curiously, we find that scanners are not used equally across the globe, but that geographic regions, organizations and networks show strong preferences of one tool over the other. Figure 3 shows a plot of the scanning hosts, with the IP address geolocalized through the Maxmind geoIP database, separated by usage of ZMap, Masscan, Unicorn and NMap. The size of the point represents the amount of probes originating from this particular region. If we compare these distributions against the geographical allocation of the IPv4 address space, we find that only NMap – the oldest scan tool and the one that is typically described in tutorials and instructional material on port scanning – actually resembles this global distribution, while all other tools heavily deviate from it. We find that ZMap is used by a limited user base but the majority of adopters each send out large volumes of requests. An analysis of the involved IP addresses using reverse DNS, Whois lookups and visiting the host showed a significant share belonging to universities and research organizations. The user base of Masscan on the other hand is much more dispersed and heterogeneous. We see two regimes in masscan-originated scan campaigns: extremely small runs generating only a few packets to highly volumetric ones, in the worst case a single IP sending out a total of 34.5 million packets over the course of one month, orders of magnitudes larger than heavy hitters using other tools which are also mostly located in South-East Asia. Also unicorn features this heavy bias towards South East Asian origins, although there exists a base population of low usage resembling the global IP allocation like in the case of NMap. Except for the case of ZMap, which comes from an academic lab publicized by a research paper and thus might explain the strong organizational/academic user base, there are no clear reasons to explain the large differences in geographical adoption. All four tools are open source and directly installable through packet managers in major Linux distributions.

B. Targeted Services

When matching the identified tool against the ports a scan campaign targets, even more biases emerge. Table I lists a selection of 10 commonly used applications on servers together with their default TCP port configuration. In our data we see that IP addresses exhibit a strong preference for a particular tool to scan the Internet. If we analyze the total amount of probes sent by a certain tool with regard to the targeted ports, we see that specific users show a strong specialization.

What is unexpected though is that comparatively strong port preference exists for the majority of users employing a certain tool. More than a third of all NMap probes target telnet



(a) ZMap (b) NMap (c) Masscan (d) Unicorn

Fig. 3. World-wide distribution of the main four scan tools.

TABLE I. PERCENTAGE OF TARGETED PORTS BY SCAN TOOL

Port (Typical Application)	NMap	ZMap	Masscan	Unicorn
7 (Echo)	0.0%	0.0%	0.09%	0.0%
22 (SSH)	2.68%	32.97%	5.03%	81.74%
23 (Telnet)	34.44%	2.37%	1.41%	0.68%
53 (DNS)	0.0%	2.09%	0.18%	0.0%
80 (HTTP)	4.52%	8.15%	6.44%	2.13%
110 (POP)	0.0%	1.38%	0.39%	0.0%
123 (NTP)	0.0%	0.04%	0.09%	0.0%
443 (HTTPS)	0.16%	11.14%	2.94%	0.05%
3306 (MySQL)	0.0%	1.14%	0.39%	0.01%
8080 (Alt HTTP)	7.21%	4.83%	3.17%	1.63%

on port 23, while a third of ZMap probes are directed at SSH on port 22. We further notice the existence of ZMap branches in the wild, each with a different hardcoded IP ID and a varying set of preferred ports. Even more biased is Unicorn, whose users direct 4 out of 5 packets to port 22. Masscan on the other hand is also an outlier in terms of targeting. Masscan users send out a massive amount of packets, directed at a massive amount of ports – even the top frequented port 80 (HTTP) accumulated less than 6.5% of all masscan-attributed probes. The reason for this behavior is still subject to research, one possible hypothesis is working relationships and knowledge exchange between actors specializing in the same goal, who then also adopt tooling used by their peers.

VI. Conclusion

In this paper, we demonstrated that is is possible to identify major scanner software based on residual artifacts in the generation of network and transport layer header fields, as well as the exploration order of hosts. Based on an analysis of probe traffic directed against a two /16 network telescope, we find that the major tools have a unique geographic footprint, differing from the global IP allocation and in case of ZMap biased towards institutional adoption. We also see that IP addresses show preference towards a particular tool and surprisingly also a general bias exists in terms of the ports tool users target. The origin for these geographic and usage biases is subject of ongoing research.

References

[1] C. Doerr, M. el Maouchi, S. Kamoen, and J. Moree, “Scan prediction and reconnaissance mitigation through commodity graphics cards,” in *IEEE Communication and Network Security (CNS)*, 2016.

[2] C. B. Lee, C. Roedel, and E. Silenok, “Detection and characterization of port scan attacks,” tech. rep., University of California San Diego, 2003.

[3] V. Yegneswaran, P. Barford, and V. Paxson, “Using honeynets for internet situational awareness,” in *HotNets*, 2005.

[4] R. Anderson, *Security Engineering*. Wiley, 2008.

[5] M. Allman, V. Paxson, and J. Terrell, “A brief history of scanning,” in *ACM SIGCOMM Conference on Internet Measurement*, 2007.

[6] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescapé, “Analysis of a “/0” stealth scan from a botnet,” in *Internet Measurement Conference (IMC)*, 2012.

[7] M. Javed and V. Paxson, “Detecting stealthy, distributed ssh brute-forcing,” in *ACM SIGSAC Conference on Computer & Communications Security*, 2013.

[8] Information Sciences Institute, “Transmission control protocol (rfc793),” tech. rep., IETF, 1981.

[9] Z. Durumeric, M. Bailey, and J. A. Halderman, “An internet-wide view of internet-wide scanning,” in *23rd USENIX Security Symposium*, 2014.