Discovering Collaboration: Unveiling Slow, Distributed Scanners based on Common Header Field Patterns

Harm Griffioen
Cyber Threat Intelligence Lab
Delft University of Technology
Delft, the Netherlands
h.j.griffioen@tudelft.nl

Christian Doerr
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
christian.doerr@hpi.uni-potsdam.de

***Abstract*— To compromise a computer, it is first necessary to discover which hosts are active and which services they run. This reconnaissance is typically accomplished through port scanning. Defense systems monitor for these unsolicited packets and raise an alarm if a predefined threshold is exceeded. To remain undetected, adversaries can either slow down the scan, and/or distribute it over multiple hosts. With each source below the threshold, the combination of all may still complete the scan efficiently. It is especially this group that is of concern: with enough resources and knowledge to execute such a coordinated activity, they will pose a more potent threat than the noisy "script kiddie". Correlating which out of 4 billion IPs potentially collaborate is however a challenging task, hence today's systems do not consider coordination beyond basic subnet aggregation.**

In this paper, we propose a method to identify and fingerprint distributed scanners based on commonalities in header fields, which are an artifact of the way fast port scanning software is built. We demonstrate that this method can effectively locate groups, and based on the monitoring logs we report on a number of new groups and tools, which have previously not been reported in the academic literature.

Fingerprints generated can ultimately be used as Indicators of Compromise to detect and mitigate scanning behavior in order to deny adversaries the possibility to learn about weaknesses of a system.

I. Introduction

Port scanning is an important technique that serves many purposes. While there are benign uses for this type of scanning, such as penetration testing, in which the security of a system or organization is tested, there are also malicious uses. Adversaries are employing it to scan the Internet looking for active systems and which services these systems run, information that could be used for a subsequent exploitation of the host.

Intrusion detection systems (IDS) or firewalls typically monitor for these unsolicited incoming packets. If the number of packets sent by a particular source IP exceeds a predefined threshold, an alarm is raised. As it is in most application scenarios impractical to set this threshold to zero due to the prevalence of IP address spoofing and thus unsolicited backscatter, there exists a small margin that adversaries could exploit for their activities. First, an attacker could limit the scanning rate to remain beneath some assumed threshold, which would mean that the scan takes a large timespan to complete. Second, as IDSes or firewalls would normally do their bookkeeping at the level of IP addresses, an advanced adversary could split up the scan and run it simultaneously from a large number of origins, each testing only a portion of the targeted addresses. Given enough source hosts in combination with a slow enough scanning speed, the adverary could systematically collect the required data without triggering an alarm. Evidently, it is these adversaries the defender should be most worried about: with enough patience and equipped with sufficient knowledge and resources to deploy and maintain such a coordinated scanning activity, they will most likely pose a more significant threat that the prototypical "script kiddie" who is sending a lot of probing traffic from a single origin at a fast rate and is subsequently detected by an IDS or firewall.

Detecting such distributed, coordinated scanning activity is a computationally challenging task for two reasons: first, the indicators for the activity are of such low volume that the scan disappears in the background noise of Internet backscatter [1]. Second, as an unknown number out of the 4 billion total IPv4 addresses could be part of such a coordinated action, the target and procedure of the scan is unknown to the defender, and the targeted systems may or may not contain some overlap, too many candidate solutions exists to be meaningfully explored. As a result, today's defenses do not consider distributed port scanning beyond basic subnet distribution, where origins are located in the same class C (or /24) network.

In this paper, we pursue an alternative approach to the problem of detecting distributed port scans. Instead of identifying hosts out of the gigantic pool of all IP addresses and incoming packets that complement each other to a plausible degree, we leverage the fact that the adversary will need to exploit the economies of scale and likely deploy the same or a similar tool across all hosts participating in the scan. As high performance scanners no longer maintain a local state of the scan but recognize returning probes by embedding identifying information into the packet itself (as we will discuss in more detail in section III), we can leverage the fact that packets sent by identical software and collaborating hosts will have data embedded in exactly the same places, encoded using the same algorithm, even though we do not know which algorithm and

data it actually is. We can test for these commonalities using basic boolean logic in quadratic runtime, which makes the identification and correlation of collaborating hosts feasible.

This paper makes two main contributions:

- We propose a new technique to find identical port scan tools and collaborating hosts based on the embedding of information in scan probes. We show that the method is very resilient to noise and able to find coordinated systems well below the Internet backscatter noise floor.
- We demonstrate the feasibility of the approach based on the port scanning activity directed against a large organizational network. Besides the identification of common port scan software, we also discover several new tools previously unknown in the literature.

The remainder of this paper is structured as follows: Section II summarizes previous work in the discovery of coordinated scanners, section III provides a brief introduction into the concept of port scanning and the way high performance port scanners are built. Based on these universal designs, section IV introduces the available angles by which distributed, stateless port scanners can be detected. Section V describes the dataset and evaluation methodology, while section VI evaluates the performance of the presented approach. Section VII presents the results from an analysis of reconnaissance traffic against a large organization, and reveals new and previously unknown port scan tools. Section VIII summarizes the results.

II. Related work

While several previous studies have reported the existence of sophisticated, distributed adversaries [2], [3], only a small body of literature has been developed towards the identification of collaborating hosts. As discussed in the introduction, the identification and extraction of k coordinated systems from a set of n elements is basically equivalent to the computation of the binomial coefficient in combinatorics (with the addition that all packets from two source IP addresses need to be checked for overlap), which does not scale advanteously. Previous literature thus has either focused on small scale evaluations with prefiltered data, or relied on several assumptions on the organization and behavior of distributed scanners to make the problem scalable.

One of the approaches to detect distributed scanners is proposed by Gates [4] and uses set cover to find relations between IP addresses. The set cover problem is a NP-complete problem [5], therefore using this algorithm is infeasible for high volumes of data. Also, the limitations of the method proposed by Gates include that only groups that hit 95% of a particular target IP range can be detected, which can be circumvented by adversaries. Robertson et al. [6] detect distributed scans as long as the IP addresses participating in these scans are located in the same subnet. It makes the assumption that the IP addresses are related when they are close together. This assumption might hold in a number of cases like scanning services such as Shodan which scan from specific ranges, but it is not always valid. Previous work shows that there are large distributed groups whose scans are not originating from the same subnet [2], [3]. In the work by Yegneswaran, Barford and Ullrich [7], the authors identify coordinated behavior by looking at destination ports and destination IP addresses, which they use to identify slow scanning malware. They find that a large amount of scans is coordinated in nature, meaning that IP addresses elicit the same behavior. The authors evaluate destination ports and IP addresses present in the packet headers, discarding other fields.

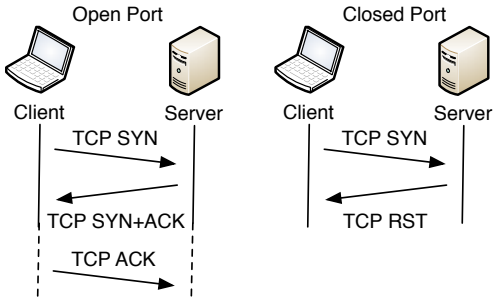
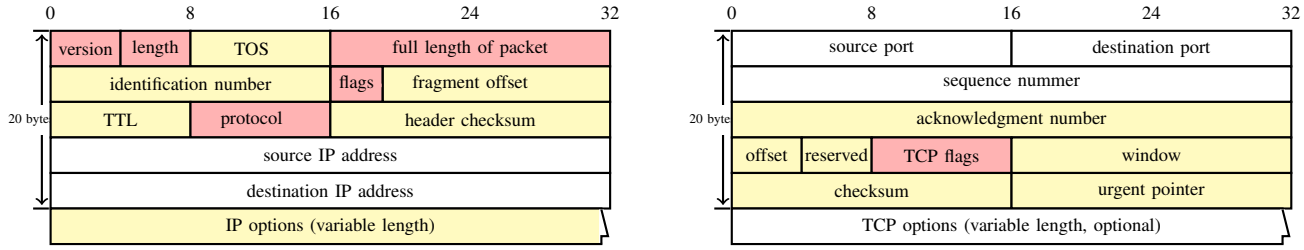


Fig. 1: The response sent to a TCP SYN frame reveals to a client whether a server port is open or closed.

As we will demonstrate in section VI, the above methods fall short to reliably detect collaborative port scans in two respects. First, when operating at Internet scale and inside a large organization, the volume of incoming data cannot be analyzed for the degree of complimentarity, nor is there usually a prefiltering available as required for Gates' set cover solution [4]. Second, the results by [2] highlight that adversaries typically do not respect the boundaries of subnets but rather distribute their activities across network ranges, ISPs and country boundaries. This reduces the applicability of heuristics, which trade the computational complexity with the assumption of logical proximity as for example [6]. As the fundamental algorithmic problem cannot be tackled, we will demonstrate in the following how the problem may be redefined, and this alternative problem does allow a computationally feasible angle for detecting collaborating hosts.

III. Port scanning

In order to determine whether a remote host runs an application behind a particular port, adversaries typically rely on port scanning as the first step in the reconnaissance process. Transport layer protocols such as TCP and UDP react differently, depending whether a system port is opened or not by an application. As shown in figure 1, the specification of the TCP protocol [9] mandates that a connection request from a client, a TCP SYN, should be answered with a TCP packet in which the SYN+ACK flags are set. A client would then close the initial handshake of the TCP protocol through a final TCP ACK, after which both client and server have established and synchronized the state of a duplex connection between them. In case the initial TCP SYN reaches a closed port, RFC793 [9] mandates the server to respond with a TCP RST packet, which instructs the client to drop the connection request as well as remove and reset its internal state. As a port scanner



(a) IPv4 header as defined in RFC 791 [8].

(b) TCP header as defined in RFC 793 [9].

Fig. 2: IPv4 and TCP headers as defined in the RFCs. Fields marked in yellow show fields that can be modified, but are not present in the response of the packet.

is not interested in actually negotiating a connection with a server but only wants to ascertain whether there is a service present and active behind a given port, it would only sent an initial TCP SYN and not complete the handshake further.

When performing a survey of active hosts and open ports, a port scanning application would successively send out TCP SYN packets to all targeted destination addresses at one or more ports, and log from which remote hosts it has received a response. During this scan, the application internally has to maintain some record about the scan packets in flight for two reasons: first, whenever a host is connected to the open Internet, almost immediately backscatter will trickle in, which is the result of continuously running DDoS attacks and IP address spoofing. If the scanner does not know to which destinations it had earlier sent TCP SYN requests, the application would misinterpret incoming backscattered TCP SYN+ACKs as responses to its scan. Second, if either the TCP SYN or the response has been lost, port scanning tools such as the commonly used nmap will resend the probe two more times before determining a destination to be unresponsive.

Clearly, maintenance of a log of open, unanswered TCP SYNs requires a significant bookkeeping effort, especially if the port scanner operates at very high speed. While long established tools such as nmap only progress slowly in such a scan, recent modern developments such as masscan or zmap are able to scan the entire IPv4 address space in less than one hour [10], [11]. This is made possible at the expense of foregoing re-transmits for timed-out responses, but these high performance scanners also no longer record where they have sent packets to and thereby avoid any overhead with storing, updating and removing records during the scan.

To separate responses from backscatter, these tools embed information into the outgoing probe that is preserved on the way back. As an example, the server's acknowledgement of the client's initial TCP SYN packet contains the 32 bit initial sequence number (ISN) used in the client's request (incremented by one). If the port scanner would choose a 32 bit random number at the beginning of the scan and use this fixed value for each scan probe, the responses could be reliably separated from incoming traffic as backscatter would only have a chance of 1 in 4 billion to use the expected random value as ISN, while the scanner has reduced the entire management storage requirement to just 4 bytes. Ghiette, Blenn and Doerr [12] report on the usage of such random session keys, but also find combinations of connection meta data as source/destination IP addresses or port numbers be used for ISN generation in common port scanning tools.

IV. Methodology

In the previous section, we have discussed how modern port scanning tools embed information in outgoing probes to enable a low cost re-identification of responses in incoming network traffic. While the tools analyzed by [12] all use one type of such ISN generation (which then allows these tools to be fingerprinted), there might be other tools unknown and unavailable to the general public, as they are custom developed by (advanced) adversaries to avoid such signature-based detection. Indeed, [13] finds in the case of SSH bruteforcing the existence of dozens of customized brute forcing software, many of them exclusively in use by a limited number of related hosts spread around the world. Aside from identifying the presence of slow, distributed scanners, we would also like to identify the existence of such customized tools, as they also indicate sophisticated adversarial knowledge and the availability of resources.

A. Detecting shared header patterns

As we have seen above, if commonly available and custom-built port scanners embed some state information into the scan probe, this could only be meaningfully done in header information that is (a) not modified by intermediate routers, (b) either entirely preserved in the return packet or modified in a predictable way such as the ISN, and (c) the modification does not break the functioning of the protocol. When we look at the header fields of the IPv4 and TCP protocols as depicted in figure 2, we directly see that only few of the header fields could be arbitrarily changed. Modifications to fields indicated in red would break the correct working of IPv4 or TCP, for example a modification of the IPv4 version field would mean the packet is no longer correctly recognized and parsed by the destination. Other fields such as the TOS, TTL or IP ID only carry meaning in one direction and are not copied over into the return, while values such as the IPv4 header checksum are modified along the way. These are indicated in yellow.

From the IPv4 header, only the data placed into the source and destination IP address fields are available in the return, both of which could be used as input to an identification scheme. Here only the source IP address could be used to a limited extent to embed some information if the adversary has control over an address range or may monitor network traffic. In the TCP header, only data placed in the source and destination port as well as sequence number is returned. While like in case of the destination IP address a modification of the destination port is meaningless, as the adversary wants to scan this particular destination, these fields could be used as input information to an identification scheme, whose input could only be placed into either source port or sequence number.

The possible fingerprint FP could thus be an arbitrary combination of source and destination address and port as well as a possible secret value, which is placed in either source address, source port or TCP sequence number. As the header fields for IP address and TCP sequence number are 32 bits while port numbers are only 16 bits in length, information could furthermore be arbitrarily shifted by an unknown offset o_i, for which we arrive at the general formula

$$FP = random \oplus (srcIP \ll o_1) \oplus (dstIP \ll o_2) \\ \oplus (srcPort \ll o_3) \oplus (dstPort \ll o_4).$$

Note that this formula is general enough to also capture cases where information is not used as part of the fingerprint. Consider the case of embedding a 32-bit FP into the ISN, with the only information being included are the dstIP and the dstPort as the most significant bits. This can be realized through *random* = 0x00, $o_1 > 32$ and $o_3 > 32$ which will eliminate the other values from the computation. To discover and identify this tool, we need to learn the values of *random*, o_1 through o_4 from incoming scan data.

In order to solve this equation and find those values for which the fingerprint FP matches a header embedded in the packet, 32 comparisons per data field are necessary, or a total of 131,072 comparisons per packet. While this is computationally infeasible to do per incoming packet, at least in software, we can test for the presence of such an algorithmic generation by comparing pairs of packets directed to different destination IP addresses or ports.

Let us suppose from now on that the fingerprint has been embedded as the initial sequence number in the TCP SYN packet, and we have obtained two sequence numbers seq_A and seq_B from the packets sent from the same source towards host A and B, targeting the same port. When we combine both through an XOR relationship,

$$\begin{aligned} seq_A \oplus seq_B = &((random \oplus (srcIP \ll o_1) \oplus (A \ll o_2) \\ &\oplus (srcPort_A \ll o_3) \oplus (dstPort \ll o_4)) \\ &\oplus ((random \oplus (srcIP \ll o_1) \oplus (B \ll o_2) \\ &\oplus (srcPort_A \ll o_3) \oplus (dstPort \ll o_4)) \\ = &(A \ll o_2) \oplus (B \ll o_2) \oplus (srcPort_A \ll o_3) \\ &\oplus (srcPort_B \ll o_3) \end{aligned}$$

all identical variables are eliminated, as $(A \oplus r) \oplus (B \oplus r) = (A \oplus B)$. This will conveniently remove unguessable items such as a random value or session key as well as remove any identical fields between probes such source IP or destination ports, thereby leaving only the remaining items to be tested. If the port scan is a horizontal scan, 35,937 comparisons are left over, as the destination port will fall out in the elimination. By picking packets hitting the same destination IP or originating from the same source, in which case these values fall out in the XOR operation, the number of equations drops to only 1089. This can be further simplified, by mathematically checking when there is only one field possibly left in the equation. If we were to mix in a value such as the source port with a particular offset, numerically this will mean that the value is multiplied by a power of 2 before added to the number. Thus, if we divide the result of a two-packet XOR by the actual value a field that we suspect was included in the equation, for example the source port,

$$\hat{o_4} = log_2(\frac{seq_A \oplus seq_B}{srcPort_A \oplus srcPort_B})$$

the resulting guess of $\hat{o_4}$ should also result in the same integer. This reduces the effort per packet pair further, but can only be performed after eliminating the other fields, reducing the search space of the equation. 33 steps are needed to check one field, and for every step the last field is also mathematically checked. Checking all fields will therefore take 66 equations when strategically picking which packets to compare.

B. Iterative filtering

Not all scanners provide these clear patterns, and sophisticated scanners that do not have these relationships are hard to cluster together in a large pool of data. The data points that are generated by slow distributed scanners are hidden significantly beneath the noise floor of other scanning activity. In order to find and detect stealthy actors, we adopt the iterative filtering approach described in [14], in which events are filtered out to show events that were not known at first.

If an adversary was to perform a distributed scan from multiple machines, it would be inefficient to create a new program for every device, as opposed to distribute the same tool over all machines. The behavior on all these devices would thus be the same, all following the rules pre-programmed in the tool. By looking at relations in packets originating from a source, and building a behavioral model from these packets, one would be able to group these IP addresses together on the tools that are used.

We define packet relations by commonly used values in the header, as well as behavioral traits such as re-transmissions and randomization frequency. The header values that can be set by the adversary without breaking the TCP protocol are the fields marked white and yellow in figure 2, which are therefore the fields we look at. Randomization and re-transmissions of these header values has been taken into account through the addition of a correlation between the number of distinct header values with regards to the total number of packets. With these fields, a profile is built per source IP.

From the profiles, pair-wise correlation scores are calculated, which are used to identify whether hosts are using the same tooling to scan the internet. After a particular set of hosts that collaborate and use a specific tool has been identified, this group is removed from the data body and the analysis is repeated. With "strong" signals and heavy senders removed, relationships between collaborating but less active hosts become then visible in the filtered data which would otherwise be overshadowed by the strong correlations. In essence, this amplifies the weak signals hidden in the noise floor. Additionally, by leveraging the fact that specific tools are already identified, we can safely remove these from the dataset without losing information about other relationships. As they are removed, new groups become visible that previously correlated heavily with other groups that are now removed. We consecutively loop until no new scanning groups can be found in the data or the identified correlations cross a threshold indicating no distinguishability from randomness.

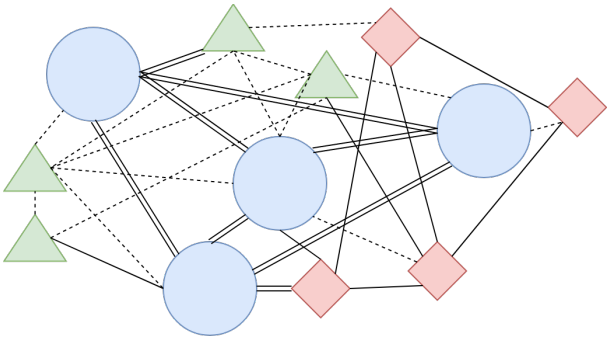
To perform this clustering we use SLPA, a graph based clustering method [15]. The graph is built by creating a node for every scanning IP address, and edges are added only if the pair-wise correlation scores between two nodes exceeds the threshold of the current step of the clustering algorithm. Figure 3 shows this process, where highly connected subgraphs are extracted as clusters from the dataset in iterative steps.

In addition to the above described detection methodologies, we identify time patterns. Clear time patterns are used to indicate collaboration, where hosts come up and go down in the same time period. To identify these patterns, we have split the dataset in sliding windows of 20 minutes, for which we will consider scanners to have started or stopped in the same time period.

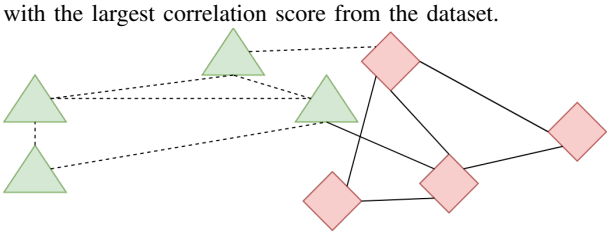
V. Data collection

In order to evaluate the detection capability of the presented approach and discover tooling used in the wild by adversaries, we utilize data from a network telescope. This telescope contains three partially populated /16 networks and collects incoming packets for approximately 65,000 unused IP addresses [1]. As these IP addresses are unused, the trace is thus devoid of any user data (which conveniently eliminates potential privacy concerns), but only contains Internet backscatter and adversarial activity directed against the ranges such as port scans. As network telescopes contain a large number of IP addresses but do not actively respond to incoming probes, they are excellently suited to detect horizontal scanners, which [16] reports is the predominant scanning mode in today's Internet.

While the telescope has been in operation for more than three years, this study is based on data collected over a timespan of two months in 2018. This timeframe accounts for approximately 864 GB of traffic and a total of over 6.5 billion packets. Between the two major transport layer protocols UDP and TCP, the vast majority of port scanning traffic is directed at TCP, which will hence be the focus of this paper. We classify packets as probing activity based on the presence of the TCP SYN [16], and do not include backscatter in the analysis.



(a) Pair-wise correlated graph, lines show the amount of correlation. As many points correlate at least a little, we iteratively filter clusters with the largest correlation score from the dataset.



(b) The same graph after removing the first high-correlating cluster, showing different clusters previously overshadowed by the correlations with the filtered group.

Fig. 3: Visual representation of proposed clustering method.

VI. Evaluation

In this section, we evaluate the performance of our proposed methodology to detect distributed scanners that deliberately act to avoid detection. As there is no prior available labeled data set, our evaluation strategy will hence be three-fold:

- First, we will establish several dimensions in which a scanner can hide and create concrete scenarios for the different hiding dimensions.
- Second, we create artificial groups of scanners based on the aforementioned scenarios. We implant these scanners within the data trace collected by the network telescope, and given this ground truth test what percentage of the embedded scanners our method is still able to identify given various degree of hiding.
- Third, we run our proposed method on the data corpus to detect scanner tools currently in adversarial use. These findings on which toolchains and degrees of coordination are used in the wild are presented in section VII.

A. Degrees of stealthiness

Goal of an adversary in port scanning is to obtain a record of the available hosts and services on the Internet in general, or a specific remote network in particular. From our earlier discussion and the review of previous work, we know that an adversary will rate-limit a scanning campaign and spread out the campaign over a number of IP addresses in an attempt to evade detection, as current defenses such as IDS/IPSes aggregate packets by IP addresses in particular time windows. While one host could be assigned multiple IPs, multiple source IPs would usually imply multiple source hosts. Multiple IPs

would in the simplest case be located in the same subnet, or be geographically distributed and hence located at the IP space of the single or multiple Internet service providers. In a coordinated distributed activity, we would assume resource reuse of the available tooling for the economies of scale as discussed before (which would then have patterns we aim to discover in section IV), as well as similar behavior of all participating hosts for example somewhat synchronized start and end times of the scan.

While all of the above would be features that a distributed, coordinated system would exhibit, a knowledgeable advanced adversary could deliberately set up the environment to avoid showing these features. Therefore, we will create distributed scanners in our evaluation dataset that use one or more of the following five different dimensions to 'hide' their presence:

1) *Source IP addresses:* The scan can be distributed over a large number of source IP addresses. In our tests, we added scanning groups consisting of 1 to 1,000,000 origins. IPs can be either located in the same subnet, in multiple subnets, or randomly placed in the IPv4 space.
2) *Packet volume:* The number of probes can also be leveraged by a scanner to remain undetected and generate only a weak signal. We embed groups sending 10,000 down to 5 packets to our 60,000 IP addresses.
3) *Header field patterns:* Aside from hard coding select header values such as ZMap's IPID 54321 [17] and encoding of state information in header fields as described in [12], tools could also randomize all fields present in the packet. This way, the scanning traffic will ultimately be indistinguishable from normal network traffic. Randomizing each field would mean that the scanner has to keep track of every packet it has sent out, in order to match them back to the scan.
4) *Sequence number:* As scanners can embed information in the sequence number, the value of the sequence number can be leveraged to find a scanning group. Sequence numbers could be static for every packet, generated through an equation, or totally random.
5) *Subnet:* One of the known detection methods is to link IP addresses in the same subnet [6]. By distributing the scanning activity over a number of subnets, an adversary could evade detection. Thus, we embed scanner groups that are located in multiple subnets in the data, with the highest level of hiding being geographically distributed.
6) *Time-based patterns:* Distributed scanners may have a coordinated on-off time, in other words are active in the same time frame during which they generate probes at random intervals. With additional effort, an adversary could let the source host be active at different, uncorrelated times, although this would waste some of the available scanning capacity of the infrastructure as source hosts remain idle.
7) *Destination IP cover:* A scanner can skip part of the destination IP space in order to not be detected by methods proposed in [4]. This means that the scanner loses some information. In order to counter this information loss, an attacker could also opt to deliberately create an overlap in scanned destinations. Then the algorithm will not flag the set of IP addresses as being part of one group.

		Low	Medium	High
1.	#Source IP addresses	1-3	1k-5k	100k-1M
2.	Packets per source IP	10k-5k	1k-500	100-5
3.	Unique header fields	Static	Few random	All random
4.	Sequence number	Static	XOR	Random
5.	Same subnet	Yes	Multiple	No
6.	Time based pattern	Yes	Partial	No
7.	Destination IP cover	Perfect	Overlap	Not full

Fig. 4: Hiding evaluation, for comparison with the state of the art, we define different hiding levels from low to high.

B. Method validation

From the overview of different hiding techniques discussed above, we have created an overview of different scanning techniques that could be used by attackers with different sophistication levels. As some configuration combinations would not be meaningful, for example it is impossible to have a single-source scanner that is highly distributed, the options along the 7 dimensions as shown in figure 4 resulted in 3060 theoretically feasible configurations for scanner behavior.

For each viable combination, port scanning traffic is injected randomly throughout the telescope data. This is necessary as there is no data set available of labeled distributed slow scanners, especially none where a wide array of hiding scenarios is included. By exhaustively generating scanning groups, we can estimate how scanning groups would have to behave in order to remain undetected, and how well the current and proposed methods work.

In total, our method is able to detect 2312 groups out of the 3060 groups that were added to the data. These 2313 successful detections are the instances where all of the source IP addresses were exactly identified by our algorithm and matched in one cluster. We can however argue whether an exact match is the best choice for both precision and recall, or whether we would be rather interested to evaluate whether we have better recall with some loss in precision. In other words, we could choose to accept some false positives in a cluster if this would significantly increase the likelihood that we identified all malicious scanning IPs in the data set. From a defense perspective, correctness of the total cluster might be less important than finding anomalies in the first place. If we go beyond evaluating only the groups that had perfect precision and allow for groups in which at least 95% has been identified in a cluster, we detect 2692 groups out of the 3060 combinations. This means with up to 5% of noise, we detect 87.97% of the added groups. While 5% noise produces a 16.4% increase in the detected groups, beyond the 5% threshold the detection rate increases slower than the added noise. Therefore, we will take a 5% noise limit as the second

TABLE I: Comparison of the proposed method and the state of the art.

Method	Found	Recall	Precision
Our method (exact)	2312	75.55%	100%
Our method (5% noise)	2692	87.97%	98.14%
Subnet method	983	32.12%	100%
Destination IP	107	3.50%	100%

variant of our method, in which analysts can trade a small amount of false positives for increased detection.

The groups that remained undetected after applying our method are the groups that are hidden in almost every dimension, as well as the groups that only have 1-5 source IP addresses and only hit a very small fraction of the telescope, as the combined signal is so weak that the method is not able to find a connection. As we can fingerprint groups based on a subset of the total set of identifiers, the method allows for the detection of groups that are attempting to hide in one dimension. The groups randomizing almost all fields, did not use a XOR relation, and were distributed over a large number of different hosts sending a low amount of packets are not found using the proposed method. While the detection method does not directly find these highly hidden groups, parts of them are still identified, as which we iteratively filter the data, so it is able to find some of these hidden groups as they are distinct in their hiding strategy.

As discussed in section II, there are two methods that can be considered the state of the art in distributed scanner detection. These are the method of grouping IP addresses by subnet [6], and the detection of scanners based on the destination IP addresses scanned [4]. The benchmark of our work is shown in table I, which shows our method outperforms the state of the art, with the destination IP method finding the least number of groups. The reason for this is that the amount of data for this comparison is sufficiently large to make the computation of the set cover infeasible. As an approximation algorithm is needed for the set cover problem, and there are many different permutations of scanners that would create such a cover, this method suffers from a large number of false positives, and is not able to find many distinct groups. The subnet method does find some groups, as some of the groups added to the data were indeed located in the same subnet. However, the method fails to detect groups in other hiding dimensions, as the subnet method relies on subnet-based thresholds, which are circumvented by scanners that are distributed.

As classifying the performance as a lump average over 3060 different combinations does not sufficiently reveal these nuances and a visualization of results across 7 dimensions is challenging, table II aggregates the results into categories of adversarial sophistication. To simplify the presentation, we can categorize each of the values from figure 4 in a level of sophistication. For example, a scanner group where all of the source IPs are consecutively placed in the same subnet would have low sophistication, while one where the IP addresses are randomly placed worldwide is highly advanced, and so on.

When we only consider scenarios where all variables are unsophisticated, both the proposed and the subnet method accomplish a perfect result and also a set cover accomplishes 59% precision. If all configuration options have medium sophistication, our method continues to identify all embedded groups while previous work drops to 35% and 0% precision. This drop can be entirely explained based on the algorithms used: the subnet-based method makes for example the strong assumption that scanners originate from the same subnet, as soon as scanner vary along this dimension nearly two thirds of the embedded scanner groups go by unnoticed, as the subnet-based method is only applicable in one third of the groups. The method is also able to detect single IP addresses based on a threshold, grouping multiple of these IP addresses across different subnets together is however not done by this method.

Our proposed methodology reaches its limit as soon as there is almost no exploitable commonality between the scan probes and between the header values. The residual 21% detection is accomplished due to the algorithm clustering these source IP addresses together because the only thing they have in common is that they have *no* clearly visible relationship. Having no commonality with others is also a commonality by itself, even though a weak one.

Table II shows that in order to thwart our method, adversaries have to reach a high level of sophistication in most dimensions. For example, from table II it can be seen that our method is able to detect the cases in which the scanner is hidden in every dimension, except for dimension #4. By not making strong assumptions on the form of a scan, it becomes harder for an attacker to evade this detection. We can therefore conclude that our method is successful in detecting different levels of hiding technique.

TABLE II: Precision based on levels of hiding. Indices based on figure 4

	Our method	Subnet	Dest. IP
Low hiding	100%	100%	59%
Medium hiding	100%	35%	0%
High hiding	21%	0%	0%
All high, #2 low	78%	12%	0%
All high, #5 low	26%	76%	0%
All high, #7 low	26%	0%	67%
All high, #7 medium	26%	0%	0%
All high, #4 medium	100%	0%	0%

TABLE III: Tools found using XOR-analysis, filtered to groups according to behavior by iterative filtering. Tool name is chosen as a unique identifier of the group.

Name	Fields in equation	Ports	#IP addresses
Unicorn [13]	key, srcIP, srcPort, dstPort	3389	246
Mirai-like [18]	dstIP	5555	6028
Dot zero	Shifted dstIP	130, 2434, 131, 2435	47
IPID	dstIP, IPID, dstPort	22	153
All	key, dstIP, dstPort, srcPort	22	25
Dst sess. key	key, dstIP	80, 443	34
Dst IP & ports	dstIP, dstPort, srcPort	21	19
Src IP & ports	srcIP, dstPort, srcPort	22	28

VII. Scanners in the wild

In the previous section, we have validated our proposed methodology to have excellent precision (98.1 - 100%) and high recall (75.5 - 87.9%), in other words while we will miss some slow groups, we will be able to reliably identify those within our detection capability. In this section, we will report on both the tooling and coordinated scanning groups discovered from the data obtained when applying the proposed methodology on scan traffic directed against the organization.

A. Tools

The two month trace revealed a total of eight tools that used combinations of the meta data within the header fields of the scan probe. All eight fingerprints embedded the scanning data within the sequence number field, while no other instances of an algorithmic generation of for example source port or source IP were observed. This is not surprising per se, as the ISN value can be flexibly set without imposing any requirement on the scanning setup: a scanner using a dynamically derived source port could not open a port but would have to listen in promiscuous mode and would risk a collision with other network applications running on the same host, while an encoding in the source IP address requires the adversary to listen to a large block of IP addresses.

Table III lists the discovered implementations, together with the fields used as part of the ISN generating fingerprint. As we see, all tooling encodes the IP address into the sequence number, which as a 32 bit value thus provides a very low likelihood for the scanner to accept incoming data by mistake. From the eight tool chains discovered, only the first two fingerprints can be matched with that of a known tool, Unicorn and Mirai, while for the remaining ones we have not been able to find any record in the existing academic literature or port scanner implementations matching these fingerprints.

B. Scanners

In total, we identified 1249 clusters of IP addresses in the data that employ the above tooling, but may be grouped into separate entities as their concrete behaviors differ, out of which we have manually inspected 50 of these groups. In the following, we will briefly outline three of these groups, their volume and a few behavioral characteristics. We used characteristics of the groups to provide them with a name.

1) Mirai-like: Several identified clusters in the data set used the destination IP directly as the sequence number in the probing packet, so that the return could be trivially identified by the scanner. As this is the same strategy Mirai used to pick its sequence numbers, we have named it as such. The IP addresses in this group send at most 20 packets to the 65,000 telescope IP addresses in a 60 minutes interval. This rate of sending packets would stay under the radar of any currently used IDS system we are aware of. Furthermore, the complete global distribution would let it go undetected by current scanner identification methods described in section II. By leveraging our detection methodology, a distributed scanning net such as Mirai could be identified before it is used for malicious purposes, and we could sooner be aware that there are anomalies in scanning traffic.

2) IPID group: In addition to the destination IP and destination port, this group uses the IP identification number in the XOR equation of the sequence number, ISN = dstIP $\oplus$ IPID $\oplus$ dstPort. This behavior is rather curious, as the IPID only has meaning in one direction and is not copied back into the return packet. In this case, the scanner most likely sets the outgoing IPID in a predictable way, as it will not be possible to identify the value on return of the packet, hence leading to our naming. Based on the behavioral model created for this group it became apparent that the IP identification number seems not be random, but rather is identical for every packet that is sent to the same destination IP and port, regardless of which source it originates from.

3) Dot zero group: The dot zero group is one of the stranger groups that have been uncovered. This group targets only IP addresses where the last byte is zero. Why only these IP addresses are scanned, and not other IP addresses is currently not known, especially as this is the network address in class C networks. Further study in what can be achieved by scanning these particular kinds of IP addresses should be done. It might be targeting Internet gateways for example, but it is unclear what this would accomplish. The group exhibits an interesting feature that was discovered by our proposed method in the iterative filtering step, in that the source port is constant for all the hosts in the same subnet. Take for example two IP addresses in the same subnet: *a.b.c.100* and *a.b.c.200*. These IP addresses will use the same source port. An IP address in a different subnet *c.b.a.100* will use a different source port, but also the same source port as any IP address in its own subnet.

VIII. Conclusion

In this work, we proposed a method to identify custom port scan tooling and coordinated scanners based on artifacts contained in the header fields. Modern port scan software no longer stores the state and the progression of network scans to avoid administrative bookkeeping on the local host, but rather embeds the data into header fields that are preserved when the target replies. This allows the scanner to differentiate between scan responses and unrelated traffic with minimal overhead.

To detect these tools and ultimately distributed slow scanners, we match all connection meta data in a pattern analysis, which becomes scalable as we match packets sent by different scanners to link those that rely on the same strategy. We demonstrate that the proposed methodology is able to detect scanners with various levels of stealthiness. When running the system on scan traffic directed against three class B networks, we were able to identify eight distinct tools based on these relations. The scanners identified seemed to be wide-spread, and was employed by a total of 1249 clusters of source addresses. Groups made deliberate efforts to trawl through the monitored networks at a slow enough rate to be not detectable by existing IDS/IPS tools, and violate the assumptions in previously proposed slow scan detection algorithms.

REFERENCES

[1] N. Blenn, V. Ghiëtte, and C. Doerr, "Quantifying the spectrum of denial-of-service attacks through internet backscatter," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, p. 21, ACM, 2017.

[2] Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," in *USENIX Security Symposium*, pp. 65–78, 2014.

[3] A. Dainotti, A. King, F. Papale, A. Pescape, *et al.*, "Analysis of a /0 stealth scan from a botnet," in *Proceedings of the 2012 Internet Measurement Conference*, pp. 1–14, ACM, 2012.

[4] C. Gates, "Co-ordinated port scans: A model, a detector and an evaluation methodology," 2006.

[5] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified np-complete graph problems," *Theoretical computer science*, vol. 1, no. 3, pp. 237–267, 1976.

[6] S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo, "Surveillance detection in high bandwidth environments," in *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1, pp. 130–138, IEEE, 2003.

[7] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet intrusions: global characteristics and prevalence," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 138–147, 2003.

[8] "RFC 971 - internet protocol." Retrieved from: https://tools.ietf.org/html/rfc791, at 18-06-2018.

[9] "RFC 973 - transmission control protocol." Retrieved from: https://tools.ietf.org/html/rfc793, at 18-06-2018.

[10] R. D. Graham, "Masscan: Mass ip port scanner," *URL: https://github.com/robertdavidgraham/masscan*, 2014.

[11] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman, "Zippier ZMap: Internet-wide scanning at 10 Gbps," in *WOOT*, 2014.

[12] V. Ghiette, N. Blenn, and C. Doerr, "Remote identification of port scan toolchains," in *IFIP International Conference on New Technologies, Mobility and Security*, 2016.

[13] V. Ghiëtte, H. Griffioen, and C. Doerr, "Fingerprinting tooling used for {SSH} compromisation attempts," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, pp. 61–71, 2019.

[14] "Finding very damaging needles in very large haystacks." Retrieved from: http://cs.ucsb.edu/ kemm/MURI/Presentations/paxson.pdf.

[15] J. Xie, B. K. Szymanski, and X. Liu, "Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pp. 344–349, IEEE, 2011.

[16] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: a comprehensive survey," *IEEE communications surveys & tutorials*, vol. 16, no. 3, pp. 1496–1519, 2014.

[17] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *USENIX Security Symposium*, vol. 8, pp. 47–53, 2013.

[18] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1093–1110, 2017.