Clustering Payloads: Grouping Randomized Scan Probes Into Campaign Templates

1st Vincent Ghiette
Delft University of Technology
Delft, The Netherlands
v.d.h.ghiette@tudelft.nl

2nd Christian Doerr
Hasso Plattner Institute for Digital Engineering
Potsdam, Germany
christian.doerr@hpi.de

***Abstract*—Over the past decade, the scanning landscape has significantly changed. Powerful tools such as Masscan or Zmap allow anyone to scan the entire Internet in a matter of hours. Simultaneously, we witnessed the emergence of stealthy scanners, which map the Internet from thousands of vantage points at a low rate attempting to forego detection.**

As scanning is typically the first step towards later intrusion, organizations need to track, understand and draw intelligence from these scan campaigns. Organizations benefit from obtaining insights into what adversaries are currently looking for, which might reveal some new vulnerabilities. Furthermore, relating IP addresses with each other participating in scan campaigns provides valuable insights into the adversary's capabilities.

In this paper, we describe a protocol-agnostic approach to extract commonalities and patterns from UDP scan traffic, relate individual scan packets regardless of whether they are sending static data or randomizing their payloads across destinations, and obtain 97% pattern accuracy with a data coverage of 96%. We apply our methodology on seven years of NTP and DNS scan traffic demonstrating that our automatic clustering provides stable tracking of strategies over time and identifies groups of source IPs with these behavioral characteristics effectively.

***Index Terms*—UDP, payload, clustering, network scans**

I. Introduction

Network scanning is often the precursor to a subsequent attack, and is used by adversaries to discover active hosts, exposed and vulnerable services, and potential weaknesses in a network [1]. As such, network scans provide useful information to defenders as it can alert them to the type of services and vulnerabilities the adversary is looking for, and provide the opportunity to strengthen and complement defenses long before a compromisation attempt will commence [2].

Two important and diverging developments have drastically changed the scanning landscape over the past decade. On the one hand, high-speed scanning tools such as Masscan or Zmap enable anyone to systematically trawl through the entire Internet from a single PC in a matter of hours and even minutes on a 10 GbE link. While these activities will be immensely noisy and obvious to anyone, they allow adversaries to keep track of hosts and open ports, for example to keep up to date information about DDoS amplifiers [3]. On the other hand, we observe a more widespread application of stealthy scanning, where adversaries are probing the IPv4 address space at a very low rate from thousands and even tens of thousands of vantage points [4], [5]. With inter-arrival times of scan probes in the order of hours or days, it is difficult for organizations to keep track of these scan campaigns and relate those IP addresses participating in one to each other with existing tools such as intrusion detection systems or SIEMs [6].

Previous work has mainly focused on detecting scans and scanning campaigns in TCP, as this transport protocol accounts for the majority of scanning traffic [7], [8]. Although TCP scans are more prevalent, scanning for devices running services on top of the UDP protocol can be lucrative, as these services are on the one hand typically abused in amplification attacks, and on the other hand it is easy to realize high-performance, low latency protocols on UDP, as exemplified by the transition of HTTP3 on UDP. As the user datagram protocol is handshake-less, this offers a different, novel angle to identify and interpret scan campaigns. While in TCP a service (or honeypot) needs to be active to complete the TCP handshake before any actual data is exchanged, in UDP the scanner has to send some payload already in the first probe packet. As a service might not respond at all unless some correctly formatted and valid request is sent, the adversary cannot simply conduct surveys of open ports but necessarily has to reveal what he is looking for, potentially even to the extent of including some specific exploit in the probe packet. This is, for instance, the case for a vulnerability found in Juniper routers [9], depending on the sent UDP payload, an attacker can reveal whether he has the intent to execute code or perform a DoS attack. This allows us to quantify port scanning and characterize scanners based on what they are trying to do and relate them based on common intent.

This angle has received almost no attention to date, and in this paper, we demonstrate that it is possible to automatically extract common scan traffic into user-interpretable patterns across any UDP protocol, human-readable or binary. Defenders can, therefore, directly assess how a particular service is targeted by an adversary and in case of a new or unmitigated exploit rollout controls. Even though stealthy scanning distributes scan probes across many hosts and over extended periods of time, any of these scan probes still has to use the same or similar packet payload. As we are matching probes based on commonalities, our approach is effective in identifying such distributed campaigns. With this paper, we

ISBN 978-3-903176-48-5

make the following three contributions:

- We propose a protocol-agnostic method that can extract commonalities across scans in unsupervised learning, and translate them into understandable templates that help the defender understand the mechanisms and potential aim of the scan. We show that the method is effective for both static scan probes as well as those that randomize header and payload components on a per-destination basis.
- We demonstrate that such templates effectively track developments for individual scanning groups as well as the scanning landscape. For instance, the template analysis clearly identifies how DNS scanners shifted their attention from the ANY to the TXT query over time.
- As scan campaigns by definition will show some form of commonality, our template method allows us to also identify scan campaigns by matching sources that send an instantiation of an abstract template. We show that this technique can match highly-distributed scans with very low inter-arrival times and link addresses even if they only occasionally emerge.

II. Related Work

Lockheed Martin shows that scanning the Internet is often the first step taken to set up a cyber attack [2]. Scanning the Internet for devices can be a lucrative endeavor for cybercriminals. Researchers scanning the Internet show that 13.8% of 3 million devices use standard login credentials [10], making them easy targets for cybercriminals. In order to get a better understanding of attackers scanning the Internet for vulnerable devices, researchers have studied scanning behaviors. Research [4], [11]–[16] shows that scanning activity can be identified and that attackers adopt different scanning tactics. Attackers target different protocols, use vertical and horizontal scanning strategies, and leverage blacklists to avoid detection. In addition, Lee et al. [17] reveal that scanners use decoy probes to hide their activities. With detecting scanning activities playing an important role in cybersecurity, Coudriau et al. [18] developed a visualization tool, and Iglesias and Zseby developed a method to identify new scanning campaigns [19].

By monitoring and analyzing the tactics used by attackers, researchers have shown the capability to identify the tools used by scanners [7], [13]. The tools are identified using time series analysis or by analyzing the header fields of the received scanning probes. Further studies are conducted showing that scans performed by malware and botnet can be identified [5], [8], [20]. The identified scans performed by malware offer an insight into which services botmasters are targeting. Antonakakis et al. [21] show that the Mirai botnet has increased the number of destination ports scanned during its lifetime in order to increase the number of devices it can infect. Next to monitoring botnet evolution, analyzing the scans performed by botnets can contribute to less evident studies such as measuring the IP churn in autonomous systems [22].

Besides monitoring the scanning activities of botnets, researchers have analyzed distributed scanning campaigns in general. Early research performed by Gates [23] shows that by measuring the overlap in scanned IP addresses, distributed scanning campaigns can be identified. Subsequent research [7], [24], [25] utilizes properties found in the TCP header to identify large distributed scanning campaigns.

Fig. 1: UDP scans only return results if a valid request is sent.

The works above show that it is possible to identify distributed scanning campaigns. However, most research only analyzes scans towards services running on top of the TCP protocol. Although 86.3% of the scans are directed towards TCP ports [7], services running on top of the UDP protocol are prone to be abused by cybercriminals setting up DDoS attacks. Ghiette and Doerr [26] show that after the Memcached vulnerability was exposed, an explosion in scanning activity followed. The UDP protocol is connectionless, forcing scanners to send scan probes containing valid payloads. In the following work, we leverage the necessity of scanners to use probes with valid payloads and introduce a protocol-agnostic method to analyze scan traffic. Using our proposed method, we offer an insight into the UDP scanning landscape, showing a change in scanning behavior over the past seven years. Similar to [7], [24], [25] which uses TCP header values, we show that the UDP payloads can be used to identify distributed scans.

III. Protocol-agnostic template extraction

In order to make sense of scanning data, it would be useful to identify common patterns in scan traffic to help defenders understand the goals of scanners even in the presence of randomization, and link origins of common scanning patterns to identify campaigns and collaborating entities. In the following, we discuss our approach to identify such patterns in a protocol-agnostic way for UDP. Before we dive into the approach, we will first briefly summarize how adversaries perform UDP scanning to motivate the requirements for template creation.

A. Randomized UDP scanning

The connectionless nature of the UDP protocol dictates how attackers may scan for devices running a service on top of the UDP protocol. If a UDP packet is sent to a remote host, the protocol specifies that the host returns an ICMP port unreachable notification in case no service is listening to the targeted port. Hence, upon receiving such an answer, the adversaries would know for certain that no service is present, and if no answer is sent, a UDP service should be present. In today's networks, this reasoning does not work anymore as notifications are typically disabled by operating systems and filtered by the network to prevent such scan surveys. To illustrate this, we sent a DNS query to port 53 to 10 million randomly chosen IP addresses on the Internet. 19,553 IPs responded to our query, 389,673 returned an ICMP port unreachable, while 95.9% of the probed IPs remained silent.

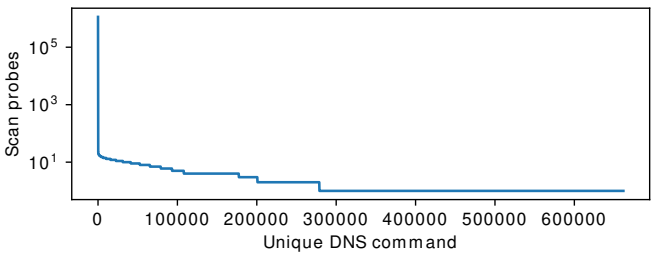


Fig. 2: Scan probes received per unique DNS command.

To determine whether a port is open or closed, scanners would need to elicit a response from a targeted service. Blindly blasting out random data would however also not provide any insight, as the software listening behind a port would probably parse incoming requests and only respond to those it understands. This thus mandates the scanner to craft valid protocol headers and payloads for each targeted protocol as shown in figure 1. A variety of scanning tools feature built-in probes for specific protocols allowing attackers to scan the Internet with little protocol knowledge, as well as allow advanced attackers to further customize the payload.

Such customization includes, for example, the randomization of certain parts of the sent header and payload, for example, the query ID in a DNS packet, which according to the Zmap documentation [27] can be effective in circumventing security measures in multi-homed systems. Creative attackers may also randomize other parts of DNS queries, such as changing the subdomain for each probe, in an attempt to evade IDS signatures or detection and filtering if repeated packet payloads are used to detect scans.

These practices drastically complicate campaign analysis, as we would not know apriori if and how scan probes would look like and how such randomization might take place, in practice these templating options by common scanning tools make each scan probe appear as unique. Previous work identified an entire range of practices, featuring basic randomized header and payload fields to complex relationships such as the encoding of IP/port information in the payload, possibly further masked with bit shifts or encrypted with session secrets [13], [28].

To visualize this immense heterogeneity in today's port scanning, let us look at scanning traffic collected at a network telescope containing over 65,000 unused IPv4 addresses between 2017 and 2021. Given the large scale of the telescope and intensive port scanning on the Internet today, it is sufficient to only include one month per year in our analysis. We perform our analysis using DNS and NTP scanning traffic, as we show in section III-B that they possess general traits exhibited by many protocols, yielding in a data set containing 104.5 million DNS and 620,7 million NTP scan probes that we will work with further in this analysis.

Figure 2 shows the distribution of unique scan probes sent towards port 53, the standard port for the DNS, in ascending order by the number of probes received. We immediately notice that the distribution of scan probes is heavy-tailed. The 28 most commonly sent DNS packets are received between 50,000 and one million times by our network telescope, while for more than 30,000 payloads, we only receive one probe. This means that randomization is a default practice and clustering identical packets in campaigns is ineffective for the bulk of scanning. Considering that our telescope ranges account for more than 65,000 IP addresses, we also see that the apparent heavy usage of those "typical" 28 payloads is not that distinctive if we receive on average 2.69 such static payloads per telescope IP per month. Effective scan detection and campaign clustering thus requires automatic detection of these packet generation templates.

B. Characterizing protocol traits

In the following, we develop a protocol-agnostic procedure to extract the templates used by scanners when generating UDP probes. Currently, IANA lists over 6,000 services running on top of the UDP protocol. Although each protocol is different, there are some shared commonalities between them. We analyze the seven most commonly scanned UDP protocols [29] and identify general features that can be found across all these protocols, which we will use to design our protocol-agnostics method to handle UDP packets in general.

(1) The first characteristic is the usage of binary or text-based data. Protocols such as CHARGEN and SIP exchange textual data while NTP uses binary data. Others such a DNS include a mix of both in their payloads. An effective method for behavior clustering should handle binary, textual, and mixed payloads equally well.

(2) The second aspect relates to the use of randomized data. As UDP is connectionless, it forces protocols to include handles to differentiate connections. Protocols often use fields with per-session randomness to do so. As an example, SIP assigns a random value to sessions, and DNS randomizes its query ID. Therefore, a clustering should be resilient to partially randomized content, if payloads are otherwise "sufficiently" similar.

(3) The third aspect relates to length, or more precisely the *lack* of length requirements. Protocols naturally differ into their packet structure and many protocols allow messages to vary in length. For instance, DNS servers will respond to a short bind command and to longer domain name queries. Similarly, SIP servers allow clients to send a user name, affecting the message length. Therefore, a method should not require any apriori knowledge and be able to relate payloads even if they differ in structure.

(4) The last feature is related to protocols allowing customized commands. Some protocols, such as NTP, are strictly defined in terms of payload content. Others, such as DNS and CHARGEN allow for arbitrary data in the payload. As we do not know "how different" a protocol typically is, this aspect should be learned from the data itself.

C. Proposed methodology

We use these four general characteristics to develop a protocol-agnostic process that allows us to extract scan tem-

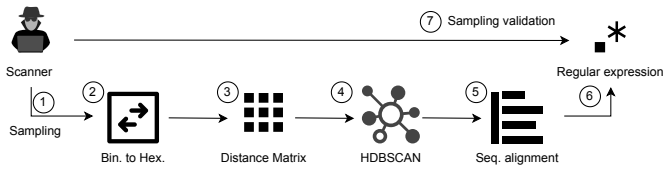


Fig. 3: Templating process

```
Clustered commands
0.0.0.0.in-addr.arpa
111.11.11.111.in-addr.arpa

Aligned commands
--0.-0.-0.--0.in-addr.arpa
111.11.11.111.in-addr.arpa

Regular expression
^.{0,2}.\..{0,1}.\..{0,1}.\..{0,2}.\.in-addr.arpa$
```

Fig. 4: Example of template extraction.

plates used by attackers regardless of the underlying protocol. Figure 3 depicts our data processing pipeline for clustering and transforming scan traffic into human-readable templates. In the following, we systematically explain each of the seven steps taken by our approach and justify our design.

(1) We begin our pipeline by randomly sampling packets from incoming scan traffic. Data sampling is necessary due to the intensity of scanning on the Internet. As state-of-the-art cluster generation has a complexity of $\mathcal{O}(n \log n)$ and require distance matrices of n^2, it is prohibitive in scenarios like Internet scanning where we receive millions of packets for popular UDP protocols in a given month. In this step, we randomly sample packets to a degree fitting our computational resources. In our work, we randomly selected 70,000 scan probes, as in theory, such sampling could miss out on certain patterns, the pipeline verifies the sampling in the final step and adjusts accordingly.

(2) We convert the payloads to a uniform representation. To accommodate for payloads with text and binary information, we transform all payloads to their equivalent hexadecimal string notation creating a uniform representation. A hexadecimal string notation allows us to use a distance metric that works on both binary and textual commands.

(3) We calculate the distance matrix as input for cluster generation. We use the Damerau-Levenshtein distance [30], which counts the minimum number of insertions, swaps, and deletions required to make two strings identical. The Damerau-Levenshtein distance may compute the similarity of strings with different lengths, which makes the algorithm agnostic to different types of protocols and the potential to introduce arbitrary protocol payloads.

(4) We cluster the sampled packets using the HDBSCAN algorithm. HDBSCAN has the nice property to cluster data with the characteristics discussed in the previous section. Furthermore, it does not require knowledge of the number of clusters to generate good results, but uses the data itself to determine a cut-off. Scanners can use templates that introduce varying degrees of randomness, for instance, a template $S1$ generating random DNS queries for subdomains of *google.com* for which the subdomain length is 5 and a template $S2$ of random queries for subdomains of *amazon.com* with a random subdomain length of 20. As the calculated Damerau-Levenshtein distances between the probes generated by $S1$ are smaller than the distances between the probes generated by $S2$, and HDBSCAN can find clusters with varying densities, HDBSCAN will form clusters grouping the commands generated by $S1$ and $S2$ in separate clusters. Additionally, we selected HDBSCAN because it can handle noise in the dataset caused by Internet backscatter.

(5) We translate the identified clusters into a common representation. We expect clusters to contain payloads of different lengths. However, different length patterns cannot be harmonized. Therefore, we apply sequence alignment using MAFFT to the clustered probes.

(6) We unify the aligned probes in each cluster into a human-readable template representing all its members. We use the aligned payloads of the clusters to generate regular expressions, specifically as it allows to define random parts in the payloads and can handle the gaps introduced by the sequence alignment process while being easily interpretable by a human. We generate the regular expression by placing a '.' in the regular expression where characters in the probes vary, the gap characters introduced by the alignment are noted by a '.{0,n}' in the regular expression, where n is the number of consecutive gaps. Characters that do not vary throughout the probes remain the same. Figure 4 illustrates the template building process using two reverse DNS lookup commands. We illustrate the process using the actual commands, not the hexadecimal translation, as it is easier to understand.

(7) In the final step, we map the remaining dataset onto the resulting templates to validate the sampling process. Although sampling is randomized, it may affect the detection rate of templates. To measure whether all templates have been identified, we identify any data that cannot be matched by the crafted regular expressions, and rerun for potentially unclassified data. In practice we find that unmatched data is negligible.

IV. Validating our approach

In section III we introduced a method to extract commands used by scanners to probe for UDP services. Our method bundles HDBSCAN on a sampled data set and explains found clusters using regular expressions. Although HDBSCAN and regular expressions are established techniques, we are not aware of previous work demonstrating its ability to cluster scanning probes. In this section, we validate whether the methodology creates meaningful clusters and templates using domain-specific knowledge and verify that despite sampling, the template generation delivers meaningful results.

Validation data set. In the following, we validate our methodology by clustering DNS probes as this protocol uses all protocol characteristics identified in section III-B, specifically both textual and binary payloads, commands can be of

TABLE I: Clustering results for three months of DNS scans.

Data set	Num. clusters	Noise points	P <1	% Clustered
2018-01	72	380	10	99.44
2018-04	53	275	7	99.59
2018-07	59	457	11	99.33

different lengths, and most importantly, the protocol allows for user input of arbitrary lengths, which can be randomized. To analyze the stability of the clustering itself, we cluster three datasets each spanning one month in 2018, believing that over the course of a few months the behavioral evolution of scanners should be minimal, allowing us to evaluate the stability of our cluster generation. On average, each dataset contains 10.1 million DNS packets.

Clustering results. When we apply sampling and clustering as explained before to the test datasets above, we obtain very high cluster assignment scores as shown in table I. HDBSCAN allows for points to be classified as noise and assigns a *P* value, between 0 and 1, to each point in a cluster representing the probability of that point belonging to the assumed cluster. We see that density-based clustering is very effective for packet data, and only for less than 0.7% of all points the *P* value is smaller than 1. More than 99.33% of packets are clustered with high confidence across each dataset.

Evaluation of generated clusters. Table I also shows that the number of clusters significantly varies each month. Clusters are also not equal in size, the top ten most significant clusters contain between 66.73% and 69.91% of the data for each clustered dataset. We manually inspected and analyzed all of the clusters generated by HDBSCAN, to verify that HDBSCAN-based clustering of payload data generated meaningful results. As mentioned earlier, scanners can either use sent static scan probes or dynamically generate their packets. Clusters containing probes generated by a static approach are trivially explained as the cluster members should only contain identical payloads. If a cluster contains probes from dynamic generation, we need to determine whether the payloads in the cluster could have been generated using the same template. We will treat static and dynamic clusters separately below.

a) Static clusters. HDBSCAN identified 72, 53, and 59 clusters for the three monthly data sets. Of the identified clusters, 62, 43, and 45 respectively are composed of identical commands. If the clustering related identical payloads correctly, there must not be any instance in the dataset where a particular static payload from a given cluster is also part of another cluster, as these should have been merged during the clustering process. Indeed, in none of the test datasets we find any instance of such overlap.

b) Dynamic clusters. The validation of clusters containing different commands requires us to ascertain whether all commands in a particular cluster can indeed be generated using the same template. HDBSCAN forms 10, 10, and 14 clusters containing different commands for the three validation data sets. In the following, we showcase the examination of the results for the month of January, and report on the other two.

TABLE II: Dynamic clusters in the Jan 2018 validation set.

Cluster ID	Number of commands	Unique commands	Unique domains	Unique QIDs
0	119	119	-	-
1	259	257	1	257
2	1003	1003	1003	1
3	1038	1032	1032	1029
4	1115	1107	1107	1103
32	4128	4005	1	4005
38	2331	2226	1	2226
12	153	2	1	2
46	141	2	1	2

We analyze the clusters semantically by parsing the binary data and translating it into the DNS packet structure as specified by RFC 1035 [31]. The clusters identified by the algorithm contain commands that differ either by the query ID, domain name, or both. Table II shows for each dynamic cluster the number of (i) commands, (ii) unique commands, (iii) commands containing a unique domain name, and (iv) those having a unique query ID.

The first type of behavior clustered together are DNS queries with a static domain but every packet containing a random query ID. Clusters 1, 38, and 32 are examples of these, but each of these clusters target a different domain name. All scan packets in each of these clusters can be generated by a template in which all of the DNS fields are fixed except for the query ID field that is randomized for each probe. The use of such a template is recommended by the manual of Zmap [27].

The second type of behavior is where every probe queries a different domain name, but otherwise the packet structure is entirely static. Cluster 2 is an example of this, where every scan packet contains a randomly-generated subdomain all in the form *xxx.xxxxxxxxx.wc.syssec.rub.de* (where x represents an alphanumerical character).

The third type of behavior is where different parts, here the domain and query ID, vary per packet. Clusters 3 and 4 are examples of this, but within each cluster the top- and second-level domain is identical. Cluster 3 contains probes for random subdomains of the *openresolvertest.net* domain, and cluster 4 for subdomains of the *openresolverproject.org*. Although both clusters use the same generator where random subdomains are composed of 9 alphanumerical characters, the pipeline treats them as separate behaviors. Although clusters 12 and 46 vary in both aspects, we find that they contain two domain names and two query IDs, and were accidentally formed in an attempt to meet the minimum cluster size of 100. If high precision results are needed, the algorithm could be rerun with lower minimum cluster sizes if some generated templates overlap. Cluster 0 does not contain domains nor query IDs, indicated in table II, as this cluster contains SIP probes that were sent towards port 53. However, all these commands are correctly related as they are generated using sipvicious [32], a scanning tool specially developed for scanning SIP servers.

Using domain-specific knowledge, we have analyzed all 184

identified clusters, of which 97.28% are valid patterns.

Evaluating the regular expressions. After validating the extracted clusters, we also verify the methodologies' fifth and sixth steps for generating regular expressions. Each expression should be unique as it represents a scan template, and no expression should overlap. Furthermore, these expressions should be as tight as possible, a regular expression representing one cluster should not be able to match any command found in other clusters. As we verify the pipeline output for the 184 clusters and regular expression, we see that every expression is unique and does not match packets from any other cluster, with the exception of those two clusters we identified as incorrectly merged due to the minimum size requirement. The remaining 182 generated expressions provide a unique representation for their clusters.

Evaluating the sampling. The final step in our evaluation is to verify that the sampling performed in the first step of the pipeline did not influence the template detection rate. We use the previously evaluated regular expressions to match probes in their respective data sets. After matching, we find that 96.04% of all the probes are mapped to a regular expression indicating that the sampling had minimal effect.

V. Longitudinal Study of Command Usage

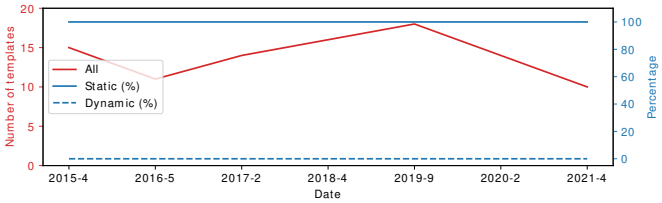
In the previous sections, we proposed and validated a methodology to cluster and extract commands used by scanners. In this section, we apply our approach to perform a longitudinal study of the commands in scan campaigns for DNS and NTP protocols between 2015 and 2021.

A. Evolution of command templates

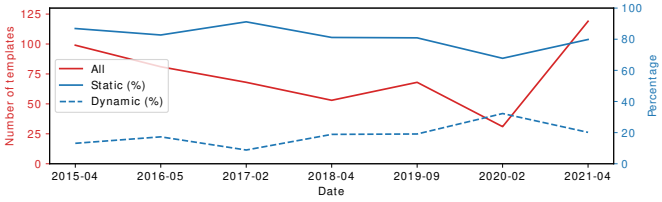
In section IV, we showed that attackers use templates to craft dynamic and static probes based on templates, each with its own advantages and drawbacks. In this section, we use our methodology to obtain which types of templates have been used over the years and how the approach to scanning changed over the course of time.

To analyze the development of scanning practices over time, we draw on the dataset of scan traffic collected by 65,000 telescope IPs over the last seven years, and let the algorithm extract the templates used by scanners in each time slot. Figure 5a shows the percentage of static and dynamic NTP commands as well as the number of unique commands in use, figure 5b depicts the same for DNS. As we see in the graph, everyone who is scanning for NTP services always does so by directing the same scan payload towards different destination addresses[1]. While each scanner behaves entirely static, there is variation between the groups of people who scan for NTP. Over the years, we see between 10 and 18 patterns being employed. The complete lack of dynamic in NTP is not per se surprising, as the protocol specification provides only little opportunity to include arbitrary data. Among the very few possibilities, NTP versions 3 and 4 allow the inclusion of three timestamps to represent the current time configurations. We verified that scanning using randomization provides valid responses in NTPv4, but scanners do not pursue this strategy.

[1]The single dynamic scanner instance in 2018 is not an exception to this rule, but a scanner who has launched a campaign sending SIP requests to NTP servers.



(a) NTP



(b) DNS

Fig. 5: Number and share of templates in use in NTP & DNS.

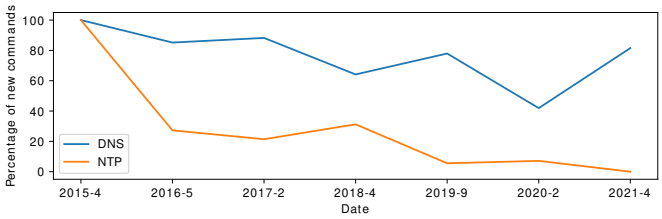


Fig. 6: Percentage of new templates in NTP & DNS.

The practice of scanning is however different in case of DNS as shown in figure 5b. For DNS, our algorithm identified typically between 53 and 121 distinctive scan patterns per year except for an outlier in 2020, and between 8% and 33% of these identified templates are dynamic and randomized. Aside from a slight increase in DNS, we find that the general tactics and preferences of scan randomization or usage of static payloads has not significantly changed over those years.

While the previous results would indicate a basically static ecosystem with no evolution at all, there is actually significant innovation on what exactly the scanners send. Figure 6 shows the percentage of new NTP and DNS commands that are seen for each year, in other words, as we see 99 DNS commands in total in 2015 as shown in figure 5b, the value of 85% depicted here for 2016 says that out of the 81 templates we discovered in 2015, 85% or 69 in total were new. As we see in case of DNS, scanners constantly update how they perform the scans, and in 2021 more than 80% of all scan packets used patterns that we did not ever see before in those past seven years. This is due to the widespread adoption of new query types (as we show in the next section), as well as evolution of the protocol.

In contrary to this, the percentage of newly observed NTP

commands is minimal and diminishes quickly. Curiously, based on manual inspection, those new templates that *are* used stem mainly from operational mistakes by scanner sending invalid NTP requests, or by scanning for a protocol other than NTP on the IANA NTP default port. Although the NTP protocol was updated three times within the last decade, none of the new features are included by scanners in their activity. As shown by [3] for case of DDoS amplification attacks, adversaries seem to form different communities each focusing particular protocols as their tools of the trade with different degrees of sophistication and techniques. A similar situation also seems to be true in case of scanning.

B. Evolution of command types

Previously, we showed that scanners probing for DNS display a significant degree of variation in their activities. The advantage of our methodology is that it creates human-readable patterns that we can interpret with respect to the potential target and intention of the activity. In the following, we report on the content and development of these templates over time.

Targeted domain name. One easily changed feature of DNS scan probes is the domain name that is being targeted. If they perform surveys of open DNS resolvers, scanners might change what domain name they query for, as commonly used ones (such as Zmap's default query for google.it) might be flagged or blocked within the destination network. If actors are searching for DNS amplification attacks, they might switch over to zones offering larger amplification or target a different authoritative name server.

We analyze the discovered templates (which might include randomization in headers or payload fields such as random subdomains) with respect to the following three features: first, how often particular domain names are targeted. Second, the percentage of source IP addresses that use a particular domain. Third, the volume of scan packets that are received containing a particular domain name. Figure 7 shows the distribution for the three metrics for our snapshot traffic from 2015 until 2021.

In each bar plot, each color represents a particular domain. As the number of domains exceeds the number of meaningful colors, every color is used multiple times in a rotating fashion. However, the same color at the same position symbolizes the same domain name across these seven years. As we are concerned with respect to global developments and not concrete targets, we also omit to break out the 179 targeted domains in a legend. Figure 7a depicts how often a particular domain name appears in a particular template, for instance, a campaign enumerating subdomains in the form of `www[0-9]{3,5}.domain.com` would be grouped into one template, a campaign randomly looking up subdomains in the form of `host-[a-z]0[0-9]+.domain.com` would be grouped into another, both however share the second-level domain domain.com in their template. The figure shows no evidence of global trends, domains appear with fluctuating "market shares" across the years, except that over time more domain names appear in the templates of DNS scanners.

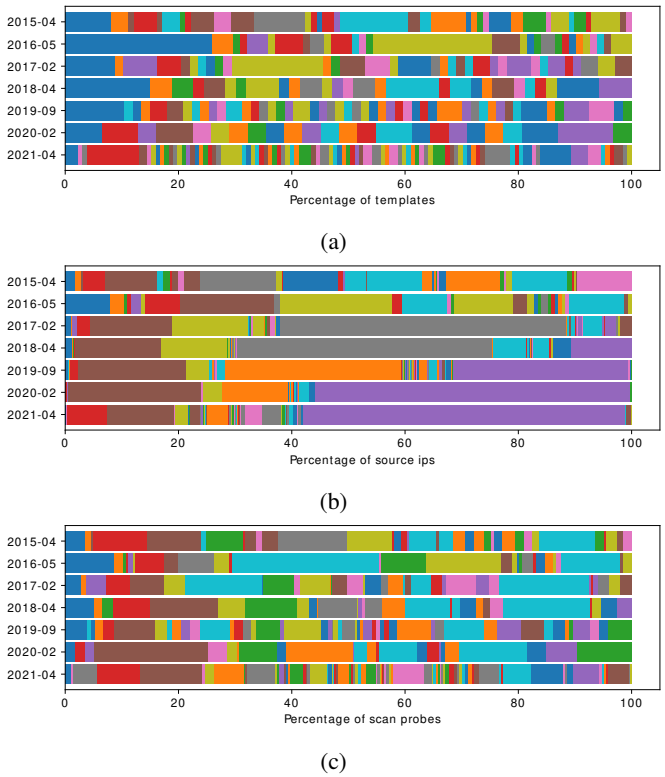
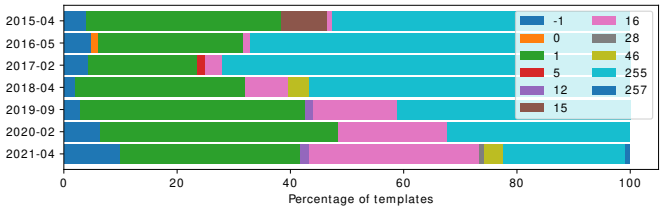


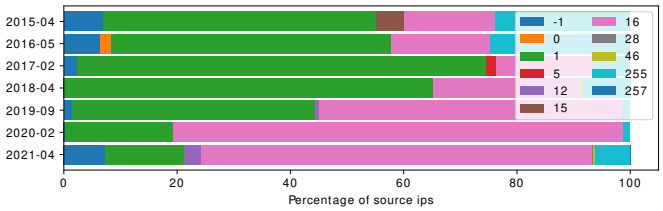
Fig. 7: Domain name usage of scanners.

When we look at the percentage of source IPs that have send a probe belonging to a particular template as shown in figure 7b, we see that although there is a lot of variety in the ecosystem in terms of what domain names are queried, the absolute bulk of sources are using on a few select templates. Of particular importance here is the purple bar, which depicts the *version.bind.* command, used to discover the software version of the most widely used DNS resolver software bind, for example to execute a particular exploit. While in 2017 only 1.8% of the source IPs sent a probe containing the bind command, in 2021 the percentage increased to 56.7%. Figure 7c shows the percentage of packets that are sent using a particular template. The figure shows that the number of received probes per template is more or less equally distributed. The different distributions shown by figures 7b and 7c show that a large number of source IPs using the same template are responsible for a relatively low amount of scanning probes, indicating the presence of large-scale distributed scanning campaigns. In section VI we show that, indeed, scanning campaigns can be identified using templates.

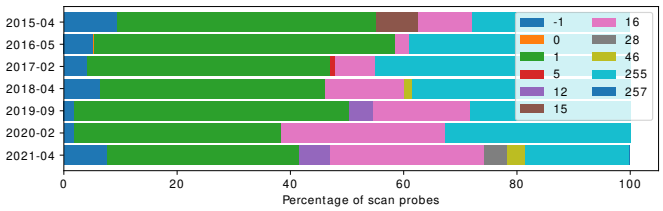
Used query types. Aside from the domain name, also the parameters of the domain query can be adjusted. In the simplest cases, this would entail the type of query, such as A, NS or ANY, but also instructions whether the target should recurse or validate DNSSec. Figure 8 depicts the development in DNS query type over the past seven years based on the same metrics as before, but given the limited number of query types



(a)



(b)



(c)

Fig. 8: Query type usage of scanners.

each is labeled by its numerical value as specified in [31].

We see very clearly how the ecosystem of scanners adapts its templates to transition from the ANY (255) query type to using the TXT (16) query type. While in 2017, 72.1% of all the identified templates used ANY, in 2021 this number has dropped to 21.6%. Templates based on ANY – the query for *any* kind of DNS record, a response typically extensive in length and thus useful for DNS amplification attacks – are successively being replaced with those using TXT, which increased from 2.9% to 30.0% during this time. TXT record fields today contain security features such as DNS keys, record signatures, or certificate authority pointers, and are thus also large enough in size to be interesting avenues for abuse.

When we study the number of sources relying on a particular template / query type combination in figure 8b, we see that this shift is basically adopted by the entire ecosystem. In 2017, 17.13% relied on TXT queries, this percentage has increased to 69.1% by 2021. Figure 8c shows the percentage of probes received for a particular query type and confirms the shift of scanners using TXT queries in favor of the ANY type.

VI. Identifying Campaigns using Probe Templates

In the previous section, we clearly identified signs of distributed scans when we looked at which sources scanned using a particular template. These coordinated activities stem from a variety of origins, for example cyber security firms, universities, as well as malicious actors scaling out port scanning to avoid detection. In this section we show that our

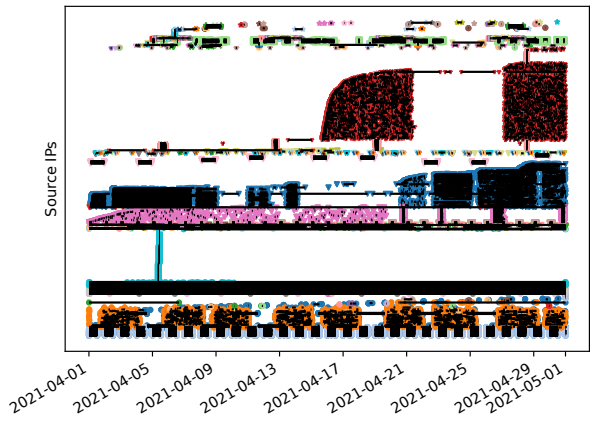


Fig. 9: Scans performed in 2021 grouped by used template.

methodology is also effective at identifying these campaigns and relating the participating IP addresses to each other.

We refer to a port scan as a sequence of probes sent by a source IP towards one or more destinations host/port combinations in a fixed time period. These individual port scans can be grouped into a scan campaign, which is the systematic scanning activity by multiple entities towards a particular goal. Scan campaigns are usually scaled out to multiple sources, as each device needs to run at less intensity and runs less risk of being detected or blocked. As these devices are under control by the same actor, they are usually managed in a comparable way, which means that the individual devices often show similar behavior, such as comparable if not identical start and end times, activity periods, or in the extreme case even similar port allocations. We showed in section V-B that scanners often rely on templates to generate their scan probes, where each source would send (randomized) instantiations of a general template, used as a blueprint to specify the scan. We can use these characteristics to identify a campaign and the devices participating in it.

To demonstrate this feasibility, we show the results of campaign extraction given scan traffic received through April 2021. We use our methodology to extract templates from this set and assign every source IP a label based on templates it instantiates. Figure 9 depicts the individual scanning activity of hosts over the course of one month, ordered by its first occurrence. The color of each activity indicates its cluster membership, the beginning and end of each scan is indicated by a black marker. We consider a scan terminated if we receive no packets from a source address for more than six hours.

As we can visually inspect, IP addresses matched on content templates also show other similar behavioral characteristics. For example, the scanning activities performed by the orange IPs all run repeated scans in coordination, and also the behavior of the red IPs is matched which may start and stop simultaneously, but show also a slow startup phase that would be difficult to find based on on/off activity only.

As the methodology can relate common behavior fuzzily, this grouping is also possible if there is little data available.

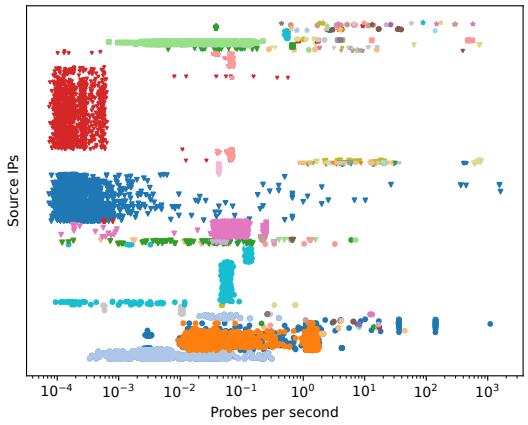


Fig. 10: Scanning speeds in 2021 grouped by template.

Figure 10 shows the speed of each source IP in terms of sent probes per second, the colors and markers match those shown earlier in figure 9. We can visually see that the source IPs using the same templates scan at the same speed. Particularly interesting is here the fact that the more sources a campaigns is using, the slower it tends to send probes. The two largest campaigns in the figure emit on average about one packet every 1 $\frac{1}{2}$ hours, compared to some of the faster campaigns at more than 10,000 packets per second. Identifying and linking scanners when operating at such a low rate is typically challenging, as they can only afford such a low rate by using a large number of sources contributing to the same goal, we can still identify them given our template methodology.

VII. Conclusion

In this work, we presented and validated a novel approach to identify the templates used by scanners. Using our approach, we can parse 96.04% of the scanning traffic and extract the used templates with an accuracy of 97.28%. We use our approach to perform a longitudinal study showing that the DNS scanning landscape has changed. Scanners have shifted from using ANY queries to TXT queries, most likely in response to the introduction of RFC 8482 [33] limiting the information gained by scanners using the ANY query. Finally, we leveraged our templating approach to identify large scanning campaigns.

References

[1] O. Al-Jarrah and A. Arafat, "Network intrusion detection system using attack behavior classification," in *IEEE ICICS*, 2014.

[2] Lockheed Martin, "The cyber kill chain," Tech. Rep., Acccessed April 2021.

[3] H. Griffioen, K. Oosthoek, P. van der Knaap, and C. Doerr, "Scan, test, execute: Adversarial tactics in amplification ddos attacks," in *ACM CCS*, 2021.

[4] S. Haas, F. Wilkens, and M. Fischer, "Scan correlation–revealing distributed scan campaigns," in *IEEE/IFIP NOMS*, 2020.

[6] M. G. Kang, J. Caballero, and D. Song, "Distributed evasive scan techniques and countermeasures," in *SIG SIDAR DIMVA*. Springer, 2007.

[5] S. Torabi, E. Bou-Harb, C. Assi, E. B. Karbab, A. Boukhtouta, and M. Debbabi, "Inferring and investigating iot-generated scanning campaigns targeting a large network telescope," *IEEE TDSC*, 2020.

[7] E. Bou-Harb, M. Debbabi, and C. Assi, "A systematic approach for detecting and clustering distributed cyber scanning," *Computer Networks*, 2013.

[8] S. Torabi, E. Bou-Harb, C. Assi, M. Galluscio, A. Boukhtouta, and M. Debbabi, "Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective," in *IEEE/IFIP DSN*, 2018.

[9] "Junos os: Remote code execution vulnerability in overlayd service (cve-2021-0254)," https://kb.juniper.net/JSA11147, acccessed 15 February 2021.

[10] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2008.

[11] H. Heo and S. Shin, "Who is knocking on the telnet port: A large-scale empirical study of network scanning," in *Asia CCS*, 2018.

[12] M. Alsaleh and P. C. Van Oorschot, "Network scan detection with lqs: a lightweight, quick and stateful algorithm," in *CCS*, 2011.

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

[14] P. S. Joshi and H. A. Dinesha, "Survey on identification of malicious activities by monitoring darknet access," in *ICSSIT*, 2020.

[15] A. Affinito, A. Botta, L. Gallo, M. Garofalo, and G. Ventre, "Spark-based port and net scan detection," in *ACM/SIGAPP SAC*, 2020.

[16] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, 2002.

[17] C. B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," *Univeristy of California, Department of Computer Science and Engineering*, 2003.

[18] M. Coudriau, A. Lahmadi, and J. François, "Topological analysis and visualisation of network monitoring data: Darknet case study," in *IEEE WIFS*, 2016.

[19] F. Iglesias and T. Zseby, "Modelling ip darkspace traffic by means of clustering techniques," in *IEEE CNS*, 2014.

[20] E. Bou-Harb, M. Debbabi, and C. Assi, "On fingerprinting probing activities," *Computers & Security*, 2014.

[21] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017.

[22] H. Griffioen and C. Doerr, "Quantifying autonomous system ip churn using attack traffic of botnets," in *ARES*, 2020.

[23] C. Gates, "Coordinated scan detection." in *NDSS*, 2009.

[24] S. Pang, D. Komosny, L. Zhu, R. Zhang, A. Sarrafzadeh, T. Ban, and D. Inoue, "Malicious events grouping via behavior based darknet traffic flow analysis," *Wireless Personal Communications*, 2017.

[25] C. Fachkha, E. Bou-Harb, A. Keliris, N. Memon, and M. Ahamad, "Internet-scale probing of cps: Inference, characterization and orchestration analysis," in *NDSS*, 2017.

[26] V. Ghiette and C. Doerr, "How media reports trigger copycats: An analysis of the brewing of the largest packet storm to date," in *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*.

[27] "Zmap manual," https://github.com/zmap/zmap/wiki/UDP-Probe-Module, acccessed 12 August 2021.

[28] H. Griffioen and C. Doerr, "Discovering collaboration: Unveiling slow, distributed scanners based on common header field patterns," in *IEEE/IFIP NOMS*, 2020.

[29] D. R. Thomas, R. Clayton, and A. R. Beresford, "1000 days of udp amplification ddos attacks," in *IEEE/APWG eCrime*, 2017.

[30] G. V. Bard, "Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric," in *ACSW frontiers*. Citeseer, 2007.

[31] P. Mockapetris, "Domain names - implementation and specification," RFC 1035, 1987.

[32] "Sipvicious," https://github.com/EnableSecurity/sipvicious, acccessed 12 August 2021.

[33] J. Abley, O. Guomundsson, M. Majkowski, and E. Hunt, "Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY," RFC 8482, 2019.