Dmap: Automating Domain Name Ecosystem Measurements and Applications

Maarten Wullink, Giovane C. M. Moura, and Cristian Hesselman
SIDN Labs
Arnhem, The Netherlands
firstname.lastname@sidn.nl

***Abstract*—Behind each Internet domain name, there is a set of entities/companies responsible for delivering the various services associated with it, such as Web hosting and e-mail. Together, they form what we refer to as *DNS ecosystem*. Currently, there is no single measurement tool designed to measure this ecosystem altogether. As a result, researchers that aim at analyzing (parts of) this ecosystem often have to spend significant amounts of time preparing and executing the multiple application measurements and post-processing their heterogeneous raw datasets. Given that time is a scare resource, this complexity diverts researcher's time from actual analysis, ultimately limiting how far many studies go. To help researchers facing this situation, we present `Dmap`, an active measurement application that reduces the complexity of executing both measurements and analysis. It does so by (i) automating the crawling of several application protocols (DNS, HTTP, TLS/SSL, SMTP, both over IPv4 and IPv6) and (ii) storing the results into a relational data base, enabling researchers to quickly perform hypothesis tests within interactive response times using SQL. `Dmap` current version has 40 classifiers that generate 166 derived features (*e.g.,* CMS detection, page language), which can be used by researchers and operators to build applications and services. We present an evaluation of `Dmap` and show three applications that it can be used for, such as profiling the Alexa 1 million domains. We use `Dmap` at SIDN (.nl registry) for research on the .nl zone and make it open-source for researchers.**

I. Introduction

The Internet Domain Naming System (DNS) provides a global hierarchical naming space in which labels are used for hosts, services, applications, and networks on the Internet [1]. It comprises one of the core services on the Internet [2].

Associated with each domain name, there is a diverse set of entities/companies providing various services/functionalities, forming what we refer to in this paper as *DNS ecosystem*. To illustrate it, consider the domain wikipedia.org in Figure 1. To register a domain, there are typically three entities involved: registrant, registrar, and registry. A *registrant* (person or company) chooses a *registrar* (*e.g.,* GoDaddy) which is accredited by the top-level domain (TLD) *registry* (Public Interest Registry for .org) to register the domain (wikipedia.org) on behalf of the registrant. After that, a registrant sets the authoritative name servers [3] ({ns0,ns1,ns2}.wikimedia.org) to answer queries to the domain. Each of these authoritative name servers in turn can be run by different DNS providers (*e.g.,* Dyn, Amazon Route53) on different networks/locations.

After that, the registrant can associate diverse *services* with the domain, such as web, and e-mail (using DNS MX records [4]). Each of these services can also be provided by different companies (*e.g.,* GMail or Yahoo for e-mail).

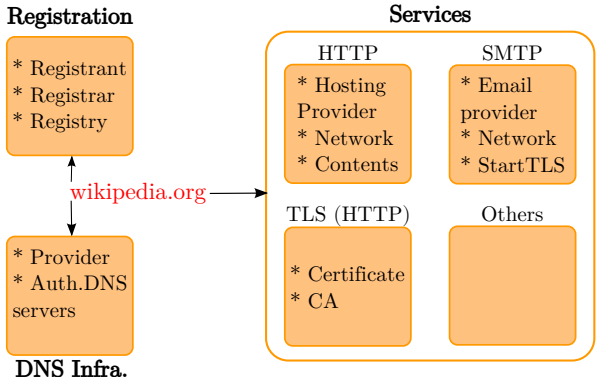

Fig. 1. Domain Name Ecosystem

This example illustrates the complexity of the DNS ecosystem. Currently, there is no open-source measurement tool that measures this ecosystem altogether. Instead, there is a set of of application-specific tools that produce heterogeneous datasets (§II). As a consequence, researchers often spend valuable time on (i) both coordinating measurements (due to heterogeneous tools) and (ii) preparing the collected data (handling different formats), instead of spending time on actual research questions. Ultimately, this complexity operates as veiled *limiting factor* in many measurements studies, by consuming time that otherwise could be spent on data analysis and hypothesis tests.

To help researchers and operators facing this scenario, we introduce *DNS Ecosystem Mapper* (`Dmap`, available at [5]), an open-source, scalable, modular, and distributed RESTFul web application [6] that *reduces the complexity* in both carrying the measurements related to the domain name ecosystem and the analyzing the collected data. It does that by (§III) (i) automating the crawling of various application layer protocols (DNS, HTTP, SMTP, and TLS, for both IPv4 and IPv6) for any given domain name and by employing a unified relational database model, which (ii) enables researchers to perform hypothesis tests using SQL syntax, obtaining results within interactive response times. Besides that, `Dmap` goes beyond current tools by enriching the raw measurements with *derived features*, which is done with the use of 40 classifiers that,

among others tasks, detect the language of a web page and determine if and which content management system (CMS) are deployed. Ultimately Dmap produces up to 166 features for a given domain name.

At SIDN (.nl top-level domain (TLD) operator), we employ Dmap to perform periodical full zone scans ($\sim$ 5.8 million domains) to support projects that focus both on improving security and stability of the DNS [7]. We assess its performance and precision using a production DNS zone (.nl, §IV).

By significantly reducing the complexity associated with the execution of measurements and data engineering, Dmap frees researchers to spend more time in the data analysis and focus on research questions. We show three applications of Dmap and show how data analysis can be easily done with SQL (§V). Finally, we make Dmap open for researchers [5].

II. Yet another scanner?

Over the last few years, the Internet measurement community has been benefiting from both fast scanners and open measurement data repositories. So one may wonder: do we need yet another scanner?

To answer this question, we first lay out the requirements for Dmap and then discuss how current tools/repositories do not fulfill them.

A. Requirements

The main requirement for Dmap is to reduce the complexity associated with both execution and analysis of measurements related to the domain name ecosystem (Figure 1). To achieve that, it needs to fulfill the following sub-requirements:

1) *Domain-centric*: Dmap should take as input a list of domains, as defined by its users, and not a list of IP addresses. It can used by perform scans on entire DNS zones or a subset of them.
2) *Automate the crawling of multiple protocols*: It should crawl various protocols (DNS, HTTP, SMTP, and TLS, currently), and generate a screenshot of the main page of a domain, if exists.
3) *IPv4 and IPv6 support*: it should measure actively the application protocols from #2 using both IPv4 and IPv6, and store the results separately.
4) *Completeness*: Dmap should aim for completeness of results, by having fail-safe mechanisms that allow for retrial of measurements in case of transient failures. Users should be able to set timeout and retry options for each measurement.
5) *Derived features using classifiers:* Dmap should produce derived features from raw datasets. Those features are those that requires extra processing, such as identification of CMSes and languages identification on Web sites, among others.
6) *SQL-based unified data model*: features from the raw measurement datasets should be extracted and then stored in unified SQL-based data model, in an external relational database management system (RDMBS). This model enables researchers and engineers to analyze large volume of data with interactive response times (seconds to minutes), which is essential for hypothesis tests and design of new applications [8], [9].
7) *Modular*: users should be able to control which application protocols should be measured (#2).
8) *Distributed*: Dmap architecture should be distributed for scalability.
9) *Open-source*: the tool should be open-source for researchers, so users have a complete understanding on how measurements are carried.

B. Current Tools and Services

Currently, there are various measurement tools that can be used to carry out Internet measurements. However, none of them carry multi-application measurements. As a consequence, users have to use distinct applications for different protocols, and some do not support IPv6. For example, Masscan [10] and Zmap [11] are tools that have decreased significantly the time required for perform IPv4 port scans, while ZMap subtools [12] addressed specific applications, such as HTTP and TLS.

While they have significant value, these problem with these tools is *fragmentation*: for each type of measurement, one needs to employ a specific tool, increasing the complexity. And each of them have of them have a distinct data format. As such, it does not fulfill sub-requirements #2 and #6.

Moreover, Masscan and ZMap are designed with a different goal: scan the entire IPv4 space. Dmap, on the other hand, is designed to scan domain names. The domain name space is different from the IP address space, and given the prevalence of web shared hosting [13] (in which many sites are hosted on the same servers), IP-wide scans cannot be used to enumerate (and measure) the domain name space, since such tools cannot determine which are the domains present in shared hosting services.

Other services such as DomainTools [14] provide similar features as Dmap. However, they are paid services and not open-source for researchers – violating requirement #9.

C. Public Data Repositories

Besides open measurement tools, Researchers have also available several large public measurement data repositories. CAIDA [15], USC/ISI's ANT [16], scans.io [17] and censys.io [18] are among the most popular ones.

Similarly to the existing tools, these datasets provide valuable data to the research community. However, they may not cover all cases. For example, at SIDN, we periodically scan our .nl zone at a frequency we find necessary. Other researchers, for example, may use Dmap to scan other publicly available zones [19] at their will. Thus, these repositories, of course, cannot cope with the user's specifics requirements, not fulfilling requirement #1. Besides, they also come in different data formats (#6) and may not support IPv6 (#3).

III. Dmap Design Principles

Different from most measurements tools, which are standalone, Dmap has a distributed client/server architecture in form

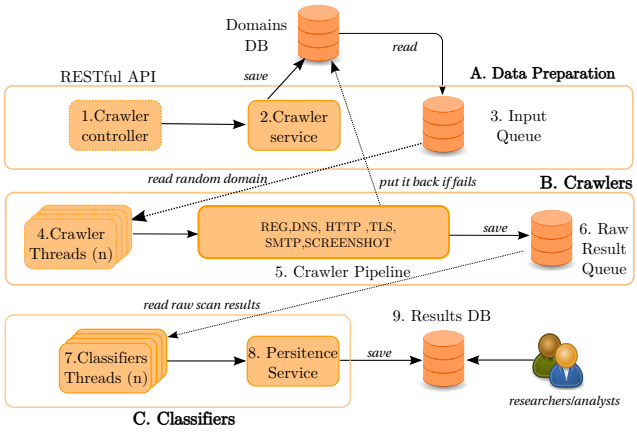

Fig. 2. `Dmap` Architecture

of a Representational state transfer (RESTFul) [6] web service developed in Java. uses Spring Boot [20], which is a Java framework that contains a series of libraries/tools that abstracts part of the complexity in writing RESTFul applications. With `Dmap`, users have full control of what, when, and what protocols to measure.

Besides that, `Dmap` uses other third-party libraries (*e.g.,* Maxmind GeoIP database [21] for annotating IP addreses, PhantomJS [22] for screenshots, the Java VM, and PostgreSQL [23] to store the final results). Even though `Dmap` has a distributed architecture, it can run on a single server as a `.jar` file.

Figure 2 shows the architecture of `Dmap`. It is divided in three main modules: Data preparation (A), Crawlers (B), and Classifiers (C). Next, we cover these in more details.

A. Data Preparation and Control (A)

The first module of `Dmap` consists of preparing the data to be used in the scans. Note that `Dmap` can also be used to scan individual domains using a web interface (also generating JSON output) but in this section we focus on how it performs scans in bulk, *i.e.,* on thousands or million domain names.

First, a user invokes the Crawler controller via an URI (step 1 on Figure 2) , using the RESTFul API. This triggers the Crawler service (2), which has two main functions: read the contents of a `csv` file containing the list of domains to be scanned and storing it into Domains DB, which is a PostgreSQL database.

Then, the same Crawler service reads n random domains from Domain DB and inserts these into the Input Queue (3), which in fact is a in-memory database used by the crawlers to start the scan. We employ Hazelcast [24] as our Input Queue, which is a lightweight open-source in-memory database.

The Crawler service also monitors the size of Input Queue as the scans take place, and once it drops below a configurable threshold, it then fetches more random domains from Domains DB and inserts then into Input Queue. We choose to place n random domains (*e.g.,*15,000) each time at Input Queue instead of all of them in order to `Dmap`'s reduce memory footprint and to minimize possible data loss during scans in case of application errors. This allow us to re-start the scan from where it stopped just by looking at the domain's states stored in DomainsDB (§III-D).

B. Crawlers Module (B)

The crawler module (B) of `Dmap` is responsible for carrying out the actual application level measurements. We first describe how they work in broad terms, and then cover the specifics of each crawler.

After storing n random domains in Input Queue (step 3), the application starts x Crawler Threads (step 4 in Figure 2, configurable by the user). Each thread then reads one domain from the Input Queue and executes the Crawler Pipeline (5) for the domain.

The Crawler Pipeline defines the sequences of measurements that will be executed. Currently, it is composed of ten stages: eight designed to measure application protocols (DNS, HTTP, TLS, and STMP), each of them carrying out measurements over IPv4 and IPv6 (thus one stage per application/IP stack version); another stage used to capture a screenshot of a website and one stage (REG) to retrieve domain name registration information (typically internal to TLD registries, but it could be extended to support third-party APIs).

Each individual crawler of the pipeline can also be optionally activated/deactivate for each measurement. For example, a user interested only in DNS information can disable all other crawlers, thus reducing the total measurement time.

`Dmap` keeps the state of each measurement. To do that, its Crawler Service (2) keeps in DomainsDB metadata that indicate which domains have been crawled, which need to be retried, which protocols need to still to be crawled.

After performing its measurements, each crawler in the pipeline stores its raw measurement results into the the Raw Result Queue (step 6), which is a Java queue implementation. We refer to this queue as "raw" since it contains different data formats, such as html (for HTTP measurements). By default, `Dmap` does not store the raw measurement results – only results as specified in our data model (§III-F) (we are currently working on implementing a module to export this raw datasets, so it can be further analyzed using other solutions such as Elasticsearch [25]).

C. Crawlers Details

We have developed each crawler of the pipeline from scratch, using a series of external third-party libraries. The reasons for doing this is that it allows us to have full control over the measurement data and processing, enabling us to extract whatever features we needed. Next we discuss each crawler.

1) `DNS` crawler: the DNS is used to store various types of records. For example, A [1] and AAAA [26] records are used to stores IP addresses (IPv4 and IPv6 respectively, the "classic" name to IP mapping). However, there are other record types too: MX records [4] specify mail exchange for the domain (*e.g.,* smtp.example.nl), while SOA specifies the start

of the zone authority. TXT records [4], on the other hand, hold descriptive data, and are use in DMARC [27] and DKIM [28] protocols. Our crawler downloads all these records, which are later processed by classifiers that annotate them with other features.

2) `HTTP` crawler: we use Apache's HttpClient [29] library, which provides an implementation of the HTTP client protocol. We have implemented, however, the logic behind the crawler, and handled a series of issues. For example, if a domain redirects to another domain or subdomain, our crawler will follow to the redirect page (up to a certain number of redirects configured by the user). Also it is able to handle many cornercases such as websites that continue streaming large amounts of data, by using timeouts and maximum bytes allowed to download.

3) `TLS` crawler: The crawler attempts to establish a HTTPS connection to the domain name, and download its X.509 certificate [30] (which is not part of the protocol). We disable all Java security controls, to prevent that exceptions are raised occur during this phase, so we are able to determine if there is an error with a certificate, what error it is (expired, self-signed, etc.).

4) `SMTP` crawler: the SMTP crawler crawls all configured mail servers (given by MX records) and verifies if they supports StartTLS (opportunistic TLS) [31]. The crawler will try to create a connection to each mail server IP address (over both IPv4 and IPv6).

5) `Screenshot` crawler: it creates a screenshot of the main web page and uses PhantomJS for that. This is a relatively expensive operation because all the website resources (*e.g.,* HTML, JavaScript, Images) must be download and then the page has to be rendered. Therefore, we disable this crawler by default to speed up large zone scans. However, users can activate as they wish.

6) `REG` crawler: we use it internally at our .nl registry, and we retrieve information such as registration date and registrar. Our open-source version has these features disabled, since they are registry only. This crawler could be extended to connect to third-party APIs or `whois` services to extract more features.

D. Fail-safe Mechanisms

Measurements results may be affected by transient errors [32], such as connectivity issues and packet loss. To handle such errors, we have built into `Dmap` various fail-safe mechanisms.

First, for each domain d, we keep in the DomainsDB metadata about the stage in which the domain is in the pipeline. If one of the crawlers in the pipeline fails, `Dmap` stores extra information associated with that – such as when it failed, and the respective error. It also keeps a counter for number of retries and a time for the next time it should be retried (both parameters configurable by the user).

We also handle different exceptions for each type of crawler. For example, for the `DNS` crawler, we do not retry if a domain name resolution was answered with an NXDOMAIN [1], which indicates the requested domain does not exist. If there is a timeout, however, `Dmap` retries again to crawl the domain after a configurate time interval.

TABLE I
SIX OUT OF THIRTY `DMAP` CLASSIFIERS AND FEATURES

Classifier	Features
TLSCert	extract X.509 cert. info and type of cert
ServerClass	OS and Web server fingerprint
Parking	if a domain is parked and/or for sale
CMSClass	CMS, shopping cart and forum detection
	if a domain has a registrar placeholder page
Business	type of site (online shop)
LetsEncrypt	if domain uses Let's Encrypt Certs

E. Classifiers Module (C)

Classifiers have two roles on `Dmap`: extract the required features from the raw measurement data and enrich this data with other derived features. Such derived features is one of the biggest differences between `Dmap` and other tools: it generates by default a richer set of features that would require researchers to manually implement them.

Table I shows 6 of these classifiers and their respective features. For example, ServerClass attempts to perform OS and Web server identification, while TLSCert reads the X.509 certificates, extract its type, CA, among others. Many domains are also "parked" [33], *i.e.,* a domain that is not developed and usually used in automatically generated advertisement.

The CMS classifier attempts to identify a content management system (CMS), such as Joomla, is in use on a website, besides detecting the presence of forums or shopping carts (used to detect online stores). To do that, we have manually inspected various CMSes and produced matching patterns for 42 of them. For example, to identify Wordpress, we look into the `generator` tag found in the HTML code of the main page and perform substring match with `Wordpress` and `WPML`. In addition, we also search for internal links with `wp-content` and `wp-includes` on their paths (these matching patterns need to be updated as new versions of CMSes are released).

Table II shows 65 features produced by HTTP classifiers, from the raw HTML code and its metadata. Features such as `httpStatus`, total number of links and internal links (`htmlLinksInt`, `htmlLinksAll`), and site language (`pageLan`) can be directly used by researchers to build classifiers to detect, for example, malicious web sites.

Currently, there are *40 classifiers* on `Dmap`. Each classifier, in turn, is associated to one crawler type only. For example, the Parking classifier (Table IV) process the raw html data from the HTTP cralwer. Users can also develop new classifiers or modify existing ones. Users cannot, however, disable individual features – they can either enable or disable crawlers (§III-C), but once a crawler is activated, their respective classifiers will generate the entire set of features.

Ultimately, the classifiers store the results into Results DB (9), an external database upon which analysts can carry out their hypothesis tests and build other applications We cur-

TABLE II
65 HTTP IPV4 FEATURES PRODUCED BY DMAP FOR EN.WIKIPEDIA.ORG.

```
domainname:"wikipedia.org",
crawlRun:764,
ipVersion:4,
url:null,
crawlName:"www.wikipedia.org",
crawlUrl:"https://www.
wikipedia.org/",
crawlDomain:"wikipedia.org",
crawlDate:"2018-05-15
T09:28:48.642+0000",
crawlStatus:0,
crawlPages:2,
crawlRetries:0,
networkLoadTime:24,
networkLoadTimeAll:[
{
url:"http://www.wikipedia.org",
type:"index",
time:24,
status:"OK"},
{ url:"https://www.wikipedia.
org/",
type:"redirect",
time:46,
status:"OK"}],
httpStatus:200,
httpBytesLen:75232,
htmlTitle:"Wikipedia",
htmlDescription:"Wikipedia is
a free online encyclopedia,
created and edited by
volunteers around the world and
hosted by the
Wikimedia Foundation.",
htmlKeywords:null,
serverEngine:null,
serverEngineVersion:null,
pageDefault:false,
pageSuspended:false,
pageLang:"mul",
pageLangMulti:false,
pageLangProb:100,
pageFingerprint: (ommit)
httpRedirect:false,
httpRedirectCount:1,
httpRedirectChain:"www.wiki
pedia.org,www.wikipedia.org",
httpRedirectHttps:true,
httpRedirectTld:false,
tldStart:"org",
tldEnd:"org",
htmlLinksAll:320,
htmlLinksInt:0,
htmlLinksExt:320,
htmlLinksImg:0,
htmlVersion:"HTML 5",
htmlFrameCount:0,
htmlSpiderBlocked:false,
appCms:null,
appCmsOther:null,
appForum:null,
appShoppingCart:null,
pageType:"Content",
parkingProvider:null,
pagePlaceholder:false,
serverOs:null,
statsWordCount:619,
trustMark:null,
secHsts:"max-age=106384710;
includeSubDomains; preload",
secPubKeyPin:null,
secPubKeyPinReport:null,
secContentSecPol:null,
cookiesCount:3,
cookiesPersistentCount:2,
privacyPolicyAvail:false,
businessCocNo:null,
businessVatNo:null,
businessBankNo:null,
businessBicNo:null,
businessPhoneNo:null,
businessAddress:null
```

rently implement ResultsDB using PostgreSQL, but for larger datasets, one can export this data to other Big data databases, that support SQL syntax, such as Impala [34] (we use Impala to analyze authoritative DNS traffic in ENTRADA[35], another open-source tool we developed).

F. Data Model

Dmap uses a SQL-based data model with 44 tables. The 166 features produced per every single input domain are distributed over six result tables – one per service: DNS (30, which 4 are nested JSON fields); HTTP (65, 2 are nested); SMTP (11, of which 1 is nested); TLS (37); REG (8) and Screenshot (15). Nested features may have a variable number of subfeatures, as `networkLoadTimeAll` in Table II. The other tables are used to storate metadata associated with the crawling (status, error, for example). Due to space constraints, we omit the complete data model here and make it available at [5].

G. Ethical and legal considerations:

As any active Internet measurement tool, Dmap users should evaluate the ethical implications of their measurements [36] and the impact it may have on the measured systems, as well as managing the collected data. We do not cover the legal issues

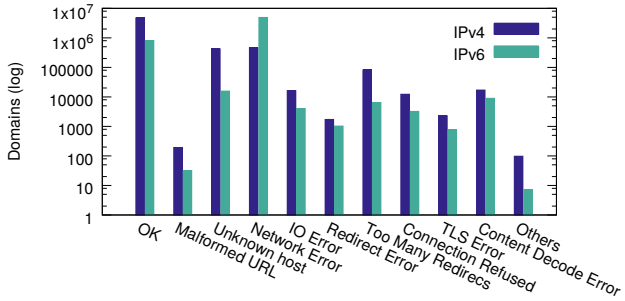

Fig. 3. HTTP crawler result codes

in this paper, but we have developed, together with our legal department, a publicly available data privacy framework [37] that conforms to both EU and Dutch laws. This framework has been implemented, including a privacy board that oversees SIDN Labs research.

IV. EVALUATION

To evaluate Dmap , we have carried out a full zone on our .nl TLD zone on September 3rd, 2017 from our corporate network. In total, 5,766,118 domains were used as input. We ran it on a single virtual machine with 8x 1.4GHz CPU cores with 32GB of RAM, and 1 Gbps shared line. We activated 6 crawlers (screenshot was disabled), with 300 Crawler threads and 50 Classifier threads (Figure 2).

A. Precision/Completeness

To determine how precise and complete Dmap is, we compare its results for DNS services against OpenIntel [38], a project that daily perform DNS scans of various DNS zones and that uses a different vantage point from ours.

On September 3rd, 2017, after scanning all .nl domains for DNS records, OpenIntel listed 913,511 domains that had a AAAA record associated to it, *i.e.,*a IPv6 address (∼15.8% of the total – not every domain needs to have an IPv6 AAAA record). Dmap, in turn, found that 913,593 .nl domains had AAAA records – a very similar figure. The small difference (82 domains) can be explained by transient changes in these domains setups while being measured.

For DNS zones, such as .nl, we can expect that not all domains have a Web page (some not used for Web, others not depeloped yet). We then show the results for the HTTP crawler over the .nl zone. As discussed in §III-B, this crawler downloads the main page associated with a domain (obtained from its A or AAAA records). Figure 3 shows the results of the HTTP crawlers. Out of 5,766,118 domains, 4,756,943 (82.49%) had an active Web page on IPv4 (OK on x axis), and 811,425 had an IPv6 page (14.07%) (note that we found that 914,593 domains had an IPv6 AAAA record associated to it, but only 811,425 had a Web server running on this address).

Fail-safe mechanism: as discussed in §III-B, we have built a fail-safe mechanism in each crawler. To evaluate its behavior, we analyze how many tries the HTTP-crawler had perform

before it could connect to the domains with available pages. We found that only 40 domains for IPv6 required at least one connection retrial (after the first failed), and only 38 for IPv4. After that, if they did not responded, we collected the error codes shown in Figure 3. As such, most connections either work for the first time or not worked at all.

B. Performance

Performance is an important feature of any Internet measurement application. In our single VM setup, `Dmap` crawled, on average, 11.34 domain per second (980,000/day), with all crawlers activated except for the Screenshot crawler. However, `Dmap` is designed to scale horizontally in terms of performance, *i.e.,* adding more crawling nodes (part B and C on Figure 2) can linearly improve the performance whenever needed. Also, depending on the user's needs, some crawlers from the pipeline can be disabled, thus reducing the total measurement time.

C. Comparison with other tools

Given the lack of comparable open-source tools that produce the same number of features (§II), it is hard to draw a direct comparison with `Dmap`.

Still, compared to other stateless IPv4 scanners such as Masscan and ZMap (which produces far fewer features), `Dmap` performance may seem humble. However, there are reasons for that: first, the number of features (up to 166) generated, which required 5 crawlers that performed tasks such as resolving domain names, establishing TLS sessions, retrieving certificates, HTTP connections over both IPv4 and IPv6, and run a series of classifiers on the data. Therefore, there is intrinsic a trade-off between number of features and performance.

This performance difference between IPv4 scanners and `Dmap` is less critical also given the differences in scan space. For example, for second-level domains (*e.g.,* example.nl and not portal.example.nl), the .com zone, which is the biggest, has $\sim$ 130 million second-level domain names. Comparatively, the IPv4 address space has more than 4 billion addresses and the IPv6 has 3.4×10^{38} addresses.

V. Applications

In this section, we present three applications for which we have used `Dmap`. We do not intend to present a comprehensive list of applications; rather we show how easy `Dmap` can be used for different use cases.

A. Profiling Alexa 1 Million

The Alexa 1 million domains list which ranks sites based on their toolbar users [39], [40]. We crawled this list of domains with `Dmap` on February 21st, 2018 and present here a descriptive analysis of the results obtained. We also make available the resulting dataset (PostgreSQL database) and SQL code for this analysis available in [5].

Table III shows a profile of the Alexa 1M domains divided by protocol, for both IPv4 and IPv6. Let's start with the DNS protocol: the row *# Domains OK* show the number of domains that had at least one of its authoritative serves (defined by NS records in DNS) with IPv6 (defined by a AAAA record associated to a NS record).

TABLE III
Alexa 1 million profiling

	IPv4	IPv6	IPv6/IPv4
DNS			
# Domains (OK)	972,155	153,485	0.16
# Unique NSes	289,014	26,127	0.09
# Unique IP	210,650	19,754	0.09
# Unique ASes	18,418	3,178	0.17
# CDN Cloudflare	117,538	115,396	0.98
HTTP			
# Domains (OK)	968,338	153,485	0.16
# HTML 5	681,757	116,066	0.17
Bytes (median)	53,889	64,735	1.20
External links (median)	7	8	1.14
Internal links (median)	67	75	1.12
Cookies (median)	1	1	1.00
TLS			
# Domains (OK)	772,455	129,443	0.17
# Let's Encrypt	165,526	10,466	0.06
SMTP			
# Domains (OK)	843,126	190,736	0.23
# Unique SMTP	501,848	24,311	0.05
# Unique IP	286,504	10,113	0.04
# Unique StartTLS	302,871	8,016	0.03

Then, we see that $\sim$289K authoritative servers (NS records, such as ns.google.com) were shared among 1M domains. These authoritative, in turned, were mapped to $\sim$210K IPv4 addresses, distributed over $\sim$18K Autonomous Systems (ASes). We select two findings from this data:

IPv6 adoption is slightly faster on SMTP: the number of domains that support IPv6 is higher for e-mail servers (0.23) than other services (0.16–0.17). We can see that one DNS Provider (Cloudflare) accounts for $\sim$117K domains (IPv4), or 12.18% of all domains crawled. Besides, we see that $\sim$60% of the unique SMTP servers support StartTLS [41], which is an extension to SSL/TLS to enable encrypted communications over a plain-text connection.

77.2% of domains deploy Web encryption (IPv4), and 1 in 5 now use Let's Encrypt: To enable web encryption, a Web site needs an X.509 certificate [30], which is issued by a Certificate Authorities (CAs). We found that 77.2% of all domains now have Web encryption over IPv4. Among the available CAs, Let's Encrypt [42] foundation has drawn significant attention over the last years since it was the first CA to provide both free X.509 certificates and automated software for that. In a previous study, we evaluated the first year of Let's Encrypt in terms of certificate *issuance* [43] for all domains and found that Let's Encrypt grew very fast in the first year. Now, we can see how many domains of the 1M list deploy Let's encrypt certificates on second-level domain. As can be seen, for IPv4, 21.4% of domains that support SSL/TLS over HTTP employ Let's Encrypt.

TABLE IV
TLS/SSL WEB DEPLOYMENT ON .NL ZONE OVER IPV4

	20170903	20180201
Zone size	5,766,118	5,801,191
TLS/SSL	2,595,281 (45.10%)	2,674,877 (46.10%)
Types of Certificates		
Unknown	382 (0.01%)	407 (0.01%)
self-signed	883,844 (34.05%)	749,033 (28.01%)
DV	1,406,072 (54.17%)	1,618,575 (60.5%)
OV	269,284 (10.37%)	272,318 (10.18%)
EV	35,699 (1.37%)	34,544 (1.29%)
Let's Encrypt DV deployed certificates share		
Let's Encrypt (%DV)	366,623 (14.12%)	523,029 (32.31%)

B. Longitudinal Studies on Web Encryption Adoption

Dmap can be used to crawl periodically the same domains and support longitudinal studies of changes in a domain name set. In this section, we show how it enables to characterize and determine how much Web encryption adoption has changed for the .nl zone.

With other tools, this rather straightforward question may take researchers significant amounts of time to both measure and analyze it. With Dmap, a researcher can used the SQL query showed in Listing 1 to retrieve the results. In our setup, it took 8s to run this query.

```
1 select count(domainname) from crawl_result_tls where
2 crawl_result_tls.tls_avail=true and
3 crawl_result_tls.crawl_run=$ID
4 and tls_https_status=200
```

Listing 1. SQL query to analyze web encryption on the .nl zone.

We choose two dates roughly 5 months apart and compare how Web encryption adoption has changed in the .nl zone: Sept. 3rd, 2017, in which there were 5.7 million in the zone file and on, Feb. 2nd, 2018, with 5.8 million domains. Table IV shows the results. In this five months, the .nl zone went from 45% to 46% of all domains with enabled Web encryption.

X.509 certificates used for Web encryption are usually offered in three types by CAs: domain validated (DV), organization validated (OV), and extended validation (EV). They all employ the same encryption measures — they differ on how the CA verifies the user's identity (e.g. if the user is the legal owner of the domain). Using a similar query to Listing 1, we can determine the type of X.509 certificates [30] employed by .nl domains. Table IV shows the results. As can be seen, there has been a drop on the number of self-singed certificates, followed by an increase in number of DV certificates (OV and EV remained relatively stable).

We can use Dmap to determine what is the market size of Let's Encrypt CA on the entire .nl zone. Table IV show these numbers. In this period, more than 150,000 sites obtained a certificate from Let's Encrypt, being currently responsible for 32.31% of all DV certificates by February 2018 (it only provides DV certificates), from 14.12% in September 2017. In total, ∼9% of all .nl domains employ Let's Encrypt, while 16.5% of the Alexa 1M employ it (Table III).

TABLE V
CMS AND WEB SERVER USAGE .NL ZONE (20170903)

CMS	Total (%)	Web Server	Total (%)
Wordpress	848,083 (73.84%)	Apache	3,397,930 (77.05%)
Joomla	99,865 (8.69%)	nginx	696,697 (15.79%)
Drupal	40,798 (3.55%)	MS ISS	249,367 (5.65%)
Blogo	20,749 (1.80%)	cloud-nginx	60,530 (1.13%)
Wix	17,861 (1.53%)	Coyote	4,237 (0.01%)
Others	121,232 (10.43%)	Others	905 (0.00%)
Total	**1,148,408**	**Total**	**4,409,666**

C. CMS and Web server Usage

CMS, such as Wordpress and Joomla, are popular solutions for managing the content of websites. They provide an abstraction layer in which the content creation is separated from the website design and coding. Due to their popularity, vulnerabilities in one specific version of a CMS can be re-used to exploit in other sites that run the same CMS version. Thus, CMS presence has been positively correlate with webserver compromise [44], [13]. Therefore, we have developed a classifier that attempts to identify if websites are running CMSes as well as web server software (§III-E).

Table V shows the results for .nl domains for the 20170903 zone scan. WordPress is the most popular CMS, and Apache is the most popular web server. Dmap can help researchers and security experts in identifying vulnerable systems, which can be notified in order to mitigate potential attacks.

VI. RELATED WORK

Dmap is the first open-source active measurements application that focus on crawling various applications, extracting a series of features from the raw datasets and producing a derived features with its classifiers, and storing the results in a relational data base. We have previously discussed existing measurement tools in §II.

There are other works that are similar but not directly related. For example, OpenIntel [38] focus on daily crawls of DNS records of multiple zones (thus comparable to our DNS crawler – §III-C). However,it does not crawl applications other than DNS. Dmap , besides DNS, also crawls HTTP, TLS, SMTP protocols as well, enriching the raw data with a series of derived features with the aid of its classifiers.

Another similar work is our previous open-source ENTRADA, which is data streaming warehousing for authoritative DNS traffic. However, ENTRADA is designed to collect and store passive measurement data, while Dmap is an active measurement tool. However, both tools provide a SQL interface for data analysis.

VII. CONCLUSION AND FUTURE WORK

Internet measurements can be complex and demand significant efforts in planning, executing, and analyzing the collected data [32]. All this "heavy lifting" must be carefully executed before a researcher can begin to answer their research questions. This is still the case for many researchers, especially when public data repositories and tools do not fulfill the

researchers' requirements. As such, this complexity is in fact a limiting factor for many studies.

In this paper, we presented Dmap, a measurement tool which its main goal is to *reduce the complexity* in both carrying measurements and analyzing the collected data by automating multi-application measurements and data preparation. Researchers (and DNS operators), thus, can profit from Dmap by spending their valuable time in hypothesis tests and data analysis. We make Dmap source-code and binaries available for researchers [5]. As future work, we intend to add other crawlers to Dmap, so new application layer protocols can be supported and develop new classifier to produce more derived features.

Overall, we hope Dmap can be useful for researchers and operators to build applications that help to improve both security and stability of global DNS.

Acknowledgments: We would like to thank Moritz Müller and the anonymous TMA reviewers for their valuable comments on paper drafts.

REFERENCES

[1] Mockapetris, P.: Domain names - concepts and facilities. RFC 1034 (1987)
[2] Moura, G.C.M., de O. Schmidt, R., Heidemann, J., de Vries, W.B., Müller, M., Wei, L., Hesselman, C.: Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In: Proceedings of the 2016 ACM Conference on Internet Measurement Conference. (October 2016) 255–270
[3] Hoffman, P., Sullivan, A., Fujiwara, K.: DNS Terminology. RFC 7719 (December 2015)
[4] Mockapetris, P.: Domain names - implementation and specification. RFC 1035 (November 1987)
[5] SIDN Labs: Dmap (May 2018) . https://dmap.sidnlabs.nl/.
[6] Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis (2000) University of California, Irvine.
[7] Hesselman, C., Moura, G.C.M., d. O. Schmidt, R., Toet, C.: Increasing DNS Security and Stability through a Control Plane for Top-Level Domain Operators. IEEE Communications Magazine **55**(1) (January 2017) 197–203
[8] Melnik, S., Gubarev, A., Long, J.J., Romer, G., Shivakumar, S., Tolton, M., Vassilakis, T.: Dremel: Interactive Analysis of Web-Scale Datasets. In: Proc. of the 36th Int'l Conf on Very Large Data Bases. (2010) 330–339
[9] Wullink, M., Moura, G.C., Müller, M., Hesselman, C.: Entrada: A high-performance network traffic data streaming warehouse. In: Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP, IEEE (April 2016)
[10] Graham, R.D.: Masscan: TCP port scanner, spews SYN packets asynchronously, scanning entire Internet in under 5 minutes. https://github.com/robertdavidgraham/masscan (February 2018)
[11] Durumeric, Z., Wustrow, E., Halderman, J.A.: Zmap: Fast internet-wide scanning and its security applications. In: Proceedings of the 22nd USENIX Security Symposium. (2013)
[12] ZMap: The ZMap Project (September 2017) . https://zmap.io.
[13] Tajalizadehkhoob, S., van Goethem, T., Korczyński, M., Noroozian, A., Böhme, R., Moore, T., Joosen, W., van Eeten, M.: Herding Vulnerable Cats: a Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. CCS '17, New York, NY, USA, ACM (2017) 553–567
[14] DomainTools: Threat Intelligence, Threat Hunting, Incident Response, Whois: DomainTools. https://www.domaintools.com/ (February 2018)
[15] CAIDA: CAIDA Data - Overview of Datasets, Monitors, and Reports. http://www.caida.org/data/overview/ (February 2018)
[16] ISI/USC: ANT Datasets. https://ant.isi.edu/datasets/index.html (February 2018)
[17] Team, C.: Internet-Wide Scan Data Repository. https://scans.io/ (February 2018)
[18] Censys: Security Driven by Data. https://censys.io/ (February 2018)
[19] ICANN: Centralized Zone Data Service. https://czds.icann.org/en (February 2018)
[20] Spring: Spring Boot (February 2018) https://projects.spring.io/spring-boot/.
[21] Maxmind: GeoIP2 City Database Demo (February 2018) https://dev.maxmind.com/geoip/geoipupdate/.
[22] PhantomJS: PhantomJS (September 2017) . http://phantomjs.org/.
[23] PostgreSQL: The world's most adavanced open source database (February 2018) https://www.postgresql.org/.
[24] Hazelcast: Hazelcast IMDG - the leading In-memory Data Grid (February 2018) https://hazelcast.org/.
[25] Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide. 1st edn. O'Reilly Media, Inc. (2015)
[26] Thomson, S., Huitema, C., Ksinant, V., Souissi, M.: DNS Extensions to Support IP Version 6. RFC 3596 (October 2003)
[27] Binet, D., Boucadair, M., Vizdal, A., Chen, G., Heatley, N., Chandler, R., Michaud, D., Lopez, D., Haeffner, W.: An IPv6 Profile for 3GPP Mobile Devices. RFC 7849 (May 2016)
[28] Crocker, D., Hansen, T., Kucherawy, M.: DomainKeys Identified Mail (DKIM) Signatures. RFC 6376 (September 2011)
[29] Apache: Apache HttpComponents (February 2018) https://hc.apache.org/.
[30] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard) (May 2008) Updated by RFC 6818.
[31] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (August 2008)
[32] Paxson, V.: Strategies for Sound Internet Measurement. In: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement. IMC '04, New York, NY, USA, ACM (2004) 263–271
[33] Vissers, T., Joosen, W., Nikiforakis, N.: Parking sensors: Analyzing and detecting parked domains. In: Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015), Internet Society (2015) 53–53
[34] Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Erickson, J., Grund, M., Hecht, D., Jacobs, M., et al.: Impala: A modern, open-source SQL engine for Hadoop. In: Proceedings of the Conference on Innovative Data Systems Research (CIDR'15). (2015)
[35] Wullink, M., Moura, G.C.M., Müller, M., Hesselman, C.: ENTRADA: A high-performance network traffic data streaming warehouse. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. (April 2016)
[36] Partridge, C., Allman, M.: Ethical considerations in network measurement papers. Communications of the ACM **59**(10) (2016) 58–64
[37] C. Hesselman, J. Jansen, M. Wullink, K. Vink, and M. Simon: A privacy framework for DNS big data applications. Technical report (2015) https://www.sidnlabs.nl/uploads/tx_sidnpublications/SIDN_Labs_Privacyraamwerk_Position_Paper_V1.4_ENG.pdf.
[38] van Rijswijk-Deij, R., Jonker, M., Sperotto, A., Pras, A.: A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements. IEEE Journal on Selected Areas in Communications **34**(6) (June 2016) 1877–1888
[39] Alexa: Keyword Research, Competitive Analysis, & website ranking. https://www.alexa.com (February 2018)
[40] Englehardt, S., Narayanan, A.: Online Tracking: A 1-million-site Measurement and Analysis, booktitle = Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, New York, NY, USA, ACM (2016) 1388–1401
[41] Hoffman, P.: SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207 (February 2002)
[42] Let's Encrypt: Free SSL/TLS Certificates. https://letsencrypt.org/
[43] Aertsen, M., Korczyński, M., Moura, G.C.M., Tajalizadehkhoob, S., van den Berg, J.: No Domain Left Behind: Is Let's Encrypt Democratizing Encryption? In: Proceedings of the Applied Networking Research Workshop. ANRW '17, New York, NY, USA, ACM (2017) 48–54
[44] M. Vasek and J. Wadleigh and T. Moore: Hacking is not random: A case-control study of webserver-compromise risk. IEEE Transactions on Dependable and Secure Computing **13**(2) (March 2016) 206–219