2016

All Your DNS Records Point to Us Understanding the Security Threats of Dangling DNS Records

Daiping Liu
Univ Delaware, Newark, DE 19716 USA;

Shuai Hao

Haining Wang

Shuai Hao
William & Mary

Recommended Citation

Liu, Daiping; Hao, Shuai; Wang, Haining; and Hao, Shuai, All Your DNS Records Point to Us Understanding the Security Threats of Dangling DNS Records (2016). *Ccs'16: Proceedings of the 2016 ACM Sigsac Conference on Computer and Communications Security*.
10.1145/2976749.2978387

All Your DNS Records Point to Us

Understanding the Security Threats of Dangling DNS Records

Daiping Liu*, Shuai Hao*†, and Haining Wang*

*University of Delaware
Newark, DE 19716, USA
{dpliu, hnw}@udel.edu

†College of William and Mary
Williamsburg, VA 23187, USA
haos@cs.wm.edu

ABSTRACT

In a dangling DNS record (Dare), the resources pointed to by the DNS record are invalid, but the record itself has not yet been purged from DNS. In this paper, we shed light on a largely overlooked threat in DNS posed by dangling DNS records. Our work reveals that Dare can be easily manipulated by adversaries for domain hijacking. In particular, we identify three attack vectors that an adversary can harness to exploit Dares. In a large-scale measurement study, we uncover 467 exploitable Dares in 277 Alexa top 10,000 domains and 52 `edu` zones, showing that Dare is a real, prevalent threat. By exploiting these Dares, an adversary can take full control of the (sub)domains and can even have them signed with a Certificate Authority (CA). It is evident that the underlying cause of exploitable Dares is the lack of authenticity checking for the resources to which that DNS record points. We then propose three defense mechanisms to effectively mitigate Dares with little human effort.

Keywords

DNS; Dangling records; Domain hijacking

1. INTRODUCTION

As one of the most critical components of the Internet, the Domain Name System (DNS) provides not only vital naming services but also fundamental trust anchors for accessing Internet services. Therefore, it has always been an attractive target to attackers [28], [42], [43]. In order to ensure the authenticity and integrity of DNS systems, tremendous efforts have been devoted to protecting both client and server mechanisms [30], [32], [52], [55]. In particular, a suite of security mechanisms like DNSSEC [27] have been deployed to secure the communication channels between DNS servers and clients. However, little attention has been paid to authenticating the links between DNS servers and those resources to which DNS records point.

New Threat. In this paper, we investigate a largely overlooked threat in DNS: a dangling DNS record (Dare), which could be easily exploited for domain hijacking due to the lack of authenticity checking of the resolved resources. A DNS record, represented in a tuple `<name, TTL, class, type, data>`, is essentially a pointer, where the `data` field points to the machine that hosts the resources for the `name` field. Similar to pointers in a program, a DNS record can also become dangling. When a service accessed by the `name` field discontinues, the domain owner will release the machine to which the `data` field points and should also purge the related DNS records. Unfortunately, in practice, domain owners often forget to do the cleaning, thus resulting in dangling DNS records. Conventional wisdom holds that Dare is by and large safe. To better understand this threat, we conduct the first comprehensive study on exploitable Dares (*unsafe Dares*) in the wild. In particular, our work reveals that Dare is a real, prevalent threat.

We initiate our study by scrutinizing the DNS specifications, during which four types of security-sensitive Dares are identified, including Dare-A, Dare-CN, Dare-MX, and Dare-NS. To exploit unsafe Dares, an adversary needs to gain control of the resources in the `data` fields of DNS records. There are two types of resources in Dares: IP addresses and domain names. We present three attack vectors that an adversary can harness to hijack these resources. (1) In the first attack vector, we observe that cloud platforms have become a popular choice for modern websites. In clouds, physical resources, especially the public IP address pool, are shared among all customers. Unfortunately, in practice, many domain administrators mistakenly trust these ephemeral and publicly allocable resources, potentially generating all types of Dares. In a sense, this attack vector is probability-based since the IP allocation in clouds is generally random. (2) Modern websites extensively use third-party services. To integrate a third-party service into a website, a domain owner needs to add an `A` or `CNAME` record in the authoritative DNS (aDNS) servers and claim the ownership of the (sub)domain on the owner's third-party service account. Any service account that successfully claims ownership of a (sub)domain can control the content of that (sub)domain. Surprisingly, most third-party services do not verify such a claim, implying that an adversary can potentially claim and control any (sub)domain that has been abandoned by its original owner. The second attack vector is thus to hunt for the Dares linked to abandoned third-party services. (3) Since a domain can expire [46], the third attack vector is simply to search for the expired domains in the `data` fields of DNS records.

Large-scale measurement study. Given the three attack vectors, we then assess the magnitude of the unsafe Dares in the wild. We conduct a large-scale measurement on four datasets, one containing the apex domains in the Alexa top 1 million spanning seven years and the other three containing the subdomains in the Alexa top 10,000, 2,700 `edu`, and 1,700 `gov` zones, respectively. For the first attack vector, we develop a simple tool called IPScouter to automatically milk IP addresses in the clouds, especially the two largest clouds, Amazon EC2 [2] and Microsoft Azure [17]. Due

CCS'16, October 24-28, 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2976749.2978387

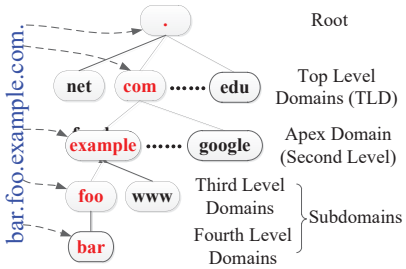


Figure 1: The hierarchy of DNS.

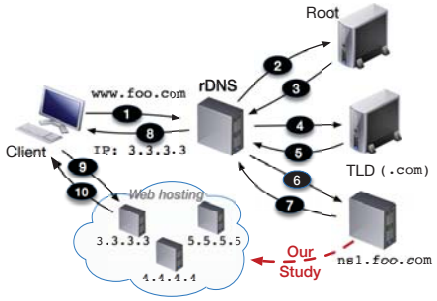


Figure 2: The workflow of DNS resolution for `www.foo.com`.

to its probabilistic nature, IPScouter cannot enumerate the whole IP address space. We therefore assess the magnitude of *potential dares* by filtering all live IP addresses. For the second attack vector, nine of the most popular third-party services are measured. For the third attack vector, we crosscheck the `WHOIS` data and the domain registrars to identify these expired domains.

In total, 791 confirmed and 5,982 potential Dares are successfully found in our measurement study. Especially, Dares exist in all four datasets, indicating a widespread threat. Even more worrisome, Dares can be found in 335 high-valued zones, including those in `edu`, `gov`, and Alexa top 10,000. By exploiting these Dares, an adversary can significantly enhance many forms of fraud activities (e.g., spamming, spear phishing, and cookie hijacking). With the emergence of automated and free Certificate Authorities (CA) like `Let's Encrypt` [15], adversaries can even have the hacked subdomains signed and set up a "genuine" HTTPS website.

Mitigations. We posit that the fundamental cause of unsafe Dares is *the lack of authenticity checking* of the ephemeral resources to which DNS records point. We thus propose three mechanisms that DNS servers and third-party services can adopt to mitigate unsafe Dares. (1) We first design a mechanism that allows aDNS servers to authenticate these machines to which `A` records point. (2) In the case of third-party services, we propose breaking the resolution chain of the dangling `CNAME` records by adopting a safer isolated name space for each user of a service. (3) Finally, we advocate that aDNS servers should periodically check the expiration of domains to which DNS records point.

Roadmap. The remainder of this paper is organized as follows. In §2, we briefly review the background of DNS. In §3, we present the problem of Dares and three attack vectors to exploit unsafe Dares. In §4 and §5, we detail the methodology and results of our large-scale measurement study, respectively. In §6, we analyze the threats posed by Dares. In §7, we propose potential mitigations. In §8, we survey related work, and finally, we conclude in §9.

2. DNS OVERVIEW

The structure of DNS is organized as a hierarchical tree, which is shown in Figure 1. The second- and sometimes third-level domains are registered by enterprises or end-users for connecting their local computing resources to the Internet. Any enterprise/user can own a domain name if it has not yet been registered by another user. The further levels of domains are usually called subdomains, typically used to designate a particular host, like web and mail servers.

The conversion between a domain name and an IP address is called DNS resolution. Figure 2 illustrates the workflow of DNS resolution when a client visits `www.foo.com` for the first time. The stub resolver on the client queries a recursive DNS (rDNS) server that can be either local or remote, i.e., outside the local network (❶). In the case of a cache miss, rDNS will initiate queries recursively to the root server, the `.com` Top Level Domain (TLD) server, and the authoritative DNS (aDNS) server of `foo.com` (❷ ∼ ❻). Finally, the authoritative server of `foo.com` will respond with the corresponding IP address of `www.foo.com` (❼ ∼ ❽). Once the client obtains the IP address, it can connect to the website hosting server (❾ ∼ ❿).

Figure 3 shows sample records on the `.com` TLD server and the aDNS server for the example described in Figure 2. Each line of the DNS data represents a resource record (RR), which is a five-tuple data structure `<name, TTL, class, type, data>`. The fields `<name, class, type>` serve as the key to `data`, and `TTL` is the time-to-live in seconds that determines the lifetime of cached DNS records.

```
;; sample portion of .com zone file.

    foo.com. NS ns1.foo.com.
    foo.com. NS ns2.foo.com.

    ns1.foo.com. A 1.1.1.1
    ns2.foo.com. A 2.2.2.2
```

```
;; sample records from ns1.foo.com

   bar.exmaple.com. CNAME www.foo.com.
   www.foo.com. A 3.3.3.3
```

Figure 3: Sample portion of a TLD zone file and DNS records on a resolver. For brevity, the TTL and class fields are omitted.

3. DANGLING DNS RECORDS

Our work is inspired by the use-after-free vulnerabilities that exploit the dangling pointers in software. The data field of a DNS record is essentially a pointer, as exemplified in Figure 4. In this example, the data field, `1.2.3.4`, points to the machine that hosts the content of `www.foo.com`. Later, when the subdomain is no longer needed, the domain owner will release the IP address. The corresponding DNS record becomes dangling if the domain owner forgets to remove it from the authoritative DNS server. In general, we define a dangling DNS record as:

Dangling DNS Record (Dare). A DNS record `r:=<name, TTL, class, type, data>` is dangling if the resource to which the `data` field points is released.

Currently, there are more than 40 types of DNS RRs. After scrutinizing the semantics of each type of DNS RR, we identify four security-sensitive records if they become dangling. These records are listed in Table 1. Obviously, not all Dares are vulnerable to be exploited. For example, given a Dare-A in Figure 4, if an adversary cannot easily obtain the IP `1.2.3.4`, this Dare-A is safe. Here we further define unsafe Dares.

Unsafe Dare. A Dare is unsafe if the abandoned resource could be manipulated by a third party other than the one who controls the `name` field.

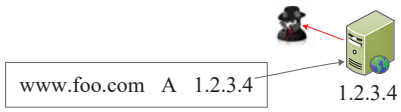


Figure 4: An example of a dangling A record.

Dare	RR	Description
Dare-A†	A	Returns an IPv4 address
Dare-CN‡	CNAME	Alias of a name to another
Dare-MX	MX	Maps to a list of message transfer agents
Dare-NS	NS	Delegate to an authoritative name server

Table 1: Types of security-sensitive dangling DNS records. †Our work currently covers IPv4 only. ‡DNAME is semantically similar to CNAME, so we do not consider DNAME separately.

In the following, we first review some key details of the DNS records in Table 1 and then present three approaches that attackers can harness to exploit the unsafe Dares.

3.1 Security Sensitive Dares

Dare-A. An `A` record maps a domain name to an IPv4 address. All requests to the `name` field of an `A` record will be directed to and handled by the host at the IP address. Thus, the domain name will be compromised if the IP address could be acquired by a third party other than the original domain name owner.

Dare-CN. A `CNAME` record specifies that a domain name is an alias for another domain name, the "canonical" domain name. For instance, `www.foo.com` in Figure 3 is the canonical domain name of its alias, `bar.example.com`. A request to the alias will be resolved to its canonical domain name, which is further resolved to an `A` record. Note that exploiting Dare-CN has almost the same effect as exploiting Dare-A.

Dare-MX. An `MX` record specifies the mail server responsible for accepting emails on behalf of the domain. In the case of multiple MX records, users can set a priority to each one and the server with the lowest value (i.e., highest priority) will be used first. In the following example, an email client will contact `a.mail.com` and `b.mail.com` first (usually in a round-robin manner); if both fail to respond, `c.mail.com` will then be contacted. Note that an `MX` record is not necessary to receive emails. When no `MX` is used, the `A` record of the domain (e.g., `foo.com`) will be treated as an implicit `MX` [14]. If a Dare-MX could be exploited, an adversary may be able to send and receive emails in this vulnerable domain.

```
foo.com. 60 MX 10 a.mail.com.
foo.com. 60 MX 10 b.mail.com.
foo.com. 60 MX 20 c.mail.com.
```

Dare-NS. An `NS` record delegates a domain to an aDNS server for answering queries about names under that domain. There also exists an `A` record to provide the IP address for the aDNS server, which is dubbed as a glue record. Normally, there are multiple `NS` records serving a single domain, and the resolvers need to choose one aDNS server for further querying. The aDNS server selection [57] can be (1) hitting the first server, (2) randomly selecting one, or (3) sorting the records based on a local-defined rule like RTTs. To force DNS resolvers to use a Dare-NS, attackers can leverage several techniques like Denial-of-Service attacks and NS pinning [39]. If a Dare-NS could be exploited, adversaries will set up a malicious aDNS and direct visitors to any IP address. Due to the transitive trust in DNS [53], the impact of Dare-NS is amplified to all those domains that directly or indirectly depend on it.

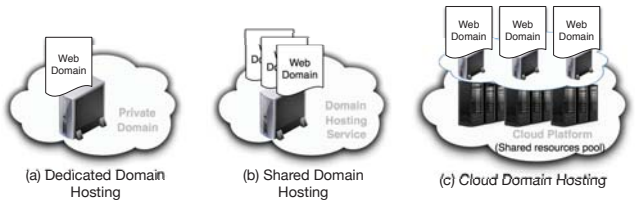


Figure 5: Three paradigms of modern domain hosting.

3.2 IP in Cloud

Every Dare in Table 1 is finally resolved to an IP address and thus adversaries can directly obtain the IP address to exploit unsafe Dares. For instance, if adversaries can obtain `1.2.3.4` in Figure 4, all subsequent requests to `www.foo.com` will then be handled by adversaries. Whether an IP address is obtainable highly depends on how a domain is hosted.

Figure 5 illustrates the three typical paradigms of modern domain hosting. In the first case, a domain is hosted on a dedicated machine with an IP allocated from the address blocks owned by the domain owner. Many large organizations like universities adopt this paradigm for most of their domains. However, the majority cannot afford dedicated hosting, and they usually host their domains using third-party services like GoDaddy [11]. In the normal configuration of these third-party services, many domains are hosted on a single server sharing the same IP address. A user only owns and controls the allocated storage space on the server. In both paradigms, a Dare-A is generally safe because adversaries cannot easily obtain the IP address to which the Dare-A points.

However, nowadays, more and more domains are migrated to the clouds. In particular, a customer can potentially obtain any public IP address from a shared IP address pool. Although the IP allocation should be random, a malicious customer can obtain the desired IP address by repeatedly allocating and releasing IP addresses. Therefore, we focus on the security threat of Dare-A in the context of cloud environments, especially the two most popular cloud platforms, Amazon EC2 [2] and Microsoft Azure [17].

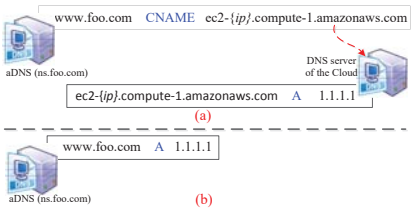


Figure 6: aDNS setups for a domain hosted in the cloud.

Amazon EC2. In Amazon EC2, users can rent virtual machines (instances) and run their own applications. By default, when an instance is initiated, it will be assigned a public IP address; when the instance is terminated, it will release the assigned IP address. EC2 also provides Elastic IP, a persistent public IP address allocated to a user's account. An Elastic IP is held by a user until she releases it. Once an Elastic IP is released, it will be recycled by EC2 and becomes re-allocable to other users immediately. Each instance that receives a public IP address is also given an external hostname in the form of `ec2-{ip}.compute-1.amazonaws.com`. Moreover, EC2 currently provides two types of platforms: EC2-Classic and EC2-VPC [3]. While they differ in many aspects, the major difference that matters to us is that a separate public IP address pool is

Dares	IP in Cloud	Abandoned Services	Expired Domains
Dare-A	√	√	
Dare-CN	√	√	√
Dare-MX	√	√	√
Dare-NS	√		√

Table 2: Summary of the attack vectors to which each type of Dare is vulnerable.

used for each type of platform. According to Amazon [24], all EC2 accounts created after December 4, 2013 can only use EC2-VPC, while EC2-Classic is merely available for accounts that have used it before on a region-by-region basis.

Once a public IP address in the cloud is obtained, a user can point its domain resource (e.g., a web server) to the IP address using either `CNAME` or `A` record, as illustrated in Figure 6. Once adversaries successfully obtain the IP address of a Dare-A, they are able to impersonate the domain resource at their will, regardless of which EC2 platform the domain resource resides in and which kind of DNS record it uses for pointing.

Microsoft Azure. Similar to EC2, the public IP addresses on Azure also fall into two categories: dynamic and reserved. A dynamic IP is allocated and then released when its associated resource, such as a virtual machine, is initiated and then terminated, respectively. To prevent its IP address from changing, a user can explicitly reserve an IP address, i.e., a static IP address. Our measurement shows that both types of public IP addresses are allocated from the same IP address pool, and any one of them becomes re-allocable immediately after being released. Furthermore, a dynamic IP can be converted to a reserved IP under a user's demand. Finally, to point a domain resource to a public IP address on Azure, the same simple technique (i.e., using either `CNAME` or `A` record) is applied.

3.3 Abandoned Third-party Services

Modern websites extensively use third-party services. For instance, they may use Mailgun [16] for email delivery and Shopify [22] for building online retail point-of-sale systems and stores. These services usually provide users a subdomain where the corresponding service is hosted. For example, when a user, Alice, subscribes the service from Shopify, she will be assigned a subdomain name, `alice.myshopify.com`, and thus her online store is accessible via this subdomain. However, in most cases, people prefer to have their stores under their own domains. To this end, each third-party service allows users to point their (sub)domains to the resource provided by the service using `A` or `CNAME` records. In the example of Shopify, Alice can set up her aDNS as follows:

```
    shop.Alice.com A 23.227.38.32
(or) shop.Alice.com CNAME alice.myshopify.com
```

In addition, Shopify's DNS server resolves all users' subdomains to a dedicated domain:

```
*.myshopify.com CNAME shops.shopify.com
```

Since all custom domains of Shopify point to the same IP address (`23.227.38.32`) or the same domain (`shops.shopify.com`), Alice also needs to claim ownership of `shop.Alice.com` on her Shopify account. In this way, Alice's store can be accessed through `shop.Alice.com`.

Later, when Alice does not want to use Shopify anymore, she can stop the service and purge the above DNS records. However, if she forgets to do the cleaning, `shop.Alice.com` will continue to be resolved to `shops.shopify.com` since most services use a wildcard to resolve user-specific subdomains (as Shopify does).

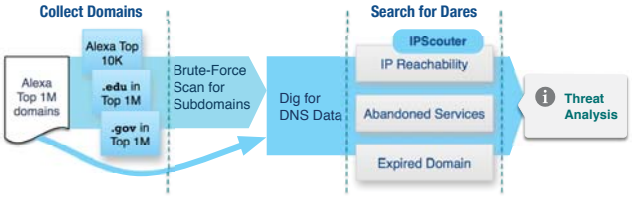


Figure 7: Methodology overview.

Now, if an adversary, Malice, knows that `shop.Alice.com` points to Shopify, he can claim ownership of it. If Shopify does not verify the claim, which is a common practice in most services, Malice can now control the subdomain, `shop.Alice.com`.

The case of email service is similar to this process. The only difference is that a user adds `MX` records, instead of `CNAME` records, for receiving emails.

A verification of domain ownership can prevent the above attacks. However, in some cases, verification is too costly, if not infeasible. For example, the Azure cloud service employs a user-specified subdomain naming scheme. To achieve the domain ownership verification, the cloud has to remember all subdomain names previously used by each user, and the induced cost is prohibitively high considering the large scale of Azure's user group.

In summary, for this attack to succeed, it requires that:

- the vulnerable domain can be resolved to a common target (e.g., IP address or domain name) and the third-party service does not verify the ownership of the vulnerable domain; *or*
- the vulnerable domain resolves to a custom target that can be obtained by any user when it is available.

3.4 Expired Domains

The `data` fields of `CNAME`, `MX`, and `NS` records all indicate domains that could expire. An adversary may re-register and abuse the expired domains. Our attack differs from previous works [46], [49], [50] in that they mainly exploit the residual trust of the expired domains, while ours abuses the trust of the *unexpired* subdomains pointing to the expired domains. Such stale records are pervasively neglected by domain administrators because (1) there could be secondary records as a means of failover (e.g., multiple `MX` and `NS` records); and (2) the services linking to the expired domains are no longer used and no one cares about updating them.

3.5 Summary

Overall, we have shown that several types of Dares can be exploited in multiple ways. Table 2 summarizes the attack vectors to which each type of Dare is vulnerable.

4. MEASUREMENT METHODOLOGY

To assess the magnitude of the Dares problem, we conduct a large-scale measurement study. The overview of our measurement methodology is shown in Figure 7. We attempt to answer the following two questions: (1) How prevalent is each type of Dares in the wild? and (2) what are the security implications of Dares?

4.1 Domain Collection

In order to comprehensively detect Dares, it is ideal to collect the DNS data for all apex domains and their subdomains. However, it is impractical to scan all domains. Since only popular ones could pose a serious threat, we build our dataset as listed in Table 3. We first obtain a list of apex domains. Here we choose a snapshot of Alexa's top 1 million domain list for each year from 2010 to 2016. These top domains are particularly attractive because a popular domain provides higher value if an adversary can control it. This set of

Dataset	Data Space
D	Unexpired apex domains in Alexa top 1M during 2010 $\sim$ 2016
S_t	Subdomains of top 10,000 general
S_e	Subdomains of top 2,700 .edu
S_g	Subdomains of top 1,700 .gov

Table 3: Evaluation set of domains.

domains is denoted as D. Note that our dataset D in essence is different from the expired domains studied in [46]. In the case of expired domains, the DNS records in all resolvers were purged, resulting in no Dares.

It is also non-trivial to obtain the complete subdomains of an apex domain. Since the majority of apex domains disallow a DNS zone transfer (i.e., a DNS query of type AXFR), we decide to use brute-force scanning to construct our subdomain list. However, it is impractical to scan all top 1 million domains. To make this search manageable, we constrain our search space to the first 10,000 domains, 2,700 `.edu` domains, and 1,700 `.gov` domains in the top 1 million domain list. We first issue a DNS zone transfer query to each of these domains, and we successfully collect the zone data for 320 domains. Based on the results of zone transfers, we then construct a word list of size 20,000 for brute-force scanning. The zone transfer results also show that the wildcard records (e.g., `*.foo.com`) are widely used in practice. In our brute-force scanning, we carefully eliminate the non-existent subdomains. In this process, we send DNS queries to about 288 million valid subdomains and about 570 thousand subdomains are successfully obtained. This subdomain dataset is denoted as $S = S_t \cup S_e \cup S_g$.

4.2 DNS Data Retrieval

Then we use the DNS tool `dig` to retrieve the DNS records of every domain in D and S. We only collect the DNS records whose types are listed in Table 1. For these types of DNS records except `A` record, we recursively issue DNS queries for the hostname in the `data` field until a query reaches (or fails to reach) an `A` record. Therefore, for each domain d in $D \cup S$, we obtain a DNS resolving chain $RC_d = \{rtype_0(d, data_0), \ldots, rtype_i(data_{i-1}, data_i)\}$. This dataset is denoted as DREC $= \bigcup RC_d$.

4.3 Searching for Dares

After the completion of DNS data collection, we then automatically search for the four types of Dares using Algorithm 1. Given the resolving chain of a domain, we recursively check the `data` field of every DNS record in the chain, as shown in Lines 7, 9, 12, 15 and 19 of Algorithm 1. The Dare type is determined by the type of the first DNS record of the chain. We next describe how we implement these checks in detail.

4.3.1 Checking A Records (Lines 7 and 9)

Both EC2 and Azure publish their public IP ranges [4], [18]. However, we still cannot know if a given IP is allocable at a specific time. Almost all cloud platforms including EC2 and Azure assign IP addresses randomly and disallow users to specify an IP to allocate. It is a challenging task to obtain a desired IP. We study this issue from the following two aspects:

- We quantify whether and how practical an attacker can overcome the random IP assignment to obtain a desired IP by scouting the IP pools.
- We then assess the potential magnitude of Dares in the wild.

Scouting IP Pools. We implement a simple tool, IPScouter, to milk the IP addresses from EC2 and Azure. Since EC2 uses two separate address pools for EC2-Classic and EC2-VPC, we set up two IPScouters, one for each address pool. The IPScouter-VPC randomly requests the IPs from all currently available regions [20]. The IPScouter-Classic requests the IPs from `us-east-1` as our account can support EC2-Classic only in this region. In both setups, only Elastic IPs are allocated via the `boto`'s [6] API `allocate_address`. Also, the IPScouter-Azure requests the IPs from the regions returned by `ServiceManagementService.list_locations()` [5]. The static IP addresses are reserved by using `create_reserved_ip_address()`. No virtual machine or service is launched in this process.

The obtained IP addresses are immediately released after being logged into `ALLOCIP`, the input to Algorithm 1. Finally, since all clouds throttle query API requests on a per-account basis, our IPScouters employ the exponential-backoff-linear-recovery strategy to control their request rates.

Algorithm 1 Search for Dares.

Input: `DREC`, `ALLOCIP`
Output: Dares (`DARES`) and potential Dares (`PDARES`)

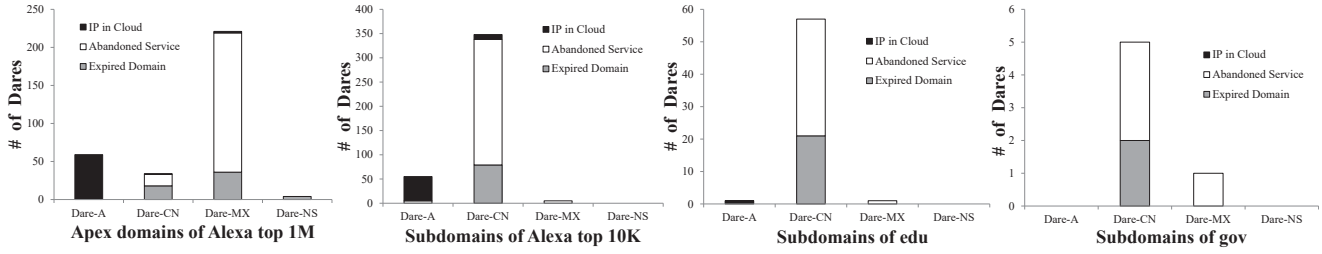
```
1:  procedure DAREFINDER(DREC, ALLOCIP)
2:      for RC ∈ DREC do
3:          daretype ← RC.rtype_0
4:          for rec ∈ RC do
5:              hostname, rtype, data ← unpack(rec)
6:              if rtype == "A" then
7:                  if data ∈ ALLOCIP then
8:                      DARES ← [daretype, rec, data]
9:                  else if likely_dareA(data) then
10:                     PDARES ← [daretype, rec, data]
11:             if rtype ∈ ["CN", "MX"] then
12:                 if domain_expired(data) then
13:                     DARES ← [daretype, rec, data]
14:                     break
15:                 if abandoned_service(data) then
16:                     DARES ← [daretype, rec, data]
17:                     break
18:             if rtype == "NS" then
19:                 if domain_expired(data) then
20:                     DARES ← [daretype, rec, data]
21:                     break
```

Potential Dares in the Wild. Our IPScouter is probabilistic by nature and many factors could affect the completeness of the milked IPs. We may not find all desired IPs in our study. For example, a cloud platform may reserve a portion of IP ranges for some time. Therefore, we scan all IPs in `DREC` to assess the potential number of exploitable Dares in the wild. Our basic assumption is that if an IP in a cloud is not alive, it has probably been released. In both EC2 and Azure, an in-use IP costs nothing, but users should pay for an unused one. Thus, we believe this assumption is valid in general. Given a set of `A` records $R = \{r_1, r_2, \cdots, r_n\}$, with $r_i =$ <name$_i$, IP$_i$> and $i \in [1, n]$, we check if they are potential Dares based on the following steps (Line 9 in Algorithm 1):

Step 1. If IP$_i$ is not in the cloud, remove r_i.

Step 2. We remove all records that are very unlikely to be dangling based on their `name` fields. For instance, a record may point to a specific service built atop an existing IaaS infrastructure like load balancing. These records are usually managed by the cloud DNS servers. If the DNS resolution is successful, it indicates that the IP is not released.

Step 3. We scan the remaining records using ZMap [34]. To reduce the scanning traffic, we prioritize the ports using a set of heuristics. For instance, the ports for HTTP and HTTPS are ranked first by default. If the `name` field starts with `ns`, it is probably a DNS server, and thus we scan port 53 first using both TCP and UDP. Note that we conduct a second scanning for all non-alive IP addresses after one month to ensure they are not transient failures.



(a) (b) (c) (d)

Figure 8: Number of confirmed Dares for each dataset.

Step 4. At this step, all remaining records are probably Dares since they are associated with these not-in-use IP addresses. We further check `archive.org` to gain more confidence on whether an archived webpage for $\texttt{name}_i$ can be found (see §5.2).

4.3.2 Checking Abandoned Services (Line 15)

We first identify a list of popular third-party services. To this end, we cluster all `CNAME` and `MX` records based on their `data` fields and then manually check all email and top 200 non-email services in terms of the cluster size. A service is selected for further checking if it (1) satisfies one of the two requirements in §3.3 (i.e., the domains using the service are vulnerable to security-sensitive Dares) and (2) provides free or free-trial accounts, which allow us for further checking. As listed in Table 4, only one email and eight non-email services meet the two pre-conditions and are chosen for further checking. The majority of those non-selected services do not provide free accounts to individuals, preventing them from being further checked. For non-email services, only several, such as Google [12] and Aliyun [1], enforce ownership verification. By contrast, we find only one email service that does not enforce ownership verification. This is probably because most email service providers try to prevent their services from being abused in spamming and phishing. A common practice of verification requires domain owners to include a random `CNAME` or `TXT` record into their aDNS's records. Since we assume that adversaries cannot control a domain's aDNS, such verification will be able to foil all attack attempts.

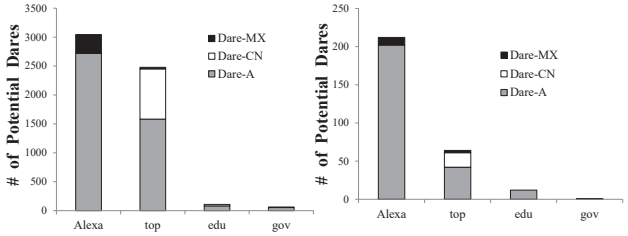
Type	Service List
`CN`	Azure cloud service (`cloudapp.net`), Shopify, Github, Wordpress, Heroku, Tumblr, Statuspage, Unbounce
`MX`	Mailgun

Table 4: Evaluated third-party services.

Then, to automatically find unclaimed domains, we build an automated tool by leveraging Selenium [21], a tool that automates web browsing. Note that we can easily single out unclaimed domains in Azure by simply finding these `CNAME` records whose `data` fields fail to be resolved to `A` records. There is no need to use the automated tool because the domains in the `name` fields of these `CNAME` records should be unclaimed in the Azure cloud service.

4.3.3 Checking Expired Domains (Lines 12 and 19)

It is straightforward to check whether a domain has expired. We first screen out the expired domains based on the `WHOIS` responses. For an expired domain, the response from `WHOIS` should be null. Since `WHOIS` is not always reliable, we then crosscheck with the popular Internet domain registrars such as GoDaddy to verify the expiration of a domain if we can re-register the domain.



(a) Amazon EC2 (b) Microsoft Azure

Figure 9: Number of potential Dares.

4.4 Limitations

While our work is able to find exploitable Dares in the wild, we cannot know whether and how many websites have already been exploited. For example, an expired domain may have already been registered by an attacker. Moreover, our study currently covers only two cloud platforms and nine third-party services. However, the Dare problem should be universal across many cloud platforms and third-party services.

5. MEASUREMENT RESULTS

In this section, we demonstrate that the problem of Dares is widespread and underplayed, even in those well managed zones like `edu` and `gov`. We first describe the general characteristics of Dares found in our measurement study and then analyze the measurement results with respect to the three attack vectors.

5.1 Characterization of Dares

Figure 8 presents the number of Dares found in the four datasets listed in Table 3. In this figure, we only count the IP addresses that are successfully obtained by our IPScouters (i.e., confirmed Dares). The remaining potential Dares are presented in Figure 9. For dataset `S`, multiple Dares in the same domain zone are counted separately. In total, we find 791 Dares and 5,982 potential Dares in the wild. As we can see, Dares exist in all four datasets, indicating a widespread problem.

It is evident that the total number of Dare-A and Dare-CN accounts for the majority of confirmed and potential Dares in the wild. This is because `A` and `CNAME` records are the most frequently used in practice. For apex domains, more than 90% delegate their aDNSes to third-party services like GoDaddy [37]. When the hosting resources are released after a website is closed, its aDNS is usually still alive and all DNS records will unlikely be deleted as the domain itself is still unexpired. The `A` and `CNAME` records for subdomains commonly link to new resources supported by the domains or external services, which often have a relatively short lifetime and can sometimes be migrated away. Due to the high churn rate of these subdomains, it poses a tedious burden for domain owners to manually keep their aDNS servers updated and consis-

tent. Therefore, in practice, these stale DNS records are usually not purged, resulting in Dares. Note that the number of Dare-CNs in dataset D is relatively small because it is normally not recommended to keep CNAME records at apex domains.

Dare-MX is mainly caused by abandoned services, and only dataset D has instances that can be exploited by the other two vectors. After examining these special instances, we find that domains in D tend to use multiple MX records pointing to different domains. For example, the DNS records of domain customizedgirl.com include

```
customizedgirl.com@ns-1057.awsdns-04.org.:
   customizedgirl.com. 60 MX 10 bridalpartytees.com.
   customizedgirl.com. 60 MX 10 customizedgirl.com.
   customizedgirl.com. 60 MX 10 shoplattitude.com.
```

Here three MX records point to three different domains with the same priority. We find that the third one, shoplattitude.com, has expired. We speculate that this is a typo, which should actually be shoplatitude.com. Since each of the three records is used in a round-robin fashion by resolvers, it is difficult for domain owners to quickly be aware of the failed record. By contrast, all MX records in a better-managed domain like those in dataset S usually point to different mail servers of the same domain.

Finally, only four instances of Dare-NS are found in our measurement, all of which are in dataset D. All instances share the same pattern of misconfiguration, with one example shown below.

```
bedshed.com.au@ns1.partnerconsole.net:
   bedshed.com.au. 3600 NS ns2.r2design.com.au.
   bedshed.com.au. 3600 NS ns1.r2design.com.au.
```

Using dig utility with +trace option, we find that the actual aDNS in .com.au TLD is ns1.partnerconsole.net, but the NS records are not updated and still point to the expired domains. The smidgen of Dare-NS in the wild is probably because the NS records are more critical, and a misconfiguration can be easily spotted. Moreover, the majority of domains have migrated aDNS to third-party services [37], which usually have well-managed servers. Unfortunately, this migration also becomes a common cause of Dare-A that can be exploited through the IP in cloud (see §5.2).

Dare	top 10K (S_t)	edu (S_e)	gov (S_g)	
Dare-A	40	1	0	
Dare-CN	260	50	5	
Dare-MX	5	1	1	
Total	277†	52	6	335

Table 5: Statistics of distinct apex domains in S with confirmed Dares. † Some domains overlap across the above three lines (e.g., a domain has both Dare-A and Dare-CN).

For dataset S, Table 5 shows the number of distinct apex domains for each type of Dare. In total, we identify Dares for 277 distinct domains in Alexa's top 10,000, 52 in edu zone and 6 in gov zone. In particular, the domains in S_t cover many types of websites as shown in Figure 10. Our results demonstrate that Dares indeed exist in almost all types of websites and thus can incur serious damages.

5.2 IP in Cloud

We now analyze the measurement results of the first attack vector in two cloud platforms: Amazon EC-2 and Microsoft Azure.

Performance of IPScouters. Figure 11 shows the number of distinct IP addresses obtained by IPScouters over time. IPScouter-Classic and IPScouter-Azure last for 14 days and IPScouter-VPC lasts for 26 days. For EC2-VPC and Azure, the number increases linearly, with about 5,000 and 2,200 new IP addresses obtained

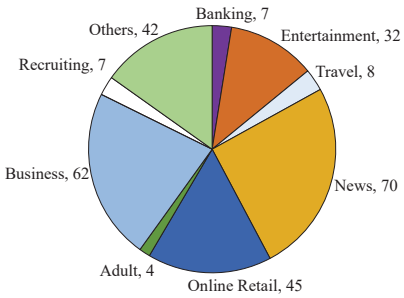


Figure 10: Categories of websites that have Dares.

daily, respectively. However, the number on EC2-Classic only rapidly increases in the first few days and then stops growing over time. The speed of IPScouters is mainly constrained by three factors: the request rate limit of the clouds, the randomness of IP allocation, and the density of IP address space. For the first constraint, we find that a five-second delay between two API calls (IP allocation or release) works fine with EC2, but at least a ten-second delay should be used in Azure. Under this configuration, IPScouters send about 7,900 and 4,300 IP allocation requests per day to EC2 and Azure, respectively.

Although it seems that IP allocation is not truly random, it is very unlikely for a cloud to re-use the recently released IP addresses. That is why on both EC2-VPC and Azure the number of daily obtained new IP addresses remains about half of the number of allocation requests sent to the clouds. This speed is fast enough to feasibly milk a large number of IP addresses in each cloud.

The significant speed decrease on EC2-Classic is probably due to the crowded IP address space. For instance, for all those domains that use EC2 in our datasets, about 69% are hosted on EC2-Classic. By contrast, both EC2-VPC and Azure have larger address space but fewer users. Note that, although the IP address spaces of EC2 and Azure include millions of IP addresses [4], [18], we speculate that only a portion of IP address space is available at any time.

We deploy only one IPScouter for each cloud platform, while adversaries may deploy IPScouter farms to significantly speed up the IP milking. Finally, the IP allocation in clouds is a complicated issue that deserves in-deep study, and we leave this exploration as our future work. In this work, our goal will be to demonstrate that the Dare problem is a real and serious threat.

Confirmed and Potential Dares. In our measurement, all confirmed Dares are from EC2, with about 93% from EC2-Classic. Considering that IPScouter-Classic milks IP addresses from only one region, more potential Dares would have been confirmed if we extended our search to other EC2 regions. Meanwhile, as shown in Figure 9, the number of potential Dares on EC2 is significantly larger than that on Azure. This is because Azure is a relatively new platform and has a much smaller market share than EC2. For instance, we find that among all domains that use clouds in our datasets, Azure hosts just one tenth. Thus, more time is required to milk desired IPs of Azure. However, this does not reduce the generality of a potential attack; the problem is universal. As the clouds become more crowded, the threat will be more serious and widespread.

By further crosschecking with archive.org, we successfully find snapshots for about 52.6% potential Dares. Thus, these domains can be claimed as true Dares with higher confidence.

We only identify a few potential Dares in edu and gov zones. Domains in these zones are mostly deployed using the paradigm of Figure 5(a), where no cloud IP is used. Besides, the majority of domains in gov zones use Rackspace [19], instead of EC2/Azure.

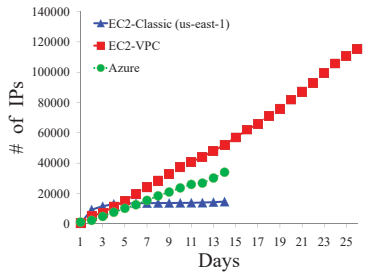


Figure 11: Number of IP addresses milked on clouds over time.

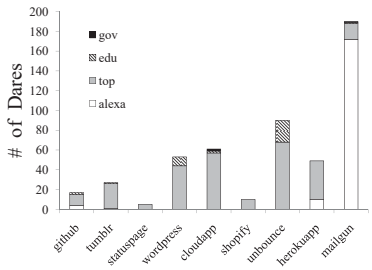


Figure 12: Number of Dares on each third-party service.

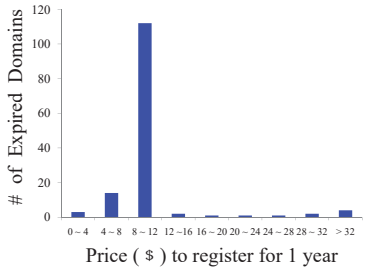


Figure 13: Prices to register the expired domains for one year.

Patterns of Dares. As shown in Figures 8(a) and 9, this attack vector can effectively exploit both apex domains and subdomains. We attempt to infer how the Dares are introduced by manually searching and checking relevant information of all confirmed Dares and 100 randomly sampled potential Dares.

While many vulnerable apex domains are toy websites with low value, more than half of those identified belong to startups for which we can find company information on CrunchBase [7], Twitter, and Github. One of the examples is described in §6.1. These startups are either closed, re-branded, or acquired by other companies. In all cases we examined, although the domain owners have released the hosting resources in a cloud, they continue to renew their domains. Such vulnerable apex domains provide valuable and attractive properties for attackers to conduct phishing and scamming.

For those vulnerable subdomains, we uncover two main causes of Dares. First, Dares are introduced due to website re-construction. One such example is `support.mediafire.com`. Previously, the "Get Support" on its homepage was linked to `www.support.mediafire.com/help`. However, it now points to the link of `www.mediafire.com/help`, and the host in the cloud for `support.mediafire.com` has been released. Clearly, the domain owner forgot to update their DNS servers with this change. In another example, `autotrader.co.uk` stopped the self-managed aDNS (`ns4.autotrader.co.uk`) and delegated the aDNS resolution to `verisigndns`. After this delegation, although they correctly updated the `NS` records, they forgot to delete the glue records. The second cause is simply that certain services have been discontinued. For instance, `books.panerabread.com` previously collaborated with Amazon to sell books. This service now seems to be closed. Again, the hosting resources are released, but its aDNS is not updated.

Cost analysis. During our study, IPScouters cost about $0.07 on EC2 and $0.005 on Azure per day. Such a low cost makes it feasible to conduct long-term IP milking. Once a desired IP is obtained, to hold it with a cheapest virtual machine, it costs $0.0115 and $0.023 per hour or $100.74 and $201.48 annually on EC2 and Azure, respectively. By contrast, the same expense would afford adversaries only several minimally effective typosquatting domains.

5.3 Abandoned Third-party Services

Figure 12 presents the number of Dares found on each third-party service, showing that Dares can be found on every service platform. Most Dares on Mailgun are in dataset D because email services are commonly hosted under apex domains. Instead, non-email services usually serve as the sub-functions of an apex domain and thus reside in the subdomains. We find that this Dare problem is quite worrisome as Dares can even be found in famous domains like `Yahoo.net` and `mit.edu`.

Patterns of Dares. While most Dares occur because the third-party services are abandoned, we find an interesting pattern in one of the services, Wordpress, as shown in the following example.

```
www2.opensky.com@ns-1448.awsdns-53.org.:
www2.opensky.com. CNAME blog.opensky.com.
blog.opensky.com. CNAME openskymerchants.wordpress.com.
```

The website of `blog.opensky.com` is still in use and its original webpage can be reached. The domain owner intends to direct `www2.opensky.com` to `blog.opensky.com` using `CNAME`. Unfortunately, this configuration fails to function properly and accessing `www2.opensky.com` will reach an error page on Wordpress. The problem lies in the fact that only `blog.opensky.com` is claimed on Wordpress, which dispatches web requests according to the initial domain name. Since `www2.opensky.com` is not claimed, Wordpress will direct all requests to the error page. An attacker can thus claim the subdomain, and all subsequent requests will then be redirected to the landing page under the attacker's control, although the `CNAME` tries to redirect to `blog.opensky.com`. While we only observe such cases on Wordpress, services like Github, Cloudapp, Shopify, and Herokuapp may also be vulnerable to this misconfiguration. The other three services, including Tumblr, Statuspage, and Unbounce, do not suffer this problem because a subdomain can be claimed if and only if it points to a specific domain like `domains.tumblr.com`.

Cost analysis. All of these services provide free or free-trial accounts. Thus, it costs adversaries virtually nothing to register many free accounts.

5.4 Expired Domains

Patterns of Dares. As our results show, many subdomains in even well-managed zones like `edu` and Alexa's top domains point to expired domains using `CNAME`. A further examination reveals three patterns into which these expired domains fall, as listed in Table 6. First, more than one-third of expired domains look quite similar to their alias subdomains. For instance, `module.rabobank.nl` points to `rabobank-hoi.nl` and `rps.berkeley.edu` points to `rpsberkeley.org`. Second, as found in [46], [50], a significant portion of subdomains point to expired external services. One example is `21vcdn.com`. The subdomain, `js.jiayuan.com`, points to the service that has stopped working since 2010. Third, we find several cases of typos. For instance, `b.ns.trnty.edu` points to `awsnds-18.net`. The domain owner obviously intends to use a `CNAME` record to redirect their previous aDNS to the one provided by Amazon AWS. This attempt fails because of a typo. The domain currently uses `NS` records to point to Amazon AWS directly, but the mistyped CNAME record still exists. The remaining 33% of expired domains basically comprise random characters.

Pattern	Examples	%
Similar to alias	module.rabobank.nl → rabobank-hoi.nl rps.berkeley.edu → rpsberkeley.org	39%
Expired external services	js.jiayuan.com → 21vcdn.com shopping.segye.com → ticketdamoa.com	21%
Typo	b.ns.trnty.edu → awsnds-18.net customizedgirl.com → shoplattitude.com	7%

Table 6: Patterns of expired domains.

Existing defense against abusive domain registration. We have re-registered all the expired examples listed in the paper (i.e., eight expired domains). After about three months, our re-registered domains are still alive and we received warning from only one domain owner. This indicates that the majority of expired domains are indeed vulnerable to be abused. Domain registrars and owners may adopt existing defense mechanisms to protect against the registration of abusive domains. First, they can disapprove those domains in malicious domain lists. However, we find that none of our identified expired domains are included in these lists. Second, they can disallow domain names that are very similar to well-known ones to be arbitrarily registered. In our datasets, we identify that this can prevent about 46% of expired domains from being exploited. Unfortunately, still about 54% of expired domains are irrelevant to vulnerable subdomains. It is difficult for registrars to determine whether such an expired domain is associated with Dares, rendering these misconducts hard to be thwarted. Therefore, more effective defense is needed to prevent abusive domain registration.

Cost analysis. These expired domains are also quite cheap to own. Figure 13 shows the prices to re-register these domains for one year. It costs less than $12 for most domains. Given the significant value of these vulnerable subdomains, this cost is negligible.

5.5 Exploiting Dares

We now determine the exploitable window of Dares. For those caused by released IP addresses in clouds and abandoned third-party services, we estimate their occurrence time by checking with `archive.org`. For expired domains, we can find their expiration date. Our results show that all Dares have a large exploitable window, ranging from three months to seven years, with over 90% being vulnerable for more than one year.

5.6 Ethical Considerations

During the process of this study, we did not conduct any adversarial activities against the scrutinized domains or the visitors to the Dares we successfully identified. We also checked with our institution's IRB and confirmed that we do not need to obtain its approval. All examples presented in this paper have either been corrected or defensively exploited by us. We have sent notifications to all affected domains, and have received responses from roughly half of them. Almost all apex domains did not reply. Although most subdomains have acknowledged our reports, only two thirds of them have taken action for remedy. Our experience is similar to the observations by Li et. al. [47].

6. THREAT ANALYSIS

Domain names serve as the trusted base in many security paradigms. For example, human users and many malicious domain detectors tend to assume an apex domain with a clean history as trusted. A user also trusts all subdomains of an apex domain with good reputation by nature. Unfortunately, our work demonstrates that such trust could be abused by adversaries to mount a number of much more powerful attacks. In this section, we describe and discuss four types of threats that could be significantly exacerbated by exploiting Dares.

6.1 Scamming, Phishing, and More

The common modus operandi that adversaries adopt in scamming, phishing, and many other forms of malicious activities includes typosquatting [44], doppelganger domains [8], and homograph attacks [41]. However, these approaches are limited in effectiveness, and vigilant users can easily spot them. Moreover, many automatic systems like EXPOSURE [29] and Notos [25] have been proposed to detect these malicious domains.

Dares can significantly enhance the effectiveness of these malicious attacks in two major ways. First, instead of registering new domains, adversaries directly abuse either subdomains or apex domains usually with a clean history and an excellent reputation. The abused domains have unchanged registration information and can even reside on the same IP addresses. Second, at an affordable cost, adversaries can target a large number of victims in a short time by leveraging services like Google AdWords. We next illustrate three case studies.

Case 1: Suspended domains getting revived. `GeoIQ.com` [10] is a web-based location analysis platform offering data sharing, risk mitigation, and real-time analysis services. The A record retrieved from its aDNS is shown below. We can see that this domain was hosted on EC2.

```
geoiq.com@ns-1496.awsdns-59.org.:
    geoiq.com. 1800 A   23.21.108.12
```

In July 2012, `GeoIQ.com` was acquired by another company and `archive.org` shows that the last snapshot of this domain was captured on August 1, 2015. This implies that the domain owners released the hosting resources in EC2 around August 2015, which was later successfully obtained by our IPScouter. However, the `WHOIS` data shows that the domain still gets renewed annually.

```
Domain Name: GEOIQ.COM
Registrar: GODADDY.COM, LLC
Updated Date: 21-sep-2015
Creation Date: 20-sep-2005
Expiration Date: 20-sep-2016
```

With a simple Google search, we can find their accounts on many platforms, including Github, Twitter, and Youtube. Adversaries could impersonate the domain and launch social engineering attacks more effectively.

Case 2: Inherited trust from apex domains. `mediafire.com`, ranked 169 in Alexa at the time of our study, is a file hosting, file synchronization and cloud storage service provider. One of their subdomains, `support.mediafire.com`, was hosted on EC2 but later was no longer used. The hosting service on EC2 was released and then successfully obtained by our IPScouters.

```
support.mediafire.com@ns-1179.awsdns-19.org.:
    support.mediafire.com. 86400 A 23.21.94.181
```

The subdomain `support` is a common practice used by many domains to provide supporting services to users. There are also many other similar cases like `jobs`, `payment`, or `shop`. If an adversary hosted malicious contents or carried out spear phishing under such subdomains, even the most vigilant users would fall victim to the attacks.

Case 3: Harvesting through Google Adwords. `Travelocity.com`, ranked 1,810 in Alexa at the time of our study, is one of the largest online travel agencies. We find that one of its subdomains points to an expired domain using `CNAME` record.

```
can.travelocity.com@pdns1.ultradns.net.:
  can.travelocity.com. CNAME travelocitycancontest.com.
```

To demonstrate how fast an adversary can spread the attacks and at what cost, we register this expired domain and direct visitors to

our subdomain using Google AdWords. To minimize the inconvenience that our study might have caused, we redirect all visitors to the homepage of Travelocity after recording the MD5 of source IP addresses. Since our interaction with the users is limited to logging the hashed IP addresses, we believe there are no ethical implications in this experiment. We run the campaign for two days, and 141 distinct IP addresses are recorded at the cost of $1.38. Adversaries could set up a fake login page or steal cookies directly. In either case, thousands of accounts could be compromised.

6.2 Active Cookie Stealing

Adversaries have multiple ways to steal and hijack cookies. One simple approach requires the traffic between users and websites to be unencrypted and adversaries to be able to monitor the traffic. This strong requirement limits the scale and feasibility of this approach for cookie stealing. For instance, almost all top websites have adopted at least partial HTTPS [54] and sensitive cookies are usually transmitted in HTTPS only (using the `Secure` flag). Alternatively, if HTTP cookies do not have the `HttpOnly` flag set, adversaries can obtain them through other means like XSS attacks. As this flag is being deployed on more websites, XSS attacks will become ineffective in cookie hijacking. By exploiting Dares, however, adversaries can actively steal cookies from world-wide users, regardless of the `HttpOnly` and `Secure` flags. This likely results in not only privacy leakage but also fully compromised accounts.

Implications. Whenever possible, cookies with sensitive account information should be scoped to trusted subdomains only. It is also unsafe to rely on the `Secure` flag to prevent cookie stealing. The `Secure` flag is known to lack integrity [58], but it was generally assumed to be secure against stealing. However, this assumption will be challenged by Dares.

6.3 Email Fraud

Email is still one of the favorite attack vectors in online fraud. The malicious emails are usually sent with authentic addresses that are not under the adversaries' control. Since adversaries cannot receive and further confirm reply emails from victims, the email attacks are open-looped. However, by exploiting a Dare, an adversary will instead be able to not only send but also *receive* emails. In particular, some popular existing anti-spam mechanisms including Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM) can be bypassed. Enhanced with these capabilities, adversaries could conduct many forms of online fraud more effectively and efficiently, from spamming, spear phishing to even abusing exclusive online membership such as Amazon Prime membership.

6.4 Forged SSL Certificate

Modern websites commonly provide critical online services over mandatory HTTPS connections, and they allow sensitive cookies to be transmitted only over encrypted connections using the `Secure` flag. For example, the following is a cookie with `Secure` flag set by `travelocity.com`:

```
Set-Cookie: JSESSION=d1b8eb43-xxx; Domain=.
    travelocity.com; Path=/; Secure; HttpOnly
```

To steal these secure cookies, an adversary has to set up an HTTPS website on the vulnerable subdomain and get it signed by a Certificate Authority (CA). To ensure the authenticity of a certificate, CA usually requires subscribers to prove the ownership of a (sub)domain. This typically involves verification via specific email addresses under the *apex* domain or those in the `WHOIS` database. Adversaries in our threat model can hardly complete this verification.

However, the emerging new Certificate Authority, such as `Let's Encrypt` [15], tends to leverage the *automated* and *free* validation

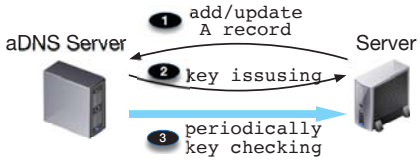


Figure 14: Authenticating Ephemeral IPs.

to simplify the process of issuing certificates. `Let's Encrypt` provides two ways for subscribers to prove the control of a domain, one of which involves provisioning an HTTP resource under the domain being signed. Unfortunately, when adversaries exploit a Dare through a cloud IP or an expired domain, they have the full access to the hosting resource of the domain and thus can pass the challenge of `Let's Encrypt`. Using this principle, we successfully have a subdomain `can.travelocity.com` [23] authentically signed.[1]

Implications. It is insufficient to use merely one single challenge for ownership verification. Considering that both aDNS (in the case of Dare-NS) and domain hosting resources could be compromised, it would seem more reliable to seek confirmation from specific emails, e.g., those in the `WHOIS` database.

7. MITIGATIONS

Almost all previous efforts, such as the Domain Name System Security Extensions (DNSSEC), attempt to protect the integrity and authenticity of DNS records returned to clients. Little attention has been paid to authenticating the resources to which DNS records point. Domain owners are commonly assumed to keep their aDNS servers updated and consistent. Unfortunately, our work has demonstrated that this assumption rarely holds in practice, and the resulted problem, Dare, is a serious and widespread threat. In this section, we propose and discuss the mechanisms that can mitigate Dares with minor manual efforts. In particular, we focus on the DNS `data` fields exploited by our three attack vectors. The key principle of these mechanisms is that *all resources* should be considered *ephemeral*.

Authenticating Ephemeral IP addresses. We propose a mechanism that allows aDNS servers to automatically authenticate IP addresses whenever an `A` record is added or updated. Figure 14 shows the workflow of the mechanism. Both aDNS and the corresponding server whose IP address is added/updated have one daemon. When the `A` record is added or updated, the aDNS communicates with the server and issues a key to it. Then, aDNS periodically checks the validation of the key. While the architecture is simple, a set of problems need to be resolved in practice, e.g., how to protect the key on the server and how much overhead is induced on aDNS. We leave the implementation and evaluation of this mechanism as our future work.

Breaking resolution chain through the aDNS of third-party services. In the case of `data` fields pointing to external services, we recommend that services like Shopify should deprecate `A` records and adopt an isolated name space in `CNAME` for each user. In our observations, all the external services except Shopify and Tumblr have deprecated `A` records. To protect the stale `CNAME` records, we define an isolated name space for each user. Since every user already has a unique account number, the services can generate

[1]The site `can.travelocity.com` is associated with a dangling DNS record and is not currently being used by the domain owner. We temporarily signed the subdomain and directed it to `www.travelocity.com`, and thus there is no break caused by our experiment.

CNAME records using the format of {*user-specified-name*}.useraccount.service.com. Multiple domains managed under the same account are assigned unique names like:

```
@aDNS of Shopify
store-1.alice.myshopify.com CNAME shops.shopify.com
store-2.alice.myshopify.com CNAME shops.shopify.com
```

Once the domain of store-1 becomes unclaimed, the record of store-1.alice.myshopify.com should be deleted from the aDNS of Shopify, and thus the dangling domain cannot be resolved.

Checking for expired domains. In existing DNS systems, only the records with expired domains in name fields will be purged from DNS servers, and those with expired domains in data fields (e.g., pointed to by a CNAME) are generally neglected. We have shown that these stale records could be exploited as a major source of Dares. We advocate that aDNS servers should periodically check the expiration of domains in data fields. Since this checking is triggered only when the expiration date is approaching, its frequency is very low and the overall overhead is trivial. Complementary to periodic checking, Alembic [46] can be used to locate potential changes in domain ownership. We are also considering to extend Alembic using the patterns listed in Table 6.

8. RELATED WORK

In past decades, significant research efforts have been devoted to studying the security of DNS. In the following, we provide an overview of previous works that are closely related to ours.

Cache Poisoning Attacks. Adversaries could exploit the flaws in a DNS server to inject incorrect entries, which will direct users to a different server controlled by adversaries. Since Bellovin [28] revealed this vulnerability in the 1990s, several mitigations [27], [31], [32], [52], [55] have been proposed. Adversaries can also tamper with DNS resolvers using spoofed DNS responses (i.e., off-path DNS poisoning [38], [39], [40], [43]). Instead of injecting entries to benign DNS servers or resolvers, more malicious resolution authorities have recently been deployed by adversaries [33], [45]. While having the same negative impact, our work is different from cache poisoning because we neither tamper with DNS resolutions nor set up malicious DNS services.

DNS Inconsistency and Misconfiguration. DNS is organized in a hierarchical tree structure, and it does not require strong consistency among nodes. Therefore, data changes in upper-level servers cannot override the cached copies in recursive resolvers. The outdated data will continue to serve users before reaching its TTL limit. The weak cache consistency could yield vulnerabilities like ghost domain names [42], i.e., the domains that have been deleted from TLD servers but still resolvable. Some approaches have been proposed to address this problem. For example, DNScup [30] proactively pushes changes in authoritative servers to recursive resolvers. However, in our study we reveal that inconsistency between DNS records and the connected resources is also prevalent and could pose serious security threats.

Pappas et al. [51] diagnosed three types of misconfigurations and found that these misconfigurations are widespread and degrade the reliability and performance of DNS. By contrast, our work uncovers a new type of DNS misconfiguration and studies its potential security threats. Although the issue of Dares has received some attention in two non-academic blogs [9], [13], our work is the first large-scale systematic study on this problem, in terms of DNS record types and the magnitude of Dares in high-value domains. We also identify two more vulnerable third-party services, one of which (i.e., Azure Cloud Service) cannot even be protected using domain ownership verification.

Malicious Domains. Finally, many works have focused on understanding and identifying malicious domain names. Jiang et al. [42] found that a malicious domain name could remain resolvable even long after it has been deleted from the upper level DNS servers or after its TTL has expired. Hao et al. [36] studied the domain registration behavior of spammers and found that spammers commonly re-register expired domains. Furthermore, Lever et al. [46] characterized the malicious re-registration of expired domains and demonstrated that the *residual trust abuse* is the root cause of many security issues. The Dares we studied in this paper can also be categorized as the residual trust abuse. The most salient difference is that Dares could be caused by not only the expired domains, but also a large number of subdomains, including those of the most well-known websites.

Recent works have also proposed approaches to distinguish benign and malicious domains. These approaches extract features from lexical representations, registration information, and the properties of name servers [25], [29], [35], [48]. Yadav et al. [56] and Antonakakis et al. [26] proposed methods to identify dynamically generated domains that are used by advanced botnets for command and control. The security threats presented in our work could significantly decrease the performance of these detectors.

9. CONCLUSION

This paper studies the problem of dangling DNS records (Dares), which has been largely overlooked, and demonstrates that Dare is a serious and widespread security threat. In order to exploit these unsafe Dares, we have presented three attack vectors, IP in cloud, abandoned third-party services, and expired domains, for domain hijacking. Then we have conducted a large-scale measurement on four datasets containing representative domains to quantify the magnitude of the unsafe Dares in the wild. We have found hundreds of unsafe Dares on even those well-managed zones like edu and Alexa top 10,000 websites. This is very worrisome because Dares can notably enhance many forms of online fraud activities, such as spamming and cookie stealing. The underlying cause of Dares is the lack of authenticity checking for resources pointed to by DNS records. To this end, we have proposed three defense mechanisms that can effectively mitigate Dares with minor human effort.

10. ACKNOWLEDGMENTS

We would like to thank our shepherd Manos Antonakakis and the anonymous reviewers for their insightful and detailed comments. This work was partially supported by ONR grant N00014-13-1-0088 and NSF grant CNS-1618117.

11. REFERENCES

[1] Aliyun. https://www.aliyun.com/.
[2] Amazon EC2. https://aws.amazon.com/ec2/.
[3] Amazon EC2 and Amazon virtual private cloud. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-vpc.html.
[4] AWS IP address ranges. http://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html.
[5] Azure SDK for python. http://azure-sdk-for-python.readthedocs.io/en/latest/.
[6] Boto: A python interface to amazon web services. http://boto.cloudhackers.com/en/latest/.
[7] CrunchBase. https://www.crunchbase.com/.
[8] Doppelganger domain. https://en.wikipedia.org/wiki/Doppelganger_domain.

[9] Fishing the AWS IP pool for dangling domains. http://www.bishopfox.com/blog/2015/10/fishing-the-aws-ip-pool-for-dangling-domains/.
[10] GeoIQ. https://crunchbase.com/organization/fortiusone.
[11] GoDaddy. https://www.godaddy.com/.
[12] Google Apps. https://support.google.com/a/answer/112038?hl=en.
[13] Hostile subdomain takeover. http://labsdetectify.wpengine.com/2014/10/21/hostile-subdomain-takeover-using-herokugithubdesk-more/.
[14] Implicit MX. http://tools.ietf.org/html/rfc5321#section-5.
[15] Let's Encrypt. https://letsencrypt.org/.
[16] Mailgun. https://www.mailgun.com/.
[17] Microsoft Azure. https://azure.microsoft.com/en-us/.
[18] Microsoft Azure datacenter IP ranges. https://www.microsoft.com/en-us/download/details.aspx?id=41653.
[19] Rackspace. https://www.rackspace.com/en-us.
[20] Regions and availability zones. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html.
[21] Selenium. http://www.seleniumhq.org/.
[22] Shopify. https://www.shopify.com/.
[23] SSL test. https://www.ssllabs.com/ssltest/analyze.html?d=can.travelocity.com.
[24] Your default VPC and subnets. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/default-vpc.html.
[25] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for DNS. In *USENIX Security*, 2010.
[26] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *USENIX Security*, 2012.
[27] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS security introduction and requirements. In *RFC 4033 (Proposed Standard)*, 2005.
[28] S. M. Bellovin. Using the domain name system for system break-ins. In *USENIX Security*, 1995.
[29] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: finding malicious domains using passive DNS analysis. In *NDSS*, 2011.
[30] X. Chen, H. Wang, S. Ren, and X. Zhang. DNScup: Strong cache consistency protocol for DNS. In *IEEE ICDCS*, 2006.
[31] D. Dagon, M. Antonakakis, K. Day, X. Luo, C. P. Lee, and W. Lee. Recursive DNS architectures and vulnerability implications. In *NDSS*, 2009.
[32] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee. Increased DNS forgery resistance through 0x20-bit encoding: Security via leet queries. In *ACM CCS*, 2008.
[33] D. Dagon, C. Lee, W. Lee, and N. Provos. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *NDSS*, 2008.
[34] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast internet-wide scanning and its security applications. In *USENIX Security*, 2013.
[35] M. Felegyhazi, C. Kreibich, and V. Paxson. On the potential of proactive domain blacklisting. In *USENIX LEET*, 2010.
[36] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck. Understanding the domain registration behavior of spammers. In *ACM IMC*, 2013.
[37] S. Hao, H. Wang, A. Stavrou, and E. Smirni. On the DNS deployment of modern web services. In *IEEE ICNP*, 2015.
[38] A. Herzberg and H. Shulman. Security of patched DNS. In *ESORICS*, 2012.
[39] A. Herzberg and H. Shulman. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org. In *IEEE CNS*, 2013.
[40] A. Herzberg and H. Shulman. Socket overloading for fun and cache-poisoning. In *ACSAC*, 2013.
[41] T. Holgers, D. E. Watson, and S. D. Gribble. Cutting through the confusion: A measurement study of homograph attacks. In *USENIX ATC*, 2006.
[42] J. Jiang, J. Liang, K. Li, J. Li, H. Duan, and J. Wu. Ghost domain name: Revoked yet still resolvable. In *NDSS*, 2012.
[43] D. Kaminsky. It's the end of the cache as we know it. In *Blackhat Briefings*, 2008.
[44] M. T. Khan, X. Huo, Z. Li, and C. Kanich. Every second counts: Quantifying the negative externalities of cybercrime via typosquatting. In *IEEE S&P*, 2015.
[45] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, and T. Holz. Going wild: Large-scale classification of open DNS resolvers. In *ACM IMC*, 2015.
[46] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis. Domain-z: 28 registrations later – measuring the exploitation of residual trust in domains. In *IEEE S&P*, 2016.
[47] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson. You've got vulnerability: Exploring effective vulnerability notifications. In *USENIX Security*, 2016.
[48] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *ICML*, 2009.
[49] T. Moore and R. Clayton. The ghosts of banking past: Empirical analysis of closed bank websites. In *FC*, 2014.
[50] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. You are what you include: Large-scale evaluation of remote javascript inclusions. In *ACM CCS*, 2012.
[51] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang. Impact of configuration errors on DNS robustness. In *ACM SIGCOMM*, 2004.
[52] R. Perdisci, M. Antonakakis, X. Luo, and W. Lee. Wsec DNS: Protecting recursive DNS resolvers from poisoning attacks. In *IEEE/IFIP DSN*, 2009.
[53] V. Ramasubramanian and E. G. Sirer. Perils of transitive trust in the domain name system. In *ACM IMC*, 2005.
[54] S. Sivakorn, I. Polakis, and A. Keromytis. The cracked cookie jar: HTTP cookie hijacking and the exposure of private information. In *IEEE S&P*, 2016.
[55] P. Vixie. DNS and BIND security issues. In *USENIX Security*, 1995.
[56] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting algorithmically generated malicious domain names. In *ACM IMC*, 2010.
[57] Y. Yu, D. Wessels, M. Larson, and L. Zhang. Authority server selection of DNS caching resolvers. In *ACM SIGCOMM CCR, 42(2): 80-86*, 2012.
[58] X. Zheng, J. Jiang, J. Liang, H. Duan, S. Chen, T. Wan, and N. Weaver. Cookies lack integrity: Real-world implications. In *USENIX Security*, 2015.