Who You Gonna Call? An Empirical Evaluation of Website `security.txt` Deployment

Tara Poteat
tpoteat3@gatech.edu
Georgia Institute of Technology
USA

Frank Li
frankli@gatech.edu
Georgia Institute of Technology
USA

ABSTRACT

The `security.txt` proposed standard allows organizations to define how security researchers should disclose security issues. While it is still proceeding through the final stages of standardization, major online services have already adopted the standard (such as Google, Facebook, LinkedIn, and Github). In this work, we conduct an empirical investigation into how websites are deploying `security.txt`. We first monitor `security.txt` adoption over a 15-month period, identifying the level of deployment for top websites. We also characterize the information being provided through `security.txt` and issues present in the provided data. Ultimately, our analysis sheds light on how the `security.txt` mechanism manifests in practice and its implications for vulnerability reporting, particularly for large-scale automated notification campaigns.

CCS CONCEPTS

• **Security and privacy** → **Web protocol security**.

ACM Reference Format:
Tara Poteat and Frank Li. 2021. Who You Gonna Call? An Empirical Evaluation of Website `security.txt` Deployment. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3487552.3487841

1 INTRODUCTION

Every year, security researchers identify thousands of new vulnerabilities and misconfigurations. Methods such as large-scale web crawling and Internet-wide scans often afford researchers the opportunity to detect en masse which Internet systems and organizations are vulnerable. While the security community has demonstrated consistency in uncovering new security problems, disclosing and remediating these issues in practice remains a distinct challenge.

While some organizations support well-publicized and concretely defined vulnerability disclosure processes, such as with bug bounty programs [9, 10], information on reporting security concerns is not commonplace for many organizations. Prior efforts into large-scale security notifications to system operators [4, 5, 15, 16, 18, 19] have identified that while such notifications can drive significant levels of remediation, one of the main barriers to successful disclosure is simply identifying a proper point of contact for affected hosts. These works have revealed that existing methods for finding contacts, such as relying on WHOIS information, attempting generic administrator email addresses (e.g., *admin@* or *hostmaster@*), or manually searching on websites and social media, are insufficient and often labor-intensive.

The `security.txt` proposed standard [7] aims to remedy this information gap by standardizing how organizations publish their vulnerability disclosure practices, guiding security researchers in reporting security concerns. Specifically, `security.txt` specifies the format of a text file to be hosted at a well-defined location on an organization's website (similar to standards such as `robots.txt` [12] and `ads.txt` [14], which publicize a website's crawling and advertising policies, respectively). The `security.txt` file contains various information fields, including points of contact, a disclosure policy, preferred languages for security reports, and the location of an encryption key to use for securing disclosure communication. First proposed in 2017, `security.txt` is still in the final stages of standardization. However, it has already been adopted by major online services for several years, such as Google, Facebook, Github, LinkedIn, and Dropbox, and is being recommended for use throughout U.S. government agencies [22]. With numerous websites already deploying `security.txt`, we consider it salient to investigate how organizations are using the standard and the characteristics of their published vulnerability disclosure information.

In this paper, we provide a large-scale empirical evaluation of `security.txt` deployment on the Alexa top 100K websites [2]. Over a 15-month period, we crawl the `security.txt` files accessible on these popular sites, measuring how deployment levels have changed longitudinally. We also investigate the nature of the information provided, including information at external references.

We observe that thousands of top websites already support `security.txt`, with adoption growing over time, especially among higher-ranked sites. Our analysis of `security.txt` content identifies common patterns in the points of contact listed for security reporting, as well as a large portion of websites providing contact information in non-standard ways. The specification also requires that `security.txt` files provide an expiration date (the only other data field required besides the contact field), yet we discover that less than 2% of websites adhere to this requirement, making it the least provided field even among various other optional fields. We also find that few sites provide a signature for their `security.txt` file, as recommended, although a fifth of sites provide encryption key information for securing disclosure communication.

Ultimately, our empirical investigation into `security.txt`'s deployment across websites sheds light on how it can be used in practice for vulnerability disclosure, notably in identifying non-standard representations of information and implications for large-scale automated security notifications to system operators.

2 BACKGROUND

2.1 security.txt Proposed Standard

The security.txt proposed standard [7] defines a method for organizations to publish their vulnerability disclosure practices, aiding security researchers in reporting discovered security issues. Similar to robots.txt [12] and ads.txt [14], used by websites to publicize crawling and advertising practices[1], security.txt specifies the format of a text file accessible on an organization's website at a well-defined location (specifically the .well-known [17] or root URL paths). For example, Google's security.txt file is available at: *https://www.google.com/.well-known/security.txt*.

The security.txt file is formatted as lines of key-value pairs (separated by colons), where the key is a field name (although free-form comments are supported as lines starting with the pound sign). There are currently eight fields defined:

(1) **Acknowledgements:** Links to a webpage where researchers can be recognized for their vulnerability reporting.
(2) **Canonical:** Lists the canonical URI of the security.txt file, useful if a researcher obtains a security.txt file through means other than directly accessing that location.
(3) **Contact (Required):** Provides information on where to report vulnerabilities, such as an email address, a phone number, or a web page with contact information.
(4) **Encryption:** Locates an encryption key that should be used for secure communication when reporting security issues. (Note, this field should not contain the key itself.)
(5) **Expires (Required):** Indicates when the data in the file should be considered stale and no longer used.
(6) **Hiring:** Links to a webpage on security-related job positions at the organization.
(7) **Policy:** Provides the location of the organization's vulnerability disclosure policy.
(8) **Preferred-Languages:** Enumerates the languages that the organization would prefer used when submitting security reports.

The standard also recommends that the security.txt file is digitally signed using an OpenPGP cleartext signature.

2.2 Related Work

To our knowledge, security.txt has not been previously empirically studied. However, prior work has empirically assessed the deployment of similar standards based on websites publishing information files at standardized locations, such as robots.txt [12] and ads.txt [14]. Sun et al. [21] measured the deployment of robots.txt files in 2007, while Kolay et al. [11] provided a larger-scale measurement and characterization of robots.txt content in 2008. Sun et al. [20] also investigated search engine biases exhibited in robots.txt policies, while Giles et al. [8] analyzed the extent to which web crawlers adhered to robots.txt policies. Bashir et al. [3] provided a longitudinal analysis of ads.txt deployment, characterizing the ad ecosystem entities listed in ads.txt files and their compliance with the specifications. We note that several of these prior works [3, 21] observed syntactic and semantic inconsistencies in how websites deployed these other standards, similar to our study's observations with security.txt.

[1]Neither robots.txt nor ads.txt are IETF standards currently, although robots.txt is also proceeding through IETF standardization. Both have enjoyed significant levels of adoption though [11, 14].

3 METHOD

To obtain data on the use of security.txt by websites, we crawled the security.txt files on the Alexa Top 100K websites [2] every 7 days over a 15-month period, from January 29, 2020 to April 19, 2021 (we had a 1.5-month measurement interruption throughout November and the first half of December, 2020).

For each measurement instance, we first downloaded the latest Alexa top list snapshot, and crawled only the top 100K ranked domains in a randomized order (to distribute load induced on crawled websites). To collect the security.txt files for these domains, we implemented a parallelized wget-based crawler that supplied a Chrome browser user-agent. We attempted to fetch the security.txt file at both a domain's root and .well-known paths. While the security.txt standard mandates that the file be accessible over HTTPS (for transport security), we attempted to connect to websites using both HTTP and HTTPS. Thus, our crawler issued four requests per domain, timing out after 5 seconds per request.

For many domains, fetching the security.txt URL returned unexpected data. The returned data was typically an HTML document (e.g., error webpage), but we also occasionally observed image files and other text files (including ads.txt and robots.txt files). We filtered all non-text files, and classified security.txt files heuristically through pattern-matching on the security.txt specified file format and defined fields. We manually inspected a random sample of 100 fetched text files that were labeled as security.txt and 100 files labeled otherwise, and did not identify any false positive or false negative classifications, giving us confidence in the accuracy of our data labeling.

For the security.txt files obtained, we parsed non-comment lines as colon-delimited key-value pairs to extract the provided fields and their values. Some fields may contain a webpage URL as a value. For security-relevant fields, we fetched these webpage URLs using the same crawling process as with accessing security.txt files, affording analysis of this external content (further details in Section 4).

4 RESULTS

In this section, we analyze the security.txt data collected. Except for our longitudinal analysis in Section 4.1, we will consider all security.txt files that we crawled across our observation period, evaluating the most recently crawled version for each domain.

4.1 Longitudinal Analysis

We first evaluate how the use of security.txt by Alexa top websites changed longitudinally.

Deployment Levels Among Top Sites. In Figure 1, we plot the proportion of websites deploying security.txt over our measurement period for different groups of websites: the top 100, top 1K, top 10K, and top 100K sites. Note that as the Alexa top list is dynamic,

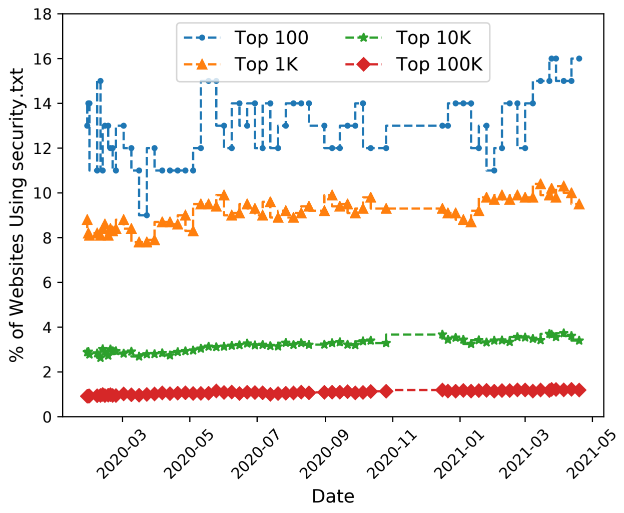

Figure 1: Deployment of security.txt over time by the Alexa top 100, top 1K, top 10K, and top 100K websites. Note that the set of websites is dynamic over time, and this graph represents the proportion of top sites using security.txt at a given time. We lack data points during a measurement outage in November and early December, 2020.

at a given time, this graph represents the top sites at that particular time (rather than a static set of sites tracked over time). Over the 15-month period of our study, the deployment levels remained largely stable for all website categories. We observe that a non-trivial proportion of top websites already deployed security.txt, with higher-ranked websites exhibiting higher deployment rates. The top 100 sites exhibited a deployment rate largely varying between 11–16% (the variation over time arises due to the small population size and regular churn in the set of top 100 sites). In comparison, 8–10% of the top 1K websites used security.txt. This percentage decreases to 3–4% for the top 10K sites, and only a percent for the top 100K. We hypothesize that higher-ranked websites may be better provisioned for managing security matters and may have greater stakes in ensuring organization security, and thus may be more likely to deploy security.txt.

Adoption Rate by Top Sites. While the proportion of top sites deploying security.txt does not exhibit significant variation during our observation period, this may reflect the churn in the top list itself, as sites that may have adopted security.txt later may change rankings or may no longer remain in the top list. To better quantify whether websites are adopting security.txt over time, we consider a static set of websites, using the set of sites and their rankings from our first crawl. However, our subsequent crawls may not have consistently measured this entire set of sites, and we may no longer measure a site after a certain point in time.

To account for this, we apply survival analysis, which is a statistical framework for characterizing the probability that a subject experiences an event (e.g., death) over time. It adjusts for right-censored data, where not all subjects may be observed until the end of data collection (i.e., subjects drop out from the observations).

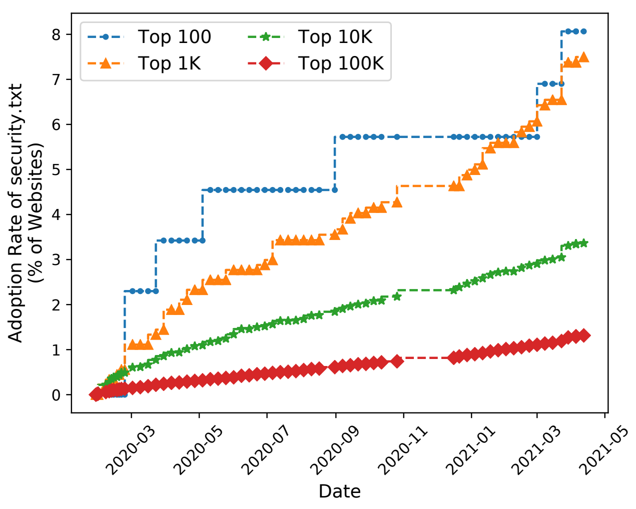

Figure 2: Adoption rate of security.txt over time by the Alexa top 100, top 1K, top 10K, and top 100K websites, using a static set of websites and rankings from our first crawl. As we do not consistently measure this entire set of sites in subsequent crawls, we plot inverted survival curves to account for sites dropping out from our measurements. We lack data points during a measurement outage in November and early December, 2020.

In our scenario, the subjects are individual websites and an event is the adoption of security.txt. We compute the Kaplan-Meier survival curve estimate, which estimates the probability of surviving (i.e., not experiencing the event) over time. In Figure 2, we plot the inverted survival curve, which represents the probability of adopting security.txt over time. We find that by tracking a static set of sites, security.txt is indeed slowly being adopted over time. We observe a higher adoption rate for higher-ranked websites, aligning with our observation that higher-ranked sites exhibit greater security.txt deployment levels.

Removal of security.txt *Files.* For all sites we observed ever deploying security.txt during our observation period, we detect that 5.2% of sites remained online yet removed their security.txt file. This proportion is small but non-trivial, suggesting that some sites may experience negative consequences from deploying the standard, perhaps receiving spam emails or low-quality security reports. Nonetheless, the vast majority of sites that have deployed security.txt continue to support it.

Changes in security.txt *File Content.* We identify that websites occasionally update their security.txt files, with 14.4% of security.txt sites ever changing their security.txt file's content. Upon manual inspection, we observe a wide range of file updates, with approximately half involving minor changes to URLs in the file and a third modifying the contact information.

Common security.txt *File Content.* Over time, we observe that many domains share the same security.txt file. Across our 15-month study, we detect a total of 7.5K distinct domains hosting a security.txt file. These domains form 2.2K clusters with identical file content. From Figure 3 in the Appendix, which plots the CDF of domain cluster sizes, we see that most clusters are small. Singletons make up 81% of clusters, and 95% of clusters have up to 4 domains. However, we do find several large clusters, with the largest containing 2025 domains. By manually inspecting these clusters, we identify that these websites are affiliated with a common organization. For example, the largest clusters are domains affiliated with Tumblr, Google, and Shoptet (a Czech e-shop hosting platform).

In our subsequent analysis, we consider both the number of security.txt domains as well as distinct security.txt files (when relevant, which approximately represents distinct organizations). We note that different domains sharing the same security.txt content do not necessarily represent the same website. For example, doubleclick.net and google.com both are associated with Google and share the same security.txt file, but security issues on one do not necessarily affect the other. Thus it is valuable to consider both domain and file granularities.

4.2 Accessing security.txt

Here, for all domains that we retrieved a security.txt file for, we characterize the location of those files[2].

URL Path. We find that 82% of domains host their security.txt file at the preferred .well-known location (with 65% of sites hosting only at that location). The remaining 18% of websites support security.txt only at the root path. Thus, both URL path locations should be checked when searching for security.txt files.

Protocol. We observe that for security.txt files located both at the root and .well-known URL paths, 90% are accessible over both HTTP and HTTPS, and an additional 3.8% are accessible only on HTTPS. However, the remaining 6.2% can only be retrieved over HTTP, mostly due to the site not supporting HTTPS itself. We note that while the standard requires the file to be accessed only on HTTPS, researchers may need to access it over HTTP as well.

4.3 security.txt Content

As discussed in Section 2.1, security.txt defines eight information fields, of which two are required (Contact and Expires). In Table 1, for all domains hosting security.txt files observed throughout our study's measurements, we list the number of domains and distinct security.txt files providing each field. In this section, we analyze these fields in more detail, focusing on those directly relevant for vulnerability disclosure.

4.3.1 *Contact and OpenBugBounty Fields.*

The Contact field is the most salient field, as the central purpose of the security.txt standard is to standardize the vulnerability reporting process. It is required by the security.txt specification, and multiple contacts can be provided.

Use of the Fields. While most sites provide a single Contact field (83% of domains, 74% of distinct security.txt files), we observe that 9% of domains (11% of files) provide two Contact fields, with a couple of domains providing up to six listed contacts. Despite being a required field, we found nearly 7% of domains and 13% of files without a Contact field listed. However, we identify that a large number of websites (12% of sites and 23% of files) contain a *OpenBugBounty* field that can link to an Open Bug Bounty program URL for disclosure [1]. This field is not defined in the security.txt proposal though. We observe that 92% of domains (84% of files) without the Contact field provide the OpenBugBounty field instead (with 6% of domains and 12% of files listing both fields). A remaining 1% of domains (2% of files) provide no structured contact information.

Field	Num. Domains	Num. Files
Contact	6988 (93.1%)	1912 (87.2%)
Expires	128 (1.7%)	86 (3.9%)
Acknowledgements	675 (9.0%)	282 (12.9%)
Canonical	1184 (15.8%)	477 (21.8%)
Encryption	1561 (20.8%)	782 (35.7%)
Hiring	4473 (59.6%)	570 (26.0%)
Preferred-Languages	1820 (24.2%)	763 (34.8%)
Policy	4005 (53.3%)	680 (31.0%)
OpenBugBounty	924 (12.3%)	495 (22.6%)
Signature	185 (2.5%)	122 (5.6%)
Total	7506	2192

Table 1: Number of security.txt domains and distinct files containing each field. The first two bolded fields are required, and the italicized OpenBugBounty field is commonly provided but not a defined field in the security.txt specification. We also list the number of signed security.txt files (the italicized Signature row), although the signature is not a distinct field.

Field Values. The security.txt specification discusses using email addresses, webpage URLs (e.g., with contact information or with a reporting form), and telephone numbers as contacts. For the Contact field values in our security.txt files, we classify them as emails, URLs, and phone numbers using regular expressions[3].

Of sites providing a Contact field, 49% of domains (75% of files) provide an email address and 49% of domains (18% of files[4]) provide a URL link. Typically only one form of contact is provided, as only 8% of domains (9% of files) provide both an email and a URL link. We also find a small number of telephone contacts (less than 1% of both domains and files), as well as empty or malformed Contact values. Upon manual inspection, we observe that several of the malformed Contact values provided emails in a divided format (e.g., *security [at] example dot org*), which is often done to prevent automatic email extraction by crawlers. This hints at concern that some security.txt adopters may have of receiving spam emails or low-quality reports at their contact points.

[2]In rare cases, a domain provides different security.txt files based on the access method. Thus, researchers may need to occasionally resolve conflicting information (potentially aggregating across files).

[3]We briefly note that security.txt specifies that each contact listed follows the standard URI prefix (i.e., "mailto:" for email, "https://" for webpages, and "tel:" for phone numbers). We find low adherence to these requirements, with over a third of domains that provide an email not using the "mailto:" URI prefix (although 99% of all contact URLs are HTTPS). Thus, these URI prefixes may not serve as reliable indicators for classifying the contact value.

[4]The largest domain clusters sharing identical security.txt files provide URLs, resulting in the significant discrepancy in the number of domains versus the number of distinct files providing a URL contact.

We analyze the most common email usernames across distinct security.txt files, finding that the most popular username by far is *security@*, appearing for 46% of emails. No other username appeared for more than 5% of emails (with the next most popular usernames being *info@*, *support@*, and *admin@*). Interestingly, *abuse@* was not a common username, which has been used by numerous prior operator notification studies [4, 5, 15, 16, 18, 19].

Shifting attention to the most common contact URLs, we observe that 76% of domains list a *hackerone.com* URL, a major vulnerability disclosure coordination and bug bounty platform. We also note that some URLs point to other bug bounty platforms as well as social media accounts (especially on Twitter). Surprisingly, *openbugbounty.org* was listed for only 24 domains' Contact field, indicating that the OpenBugBounty field is typically used instead of the required Contact field for such URLs. We note that all of the OpenBugBounty fields list an *openbugbounty.org* URL (except one domain with an empty field value). For other URLs, it is challenging to analyze their content in an automated fashion, due to the lack of standardized structure. By manually investigating a random sample of these URLs, we observe that some URLs are to webpages with contact forms, while others list additional information about contacting the organization.

4.3.2 *Expires.*

Besides the Contact field, the only other required field is the Expires field.

Use of the Field. Surprisingly, we observe that only 128 (1.7%) domains and 86 (3.9%) distinct files provide an expiration field in their security.txt file, despite being mandated. This makes the Expires field the least utilized field, suggesting that organizations do not find it valuable to consider while likely incurring some management overhead (e.g., in updating the security.txt file once expired).

Field Values. The security.txt specification recommends that the expiration date is no further than a year away, to avoid extensive file staleness. For files providing an Expires date, we compare that date with the latest date that we fetched the file. We find that 83% of files provide an expiration date within 365 days. However, we observe some abnormally distant expiration dates, including 5 domains with expiration periods exceeding a decade. We found only 1 file with a past expiration date, having expired 6 days earlier (but this file was subsequently updated).

4.3.3 *Encryption and Signatures.*

The Encryption field provides the location of an encryption key to use for securing vulnerability reporting communication with the organization. Related, organizations are encouraged to sign their security.txt files, although the signature is not a distinct field (rather the file should be signed with an OpenPGP cleartext signature). We consider both cryptographic-related content types here.

Use of the Fields. We find that 21% of domains (36% of files) provide the Encryption field, although the field is empty for 20% of these domains (and 25% of these files). Signing the security.txt file is uncommon though, with only 185 (2.5%) domains (5.5% of files) containing an OpenPGP cleartext signature.

Field Values. Of domains providing non-empty information in the Encryption field, 91% list a web URL (93% of distinct files do likewise). An additional 4% of domains (3% of files) list an *openpgp4fpr* URI [13], which references a public key's fingerprint for lookup. For remaining domains, we observe other key fingerprinting formats, email addresses (presumably to contact for the public key), human-readable descriptions (e.g., *jcb.com* asking for non-sensitive initial communication), and two domains providing the public key content in ASCII-armored format directly in the Encryption field itself (which is forbidden by the standard). We note that no domains provided a DNS record reference, one of the formats mentioned in the security.txt specification.

For the 544 distinct security.txt files listing a web URL in the Encryption field, we crawled those URLs to try obtaining the public keys (receiving an HTTP error status for 9% of URLs requested). Of the supposed key files fetched, we detect that 29% are not standard OpenPGP public keys. Half of these key files are empty, and upon manual inspection, the vast majority of the remaining half are HTML webpages with more information about the organization's key, often listing the public key within the webpage. Among public keys successfully crawled, 95% are RSA keys (versus DSA keys). We note that all but 5% of keys are greater than 1024-bits, which is recommended for both RSA and DSA. Thus, the vast majority of organizations that do publish a public key provide a strong one.

For the 185 domains with signed security.txt files, we observe that all but 7 use SHA256 or SHA512 as the signature's hashing algorithm (as recommended due to weaknesses in SHA1 and MD5). All but 10 domains supply an Encryption field value, so we attempted to verify these signatures using the public key accessible there. We were not able to fetch a standard public key file for 30 domains providing the Encryption field (17%), and successfully verified the security.txt signatures for 125 domains (71%). Among the remaining domains, we observe cases of failed signature verification as well as malformed key or signature data. Thus, many domains that do sign their security.txt files provide the corresponding public key in the Encryption field, although automated verification encountered a non-trivial number of failure conditions.

4.3.4 *Preferred-Languages.*

As security researchers may come from a variety of backgrounds, organizations can suggest preferred languages for receiving vulnerability reports through the Preferred-Languages field.

Use of the Field. 24% of domains (35% of files) list this field.

Field Values. We observe that 99% of domains (97% of files) list English as one of the preferred languages, the most common by far. Czech was listed the second most frequently for domains (39%), but only for 1% of distinct files[5]. German ranks third among domains (6%) and second among distinct files (9%). This suggests that English serves as a universal language for reporting security concerns (although this field is not present in the majority of security.txt files), aligning with prior investigation into different languages used for security notifications [15].

[5]The discrepancy between domains versus distinct files with Czech as a preferred language stems from the cluster of *shoptet.cz*-associated domains with identical security.txt files, one of our data's largest clusters.

4.3.5 Policy. The Policy field allows an organization to provide further vulnerability disclosure information.

Use of the Field. We find that 53% of domains (31% of files) list a Policy field. However, for 7% of these domains and 27% of distinct files with this field, the value is empty.

Field Values. Of those providing a non-empty Policy field, nearly all (over 99% of both domains and files) list a webpage URL pointing to policy information. We note that several policy URLs are to *hackerone.com* and *bugcrowd.com* pages, both bug bounty platforms, as well as to Github documents. However, no policy URL domain occurred in more than 5% of files, and overall nearly all policy URL domains were distinct. Assessing the policy content on these external URLs in an automated fashion is difficult, as the content is human-readable rather than machine-parsable. We briefly discuss case studies of several domains' policies in Appendix A.

4.3.6 Other Fields. `security.txt` files allow for several other fields that do not relate directly to vulnerability disclosure, but relate to other aspects of security operations. We note that a quarter of distinct `security.txt` files list a Hiring field, indicating demand for hiring security professionals by many organizations. We also observe that 9% of domains (13% of files) provide an Acknowledgement field, hopefully incentivizing/rewarding researchers who responsibly disclose issues. For the Canonical field, 16% of domains (22% of files) list this information.

5 DISCUSSION

In this paper, we conducted a longitudinal empirical evaluation of `security.txt` deployment on top websites. We tracked the adoption rate of the standard over a 15-month period, and characterized the information being provided through `security.txt` files. Our findings highlight issues present in the provided data, as well as lessons for vulnerability reporting, which we discuss further here.

Compliance with the Standard. Through analyzing the content of the `security.txt` files deployed on top websites, we identified various discrepancies between the information provided and what the standard specifies. Most notably, the Contact field provides critical vulnerability reporting information but was not always provided, as required. In many such cases, the `security.txt` file listed a non-standard OpenBugBounty field instead, even though the field is only used to list an *openbugbounty.org* URL that would be suitable for listing in the Contact field. Besides the Contact field, the Expires field is the only other field required. The `security.txt` specification argues that stale `security.txt` information can result in vulnerability reports not being received by an organization or being sent to the wrong contact, thus the need to invalidate old `security.txt` files. Yet we found that few sites provided an expiration date. In fact, it was the least used field by far, suggesting that organizations consider the management overhead of maintaining an up-to-date file as outweighing the risks introduced by outdated data. This calls into question the value of this field, and whether a different mechanism is needed for revoking `security.txt` content. We observed further discrepancies (e.g., empty or malformed fields, incorrect signatures or keys) that highlight a diversity in how `security.txt` manifests in the wild.

Automated Uses of `security.txt`. `security.txt` was explicitly designed to be machine-parsable [7]. As a result, a compelling use case is for large-scale automated vulnerability reporting, which prior research has demonstrated can drive significant levels of remediation, but often encounters challenges with discovering contacts [4, 5, 15, 16, 18, 19]. In addition, `security.txt` provides richer information than data sources used in the prior large-scale security notification efforts, which primarily relied on WHOIS records (now largely gutted of useful contact information due to GDPR [6]). We note though that the various discrepancies in field and value formats discussed throughout our study inhibit reliable automated processing, and security researchers using `security.txt` files for automated notification efforts will need to account for these differences (e.g., searching for contacts in both the Contact and OpenBugBounty fields). In addition, the information on external links (such as in Contact, Encryption, and Policy fields) is directly relevant for vulnerability reporting but rarely machine-parsable, also limiting how effectively `security.txt` files can be used for automated purposes. Expanding the `security.txt` specification to consolidate some of this information directly into the `security.txt` files (while maintaining a machine-parsable format) would better support automated vulnerability notification efforts. While supporting automation may raise concerns about receiving spam, we argue that spammers can already fruitfully mine contact information from existing `security.txt` files, and the further information would support better legitimate reporting.

Lessons for Vulnerability Reporting. The information presented in `security.txt` files provides insights useful for security researchers performing vulnerability disclosure. For example, we identified a diversity in contact types. Besides using email, a researcher may need to reach an organization through a program hosted on a common bug bounty platform, via social media, or through report forms on the organization's website. We found that if using email, the *security@* username was predominant, and may be fruitful to use when attempting to reach administrators of a domain lacking a clear contact (whereas prior notification efforts [4, 16, 18, 19] tested various other generic email usernames such as *abuse@* or *hostmaster@*). We also observed that English appeared to be a nearly-universally accepted language for reporting (aligning with prior security notification findings [15]), and many organizations do support encrypted communication. In addition, many websites publish a vulnerability disclosure policy that researchers should consider when reporting.

Future Directions. Our study provides an initial look at the real-world use of `security.txt`. Future work can continue monitoring its adoption over time, and also investigate the information it provides in more depth. In particular, the external information linked to from `security.txt` files may provide a rich data source to evaluate the types of information organizations prefer to be reported, as well as their vulnerability disclosure policies. Research into automated extraction of this information could help support `security.txt`'s use in large-scale automated notifications (as discussed above). Also, public tools could be developed for web administrators to self-assess the conformity of their `security.txt` deployments. Overall, `security.txt` adoption has been limited to date though, so additional efforts in raising awareness of the standard and incentivizing deployment are needed for it to reach its full potential.

REFERENCES

[1] 2021. Open Bug Bounty. https://www.openbugbounty.org/.

[2] Alexa. 2021. Top Sites. https://www.alexa.com/topsites.

[3] Muhammad Ahmad Bashir, Sajjad Arshad, Engin Kirda, William Robertson, and Christo Wilson. 2019. A Longitudinal Analysis of the Ads.Txt Standard. In *ACM Internet Measurement Conference (IMC)*.

[4] Orcun Cetin, Carlos Ganán, Maciej Korczynski, and Michel van Eeten. 2017. Make Notifications Great Again: Learning How to Notify in the Age of Large-Scale Vulnerability Scanning. In *Workshop on the Economy of Information Security (WEIS)*.

[5] Zakir Durumeric, Frank Li, James Kasten, Nicholas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, David Adrian, Vern Paxson, Michael Bailey, and J. Alex Halderman. 2014. The Matter of Heartbleed. In *ACM Internet Measurement Conference (IMC)*.

[6] Anthony Eden. 2019. GDPR and WHOIS Privacy. https://blog.dnsimple.com/2019/04/gdpr-and-whois-privacy.

[7] Edwin Foudil and Yakov Shafranovich. 2021. A File Format to Aid in Security Vulnerability Disclosure (draft-foudil-securitytxt-11). https://datatracker.ietf.org/doc/html/draft-foudil-securitytxt-11.

[8] C. Lee Giles, Yang Sun, and Isaac G. Councill. 2010. Measuring The Web Crawler Ethics. In *International Conference on World Wide Web (WWW)*.

[9] Github. 2021. GitHub Security Bug Bounty. https://bounty.github.com/.

[10] Google. 2021. Vulnerability Reward Program (VRP) Rules. https://www.google.com/about/appsecurity/reward-program/.

[11] Santanu Kolay, Paolo D'Alberto, Ali Dasdan, and Arnab Bhattacharjee. 2008. A Larger Scale Study of Robots.Txt. In *International Conference on World Wide Web (WWW)*.

[12] Martijn Koster, Gary Illyes, Henner Zeller, and Lizzi Harvey. 2020. Robots Exclusion Protocol (draft-koster-rep-04). https://datatracker.ietf.org/doc/html/draft-koster-rep.

[13] Wiktor Kwapisiewicz. 2021. openpgp4fpr URI scheme. https://metacode.biz/openpgp/openpgp4fpr.

[14] IAB Technology Laboratory. 2021. Ads.Txt - Authorized Digital Sellers. https://iabtechlab.com/ads-txt/.

[15] Frank Li, Zakir Durumeric, Jakub Czyz, Mohammad Karami, Michael Bailey, Damon McCoy, Stefan Savage, and Vern Paxson. 2016. You've Got Vulnerability: Exploring Effective Vulnerability Notifications. In *USENIX Security Symposium*.

[16] Frank Li, Grant Ho, Eric Kuan, Yuan Niu, Lucas Ballard, Kurt Thomas, Elie Bursztein, and Vern Paxson. 2016. Remedying Web Hijacking: Notification Effectiveness and Webmaster Comprehension. In *World Wide Web Conference (WWW)*.

[17] Mark Nottingham. 2019. RFC 8615: Well-Known Uniform Resource Identifiers (URIs). https://datatracker.ietf.org/doc/html/rfc8615.

[18] Ben Stock, Giancarlo Pellegrino, Frank Li, Michael Backes, and Christian Rossow. 2018. Didn't You Hear Me? Towards More Successful Web Vulnerability Notifications. In *Network and Distributed System Security Symposium (NDSS)*.

[19] Ben Stock, Giancarlo Pellegrino, Christian Rossow, Martin Johns, and Michael Backes. 2016. Hey, You Have a Problem: On the Feasibility of Large-Scale Web Vulnerability Notification. In *USENIX Security Symposium*.

[20] Yang Sun, Ziming Zhuang, Isaac G. Councill, and C. Lee Giles. 2007. Determining Bias to Search Engines from Robots.txt. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*.

[21] Yang Sun, Ziming Zhuang, and C. Lee Giles. 2007. A Large-Scale Study of Robots.Txt. In *International Conference on World Wide Web (WWW)*.

[22] Bryan Ware. 2021. Improving Vulnerability Disclosure Together (Officially). https://www.cisa.gov/blog/2020/09/02/improving-vulnerability-disclosure-together-officially.

[23] Wordfence. 2021. Wordfence: WordPress Security Plugin. https://www.wordfence.com/.

A ORGANIZATION VULNERABILITY DISCLOSURE POLICIES

Here we provide a brief case study of several organizations' vulnerability disclosure policies, based on the webpages linked to in the organization's `security.txt` Policy field.

- *login.gov*: The policy for the US government website lists the types of security research permitted and the systems that can be evaluated, details on how to report vulnerabilities (including the information to provide), and the 90-day coordinated disclosure timeline.
- *walmart.com*: Walmart's policy page discusses supporting responsible disclosure without the threat of legal action when complying. It additionally lists actions that would be considered non-compliant with responsible disclosure. It provides a form for a Bugcrowd vulnerability report, giving clear guidance on the information requested.
- *wordfence.com*: Wordfence, the popular WordPress security plugin [23], provides a policy page with instructions on how to contact its security team (including encouraging encrypting email communications), and the step-by-step disclosure and remediation process (including permissible vulnerability publication should a vulnerability report remain unacknowledged).

For several domains with well-known in-house bug bounty programs, such as Google [10] and Github [9], we observe their `security.txt` files' policy URLs linking to their own bug bounty programs' pages.

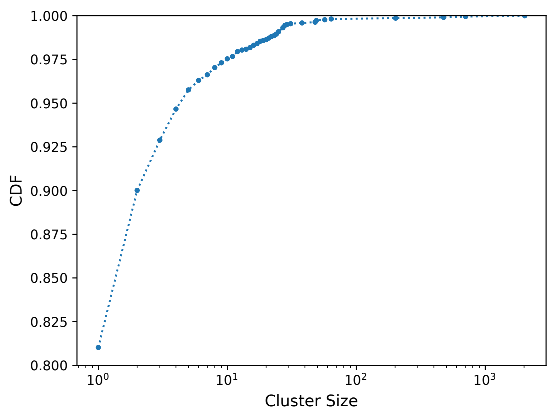

Figure 3: CDF of domain cluster sizes, where clusters are domains sharing identical `security.txt` files. Note that the y-axis begins at 0.8.