security.txt Revisited: Analysis of Prevalence and Conformity in 2022

TOBIAS HILBIG, THOMAS GERAS, ERWIN KUPRIS, and THOMAS SCHRECK,
HM Munich University of Applied Sciences, Germany

Determining the correct contact person for a particular system or organization is challenging in today's Internet architecture. However, there are various stakeholders who will need to have such information, such as national security teams, security researchers, or Internet service providers, among others. To address this problem, RFC 9116, or better known as "security.txt," was developed. If implemented correctly, then it can help these stakeholders in finding contact information to be used to notify an organization of any security issues. Further, there is another proposal called "dnssecuritytxt," which uses DNS records for this purpose.

In this research article, we evaluated the prevalence of websites that have implemented security.txt and their conformity with the standard. Through a longitudinal analysis of the top one million websites, we investigated the adoption and usage of this standard among organizations. Our results show that the overall adoption of security.txt remains low, especially among less popular websites. To drive its acceptance among organizations, security researchers, and developers, we derived several recommendations, including partnerships with vendors of browsers and content management systems.

CCS Concepts: • **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; **Network security**;

Additional Key Words and Phrases: Security.txt, internet scanning, vulnerability disclosure, Incident Response

ACM Reference format:
Tobias Hilbig, Thomas Geras, Erwin Kupris, and Thomas Schreck. 2023. security.txt Revisited: Analysis of Prevalence and Conformity in 2022. *Digit. Threat. Res. Pract.* 4, 3, Article 36 (October 2023), 17 pages.
https://doi.org/10.1145/3609234

1 INTRODUCTION

In the age of rapidly evolving digital information systems, the ability of security researchers to quickly and easily notify an organization of any security issues they may have discovered is crucial. Organizations should help benevolent security researchers in effortlessly disclosing security issues to prevent threat actors from abusing them. However, these researchers are presented with another challenge once they find a vulnerability: "How do I contact someone responsible in case I want to report a security issue that I found in their product?"

There are various methods that security researchers might use to notify the responsible party about security issues. These include messaging standard email addresses such as *security@*, using contact forms on an organization's website, or searching for contact information in centralized databases like FIRST's Incident Response

T. Hilbig, T. Geras, E. Kupris, and T. Schreck contributed equally to this research.

Authors' address: T. Hilbig, T. Geras, E. Kupris, and T. Schreck, HM Munich University of Applied Sciences, Munich, Germany; emails: tobias.hilbig@hm.edu, thomas.geras0@hm.edu, erwin.kupris@hm.edu, thomas.schreck@hm.edu.

2576-5337/2023/10-ART36
https://doi.org/10.1145/3609234

database [7]. All of these options, however, suffer from shortcomings and a lack of adoption within the IT landscape [16, 22, 23].

An Internet-Draft titled "security.txt" was published in 2017 by Edwin Foudil. It describes a simple text file that can be placed on an organization's website and aims to resolve the issues found with previous methods. By providing all the necessary information in a standard format and location, both humans and automated software can leverage it for vulnerability notifications. Over the years, lots of prominent websites have adopted security.txt and government agencies around the world started to recommend its usage [10]. In April of 2022, the IETF standardized security.txt as RFC 9166 [9].

Our work aims to evaluate the prevalence and correctness of security.txt usage on the Internet in 2022. Therefore, this article presents the first study on security.txt after its adoption as an RFC. Consequently, we want to answer two research questions: (1) How did the adoption rate of security.txt change during 2022? (2) How well do deployed security.txt files adhere to the standard?

For the first question, we scanned the one million highest-ranked websites based on the Tranco list [15] over a time of 55 weeks and evaluated the percentage of websites that deploy a security.txt file. To answer the second question, we analyzed all downloaded files, evaluating the standard's technically and semantically correct usage. We further evaluated a new standard called "dnssecuritytxt" [3].

The remainder of this article is structured as follows: We present background information on security.txt and alternative methods for vulnerability disclosure in Section 2 and examine related work in Section 3. Our methodology and the scanner's architecture are explained in Section 4. We show our results in Section 5 and discuss them along with limitations and future work in Section 6. Finally, Section 7 concludes this article.

2 BACKGROUND

Various methods have been developed for vulnerability disclosure over time. In addition to these, we detail the security.txt standard in the following sections. All of these options have one or more shortcomings: Without prior experience, researchers might not know specific methods exist, and it can be hard to select a suitable method. Furthermore, these methods did not gain enough traction within the IT industry so that security researchers can reliably use them as the standard way for vulnerability notifications.

2.1 RFC 2142 Emails

Standard email addresses, such as *security@*, *abuse@*, *info@*, *webmaster@*, and so on, might be used to disclose vulnerabilities. These email addresses were standardized in 1997 in RFC 2142 [5]. However, there is no guarantee that an organization uses or maintains standard email addresses or that these highly sensitive emails reach the right personnel. Previous studies have shown that their adoption is fairly low even though the standard has been around for over two decades [22, 23].

2.2 SECURITY.md

GitHub pioneered another method of publicizing information on the vulnerability disclosure process. Here, software projects might place a *SECURITY.md* file in the top-level directory of their repository, which details the project's rules and process for disclosing vulnerabilities [11]. Similarly, a project might deploy a *BUG-BOUNTY.md* file to draw attention to its bug bounty program and clarify its expected procedure. *SECURITY.md* files can be a great resource for manual disclosure but lack a standard format allowing for automated notification campaigns.

2.3 Contact Forms and Social Media

Other methods of directly contacting an organization can include using a contact form on their website or reaching out to social media accounts. However, locating and filling out contact forms on websites and messaging social media accounts is a cumbersome process that does not scale well for automated disclosures.

2.4 dnssecuritytxt

dnssecuritytxt is an emerging method to publish contact information via DNS and was published on March 25, 2021. The admin can either use a subdomain called *_security* or add the information to the apex of the domain. These two options are discussed in detail in Reference [3].

The mandatory *security_contact* record is supposed to contain contact information while the *security_policy* record should include a link to a security policy [3]. Similar to security.txt, the contact information must contain either an email address or a URL to a website. If more communication channels are possible, then multiple records may be added. As for the *security_policy*, it is not specified how to handle multiple entries. For both fields, the information is published as a *TXT* record that is limited to 255 characters.

2.5 Centralized Methods

Security researchers might use centralized databases to find contact information as an alternative to the stated decentralized methods. For instance, the WHOIS database might include the contact information of the domain operator. Nowadays, most of the information is protected by privacy laws such as the GDPR in the European Union [24]. The **Forum of Incident Response and Security Teams (FIRST)** maintains a database with information about **CERTs/CSIRTs (Computer (Emergency/Security Incident) Response Team)** of member organizations [7]. FIRST offers a beta version of a public API for security researchers to access their database to find the relevant member CERT/CSIRT. A FIRST membership comes with significant costs and the requirement to operate a CERT/CSIRT. This is one of the reasons why this option is not feasible for smaller organizations.

2.6 Security.txt

RFC 9116 defines a standardized way for organizations to publish information about their vulnerability disclosure process through a text file called security.txt placed on their website. Researchers who discover a security issue related to the organization can use this information to easily find the correct contacts to notify about their findings. The usage of security.txt was first proposed as an Internet-Draft by Edwin Foudil in 2017 and quickly gained traction in the information security industry [4]. In April 2022, security.txt took the second step of the IETF standardization process [2] and was promoted to an RFC [9].

Security.txt was designed to be human-readable and parsable by machines at the same time to enable large-scale vulnerability notification campaigns. Therefore, the standard specifies two locations for security.txt files: the root directory, i.e., *domain.tld/security.txt*, or the *.well-known* directory as defined in RFC 8615 [17], i.e., *domain.tld/.well-known/security.txt*. In case security.txt is present in both locations, the file in the well-known directory must be preferred [9].

The RFC strictly enforces the use of HTTPS when retrieving the file to guarantee its integrity. Additionally, every web URI in the security.txt file must start with *https://* to ensure this requirement is met along the process of gathering information. The content type must be set to *text/plain* and the file must be encoded as *utf-8*. Finally, a security.txt file is only valid for the exact domain where it is found and not for any of its sub- or parent-domains [9].

Each line of a security.txt file contains a key-value-pair separated by a colon, a comment denoted by a hashtag sign at the start of the line, or an empty line. Additionally, the file can be signed using PGP. The RFC currently lists eight different keys, or so-called “fields.” Only the *Contact* and *Expires* fields are mandatory, while all other fields are optional. The following lists the standard fields according to the RFC [9]:

Contact: This field can contain links to contact web pages, email addresses and telephone numbers. The value of the contact field needs to be a syntactically correct URI as described in RFC 3986 [1]. Therefore, contact values usually start with *https://*, *mailto:*, or *tel:*. Multiple Contact fields are allowed and are ordered by preference.

Expires: This field contains the time and date the security.txt file expires at. It should be formatted according to ISO.8601 as defined in RFC 3339 [14]. An expired security.txt file can be considered stale and should not be trusted anymore. The RFC recommends for expiry dates to not exceed a date of twelve months into the future. This field must not appear more than once per security.txt file.

Canonical: The location of the security.txt file on this particular web server as a web URI. This serves as an additional trust mechanism if the file is digitally signed.

Policy: Here, an organization can link to its vulnerability disclosure policy so researchers can adhere to their specific process and report issues responsibly.

Encryption: This field references the key that should be used to encrypt messages to the organization. The security researcher has the additional responsibility of authenticating the key. The RFC specifies that the encryption key itself should not be used as the value for this field. Instead, a reference to the key file should be used, such as a web link, the OPENPGPKEY DNS record of the domain, or the key's fingerprint.

Preferred-Languages: A comma-separated list of languages that the organization prefers when receiving vulnerability notifications. Languages should be formatted as short-form tags defined in RFC 5646 [18] and do not have to be ordered according to preference. This field must appear at most once per file.

Acknowledgments: This field contains a link to a web page where the organization recognizes the help of security researchers by listing their names and security-related contributions.

Hiring: This field can be used to link open job postings in security-related positions at the particular organization.

The RFC allows for additional fields to be added in the future. This process is handled by the **Internet Assigned Numbers Authority (IANA)**, which has a registry of valid fields for a security.txt file [13]. Finally, the standard recommends digitally signing the security.txt file using an OpenPGP clear-text signature. In this case, the *Canonical* field is recommended to be included in the file so its location can be authenticated as well.

3 RELATED WORK

To the best of our knowledge, only two studies have evaluated the adoption of security.txt across the Internet.

Poteat and Li [19] analyzed the existence and contents of security.txt files and monitored their adoption rate over a span of 15 months from January 2020 to April 2021. They found that security.txt files were deployed on only a small percentage of websites, with higher-ranked ones being more likely to adopt the standard. Their results show that 16% of the top 100, 10% of the top 1k, 4% of the top 10k, and around 1% of the top 100k websites deployed security.txt. These percentages denote the maximum deployment rate per grouping throughout their study.

In contrast to our work, the authors used a dynamic set of websites taken from the Alexa Top 100k list, which is subject to frequent change and portrays a comparably small view of the overall Internet landscape. Additionally, their longitudinal investigation was conducted before security.txt became an Internet-Standard and during a period when the draft changed regularly. For example, the *Expires* field was made mandatory in version 10 of the draft, which was published in August 2020 [8], right in the center of their measurement period. This might have made evaluating the compliance of security.txt files harder, because website operators might not have caught up with the most recent version of the draft yet.

Similar to our investigation, Findlay and Abdou [6] scanned a static set of websites using the Tranco list [15]. Their research focused on the overall adoption of security.txt and the websites' compliance across different versions of the Internet-Draft. Along with results comparable to the study of Poteat and Li [19], they found that most websites implemented outdated versions of the security.txt draft. However, this second, more recent study was conducted over the span of only one week at the end of 2021, with one crawl per day. As their results show no significant changes between these crawls, their analysis is based only on the last scan. This study could therefore not determine any long-term changes in the adoption rate of security.txt.

Before security.txt was introduced as an Internet-Draft, security researchers had to rely on other means of finding contact information of system administrators to notify them about vulnerabilities they found. In their 2016 study, Li et al. [16] evaluated different means of notifying administrators about vulnerabilities found in their systems. They differentiated between direct communication channels, such as mailing the contact found

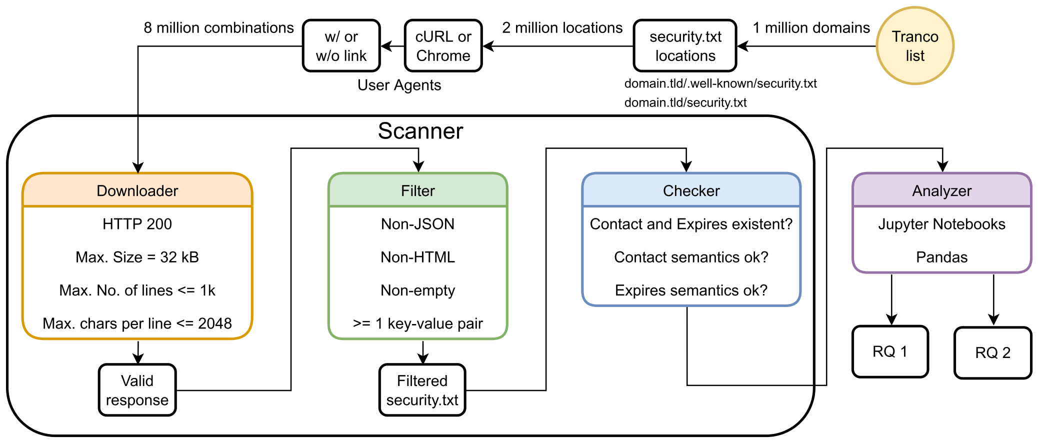

Fig. 1. Pipeline showing the architecture of our scanning and analysis framework.

in the WHOIS database, and indirect channels, i.e., using an intermediary, such as national CERTs. Their results show that directly reaching out to the WHOIS contacts with detailed information results in higher remediation rates compared to using indirect channels.

Stock et al. [23] identified numerous vulnerable websites and examined additional channels to contact the website operators, e.g., the generic email addresses *security@*, *abuse@*, and *webmaster@*. Similar to Li et al., they found that these direct communication channels via email were more effective, because intermediaries may not relay the disclosed information to the website operators. However, their results show that these standardized email addresses were only scarcely adopted, especially by less popular websites, which led to an unsatisfactory remediation rate.

In a subsequent study in 2018, Stock et al. [22] evaluated obstacles in large-scale vulnerability disclosure campaigns. Building on previous works, they only used direct communication channels. They explored additional ones besides emails, such as filling out contact forms, mailing letters to organizations' postal addresses, messaging organizations' social media accounts, and calling publicly available telephone numbers. They found these additional channels to require a lot of manual effort while at the same time resulting in too little of a success rate and therefore determined them to be ineffective from a cost perspective.

The three aforementioned studies on alternative methods for vulnerability disclosure stress the need for a standard way of establishing a communication channel between security researchers and operators—a problem that security.txt aims to resolve.

4 METHODOLOGY

This section describes our methodology by giving the necessary definitions and explaining the architecture we built for our scanner. A detailed explanation for each step of the pipeline is given in the following sections. Figure 1 visualizes the overall process. Our scanning framework consisted of four components: The downloader ingested a list of URLs and a set of user agents, tried to download each file, and saved all results for later analysis. The filter module removed non-security.txt files from further processing. Then, the checker module extracted all information from the filtered set of files. Finally, the analyzer component was used to aggregate the information for presentation and visualization. All components were custom-developed using Python 3. For the final analysis, Jupyter notebooks were used.

4.1 Hardware and Infrastructure

All scanning and analysis tasks ran on one virtual machine with 16 cores and 96 GiB RAM. The virtual machine ran Debian 11 on Kernel 5.10, with automatic security and software updates enabled. The hypervisor was running on dual socket AMD EPYC 7352 24-Core CPUs. The Internet connection with a symmetrical speed of 2.5 Gbps was provided by the Leibniz Supercomputing Centre. The DNS resolver was provided by the Munich University of Applied Sciences.

4.2 Timeline and Procedure

Our scans ran from December 26, 2021 to January 15, 2023, covering 55 weeks with 42 successful scans. We automatically started scans at midnight on Saturdays, except for one scan on October 23, 2022, which was started manually at 5:58 p.m. Each scan lasted about 30 h using the aforementioned hardware. We used the Tranco list[1] [15], generated on February 6, 2021, as the basis for our scanning. This list consists of the top one million websites together with their rank. The time limit for each request was set to 10 s, a value empirically determined in a previous study; see Findlay and Abdou [6]. Our list was generated nearly 12 months before we began the scanning process. This decision was made on purpose, as we wanted to compare our results to previous (unpublished) work at that time. The Tranco list was created with the explicit goal of ensuring the stability of the list over time. We calculated the size of the intersection set of domains between our list and the list from December 26, 2021, for three groups: top 10k (8,448 entries), top 100k (74,142 entries), and top 1M (683,537 entries). As the intersection rate is quite high, especially at the top of the list, we conclude that using a more recent list would have only marginally altered the results.

4.3 File Locations

We attempted to download security.txt files from both allowed locations: *domain.tld/.well-known/security.txt* and *domain.tld/security.txt*. For our analysis, both locations were merged. In case a file was found in one location only, this file was used. If files were found in both locations, then we used the one from the *.well-known* directory. In accordance with the RFC, we only issued requests using the HTTPS protocol.

4.4 User Agents and Opt-Out Process

We used two user agents for scanning: Chrome[2] and cURL.[3] Each user agent was used with and without a suffix containing a URL pointing back at our scan server.[4] As the user agent is commonly logged by web servers, administrators were able to visit our web page for additional information about our research and contact details. Via this mechanism, organizations could decide to opt-out of future scans. During the scanning period, one domain operator decided to do so. Therefore, this domain was excluded from future scans and all analyses. We used no other HTTP headers in the request except for *Accept: */**.

4.5 Definitions

One might argue that a security.txt file can only be considered to be valid if it fully conforms to the ABNF definition in Section 4 of the specification [9]. Doing so would exclude a large number of files with minimal syntactical or semantical errors from the analysis. Such files could still be processed by both humans and less strict machine parsers. Therefore, our definitions are as follows:

valid response: A response from a web server that was considered to be valid by the *downloader* module, as described in Section 4.6. In this case, the request was successful, a file was downloaded, and it did not exceed the size limitations of the security.txt specification.

filtered security.txt file: A security.txt file that passed the *filter* module, as described in Section 4.7. The filter module removed files that are no security.txt files, such as HTML pages, JSON objects, and binary data. It also checked for at least one key-value-pair, therefore these files were highly probable to be security.txt files.

[1] Available at https://tranco-list.eu/list/9622
[2] Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36.
[3] curl/7.80.0.
[4] ### This is a crawler for security.txt files. For more information visit https://securitytxt-scan.cs.hm.edu/###

4.6 Downloader

The downloader module ingested the list of URLs to be scanned. The order of the list was randomized before the download procedure started. Eight requests were issued in sequence for each domain, one for each combination of user agent and file location. A response from a web server was considered to be valid if the HTTP response code was 200 and the body of the response was a syntactically correct security.txt file as stated in Section 5.4 of the RFC [9], i.e., the number of lines was at most 1,000, the number of characters per line was at most 2048 and the overall size was at most 32 KBs. Furthermore, any server response was downloaded so that no pre-filtering by Content-Types occurs. This ensured that no security.txt files were excluded because of misconfigured servers. Erroneously included non-security.txt files were later excluded by the *filter* module. The download was done with 200 threads in parallel. This number was empirically determined – fewer threads reduced the download speed, and more threads only increased the memory footprint of the application. In addition to the retrieved files, we stored response codes, user agents, locations, and the exact time of each request in a CSV file for later analysis.

4.7 Filter

After scanning, retrieving, and downloading the requested sites' content, it was essential to remove all non-security.txt files to answer the first research question. Removing non-security.txt files was performed as a three-stage process in the filter module. The filter module ingests all downloaded files and sorted them into five categories: security.txt files, empty files, files containing comments, files containing the at sign, and all other files.

The python-magic [12] library was utilized in the first step of the filtering process. All files detected as HTML, Image, GIF, SVG, JSON, XML, PHP, SGML, or binary data were removed. Empty files and files containing only white space were also excluded in this step.

Four methods were employed in the second step for more in-depth content analysis. The first method utilized the python JSON library. If no error occurred while loading a file, then the file was assumed to contain a valid JSON object. Additionally, the second method was built upon the beautiful soup library, which reviewed if a file contains HTML tags. Furthermore, the third method checked whether files consisted only of white space characters. Finally, the fourth method checked if a file conforms to the typical security.txt structure, i.e., if it contained at least one colon.[5] All files except the ones detected with the last method were excluded from further analysis.

Ultimately, files that did not pass the regular expression method were analyzed in a third step for the existence of hash or at signs. In case such characters were found, they were added to the first category of files. All other files were saved for manual screening. Analysis of a random sample of 157 files from the first scan confirmed that the filter model worked as designed. None of the excluded files were security.txt files, although most of the files containing an at sign indeed contained at least an email address that could be used to contact the respective organization. This filtering process guaranteed that no security.txt files were removed erroneously. Consequently, after filtering all the downloaded files, we assumed that the remaining files were security.txt files with a high probability.

4.8 Checker

To answer the second research question, the security.txt files had to be examined from a semantic perspective. This investigation was conducted through the semantic checker module. This module ingested all files the *filter* module classified as security.txt files. The required fields *Contact* and *Expires* were extracted in the first step of the semantic checker module. Since it is permitted to specify multiple *Contact* fields, all *Contact* values were extracted and separated into emails, URLs, and telephone numbers. The values of the optional fields were also extracted, though not validated for their semantic correctness. After the extraction, the semantic checks for the

[5]Via the following regular expression: (.*):(.*).

email and URL values in the *Contact* field and the value of the *Expires* field were conducted. This module stored all extracted information in a CSV file for later analysis.

For the semantic check of the emails, several regular expressions were used. The outcome from the semantic check of an email could either be *whitespaces*, *brackets*, *missing_mail_to*, *invalid*, or *valid*. *Whitespaces* meant that there were spaces in-between *mailto:* and the email. Sometimes, *at/dot* or *[at]/[dot]* were used instead of at signs or dot characters. Therefore, they received the status *brackets*. *Missing_mail_to* indicated that the mandatory prefix *mailto:* was not present. *Invalid* emails did not pass the semantic check, and *valid* emails fulfilled all requirements of the RFC. All email values except those classified as *invalid* can be read by humans.

Like the emails, URLs were also analyzed semantically using several regular expression patterns. The result of the semantic check of an URL was *http* when the URL contained *http://*. However, if neither *http://* nor *https://* was detected, then the result of the semantic check was *no_protocol*. Furthermore, the outcome could be *invalid* when an URL did not pass the semantic check. Nonetheless, if all requirements were satisfied, then the result was *valid*.

The dateutil library was used besides regular expressions for the semantic check of the *Expires* field. The *Expires* field was investigated semantically for two aspects. The initial analysis was concerning the format. An expiring date could be *rfc3339_conform*, *parsable*, or *invalid*. *Expires* fields that passed the regular expression were classified as *rfc3339_conform*. The remaining ones were analyzed with the library. If the library was able to detect a date, then it was classified as *parsable*, otherwise as *invalid*. The second analysis evaluated whether the specified expiring date was *expired* or *not_expired*. The difference between the expiring date of the security.txt file and the timestamp of the file retrieval was calculated with second precision. Since fields classified as *invalid* could still be human-readable, the results of this semantic check must be considered as a lower bound.

We did not consider the rest of the file, including the optional fields, as relevant for our definition of a valid security.txt file. Thus, we did not semantically check the remaining fields or invalidated files based on additional, non-standardized fields.

An additional step in evaluating the compliance of security.txt files would be to verify the signatures some deployments use. The reasons why we chose to not include signature validations in our methodology were twofold: On the one hand, the standard recommends using clear text OpenPGP signatures but does not enforce this. Any other signature method might be used instead without violating the formatting rules the RFC introduces. On the other hand, RFC 9116 explicitly states that security researchers must not assume that the key that might be referenced in the *Encryption* field is the one being used for signing the security.txt file. Instead, any other key could theoretically be used for signing the message without any reference in the file. The added value of this analysis could also be considered limited, as manually checking a small, random subset of the downloaded files revealed that nearly all of them are unsigned.

4.9 Analysis

The last step of our pipeline was the analysis module. To that end, three Jupyter notebooks employing the pandas library were used. The first notebook calculated all information related to the first research question based on the results of the filter module, i.e., the deployment rates. The second and third notebook aggregated the results of the checker module, allowing us to answer the second research question.

4.10 dnssecuritytxt

A different approach was employed to analyze dnssecuritytxt. Analyzing DNS data, especially historical DNS data, is often done using so-called passiveDNS databases. In passiveDNS, sensors are deployed globally next to DNS resolvers, preferably large DNS resolvers. Those sensors record the DNS resolver's answers and save this information in a database. Afterwards, this database can be queried to see the record name, record type, and record data. The caveat of this approach is that domains that have never been queried are not part of the database.

Table 1. Number of Filtered Security.txt Files by User Agents and Location, Summarized Over All 42 Scans

Location	User Agent	w/ link	w/o link	Difference
.well-known	cURL	208,960	208,565	−0.19%
	Chrome	209,116	210,992	0.90%
root	cURL	141,291	142,201	0.64%
	Chrome	141,358	143,369	1.42%

However, this approach gave us the opportunity to query all records in the passiveDNS database without limiting ourselves to a fixed list of domains.

For our research, we were kindly given access to an extensive passiveDNS database operated by the company Farsight [21]. We queried information regarding dnssecuritytxt using the open-source tool *dnsdbflex* [20]. We used the timeframe between October 1, 2022 and January 28, 2023 for our queries and analyzed the answers.

5 RESULTS

The following presents our results for both research questions we defined. We first investigate how to best access the security.txt files as well as their overall deployment rate before validating their contents semantically in Section 5.3.

5.1 Access

Successfully retrieving security.txt files depends on a number of factors. Therefore, we analyzed how the user agent impacts the response rates. Our analysis also includes a comparison between the allowed locations at which a security.txt file can be deployed.

User Agents: Over all scans with all user agents and both locations, the average success rate, i.e., the number of valid responses as defined in Section 4.5, was 3.8%. The highest success rate was achieved using the cURL user agent and downloading the security.txt file from the root directory at 4.1%. With cURL as user agent, a link to our web page and targeting the *.well-known* directory, the success rate 3.5%. While cURL without a link received the most technically correct responses, it was essential to consider the subset of files that were also semantically correct security.txt files as explained in Section 4.7.

The number of received, filtered security.txt files by user agent and location combination, based on all 42 scans, is shown in Table 1. We state that, in general, retrieval without a link to our scan server was more successful, with around 1 percentage point more files retrieved, the only exception being the cURL user agent at the root directory, where retrieval was 0.19% points less successful. Retrieval using Chrome as user agent was more successful with a 0.5% point higher retrieval rate. Taken together, scanning without a link and with the Chrome user agent resulted in 1.1% points higher retrieval rate. All following analysis is therefore based on using the Chrome user agent without a link to our web page.

Figure 2 depicts the types of errors we observed during the latest scan on January 15, 2023. All percentage values are based on the total number of eight million requests. The errors are grouped by type: *Request Errors* denotes cases where the request itself failed to complete. These failures are split to show exact error codes from the *python-requests* library. *Request Success* shows cases in which the request completed successfully, i.e., the web server returned some data. More than 50% of all requests contained error codes in the response. These are grouped according to their HTTP status code. In about 10% of cases, a file was transmitted. This group is further detailed, showing the result of the syntax check. The figure shows that about one-third of requests failed without any valuable data being transferred. Most successful requests received HTTP 4XX response codes. About half of the files transferred failed the syntax check. Finally, about 4.5% of requests ended with a valid response as defined in Section 4.5, i.e., files that could be security.txt files.

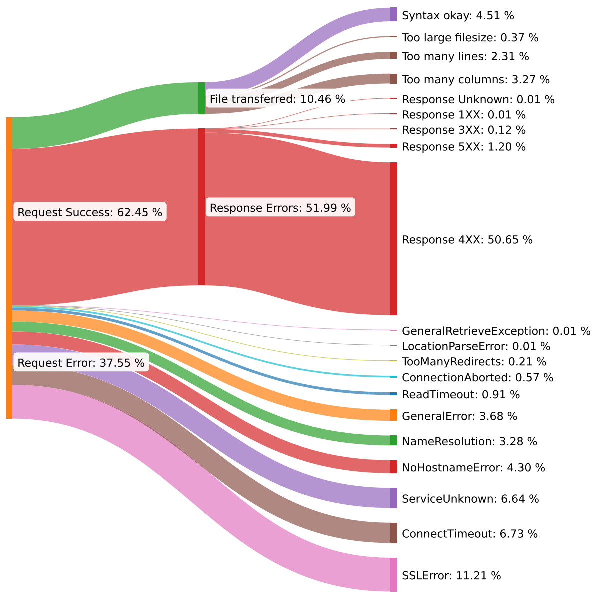

Fig. 2. Error types observed during the latest scan on January 15, 2023.

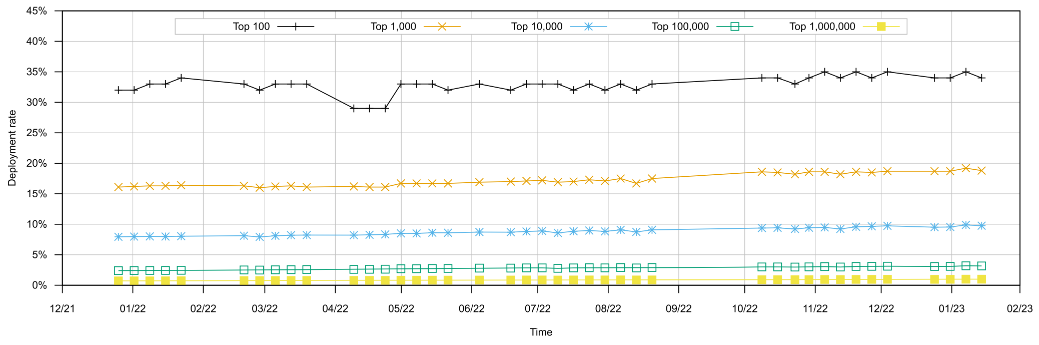

Fig. 3. Deployment rate of security.txt files, according to rank groups.

Location: During our latest scan on January 15, 2023, we received security.txt files from 8,446 domains. On 2,394 (28.3%) web servers, the file was only available in the root directory, on 4,550 (53.8%) it was only located in the *.well-known* directory. On 1,502 (17.7%) web servers, the file was offered in both locations.

5.2 Deployment

For a more detailed representation of the results, the deployment rates were analyzed as groups of the top 100, top 1k, top 10k, top 100k, and top 1M websites by rank.

Figure 3 shows an overview of the deployment rates. The x-axis shows the time, and the y-axis the percentage of websites with a filtered security.txt file. Table 2 shows a comparison of the deployment rates per group at out first and our last scan. Additionally, the changes in deployment rates between these scans are listed, using both absolute numbers and percentages relative to the group's size. In the top 100, security.txt was deployed, on average, in 32.7% of websites, a percentage that remained largely stable throughout our study. Small variations might be attributed to temporary downtimes, e.g., for maintenance reasons.

Table 2. Comparison of Deployment Rates of Security.txt Files, According to Rank Groups

Group	First	Last	Change (rel)	Change (abs)
100	32.0%	34.0%	6.3%	2
1k	16.1%	18.8%	16.8%	27
10k	7.9%	9.8%	22.9%	182
100k	2.4%	3.2%	31.3%	763
1M	0.7%	1.0%	28.6%	2,803

First and *Last* refer to the first and last scans on December 26, 2021 and January 15, 2023.

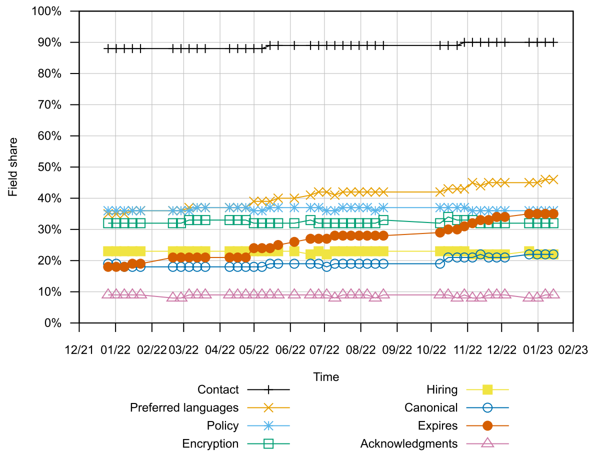

Fig. 4. Distribution of fields found in security.txt files.

It should be noted that some of the scans did not complete successfully, resulting in gaps in the figure, most notably around September 2022. We adapted our implementation of the downloader component to reduce the likelihood of such failures for future scans.

After security.txt became an RFC in April 2022, we saw a slight but steady increase in the adoption for all groups except the top 100 websites. Overall, a notable increase in adoption during the 55 weeks long scanning period can be observed. While the adoption rate of higher-ranked websites in the top 100 groups only increased by 6.3%, the top 100k group increased by 31.3%.

We also analyzed the deployment of dnssecuritytxt. For the subdomain *_security.* we found two organizations using this record to provide contact information. We conclude that this option of the standard is not widely known in the community. The usage of dnssecuritytxt in the apex of a domain is higher. In total, we found 63 records with *security_mail* information and 24 records for *security_policy*. The analysis of the entries shows that a total of 66 individual organizations are using this standard. However, some organizations only publish *security_policy* but no contact details. This violates the current draft's specification.

5.3 Content

Several aspects of the deployed security.txt files were investigated to answer our second research question.

Existence of Fields: Figure 4 shows the frequency of the existence of all fields specified in the standard. The x-axis shows the time, while the y-axis shows the percentage of security.txt files of each scan containing the fields. A *Contact* field existed in an average of 89% of all files. This percentage was very stable over the whole

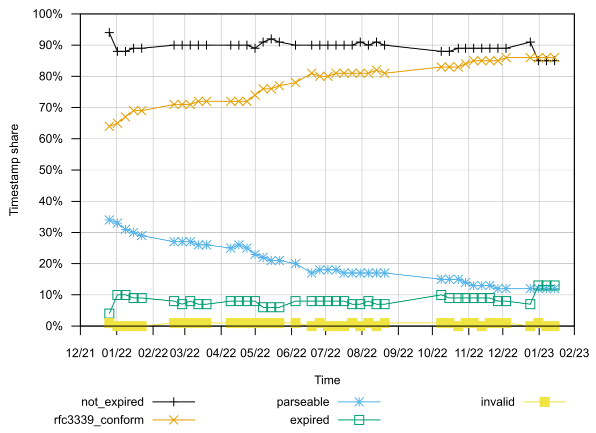

Fig. 5. Results of the expiry date evaluation.

scanning period. The *Expires* field, introduced in a later version of the standard, was found in 18% of the collected security.txt files in the first scan and in 35% in the latest scan. This shows that there is a slow adaption of the new version of the standard. There were a few select cases in which a security.txt file only had an *Expires* and no *Contact* field. However, we only saw this 18 times per scan on average and thus determined this case to be insignificant for our analysis.

For the optional fields, the *Preferred-Languages* field was most common, with presence increasing rapidly from 35% of security.txt files at the beginning of our study to 46% in the latest scan. The second most commonly used field was the *Policy* field, starting at around 36% but remaining comparably stable with a maximum of 38%. The field *Encryption* was present in 32% of security.txt files, also remaining fairly constant throughout our investigation. Thus, its adoption rate was recently surpassed by the rate of the *Expires* field. The fields *Hiring* and *Canonical* were present in an average of 23% and 19% of security.txt files, respectively. Both of their deployment rates remained fairly stable as well. Finally, the least common field was *Acknowledgements*. Its adoption rate remained largely stable between 8.3 and 9.1%.

Conformity of Contact Fields: Besides investigating the existence of required and optional fields, the values of the *Contact* field were analyzed. We found that only 0.6% of files, on average, listed a telephone number as a contact. This percentage largely remained stable, ranging between 0.6% and 0.8%.

Most emails, 63.5% on average, were valid emails, i.e., they were accepted by a regular expression for emails. The percentage of valid emails increased over time, starting from 60% at the beginning of our study and reaching its maximum at 68% in the latest scan. An average of 34.8% of the emails were missing the required prefix *mailto:*. Over the period of our study, the share of emails without *mailto:* steadily decreased, starting at 39% and reaching its minimum at 30%. Therefore, a clear trend toward standard-conform referencing of email addresses can be derived. Only 0.9% of emails contained spaces between *mailto:* and the email address. Furthermore, emails with brackets were only present in 0.8% of entries recognized as an email. Both of these percentages largely remained constant throughout our investigation.

The results of the semantic check of the URLs show that almost every URL found in a *Contact* field was valid. HTTP-URLs were only used in 0.1% of cases and none of the extracted URLs were invalid or missing the protocol altogether.

Conformity of Expires Fields: Figure 5 shows the results of the semantic check of the expiry dates. The x-axis shows the time, and the y-axis the percentage of error types observed during the evaluation of the *Expires* field. In addition, the percentages for expired and non-expired fields are given. At the beginning of our study, 65% of the expiry dates were compliant with the standard. This percentage increased significantly, reaching its maximum

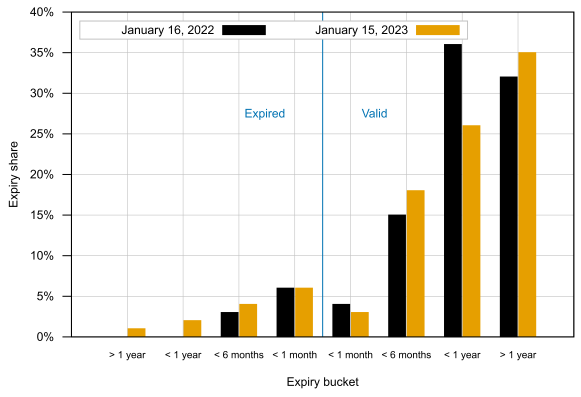

Fig. 6. Distribution of expiry dates for the first and last scan.

of 86% in our latest scans. Non-compliant dates that could still be parsed decreased from 34% at the end of 2021 to only 12% at the end of our investigation. Additionally, a relatively constant percentage of merely 1.1% of dates were invalid. This depicts a clear tendency toward conformity with the standard.

Overall, most dates were not expired at the time of testing, and the number of expired dates remained largely stable at about 8.8%. Interestingly, two significant drops in non-expired dates happened around New Year's Eve. This finding can likely be attributed to website operators setting the expiration date to the last day of the year.

Finally, Figure 6 shows the distribution of expiry dates for the scans done on January 16, 2022 and January 15, 2023. According to the standard, the expiry date must be set at most one year in the future. This applied to 56.4% of dates in the first and 48.9% in the last scan. Therefore, a majority of security.txt files with an *Expires* field did not adhere to the standard's specification in this regard. However, as our latest scan took place in January 2023, website operators might not have updated their security.txt files with new expiration dates. The expiry date was set to a time in the past in 10.9% and 15.1% of cases. In this subset, the average time since expiry was 159 days, so roughly half a year.

When analyzing our complete data set, the oldest expiry date was over 53 years ago and represents the day of the Unix epoch. On average, non-expired expiry dates were set about 4 years into the future. However, this value is highly influenced by extreme cases that set their expiry dates to up to 970 years into the future.

Most common contact domains: We analyzed the *Contact* fields in all security.txt files from the most recent scan on January 15, 2023. The domains of *Contact* fields containing websites were extracted and grouped. Some of the most used domains were *hackerone.com*, *g.co*, *www.bmwgroup.com*, and *twitter.com*. One explanation for this distribution might be that large organizations publish their information on a multitude of web servers at the same time if they decide to use security.txt.

dnssecuritytxt: The contents of dnssecuritytxt entries we analyzed only revealed one interesting fact. In total, four organizations used an incorrect UTF-8 encoding within the "TXT" record. In all cases, a security researcher could still get the correct contact information. However, for automation use cases, this may result in errors.

6 DISCUSSION

In the following sections, we discuss and interpret our results before detailing the limitations of our study and providing guidance for future work in this research field.

Deployment: Our results show that most websites have not yet adopted security.txt. While about one-third of the top 100 websites have a security.txt file deployed, these numbers shrink drastically when looking at larger groups of lower-ranked websites. As expected, this shows that adoption is generally higher for more popular websites. The organizations behind these websites presumably have more resources to spend on IT security and thus, are more likely to adhere to security standards and best practices such as security.txt. Furthermore, we see that the adoption of the standard is gaining the most traction among lower-ranked websites. While the adoption is rather low in this group, it also offers the largest potential for improvement. Even though the absolute numbers seem low, security.txt is still in its early stages, especially because it was only recently standardized. Nevertheless, the increase in adoption rates is a promising development.

Standardization: Our results show that security.txt is slowly gaining traction within the IT industry, especially since its standardization in April 2022. While the data shows no sudden leaps in adoption, the deployment of security.txt increased slightly faster after the standardization process concluded. This might signify that operators started to become more attentive to the standard and its benefits.

Comparison with Previous Studies: Our results align well and confirm the findings from previous studies about security.txt, which we discussed in Section 3. For example, our analysis of the deployment rate in the top 100k group can be compared to those of Poteat and Li [19] from May 2021. In their work, the adoption rate was about 1%, while we measured about 3.1% in our latest scan. In their study, a maximum of 16% of the top 100 websites offered a security.txt file while our maximum was at 35%. While the list of analyzed domains was different, the percentages are still somewhat comparable, as the rate of change in the top 100 is relatively low. Thus, the number of security.txt files more than doubled within less than two years.

Findlay and Abdou [6] employed a stricter definition of a valid security.txt, e.g., they required the content types to be *text/plain* only. Therefore, they determined the rate of security.txt adoption in the top one million group to be 0.49% while the rate of our latest scan stood at about 1%.

User Agents: Additionally, we conclude that the user agent used for scanning has a marginal impact at best. While a few web servers might block scans issued by unknown or certain non-browser user agents, our results prove that, for large-scale notifications campaigns, these differences are negligible. However, security researchers who aim to receive the maximum amount of possible security.txt responses should use a browser-based user agent with a recent version.

Existence of Fields: To discuss the correct usage of the standard, we first focus on the existence of mandatory fields. The *Contact* field is the most prevalent field by far, ranging at around 90% adoption rate in deployed security.txt files. This is unsurprising, because publishing contact information for vulnerability disclosure is the main purpose of the security.txt standard. For the remaining 10% of files, security researchers would need to resort to other methods of finding the contact information for the responsible organization.

The second mandatory field, the *Expires* field, is present in roughly a third of security.txt files. Consequently, at least two-thirds of security.txt files do not adhere to the RFC's specification. This number used to be even lower, as our scans at the beginning of this study as well as prior work have shown. Interestingly, the adoption rate of the *Expires* field was the second lowest rate of any of the eight standard fields at the end of 2021. However, in the most recent scan, the *Expires* field is close to being the third most deployed security.txt field, showing the steep increase in standard-conform adoption of this field. This clearly shows that the willingness of organizations to adhere to the specification is increasing rapidly, especially since the draft became an official RFC.

Furthermore, the existence of the optional fields reveals exciting facts: Most optional fields have a rather stable adoption rate throughout our study. However, the *Preferred-Languages* field experienced a steep increase in adoption, similar to the *Expires* field. We conclude that organizations are eager to receive notifications about vulnerabilities in their preferred language. For the other optional fields, interesting findings include that nearly a third of organizations who deploy a security.txt list a policy URL and prefer communication to be encrypted.

Field Conformity: Our results show that a significant number of websites do not follow the specification with regard to the *Contact* field, most prominently in the case of email addresses. One of the reasons might be the prevention of spam. Therefore, the standard would benefit from a method for securely publishing such information. The very low percentage of telephone numbers is to be expected, since maintaining a hotline for incident disclosure purposes is costly. Some security.txt files also contained an invalid *Expires* field. While nearly every field was machine-readable, some did not follow the format enforced by the standard. We believe that some website operators input the expiry date manually. Ideally, such information should be automatically generated to prevent such mistakes. In addition, we found that one in ten files had an expiry date in the past. Website operators might deploy security.txt and fail to maintain it later on. Notifying operators about soon to be expired files could be a valid strategy to counter this problem.

6.1 Limitations

Throughout our investigation, we found that a few web servers blocked scan requests for some of our tested user agents but not for others. Facebook.com, for example, blocked certain scans through cURL with an adjusted user-agent header. While blocking spoofed user-agents from accessing regular website content might be a reasonable security decision, we argue that access to the security.txt file should nevertheless be allowed for any client application.

Our scanning architecture might have excluded some security.txt files on misconfigured web servers. For example, we only scanned using the HTTPS protocol, validated HTTPS certificates, and did not account for non-standard locations of the security.txt file. However, prior work showed that some websites offered a security.txt file using only HTTP and not HTTPS. Even though the aim of our study was to include the maximum amount of files that somewhat resemble a security.txt file, these examples of misconfigurations would have only marginally impacted our results.

Finally, our scanning architecture could have accounted for more errors in the security.txt files that led to our implementation discarding them, even though a human researcher could have still used the presented information. Our analysis could have included misspelled fields that a human researcher could reasonably understand. We argue that the effort of accounting for every possible crawler detection technique, misconfiguration of web servers or files diverging from the standard would have been disproportionate to the marginal benefit for our analysis.

6.2 Future Work

We envision multiple directions for future work in the context of security.txt:

Security.txt was accepted as an RFC less than a year ago in April 2022. A longitudinal study covering a longer period of time, for example the next five years, can be done to measure the acceptance of the standard. As the most popular websites undergo changes, specific considerations for the analysis and presentation of the results would be necessary. The results of any large-scale study about security.txt can be analyzed further by integrating external data sources. For instance, the websites could be grouped by industries, sectors, countries, or other factors to determine differences between those groups. Furthermore, instead of passively scanning the Internet, it would be possible to use the contact information stated in the security.txt files to reach out to the security teams. Such a study could clarify if the information is correct and if security teams respond to notifications in a timely manner.

We consider security.txt to be an excellent method for decentralized, automated, and simple security vulnerability disclosure. Therefore, we propose the following ideas to drive the adoption of this standard in the future. One possibility is to partner with browser vendors. Browsers could, for example, automatically check for and validate a website's implementation of the standard. Building on top of that, the browser could use information from security.txt files to improve the already existing vulnerability reporting functionality built into the application.

Other recommendations we derive focus on the organization's side of the vulnerability disclosure process through the use of security.txt files. The adoption is especially low for lower-ranked websites. This could be caused by a lack of knowledge about the standard or a lack of operational skills and resources to implement and maintain the standard. Therefore, seamless and near-automatic implementation of security.txt through content management systems such as Drupal or Wordpress would be beneficial. Even though extensions for this purpose already exist, it would be preferable for these systems to offer the operator a native way to enable security.txt when deploying such systems.

Automation is another important cornerstone for improving adoption. The development of tools for automated generation, deployment, and notification in case the file expires can lessen the burden associated with maintaining a compliant security.txt file.

7 CONCLUSIONS

For researchers, finding the correct contact information of an organization for disclosing information about vulnerabilities is a difficult task. In this longitudinal study, we evaluated the adoption and correct usage of security.txt, a recently standardized method that aims to improve the disclosure process by publishing contact information in a simple, standardized way.

Our results show that the overall adoption rate of security.txt remains low, especially among lower-ranked websites. However, we find a slight but steady increase in overall adoption throughout our longitudinal investigation, especially after the standard became an RFC in April 2022. When it comes to conformity with the formatting and rules enforced by the standard, our work shows that a significant percentage of websites that deploy a security.txt file do not adhere to the RFC, including popular websites. Based on our results, we envision a number of directions for future work and propose ideas for improving the adoption of this standard. Such options include automated generation and maintenance and the integration of security.txt into popular frameworks and applications.

In conclusion, the increasing adoption of the security.txt standard is a promising step toward improving web security and facilitating communication between security researchers and website owners.

COMPETING INTERESTS

All authors declare that they have no conflicts of interest.

ETHICAL ASPECTS

The design and execution of this study was cleared by the ethics board of HM Munich University of Applied Sciences. Since all data used in this study was collected from public sources, no potential tangible harm to any person's well-being is to be expected. The data does not reveal private or confidential information about individuals.

DATA AVAILABILITY

The raw data gathered during scanning that support the findings of this study and the source code of the application used for download, analysis, and presentation are available from the authors upon reasonable request.

ACKNOWLEDGMENT

We want to thank Farsight Security for the access to their passiveDNS database.

REFERENCES

[1] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. 2005. *Uniform Resource Identifier (URI): Generic Syntax*. STD 66. RFC Editor. Retrieved from http://www.rfc-editor.org/rfc/rfc3986.txt.

[2] Scott O. Bradner. 1996. *The Internet Standards Process—Revision 3*. BCP 9. RFC Editor. Retrieved from http://www.rfc-editor.org/rfc/rfc2026.txt

[3] John Carroll and Casey Ellis. 2021. dnssecuritytxt: A standard allowing organizations to nominate security contact points and policies via DNS TXT records. Retrieved from https://dnssecuritytxt.org/

[4] Catalin Cimpanu. 2017. Bleeping Computer: Security.txt Standard Proposed, Similar to Robots.txt. Retrieved from https://www.bleepingcomputer.com/news/security/security-txt-standard-proposed-similar-to-robots-txt/

[5] Dave Crocker. 1997. *Mailbox Names for Common Services, Roles and Functions*. RFC 2142. RFC Editor.

[6] W. Paul Findlay and Abdelrahman Abdou. 2022. Characterizing the adoption of security.txt files and their applications to vulnerability notification. In *Proceedings of the Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*. The Internet Society, United States. https://doi.org/10.14722/madweb.2022.23014
[7] FIRST. 2022. Incident Response Database. Retrieved from https://www.first.org/global/irt-database
[8] Edwin Foudil and Yakov Shafranovich. 2020. *A File Format to Aid in Security Vulnerability Disclosure*. Internet-Draft draft-foudil-securitytxt-10. Internet Engineering Task Force. Retrieved from https://datatracker.ietf.org/doc/draft-foudil-securitytxt/10/. Work in Progress.
[9] Edwin Foudil and Yakov Shafranovich. 2022. *A File Format to Aid in Security Vulnerability Disclosure*. RFC 9116. RFC Editor.
[10] Edwin Foudil and Yakov Shafranovich. 2022. security.txt: A proposed standard which allows websites to define security policies. Retrieved from https://securitytxt.org/
[11] GitHub. 2022. GitHub: SECURITY.md. Retrieved from https://github.com/github/.github/blob/master/SECURITY.md
[12] Adam Hupp. 2023. GitHub: python-magic. Retrieved from https://github.com/ahupp/python-magic
[13] IANA. 2022. security.txt Fields. Retrieved from https://www.iana.org/assignments/security-txt-fields/security-txt-fields.xhtml
[14] G. Klyne and C. Newman. 2002. *Date and Time on the Internet: Timestamps*. RFC 3339. RFC Editor.
[15] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS'19)*. DOI: https://doi.org/10.14722/ndss.2019.23386
[16] 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 *Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16)*. USENIX Association, 1033–1050.
[17] Mark Nottingham. 2019. *Well-Known Uniform Resource Identifiers (URIs)*. RFC 8615. RFC Editor.
[18] Addison Phillips and Mark Davis. 2009. *Tags for Identifying Languages*. BCP 47. RFC Editor.
[19] Tara Poteat and Frank Li. 2021. Who you gonna call? an empirical evaluation of website security. txt deployment. In *Proceedings of the 21st ACM Internet Measurement Conference*. 526–532.
[20] Farsight Security. 2022. *dnsdbflex—Github Website*. Retrieved from https://github.com/farsightsec/dnsdbflex
[21] Farsight Security. 2022. *Farsight Security DNSDB*. Retrieved from https://www.farsightsecurity.com/solutions/dnsdb/
[22] Ben Stock, Giancarlo Pellegrino, Frank H. Li, Michael Backes, and Christian Rossow. 2018. Didn't you hear me?—Toward more successful web vulnerability notifications. In *Proceedings of the Network and Distributed System Security Symposium*.
[23] Ben Stock, Giancarlo Pellegrino, Christian Rossow, Martin Johns, and Michael Backes. 2016. Hey, you have a problem: On the feasibility of large-scaleweb vulnerability notification. In *Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16)*. USENIX Association, 1015–1032.
[24] The European Union. [n.d.]. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

Received 14 June 2023; accepted 26 June 2023