SPOOFY, A PYTHON TOOL FOR LOCATING VULNERABLE DOMAINS

Samantha Jackson
Department of Science
Texas A&M University-San Antonio
San Antonio, TX 78224
sjack012@jaguar.tamu.edu

Jonathan Davis
Department of Science
Texas A&M University-San Antonio
San Antonio, TX 78224
jdavi074@jaguar.tamu.edu

Nico Gibson
Department of Science
Texas A&M University-San Antonio
San Antonio, TX 78224
ngibs01@jaguar.tamu.edu

May 14, 2025

ABSTRACT

Phishing, particularly in the form of Business Email Compromise (BEC) scams, remains a significant threat in cybersecurity. To combat domain spoofing, organizations increasingly rely on DNS-based authentication protocols such as SPF, DKIM, and DMARC. This paper presents Spoofy, a lightweight Python tool designed to identify vulnerable domains by analyzing their SPF and DMARC configurations. Spoofy evaluates each domain against 198 possible configuration patterns and assigns a 'spoofability' score to quantify its susceptibility to spoofing attacks. Real-world testing using federal government domain data demonstrates Spoofy's effectiveness in identifying domains with weak or misconfigured DNS authentication. These findings highlight persistent security gaps and underscore the need for robust DNS configurations. Furthermore, this paper proposes future enhancements to Spoofy, including support for DKIM record scanning, to further strengthen its domain security assessment capabilities.

1 Introduction

The FBI defines spoofing and phishing as key components of Business Email Compromise (BEC) scams. Spoofing refers to a cybercrime in which an individual uses a vulnerable domain to disguise themselves as a legitimate network or user. A phishing scam is a kind of spoofing attack that allows threat actors to send emails while appearing as a legitimate user. This disguise can convince recipients that a phishing attack is a legitimate email, when in reality it is being used by a threat actor to gain access to proprietary data. The technical aspects of an effective phishing attack are more complicated than using a domain that looks legitimate because it is dependent on a network's domain name systems (DNS) protocol [1].

DNS protocols are useful tools that allow names to be assigned to specific IP addresses. For example, the domain name 'us.gov' is associated with the IP '23.22.13.113'. Within a domain, there exist three important email authentication methods that allow network administrators to control how emails are processed, and ideally, prevent spoofing. These authentication methods stored in the DNS make up the DNS-based email authentication triad:

- **SPF (Sender Policy Framework):** A list of IP addresses of all servers that are allowed to send emails from the domain.

- **DKIM (DomainKeys Identified Mail):** Allows domain owners to 'sign' emails from their domain. DKIMs utilize a public-private key pair where a new private key is generated as a digital signature for any email sent through said domain.
- **DMARC (Domain-based Message Authentication, Reporting and Conformance):** Tells the email server what to do given the results of SPF and DKIM. It can instruct mail servers to quarantine emails that fail these checks, reject emails, or deliver them.

Given the combination of these specific authentication methods we could ideally determine whether or not a domain is vulnerable to spoofing. Utilizing all possible combinations of SPF and DMARC parameters should allow us to determine how easy it is to spoof a domain without necessarily considering DKIM. That is where Spoofy's capabilities come in.

2 Spoofy's Methodology

Spoofy is a Python tool primarily developed by MattKeeley [2]. In order to determine the spoofability status of a domain this tool utilizes a master table that accounts for 198 possible variations of SPF and DMARC setups all of which were tested using Microsoft 365 [3]:

Table 1: SPF and DMARC Configuration Codes and Descriptions

SPF/DMARC	Code	Description
SPF	-all	Strictest setting, only allowing authorized IPs.
SPF	ãll	Soft fail, unauthorized emails may be accepted with a flag.
SPF	+all	Allows all emails.
SPF	?all	Neutral, SPF doesn't enforce a pass or fail state.
DMARC	p=none	No enforcement of SPF/DKIM failure states.
DMARC	p=quarantine	Emails that fail SPF/DKIM are sent to the spam folder.
DMARC	p=reject	Emails failing SPF/DKIM are rejected.
DMARC	aspf=r / aspf=s	Alignment modes for SPF; relaxed (r) or strict (s).
DMARC	sp=none	Same as p=, but applies to designated subdomains.
DMARC	sp=quarantine	Same as p=, but applies to designated subdomains.
DMARC	sp=reject	Same as p=, but applies to designated subdomains.

Given these conditions this table breaks down the potential for spoofability into 'spoofability codes' 1-8 using the 'is_spoofable' method based on the 'modules/spoofing.py' file. The reference table contains 198 combinations of DMARC and SPF configurations that result in a code from 1-8 where 1 is completely spoofable and 8 is impossible to spoof.

3 Initial Testing

Spoofy can be utilized on any operating system that supports Python 3 scripts. This initial testing will utilize the ARM64 2024.4 Kali Linux distribution in VMWare to demonstrate Spoofy's capabilities. Spoofy can be downloaded from the Github page using 'git clone https://github.com/MattKeeley/Spoofy'. All of the required dependencies can be installed using the command 'pip install -r requirements.txt' Once Spoofy is installed it can be executed using the format './spoofy.py [-h] (-d D | -iL iL) [-o stdout, xls] [-t T]:

- **-d:** Processes a single domain.
- **-iL:** Processes a file containing a list of raw domains to process.
- **-o stdout:** Default output format.
- **-o xls:** Output .xls format for domain analysis.
- **-t:** Sets the number of threads to use (default: 4).

For an initial test of Spoofy our team utilized the current-full.csv data from the dotgov-data developed by cisagov on GitHub [4]. This dataset contains a list of all the currently known federal domains. This initial testing resulted 271/12824 vulnerable domains. Though this initial testing showed that Spoofy only read domain information by each line in a .csv file instead of searching for domain data specifically. This made the initial data largely unusable and is a problem that needs to be addressed.

4 Comparison of Spoof-ability Tools

While Spoofy identifies vulnerable domains using SPF and DMARC record analysis, a comparative evaluation against existing tools highlights key functionalities that could significantly improve its utility and effectiveness. This section presents a comparative analysis for two established tools used for domain-based security evaluation: Domain-Security-Scanner developed by the Global Cyber Alliance and MXToolBox. Both tools offer a broader inspection of domain security configurations offering a reference point for understanding Spoofy's limitations and future development goals. The table below summarizes the comparative features of each tool:

Table 2: Feature Comparison of Spoofy and Related Tools

Tool	SPF	DKIM	DMARC	TLS	Notes
Spoofy	Y	N	Y	N	CLI tool; unique spoofability scoring via DNS config patterns.
MXToolBox	Y	Y	Y	Y	Web-based; extensive diagnostics and blacklist functions.
Domain-Security-Scanner	Y	Y	Y	Y	CLI/Web; usability hindered by Docker issues but includes TLS and DKIM.

4.1 MXToolBox Network Tools

MXToolBox is an online diagnostic tool used for email and network troubleshooting [5]. It has a suite of utilities including email and DNS troubleshooting, network diagnostics, performance and security analysis. These use cases make it an ideal comparison for Spoofy. MXToolBox is capable of checking for SPF, DKIM, and DMARC records. These settings, as established, determine the spoofability of a domain. Additionally, the 'MX check' and 'Blacklist Check' functions can give a good picture of how a domain functions. Specifically, what IPs the chosen domain is blacklisting.

In comparison to Spoofy, MXToolBox demonstrates key advantages in its range of functionality. It performs accurate detection of DMARC and SPF policies while also being able to correctly identify DKIM signatures. This information will allow an experienced user to determine whether or not a domain is vulnerable to spoofing. However, MXToolBox does not compute spoofability scores or provide a breakdown of configuration combinations in the way Spoofy does. Spoofy's unique approach offers a more targeted evaluation metric that can assist security analysts in prioritizing risk across large domain datasets.

MXToolBox's inclusion of DKIM analysis highlights a critical limitation in Spoofy's current implementation. Without DKIM checks, Spoofy cannot deliver a complete assessment of the email authentication triad (SPF, DMARC, and DKIM). This gap, especially with increased adoption of DKIM represents a priority for future development.

4.2 Domain-Security Scanner

The Domain-Security-Scanner developed by the Global Cyber Alliance, is an open-source utility capable of evaluating SPF, DKIM, and DMARC records [6]. The tools is available as a command-line interface and as a web interface. One distinguishing feature is the inclusion of a '–checkTLS' option that allows for inspection of TLS support.

Despite its utility, the Domain-Security-Scanner suffers from some usability issues. During testing, the Docker implementation would fail to launch. Hindering the tool's practical utility. This tool's web-bsaed version is operation but suffers from slow response times and less features compared to the command-line interface version [7].

Functionally, the Domain-Security-Scanner serves as a more expansive, but less reliable comparison to Spoofy. It provides DKIM and TLS diagnostics whereas Spoofy is limited to SPF and DMARC diagnostics. Moreover, Domain-Security-Scanner offers a more advisory output model that suggests configuration changes. Spoofy, by comparison, presents raw data alongside spoofability scores but does not currently include configuration recommendations.

4.3 Implications for Spoofy's Development

Both tools underscore a common deficiency in Spoofy in the lack of DKIM analysis. This omission is particularly significant given that (as of 2019) on the lower-end 13% of all domains have adopted DKIM. Though this is higher than domains that utilize both SPF and DMARC at 9% [8]. A number that is only projected to get larger with time as more providers take domain security seriously. In light of these comparisons, Spoofy's development should prioritize:

- **DKIM Scanning:** Public key lookup and signature alignment verification.
- **TLS Configuration Checks:** Evaluation of transport-layer security in parallel with DNS-based authentication.
- **Policy Recommendation:** Providing actionable guidance for mis-configured or permissive domains.

5 Software Construction

The construction of Spoofy was guided by principles of modularity, scalability, and clarity to facilitate DNS-based domain analysis across large datasets. This section provides an overview of the system's design architecture, original functionality, and the development methodology that enabled effective record parsing and spoofability evaluation. It also covers enhancements made to expand Spoofy's functionality, including support for DKIM resolution and input sanitation. Through this breakdown, we demonstrate how the tool's structure supports reliable security assessments while remaining lightweight and extensible for future updates.

5.1 Requirements

As previously established, Spoofy was developed to identify vulnerable domains based on DNS configurations, particularly, SPF and DMARC records. The tool has satisfied several core functional requirements including:

1. **Retrieving and Parsing DNS-based Authentication Records**: Grab SPF, DMARC, and BIMI, for any valid domain.
2. **Evaluation of Spoofability**: Utilizes a matrix of known SPF/DMARC combinations to assign a spoofability score from 1 to 8.
3. **Efficiency for Large Domain Lists**: Achieved through a multithreaded implementation to minimize runtime.
4. **Readable Output**: Create console logs and structured Excel exports for further analysis.

Additionally, extensibility was a core non-functional requirement to allow future enhancements. This is particularly valuable for adding new modular features that will still follow Spoofy's core requirements. In our case, two modules: `dkim.py` and `clean.py`.

5.2 Original Design

A clear understanding of Spoofy's internal architecture is crucial for assessing its modularity, extensibility, and maintainability. The tool is organized into a set of Python modules within the `/modules` directory, each handling a distinct aspect of domain-based email security analysis. Core modules manage DNS resolution, SPF parsing, DMARC policy extraction, and spoofability scoring, while utility modules handle syntax validation and reporting.

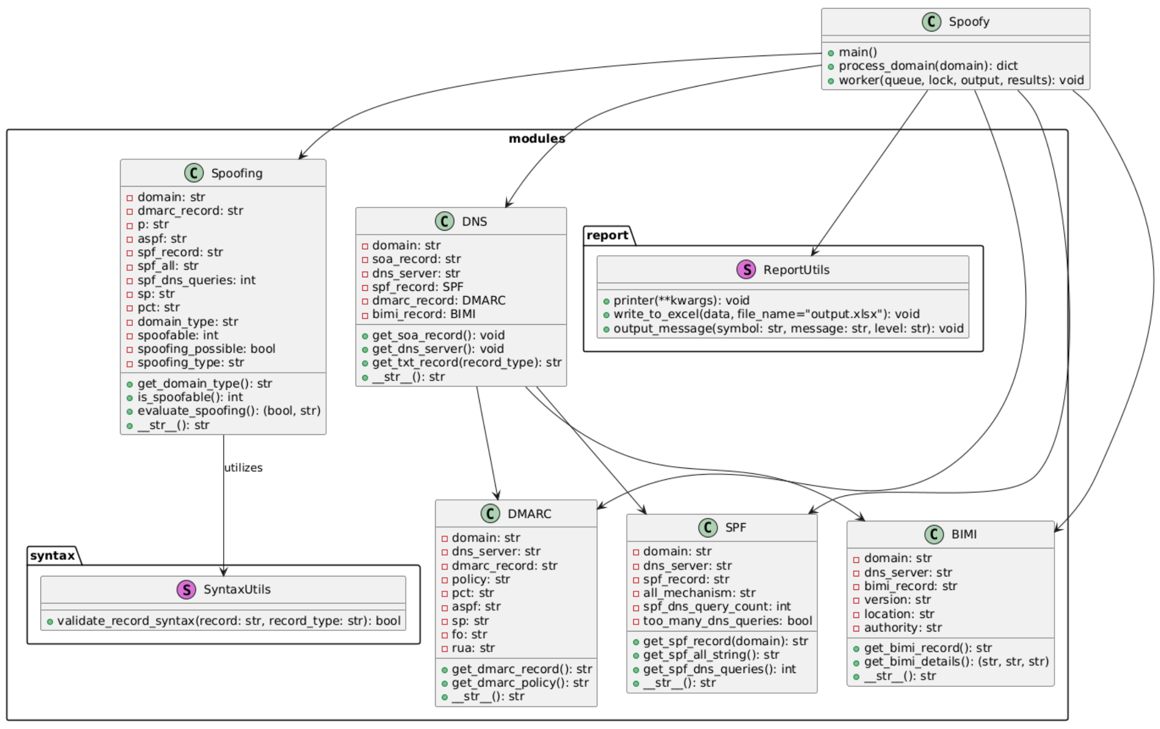


Figure 1: UML Class Diagram of Spoofy's Initial Architecture.

As illustrated in Figure 3, the `spoofy.py` driver script orchestrates the execution flow by coordinating interactions between modules. The `DNS` class acts as a central hub, identifying authoritative name servers and initializing SPF, DMARC, and BIMI lookups. Each of these protocols is encapsulated in its own dedicated class, supporting separation of concerns and facilitating isolated testing.

Spoofy's spoofability analysis is encapsulated within the `Spoofing` class, which synthesizes SPF and DMARC parameters to generate a numerical spoofability score based on a heuristic model. Supporting utilities such as `syntax.py` and `report.py` enhance the design by modularizing record validation and output formatting, respectively.

This architecture supports concurrent processing of domain data through multi-threaded execution and offers a solid foundation for future enhancements.

5.3 Design Enhancements: DKIM Integration and Data Preprocessing

In response to the limitations identified during the initial testing and comparative analysis, Spoofy's architecture was extended to include DKIM validation and input data sanitation capabilities. These enhancements are encapsulated within two new modules: `dkim.py` and `clean.py`, respectively.

The DKIM module allows Spoofy to evaluate DKIM records by probing for common selectors, parsing version and algorithm metadata, and analyzing the public key. This addition brings Spoofy closer to complete DNS-based authentication coverage and provides a foundation for future DKIM alignment checks.

The CleanUtils utility module introduces a preprocessing layer that extracts and sanitizes domain names from raw or poorly structured input sources, such as large CSVs. This ensures consistency and integrity in batch processing, minimizing the chance of unusable data due to malformed inputs.

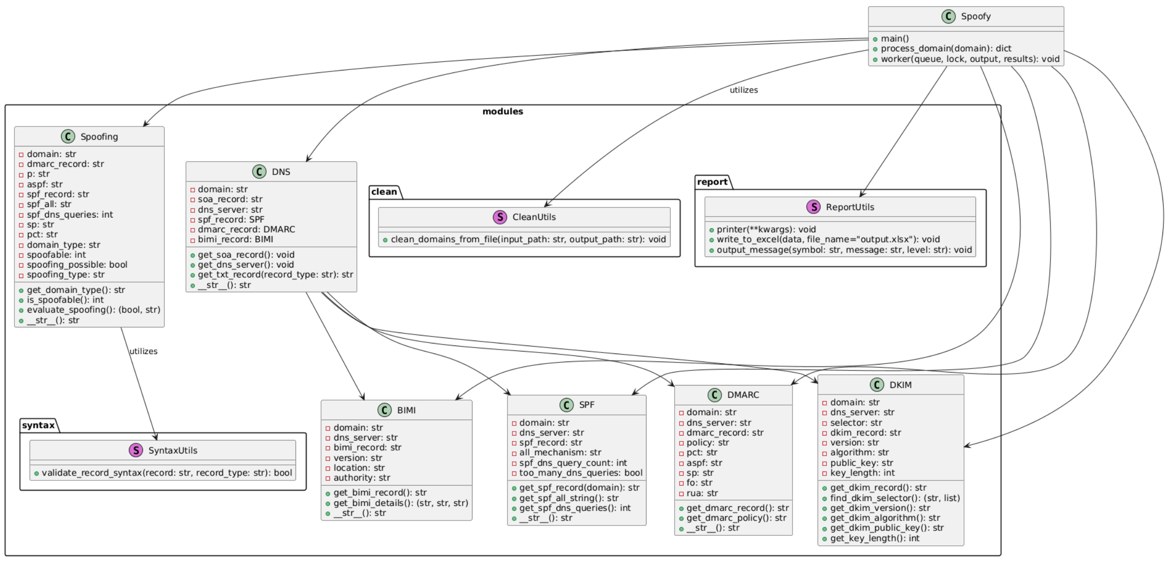


Figure 2: UML Class Diagram of Spoofy's New Architecture.

These additions demonstrate Spoofy's scalability and modularity. By decoupling preprocessing and DKIM validation into distinct, testable modules, the updated design easily adhered to principles of modularity and extensibility.

5.4 Implementation

The new modules introduced in Spoofy adhere to the project's principles of modularity and clarity. Their implementation focuses on lightweight, dependency-aware functionality to ensure interoperability within the existing framework.

5.4.1 DKIM Module

The DKIM module defines a `DKIM` class that attempts to locate and parse DKIM records for a given domain. Because DKIM records are indexed by selector names, the module uses a list of common selectors (`default`, `google`, `selector1`, `mail`, `spf`, `dkim`) to construct DNS TXT queries like `selector._domainkey.domain.com`. When a matching record is found, fields such as `v` (version), `k` (algorithm), and `p` (public key) are parsed.

The public key, encoded in Base64, is decoded and deserialized using Python's `cryptography` library. The resulting key object allows Spoofy to calculate the bit-length of the RSA key and flag configurations that fall below secure thresholds (e.g., less than 1024 bits).

5.4.2 Clean Utility Module

The Clean utility module introduces a function `clean_domains_from_file` which accepts a text or CSV file and outputs a standardized domain list. For CSV files, it assumes domain entries exist in the first column. For other formats, a regular expression is used to extract domain patterns line-by-line.

The implementation includes duplicate filtering, whitespace trimming, and extension checking to ensure only syntactically valid domain names are retained. Output is written to a new file, preserving the original dataset. This preprocessing step is especially valuable when using open government datasets or network telemetry feeds that may include noise or formatting inconsistencies.

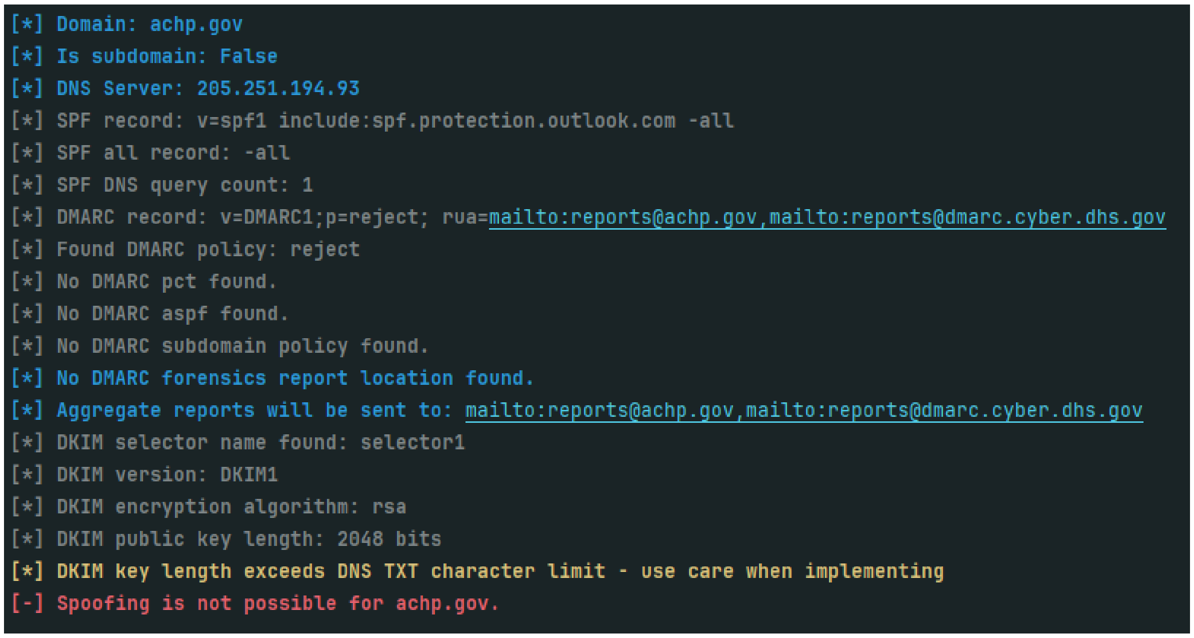
5.5 Testing

In order to test our new DKIM module we created a unit test `testing/dkim-test.py`. This script runs seven unit tests that check for correct behavior in the DKIM module. The results of these tests demonstrated correct functionality within the DKIM module with all tests returning an OK. The tests we ran are as follows:

1. **test_initialization:** Test the proper initialization of a `DKIM` object.

2. **test_find_dkim_selector_found:** Test a successful case of DKIM selection where a selector is found.
3. **test_find_dkim_selector_not_found:** Test a case where DKIM selection fails and no selector is found.
4. **test_parse_dkim_record_parts:** Test the parsing functions for the various parts of a DKIM record.
5. **test_key_length_calculation:** Test the RSA key length calculation functionality.
6. **test_error_handling_in_get_key_length:** Test error handling during RSA key length calculation.
7. **test_dns_exception_handling:** Test handling of DNS-related exceptions.

Functional tests indicate the correct functionality of the DKIM module. When running the new Spoofy script on the `current-federal.csv` dataset we can see the functionality of the DKIM module even with the limited amount of hard-coded selectors. The example output here shows that the DKIM module was capable of finding a selector and using that to determine the DKIM version, encryption algorithm, and the public key length as well as giving some suggestions. Additional testing was also done on individuals domains which indicate similar functionality [9].



```
[*] Domain: achp.gov
[*] Is subdomain: False
[*] DNS Server: 205.251.194.93
[*] SPF record: v=spf1 include:spf.protection.outlook.com -all
[*] SPF all record: -all
[*] SPF DNS query count: 1
[*] DMARC record: v=DMARC1;p=reject; rua=mailto:reports@achp.gov,mailto:reports@dmarc.cyber.dhs.gov
[*] Found DMARC policy: reject
[*] No DMARC pct found.
[*] No DMARC aspf found.
[*] No DMARC subdomain policy found.
[*] No DMARC forensics report location found.
[*] Aggregate reports will be sent to: mailto:reports@achp.gov,mailto:reports@dmarc.cyber.dhs.gov
[*] DKIM selector name found: selector1
[*] DKIM version: DKIM1
[*] DKIM encryption algorithm: rsa
[*] DKIM public key length: 2048 bits
[*] DKIM key length exceeds DNS TXT character limit - use care when implementing
[-] Spoofing is not possible for achp.gov.
```

Figure 3: Example output with the new DKIM module implemented on the domain 'achp.gov'.

6 Experiment and Analysis

This experiment evaluated the email authentication configurations of **1,353** federal domains listed in `current-federal.csv`, using our new version of the Spoofy tool [10]. Spoofy was previously capable of analyzing SPF and DMARC records; this experiment introduced new support for retrieving and parsing **DKIM** records, enabling a more comprehensive assessment. Here were our objectives:

- Practically test newly implemented DKIM record resolution, parsing, and RSA key length analysis.
- Validate existing SPF and DMARC analysis modules against a federal domain dataset.
- Export structured results to `current-federal.xlsx`.

Similarly to our initial testing we run the command `./spoofy.py -iL current-federal.csv -o xls -t 8`. Running Spoofy on the current-federal.csv (containing 1353 domains) outputting to xlsx and using eight threads. This output resulted in a file `output.xlsx` which was then imported into Google Sheets for analysis. Here were the results:

- **SPF:** Found in **1,175** (86.8%) domains. Most used a single "all" mechanism.
- **DMARC:** Present in **1,144** (84.6%) domains, but only **467** defined a `pct` value and fewer included `aspf` or `sp` tags.

- **DKIM:** Implemented in **266** (19.7%) domains with six predefined selectors, with public key lengths ranging from **1024** to **2048** bits all with RSA algorithms. Weak or malformed records were flagged.
- **BIMI:** Rarely implemented; only **4** (0.2%) domains had valid BIMI records.
- **Spoofing Analysis:** Approximately **234** (17.2%) domains were considered spoofable under the scoring model, with varying level of protection.

Though these results show the utility of these new Spoofy additions there are some clear issues with the script. For one, datasets larger than 1000 domains introduce more risk for potential errors and take significantly longer to compile into a usable dataset. When running an output to xlsx there is no indication of how close Spoofy is to completion. Introducing a loading indicator to confirm progress might help with this issue.

7 Summary and Future Work

In this work, we presented an extended version of Spoofy, a lightweight and modular Python tool for evaluating domain-level email authentication configurations. Originally developed to assess spoofability based solely on SPF and DMARC parameters, our version of Spoofy introduces support for DKIM parsing, public key analysis, and input sanitation via two new modules—`dkim.py` and `clean.py`. These enhancements brought Spoofy closer to full coverage of the DNS-based email authentication triad, providing a more accurate picture of domain vulnerability.

Empirical testing on 1,353 U.S. federal domains demonstrates the effectiveness of these improvements. While SPF and DMARC adoption remains high, DKIM presence was detected in only 20% of domains using six common selectors, suggesting wider but undiscovered deployment due to selector obfuscation. BIMI, by contrast, was nearly absent. Spoofability scores revealed that roughly 17% of domains remain susceptible to impersonation based on their DNS configurations.

Compared against more established diagnostic tools like MXToolBox and the Domain-Security-Scanner, Spoofy distinguishes itself through its scoring model, open-source accessibility, and extensibility. However, limitations such as incomplete DKIM selector probing, lack of TLS evaluation, and minimal policy guidance remain areas for future development.

Additions to the DKIM module in particular should focus on dynamically searching for selectors. During our research we located a tool dkimscan.pl by ryancdotorg on GitHub [11]. This Perl tool utilizes an extensive list of common selector and selector patterns. The ideas this tool presents would be useful to our research if it was implemented in an efficient way, and would give a more accurate picture of DKIM's presence in domain security.

Ultimately, Spoofy's evolution underscores the tension between rapidly evolving threat landscapes and the slower pace of protocol adoption. As organizations continue to harden their digital infrastructure, tools like Spoofy can serve not only as detection engines but as educational platforms that clarify the often opaque mechanics of domain authentication.

8 Conclusion

The updated Spoofy tool successfully integrates DKIM analysis, complementing existing SPF and DMARC modules. Results show that while SPF and DMARC have been widely adopted across federal domains, DKIM implementation is still lacking in many cases, and BIMI adoption remains negligible. These findings highlight ongoing gaps in email authentication practices that need to be addressed to reduce susceptibility to spoofing and impersonation attacks.

References

[1] Federal Bureau of Investigation. (n.d.). *Spoofing and phishing*. FBI. `https://www.fbi.gov/how-we-can-help-you/scams-and-safety/common-frauds-and-scams/spoofing-and-phishing`

[2] Keeley, M. (2024). *Spoofy* [GitHub repository]. GitHub. `https://github.com/MattKeeley/Spoofy/tree/main`

[3] Keeley, M. (2024). *Spoofy Master Table* [Excel spreadsheet]. GitHub. `https://github.com/MattKeeley/Spoofy/blob/main/files/Master_Table.xlsx`

[4] Cybersecurity and Infrastructure Security Agency. (2024). *Dotgov-data* [Dataset]. GitHub. `https://github.com/cisagov/dotgov-data/tree/main`

[5] MXToolbox. (n.d.). *MXToolbox SuperTool*. https://mxtoolbox.com/SuperTool.aspx

[6] Global Cyber Alliance. (2024). *Domain Security Scanner* [GitHub repository]. https://github.com/globalcyberalliance/domain-security-scanner

[7] Global Cyber Alliance. (n.d.). *DMARC Guide and Scanner*. https://dmarcguide.globalcyberalliance.org/

[8] Tatang, D., Zettl, F., & Holz, T. (2021). The evolution of DNS-based email authentication: Measuring adoption and finding flaws. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '21)*, 354–369. https://doi.org/10.1145/3471621.3471842

[9] Gibson, N. (2024). *DKIM module demo* [Video]. YouTube. https://youtu.be/1K0c0Q6JkQo?si=fWeJclXzNhyzORoF

[10] Davis, J., Jackson, S., & Gibson, N. (2024). *SpoofyDKIM* [GitHub repository]. GitHub. https://github.com/Erosssore/SpoofyDKIM

[11] Ryan, C. (2017). *dkimscan.pl* [GitHub repository]. GitHub. https://github.com/ryancdotorg/dkimscan