

Texas A&M San Antonio

Spoofy, a Python tool for Locating Vulnerable Domains

Samantha Jackson, Nico Gibson, Jonathon Davis

Abstract

This paper explores Spoofy, a lightweight Python tool designed to analyze spoofability of domains by assessing their SPF and DMARC configurations. Phishing is a critical threat in cyber security, often exploited in the form of Business Email Compromise (BEC) scams. Utilizing DNS-based authentication methods-SPF, DKIM, and DMARC-organizations can mitigate the risks associated with domain spoofing. Spoofy evaluates domains against 198 possible SPF/DMARC configurations to categorize their vulnerability into spoofability scores. This paper serves to outline Spoofy's methodology, practical applications, and its effectiveness in identifying vulnerable domains, particularly government domains. Through real-world testing with federal domain data, Spoofy identifies potential security risks; this reinforces the importance of strong DNS-configuration.

Introduction

This paper is oriented towards the programmer who has an interest in DNS spoofing and may be attempting to locate vulnerable domains using Spoofy. This tool should not be used for any illegal purposes, but rather to document a variety of domains. It is the end user's responsibility to obey all local, state, and federal laws; do not utilize this information to exploit domains. Spoofy is a light-weight, accurate solution for bulk DNS lookups that can be output in *.txt and *.xls formats.

About Spoofing and Phishing

The FBI defines spoofing and phishing as key components of Business Email Compromise (BEC) scams. Spoofing refers to a cybercrime wherein an individual utilizes 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 where in reality it is being utilized 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 heavily dependent on a network's domain name systems (DNS) protocol.

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 exists three email authentication methods that allow network administrators to control how emails are processed, and ideally, prevent spoofing. These authentication methods stored in the DNS include:

- 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): Enables 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. That is where Spoofy's capabilities come in.

Spoofy's Methodology

Spoofy is a Python tool primarily developed by MattKeeley. 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:

SPF/DMARC	Code	Description
SPF	-all	Strictest setting, only allowing authorized IPs.
SPF	~all	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 codes in this table generally correspond to the master table’s coloration chart where green is considered spoofable, yellow could potentially be spoofed, and red is not spoofable:

Spoofability Code	Meaning
0	Completely spoofable (No SPF & no DMARC enforcement).
1	Low spoofability (Strict SPF and DMARC settings).
2	Moderate spoofability (SPF -all with relaxed DMARC).
3	Somewhat spoofable (DMARC applied to partial emails).
4	Medium spoofability (+all SPF, weak DMARC settings).
5	Moderate spoofability (DMARC weak, SPF strict).
7	Somewhat protected (SPF soft-fail, DMARC weak).
8	Strong protection (SPF -all + DMARC p=reject).

Utilizing these codes, Spoofy generates an output based on any given number of domains.

Getting Started

Spoofy can be utilized on any operating system that supports Python 3 scripts. This documentation 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>’. Spoofy utilizes five python package requirements; these can be downloaded using ‘pip3 installations -r requirements.txt’ if your current OS environment doesn’t have these packages, alternatively you can use venv to install these

packages if you don't want to install packages to the OS. Most Linux distributions have these packages installed anyways.

```
(rose@rose)-[~/Desktop]
$ git clone https://github.com/MattKeeley/Spoofy
Cloning into 'Spoofy' ...
remote: Enumerating objects: 631, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (75/75), done.
remote: Total 631 (delta 37), reused 24 (delta 20), pack-reused 536 (from 2)
Receiving objects: 100% (631/631), 1007.57 KiB | 2.75 MiB/s, done.
Resolving deltas: 100% (339/339), done.
```

```
GNU nano 8.2 requirements.txt
colorama
dnspython >= 2.2.1
tldextract
pandas
openpyxl
```

Once Spoofy is installed along with all its packages you can begin testing domains. Just ensure that you have cd into the directory you cloned 'cd Spoofy'. Spoofy utilizes a relatively simple command that calls the Python executable spoofy.py. It follows the format: './spoofy.py [-h] (-d D | -iL iL) [-o {stdout,xls}] [-t T]'. You can test domains in different ways:

- -d D: Processes a single domain
 - Example: ./spoofy.py -d us.gov
- -iL iL: Processes a file containing a list of domains
 - Example: ./spoofy.py -iL fed-domains.csv

Outputs, if non-specified, will be in stdout (default) format. You can select two different formats using the modifier -o {stdout/xls}. *.xls format is useful for database analysis and is compatible with Excel.

Using the -t modifier you can specify the amount of threads Spoofy utilizes. By default it will use four threads. This can be particularly useful if you're testing a large list of domains, say

a *.csv file of several thousand domains. Doubling the computational power will save significant amounts of processing time.

For users that are confused or need a refresher on the utilization of these tools you can use the command ‘./spoofy.py -h’. Alternatively for a more comprehensive overview you can use ‘nano README.md’.

```
## HOW TO USE

`Spoofy` requires Python 3. Python 2 is not supported. Usage is shown below:

console
Usage:
  ./spoofy.py -d [DOMAIN] -o [stdout or xls] -t [NUMBER_OF_THREADS]
OR
  ./spoofy.py -iL [DOMAIN_LIST] -o [stdout or xls] -t [NUMBER_OF_THREADS]

Options:
  -d : Process a single domain.
  -iL : Provide a file containing a list of domains to process.
  -o : Specify the output format: stdout (default) or xls.
  -t : Set the number of threads to use (default: 4).


Examples:
  ./spoofy.py -d example.com -t 10
  ./spoofy.py -iL domains.txt -o xls

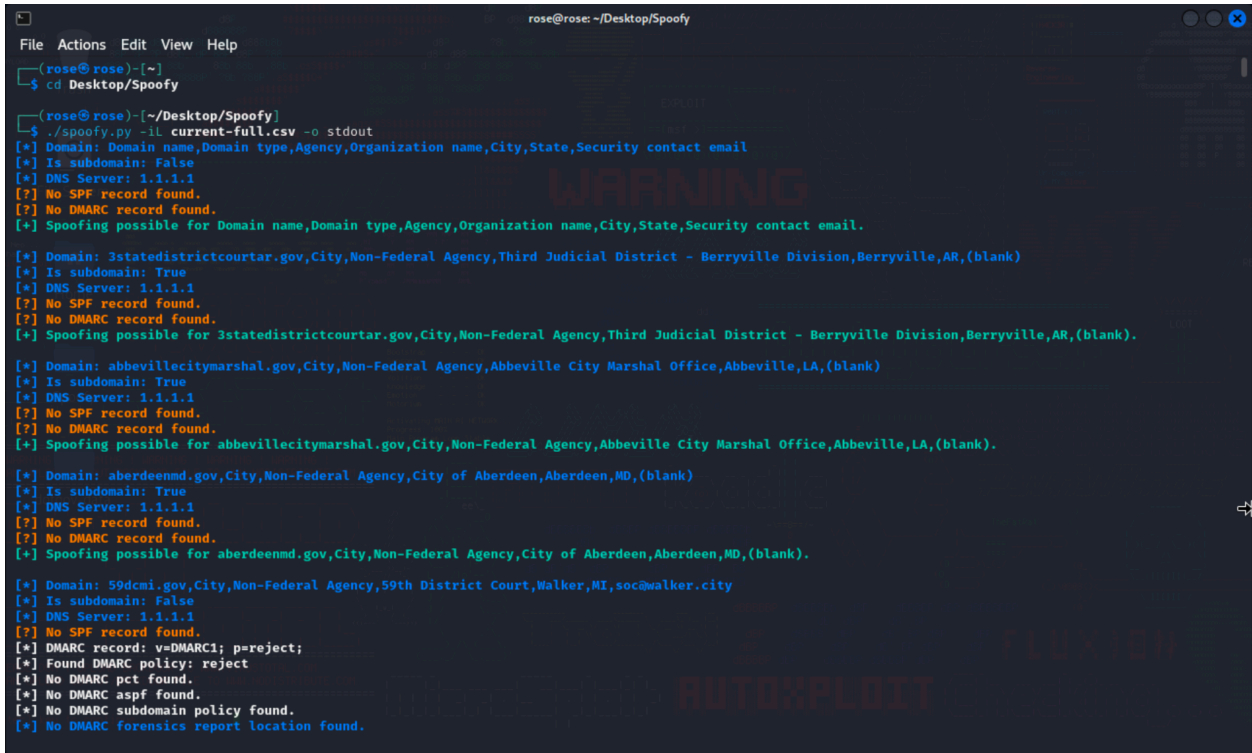
Install Dependencies:
  pip3 install -r requirements.txt
```

Practical

For our test of Spoofy we will be utilizing the [Dotgov-data](#) current-full.csv database. This file contains a list of all the current federal domains. Spoofy doesn't do well with files outside of its own file path so we will simply put the current-federal.csv file into the Spoofy folder.

Checking these files for spoofability is incredibly practical because federal domains have an air of authority to them that could fool users into thinking phishing emails are legitimate. For our command input we will use ‘./spoofy.py -iL current-full.csv -o stdout’ this will allow us to see

the output on the terminal. Additionally we will do an output for .xls format we will use “./spoofy.py -iL current-full.csv -o xls”. As you can see Spoofy found 271 domains that were potentially spoofable  Spoofable *.gov Domains .



```
rose@rose: ~/Desktop/Spoofy
File Actions Edit View Help
(rose@rose)~$ cd Desktop/Spoofy
(rose@rose)~/Desktop/Spoofy$ ./spoofy.py -iL current-full.csv -o stdout
[*] Domain: Domain name,Domain type,Agency,Organization name,City,State,Security contact email
[*] Is subdomain: False
[*] DNS Server: 1.1.1.1
[?] No SPF record found.
[?] No DMARC record found.
[+] Spoofing possible for Domain name,Domain type,Agency,Organization name,City,State,Security contact email.

[*] Domain: 3statedistrictcourtar.gov,City,Non-Federal Agency,Third Judicial District - Berryville Division,Berryville,AR,(blank)
[*] Is subdomain: True
[*] DNS Server: 1.1.1.1
[?] No SPF record found.
[?] No DMARC record found.
[+] Spoofing possible for 3statedistrictcourtar.gov,City,Non-Federal Agency,Third Judicial District - Berryville Division,Berryville,AR,(blank).

[*] Domain: abbevillecitymarshal.gov,City,Non-Federal Agency,Abbeville City Marshal Office,Abbeville,LA,(blank)
[*] Is subdomain: True
[*] DNS Server: 1.1.1.1
[?] No SPF record found.
[?] No DMARC record found.
[+] Spoofing possible for abbevillecitymarshal.gov,City,Non-Federal Agency,Abbeville City Marshal Office,Abbeville,LA,(blank).

[*] Domain: aberdeenmd.gov,City,Non-Federal Agency,City of Aberdeen,Aberdeen,MD,(blank)
[*] Is subdomain: True
[*] DNS Server: 1.1.1.1
[?] No SPF record found.
[?] No DMARC record found.
[+] Spoofing possible for aberdeenmd.gov,City,Non-Federal Agency,City of Aberdeen,Aberdeen,MD,(blank).

[*] Domain: 59dcmi.gov,City,Non-Federal Agency,59th District Court,Walker,MI,soc@walker.city
[*] Is subdomain: False
[*] DNS Server: 1.1.1.1
[?] No SPF record found.
[*] DMARC record: v=DMARC1; p=reject;
[*] Found DMARC policy: reject
[*] No DMARC pct found.
[*] No DMARC aspf found.
[*] No DMARC subdomain policy found.
[*] No DMARC forensics report location found.
```

Conclusion

Spoofy provides a lightweight and reliable method to scan many domains and subdomains for vulnerability to spoofing. By leveraging a comprehensive master table of SPF and DMARC configurations, Spoofy offers an effective method to evaluate a domain's spoofability. This makes Spoofy a useful tool to identify and mitigate security risks.

References

Keeley, M. (n.d.). *Spoofy* [GitHub repository]. GitHub.

<https://github.com/MattKeeley/Spoofy/tree/main>

Federal Bureau of Investigation. (n.d.). *Spoofing and phishing*. FBI. Retrieved

<https://www.fbi.gov/how-we-can-help-you/scams-and-safety/common-frauds-and-scams/spoofing-and-phishing>

Cloudflare. (n.d.). *DMARC, DKIM, and SPF explained*. Cloudflare.

<https://www.cloudflare.com/learning/email-security/dmarc-dkim-spf/>

Cybersecurity and Infrastructure Security Agency. (n.d.). *Dotgov data* [GitHub repository].

GitHub. <https://github.com/cisagov/dotgov-data/tree/main>