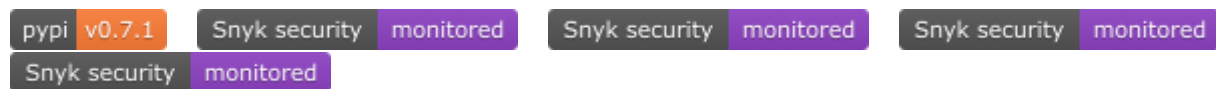

Pushing HTTPS



`pshtt` (“*pushed*”) is a tool to scan domains for HTTPS best practices. It saves its results to a CSV (or JSON) file.

`pshtt` was developed to *push* organizations — especially large ones like the US Federal Government :us: — to adopt HTTPS across the enterprise. Federal agencies must comply with M-15-13, a 2015 memorandum from the White House Office of Management and Budget, and BOD 18-01, a 2017 directive from the Department of Homeland Security, which require federal agencies to enforce HTTPS on their public web services. Much has been done, but there’s more yet to do.

`pshtt` is a collaboration between the Cyber and Infrastructure Security Agency’s National Cybersecurity Assessments and Technical Services (NCATS) team and the General Service Administration’s 18F team, with contributions from NASA, Lawrence Livermore National Laboratory, and various non-governmental organizations.

Getting started

`pshtt` can be installed as a module, or run directly from the repository.

Installed as a module

`pshtt` can be installed directly via pip:

```
1 pip install pshtt
```

It can then be run directly:

```
1 pshtt example.com [options]
```

Running directly

To run the tool locally from the repository, without installing, first install the requirements:

```
1 pip install -r requirements.txt
```

Then run it as a module via `python -m`:

```
1 python -m pshtt.cli example.com [options]
```

Usage and examples

```
1 pshtt [options] DOMAIN...
2 pshtt [options] INPUT
3
4 pshtt dhs.gov
5 pshtt --output=homeland.csv --debug dhs.gov us-cert.gov usss.gov
6 pshtt --sorted current-federal.csv
```

Note: if INPUT ends with `.csv`, domains will be read from the first column of the CSV. CSV output will always be written to disk (unless `-json` is specified), defaulting to `results.csv`.

Options

1	<code>-h --help</code>	Show this message.
2	<code>-s --sorted</code>	Sort output by domain, A-Z.
3	<code>-o --output=OUTFILE</code> <code>"")</code>	Name output file. (Defaults to <code>"results</code>
4	<code>-j --json</code>	Get results in JSON. (Defaults to CSV.)
5	<code>-m --markdown</code> <code>CSV.)</code>	Get results in Markdown. (Defaults to
6	<code>-d --debug</code>	Print debug output.
7	<code>-u --user-agent=AGENT</code>	Override user agent.
8	<code>-t --timeout=TIMEOUT</code>	Override timeout (in seconds).
9	<code>-c --cache-third-parties=DIR</code> <code>directory to cache it in.</code>	Cache third party data, and what
10	<code>-f --ca-file=PATH</code>	Specify custom CA bundle (PEM format)

Using your own CA bundle By default, `pshtt` relies on the root CAs that are trusted in the Mozilla root store. If you work behind a corporate proxy or have your own certificates that aren't publicly trusted, you can specify your own CA bundle:

```
1 pshtt --ca-file=/etc/ssl/ca.pem server.internal-location.gov
```

What's checked?

A domain is checked on its four endpoints:

- `http://`
- `http://www`
- `https://`
- `https://www`

Domain and redirect info

The following values are returned in `results.csv`:

- **Domain** - The domain you're scanning!
- **Base Domain** - The base domain of **Domain**. For example, for a Domain of `sub.example.com`, the Base Domain will be `example.com`. Usually this is the second-level domain, but `pshtt` will download and factor in the Public Suffix List when calculating the base domain. (To cache the Public Suffix List, use `--suffix-cache` as documented above.)
- **Canonical URL** - One of the four endpoints described above; a judgment call based on the observed redirect logic of the domain.
- **Live** - The domain is "live" if any endpoint is live.
- **HTTPS Live** - The domain is "HTTPS live" if any HTTPS endpoint is live.
- **HTTPS Full Connection** - The domain is "fully connected" if any HTTPS endpoint is fully connected. A "fully connected" HTTPS endpoint is one with which `pshtt` could make a full TLS connection.
- **HTTPS Client Auth Required** - A domain requires client authentication if *any* HTTPS endpoint requires it for a full TLS connection.
- **Redirect** - The domain is a "redirect domain" if at least one endpoint is a redirect, and all endpoints are either redirects or down.
- **Redirect to** - If a domain is a "redirect domain", where does it redirect to?

Landing on HTTPS

- **Valid HTTPS** - A domain has "valid HTTPS" if it responds on port 443 at the hostname in its Canonical URL with an unexpired valid certificate for the hostname. This can be true even if the Canonical URL uses HTTP.
- **HTTPS Publicly Trusted** - A domain is "publicly trusted" if its canonical endpoint has a publicly trusted certificate.
- **HTTPS Custom Truststore Trusted** - A domain is "custom truststore trusted" if its canonical endpoint has a certificate that is trusted by the custom truststore.
- **Defaults to HTTPS** - A domain "defaults to HTTPS" if its canonical endpoint uses HTTPS.
- **Downgrades HTTPS** - A domain "downgrades HTTPS" if HTTPS is supported in some way, but its canonical HTTPS endpoint immediately redirects internally to HTTP.
- **Strictly Forces HTTPS** - This is different than whether a domain "defaults" to HTTPS. A domain "Strictly Forces HTTPS" if one of the HTTPS endpoints is "live", and if both HTTP endpoints are either down or redirect immediately to any HTTPS URI. An HTTP redirect can go to HTTPS on another domain, as long as it's immediate. (A domain with an invalid cert can still be

enforcing HTTPS.)

Common errors

- **HTTPS Bad Chain** - A domain has a bad chain if either HTTPS endpoint contains a bad chain.
- **HTTPS Bad Hostname** - A domain has a bad hostname if either HTTPS endpoint fails host-name validation.
- **HTTPS Expired Cert** - A domain has an expired certificate if either HTTPS endpoint has an expired certificate.
- **HTTPS Self-Signed Cert** - A domain has a self-signed certificate if either HTTPS endpoint has a self-signed certificate.
- **HTTPS Probably Missing Intermediate Cert** - A domain is “probably missing intermediate certificate” if the canonical HTTPS endpoint is probably missing an intermediate certificate.

HSTS

- **HSTS** - A domain has HTTP Strict Transport Security enabled if its canonical HTTPS endpoint has HSTS enabled.
- **HSTS Header** - This field provides a domain’s HSTS header at its canonical endpoint.
- **HSTS Max Age** - A domain’s HSTS max-age is its canonical endpoint’s max-age.
- **HSTS Entire Domain** - A domain has HSTS enabled for the entire domain if its **root HTTPS endpoint** (*not the canonical HTTPS endpoint*) has HSTS enabled and uses the **includeSubDomains** flag.
- **HSTS Preload Ready** - A domain is HSTS “preload ready” if its **root HTTPS endpoint** (*not the canonical HTTPS endpoint*) has HSTS enabled, has a max-age of at least 18 weeks, and uses the **includeSubDomains** and **preload** flag.
- **HSTS Preload Pending** - A domain is “preload pending” when it appears in the Chrome preload pending list with the **include_subdomains** flag equal to **true**. The intent of **pshtt** is to make sure that the user is *fully* protected, so it only counts domains as HSTS preloaded if they are *fully* HSTS preloaded (meaning that all subdomains are included as well).
- **HSTS Preloaded** - A domain is HSTS preloaded if its domain name appears in the Chrome preload list with the **include_subdomains** flag equal to **true**, regardless of what header is present on any endpoint. The intent of **pshtt** is to make sure that the user is *fully* protected, so it only counts domains as HSTS preloaded if they are *fully* HSTS preloaded (meaning that all subdomains are included as well).

-
- **Base Domain HSTS Preloaded** - A domain's base domain is HSTS preloaded if its base domain appears in the Chrome preload list with the `include_subdomains` flag equal to `true`. This is subtly different from **HSTS Entire Domain**, which inspects headers on the base domain to see if HSTS is set correctly to encompass the entire zone.

Scoring

These three fields use the previous results to come to high-level conclusions about a domain's behavior.

- **Domain Supports HTTPS** - A domain 'Supports HTTPS' when it doesn't downgrade and has valid HTTPS, or when it doesn't downgrade and has a bad chain but not a bad hostname (a bad hostname makes it clear the domain isn't actively attempting to support HTTPS, whereas an incomplete chain is just a mistake.). Domains with a bad chain "support" HTTPS but user-side errors can be expected.
- **Domain Enforces HTTPS** - A domain that 'Enforces HTTPS' must 'Support HTTPS' and default to HTTPS. For websites (where `Redirect` is `false`) they are allowed to *eventually* redirect to an `https://` URI. For "redirect domains" (domains where the `Redirect` value is `true`) they must *immediately* redirect clients to an `https://` URI (even if that URI is on another domain) in order to be said to enforce HTTPS.
- **Domain Uses Strong HSTS** - A domain 'Uses Strong HSTS' when the `max-age` ≥ 31536000 .

General information

- **IP** - The IP for the domain.
- **Server Header** - The server header from the response for the domain.
- **Server Version** - The server version, as extracted from the server header.
- **HTTPS Cert Chain Length** - The certificate chain length for the canonical HTTPS endpoint.
- **Notes** - A field where free-form notes about the domain can be stored.

Uncommon errors

- **Unknown Error** - A Boolean value indicating whether or not an unexpected exception was encountered when testing the domain. The purpose of this field is to flag any odd websites for further debugging.

Troubleshooting

DNS blackhole / DNS assist

One issue which can occur when running [pshtt](#), particularly for home/residential networks, with standard ISPs is the use of “DNS Assist” features, a.k.a. “DNS Blackholes”.

In these environments, you may see inconsistent results from [pshtt](#) owing to the fact that your ISP is attempting to detect a request for an unknown site without a DNS record and is redirecting you to a search page for that site. This means that an endpoint which *should* resolve as “not-alive”, will instead resolve as “live”, owing to the detection of the live search result page.

If you would like to disable this “feature”, several ISPs offer the ability to opt out of this service, and maintain their own instructions for doing so:

- AT&T
- FIOS

Who uses pshtt?

- GSA maintains Pulse, a dashboard that tracks how federal government domains are meeting best practices on the web. Pulse is open source.
- The Freedom of the Press Foundation runs [securethe.news](#), a site that aims to “track and promote the adoption of HTTPS encryption by major news organizations’ websites”. Secure the News is open source.
- DHS issues HTTPS Reports to federal executive branch agencies.

Acknowledgements

This code was modeled after Ben Balter’s site-inspector, with significant guidance from Eric Mill.

Contributing

We welcome contributions! Please see [CONTRIBUTING.md](#) for details.

License

This project is in the worldwide public domain.

This project is in the public domain within the United States, and copyright and related rights in the work worldwide are waived through the CC0 1.0 Universal public domain dedication.

All contributions to this project will be released under the CC0 dedication. By submitting a pull request, you are agreeing to comply with this waiver of copyright interest.