Internet Infrastructure Health Metrics Framework

Phase 3: Methodology and Scoring

December 31, 2023

Introduction

This work is the latest addition to CyberGreen's Internet Infrastructure Health Metrics Framework (IIHMF). The IIHMF is an initiative dedicated to investigating, measuring, and scoring the security of six components in order to gauge the Cyber Public Health of internet infrastructure. This phase of development - phase 3 - has proven to be more complex than phase 2 because we have started measuring more intricate aspects of Cyber Public Health. As a result, both our scans and this report are inherently more complicated.

This report outlines the methodology used in the measurement and scoring of security protocols, certificates, and email security. It builds on the previous phase of work that implemented measurement and scoring for DNS, routing security, and open services. Collectively, these six components constitute integral elements of internet infrastructure.

Dataset of Domains for DNS, Email, Security Protocols, and Certificates

This document uses "domain" to refer to a single subdomain within the effective top-level domain (eTLD) (e.g. if .com is the eTLD, example.com is a domain), and "hostname" to refer to both domains and subdomains (e.g. www.example.com and example.com). In phase 2, we used a dataset of government hostnames to measure DNS security. This dataset was synthesized and merged from publicly available datasets published in prior research efforts. In phase 3, we have extended the dataset by aggregating government hostnames from additional publicly available top million datasets and by identifying Subject Alternative Names (SANs) from TLS certificates for domains in the earlier dataset.

Top Million dataset extraction and combination

The original dataset was created through a combination of web crawling, Google dorking, crowdsourcing through Amazon MTurk, and filtering on large datasets such as the Majestic Million, Tranco, Cisco Umbrella, and Alexa. These datasets are different because the way they are gathered differ. For example, Alexa reports on domains accessed via Alexa. For an explanation of the differences see Pochat *et al.* (2018).

Since these large datasets used to create the dataset of existing works were accessed in 2019, we revisited each of those top million datasets in August 2023. In addition to the popular datasets used in previous research, we supplement our dataset creation using the Builtwith, Cloudflare Radar, and Domcop datasets.

To ensure we only include government hostnames, we crosscheck the eTLD of each hostname against a list of pre-generated and well known government eTLDs. This list was compiled by combining country domains (such as .ly, .us, .au) with common government extensions (such as

.gov, .gob, and .go). Since few governments such as the Canadian provincial governments do not follow this exact format, we manually add them along with special federal (.fed), and military (.mil) ccTLDs. Several databases were maintained by their respective governments. For example, the UK maintains a list of .gov.uk hostnames that is updated annually. Similarly, the US government maintains a database for .gov hostnames, and a separate database for other TLDs. We were unable to find an official database for all .mil TLDs, but we did find a website maintained by the US army that listed army-associated .mil sites. In addition, we incorporated the Know-Nepal and Dotmil databases, which are unofficial government hostname lists created by researchers. For datasets obtained from governments or from prior research, we do not apply any additional filtering, such as removing .com sites, to prevent the removal of entries which are government websites using commercial TLD suffixes such as .com.

SAN based dataset extension

The Subject Alternate Name (SAN) is a field of TLS certificates that specifies additional hostnames for which the certificate is valid. Matches can be denoted with exact hostnames (e.g. example.com to match example.com) and with wildcards (e.g. *.example.com to match both ex1.example.com and ex2.example.com). We utilize this field of the certificate to uncover additional government hostnames.

On August 31, 2023, we retrieved TLS certificates for all government hostnames aggregated so far. To handle wildcard hostnames, we stripped away the wildcards and retained the remaining DNS SAN entry. From the SAN fields, we retrieved a total of 166304 unique hostnames which, when combined with the dataset of 323164 government hostnames, result in the final dataset of 401216 unique government hostnames – now being used to compile scorecards in the IIHMF.

Security Protocols

Security Protocols: Why They Matter

When we discuss security protocols, we mean both TLS (the successor to SSL) and also SSH or IP security. In this phase of the project, we are only measuring TLS, and want to discuss it in that larger context.

The usage of security protocols such as TLS on the web makes today's browsing experiences secure and trustable due to the authenticity, integrity, and confidentiality properties. The usage of secure communication protocols ensures that the data in transit is protected from modification or observation by active or passive network adversaries. Such network adversaries could collect sensitive information or credentials.

How Security Protocols Work

Today’s internet uses the standardized Transport Layer Security (TLS) protocol which encrypts communications between two endpoints on the internet – typically a client and a server. It uses a combination of public key cryptography to securely generate and exchange a session key which is used with symmetric key cryptographic to protect the data transmissions. TLS has evolved over the past 25 or so years to remove various security flaws and to update cryptographic algorithms.

The collection of cryptographic algorithms within each protocol are known as cipher suites. Each cipher suite consists of a set of algorithms for key exchange, encryption and message authentication (MAC).

Cyber Public Health and Security Protocols

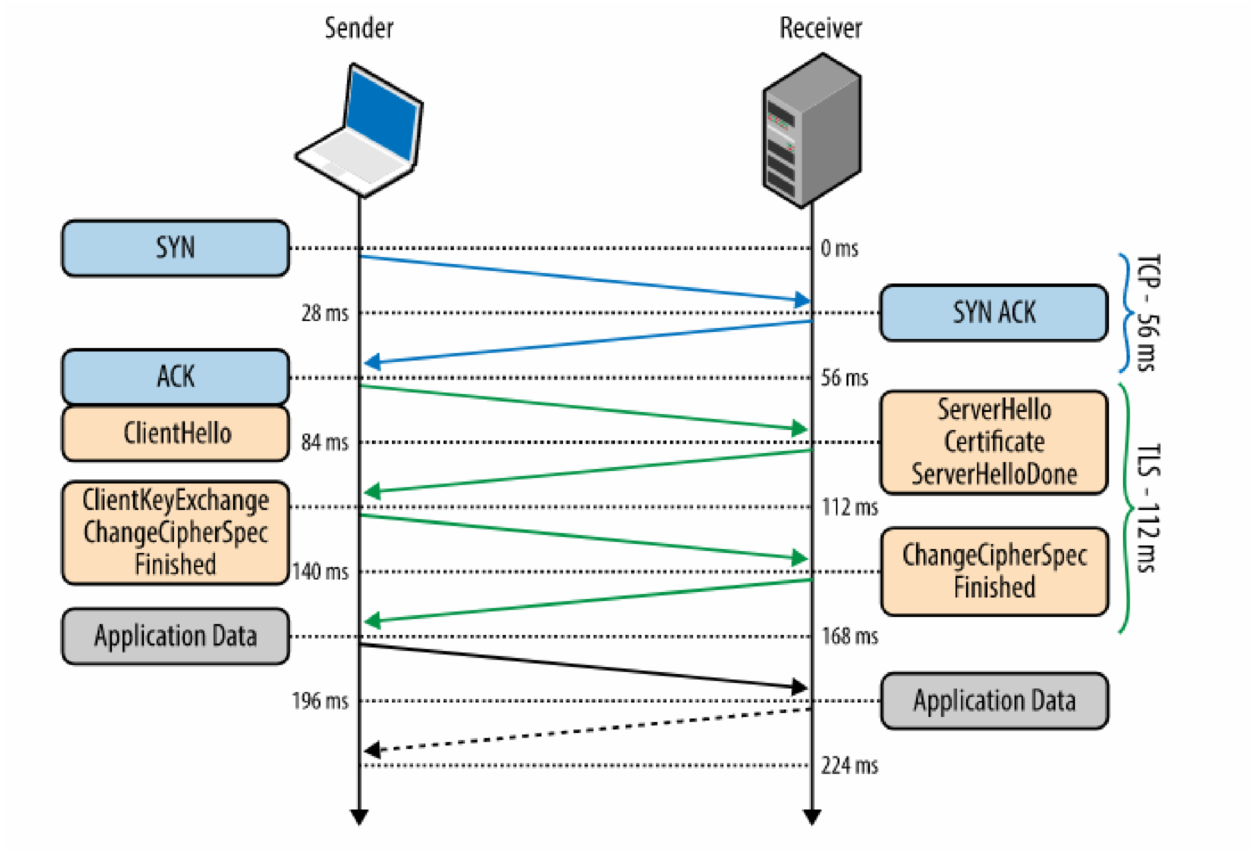
Protocols are agreements between hosts on how they communicate, and a system can only use a protocol if it’s supported by all the parties involved in the communication. If one host has older software which only supports TLS1.0, then either of the hosts must refuse to communicate with it, or all hosts must support that older version. Therefore, many libraries (and thus tools) will support older, less secure versions. This increases the risk of “downgrade” attacks, and makes the software more complex to test and maintain than if the older versions could be dropped. These are not theoretical concerns. The Go TLS library chooses to not support SSL (the predecessor to TLS). PCI has required TLS 1.2 since 2018; NIST since 2019 (https://github.com/golang/go/issues/45428). Support for older versions, therefore, may indicate an unmaintained system or might serve a set of very out-of-date clients that require support for these outdated protocols. Such out-of-date clients are likely vulnerable to many other problems, and are thus quite concerning in and of themselves.

As such, the slowest to upgrade impacts a community, not just their own health.

How We Measure Security Protocols

A feature of TLS is that it can be used to secure many protocols. Today, it is used to secure both web and email communications. In the figure below, a typical client (sender) to server (receiver) interaction is represented. TLS is generally used after a TCP connection is established (shown in blue), which consists of several messages that create a secure cryptographic “pipe” (shown in orange). Then, application data (shown in gray), such as web or mail messages, are sent through that pipe.

One crucial part of TLS is the use of certificates. Certificates show the authenticity of the receiver; they ensure that the machine “receiver” represents the party that the sender thinks it represents. For more information, see section “How Certificates Work”.



source: *High Performance Browsing Network*

Diagram of TLS Handshake Protocol

Security Protocols Scoring Overview

We score the security of TLS by the following factors:

1. Protocol Version: TLS 1.0, TLS 1.1, TLS 1.2 and TLS 1.3 are currently scanned.
2. Certificate(s) validity (see below)
3. Support for IPv4 and IPv6
4. TLS Protocol Cipher Suite Support
5. Use of Certificate Authority Authorization records

This list is roughly based on RFC 9325, combined with SSLLabs' perspective on Cipher Suites.

Certificates

Certificates: Why They Matter

Certificates prove the authenticity of the presenting party in a two party communication. Typically, certificates are presented by servers to their clients in an effort to prove the authenticity of the connection (to a specific hostname) to the client. Certificates tie public keys to hostnames, enabling key agreement in a TLS transaction. This is used to secure both web and mail communications.

Good certificates enable secure communications, while a lack of correctness or authenticity in the certificate due to mis-issuance, expiry, or incorrect cryptography indicate misconfigurations, or can hint towards the presence of active network adversaries attempting to tamper network traffic.

How Certificates Work

Certificates are issued by many Certificate Authorities (CA) located around the world which are trusted by the browsers, operating systems, and software running on user devices. Certificates are cryptographic files which are issued by a CA that include a public key, a hostname, and a signature indicating that the CA believes they should be tied together. Modern practice is that certificates should be somewhat short lived (90 days) to constrain the impact of a theft of a certificate. However, current internet standards such as RFC9325 do not take a position, and so we do not alter scores for this.

Certificates obtained by the client contain a *chain* of intermediate certificates indicating the web of trust involved in validating the hostname. A client receiving the certificate checks that the signatures of the certificates match the contents, and the chain is valid. Additionally, the client performs the check that the hostname that the client is connecting to is included in the certificate and that the certificate is not expired or revoked by the certificate authority. We check each of these things.

Cyber Public Health and Certificates

Similar to how email and TLS protocol versions have effects on the community, widespread certificate issues also impact how secure the community can be. In an ideal world, a mismatch between hostnames and certificates would be a rare issue that could be handled by a browser or email software by blocking the connection. But when thousands of invalid website certificates have this problem, and they are on government web infrastructure, such a block would interfere with citizens and businesses getting government services. So, again, the least secure configuration keeps all systems less secure.

How We Measure Certificates

To understand the DNS, TLS, and mail ecosystems together, we built a custom scanner implemented in Go. For each TLS scan on a hostname, we first look up and store A, AAAA and CAA records. Then, for each resolved IP, we try to connect on port 443 via TCP, and if we can, we initiate a TLS connection. Assuming that succeeds, we collect the TLS certificate.[1]

From the TLS certificate, we retrieve and store the security relevant fields (e.g. SAN, chain of trust, public key type, etc.) Additionally, we determine the supported TLS versions and cipher suites of a server by creating TLS connections with only one cipher suite option, effectively forcing the server to utilize the cipher suite. We create a new TLS connection for TLS versions 1.0-1.3.

For security reasons, Go does not support SSL (the predecessor to TLS) and as such we do not scan for it. TLS 1.0 was formally specified in 1999.

Certificates Scoring Overview

Our assessment of both web and email security will include a subscore which is based on the security qualities of certificates. Both those subscores will be calculated with identical algorithms running on different data. It will be possible for a domain to have a high certificate score for web and a low score for email.

Scoring includes:

1. Validity. A certificate is considered valid if it is:
   a. Cryptographically valid i.e. signature validates content using public key
   b. Valid for the duration of the certificate issuance
   c. Issued by a generally trusted certificate authority[2]
2. The host's public key type
3. The host's public key length
4. Number of SAN entries included in the certificate and their domain ownership
5. The signature algorithm used by the CA
6. The certificate chain length and validity

[1] A hostname may well have different certificates for mail and web services, and mail and web servers use different ports and protocols to retrieve them.
[2] Specifically, as validated by Go, which tracks that used by Debian Linux and Mozilla.

Email Security

Email Security: Why It Matters

The effects of email-related attacks can be far-reaching if unauthorized access or impersonation is used with an authoritative domain (e.g., government, critical infrastructure) to glean sensitive information from others. Further, compromised email accounts can be used to send real emails to other related organizations to further spread compromise.

Message Forgery: This active attack is often instantiated in falsification of messages such as emails to create phishing campaigns that seek to make recipients of the email disclose and or verify sensitive information.

Message Diversion/Deletion: An active attack where legitimate messages are removed before they can reach the desired recipient or are redirected to a network segment that is normally not part of the data path.

Message Modification: This active attack is one where a previous message has been captured and modified before being retransmitted. The message can be captured using a man-in-the-middle attack or message diversion.

Message Leakage: Message leaks can happen through active and passive attacks. During an active attack, messages sent in plaintext between mail servers, or between the client and the mail server can be observed by a network adversary to whom sensitive message information is revealed. Passive message leakage attacks happen due to compromise of mail servers, or user accounts which reveal the interactions between mail senders and receivers.

Cyber Public Health and Email Security

“Public health is the science and art of preventing disease, prolonging life, and promoting physical health and efficiency through organized community efforts … and the development of social machinery which will ensure to every individual in the community a standard of living adequate for the maintenance of health.” (Winslow)

Email touches on many elements of this. Email “diseases”, such as spam and phishing, are scourges that make it harder to use email for many of the goals of businesses and governments. Additionally, email suffers from both bad actors dragging things down, and “lazy” actors making ratchet improvements difficult, because they will be punished for “laziness,” or perhaps less judgmentally, lacking budget, skills, or facing other inhibitors to improvement. For more information, see our Cyber Belief Model technical report.

Thus, organizations cannot effectively secure themselves by themselves. If CyberGreen uses DMARC, but ASEAN does not check it, ASEAN is less secure. If ASEAN uses DMARC but

CyberGreen does not set it, ASEAN is less secure in our communications. Moreover, if many domains don't set these things, it is harder to reject mail which is not validated. An organized community effort is most worthwhile when most domains are using validation.

How Email Security Works

Email security protocols provide a mix of authentication, integrity, and confidentiality. Authentication prevents spoofing, integrity prevents tampering, and confidentiality prevents eavesdropping/observing.

1. Channel encryption, such as TLS, prevents network adversaries from observing or tampering with email in transit.

2. Protocols such as the Sender Policy Framework (SPF) help prevent email spoofing and provide authentication. This is done through DNS entries indicating which mail servers and corresponding IP addresses are permitted to send email on behalf of a domain. SPF is often associated with DomainKeys Identified Mail (DKIM). Testing for DKIM is more intrusive and nuanced and in IIHMF Phase 3, we do not test DKIM.

3. Domain-based Message Authentication, Reporting and Conformance (DMARC) is a security protocol that builds on top of SPF and DKIM to report on attempts to spoof.

4. SMTP MTA Strict Transport Security (MTA-STS) is a security standard that ensures that TLS connections are always used and provides a mechanism allowing servers to refuse message delivery to servers without trusted certificates.

How We Measure Email Security

To understand the DNS, TLS, and mail ecosystems together, we built a custom scanner implemented in Go. For each TLS scan on a hostname, we first look up and store A/AAAA records. Then, for each resolved IP, we connect on TCP ports 25, 465, 587, and 2525 which are used by mail clients. Upon success, we attempt a TLS connection by using the SMTP command for STARTTLS.

We currently scan for the mail servers' ability to use secure protocols during communication through the support for and usage of STARTTLS, and the configuration of SPF, DMARC, and MTA-STS policies but do not assess either message authentication via DKIM or encryption through OpenPGP or S/MIME.

Our mail scanner is built on top of the TLS scanner. Our scanner queries the authoritative DNS resolvers for each hostname for associated Mail Exchange (MX) records. Then, for each MX record, we resolve its associated IP addresses for both IPv4 and IPv6 networks. We create an

SMTP connection to each IP address and store the response banner for those servers which return a successful service ready state code (220). Next, we attempt to create a TLS connection using STARTTLS. Mail TLS scans mirror TLS scans described in the Security Protocols section of this document, except for a few additional SMTP-specific exchanges prior to the creation of the secure session. Mail servers are typically configured on ports 25, 465, 587, and 2525 which are scanned for each IP address associated with a hostname.

We also retrieve data on several other mail-specific security protocols. We query DNS for each mail hostname for the Sender Framework Policy (SPF) record and resolve and store any recursive lookups. MTA-STS uses a combination of DNS records which point to web pages. also specifies these mail servers on a HTTPS delivered page rather than DNS. We store both the DNS TXT record and HTTPS pages.

Finally, we perform a DNS lookup on each port+mail hostname pairing for a TLS Authentication (TLSA) record. TLSA records store a hash that matches the hash of TLS certificates from corresponding mail servers. The idea behind TLSA records is to place trust in DNS to securely verify the authority of a mail server to send mail. We scan on the same ports as SMTP (ports 25, 465, 587, and 2525).

We attempt to balance our scans to minimize intrusiveness while gathering data. We also attempt to find a balance between speed, comprehensiveness and accuracy. The scanners we built are minimally-intrusive. For example, we could send test emails to observe responses, but the additional information seems potentially unbalanced in comparison to the increased intrusiveness. To accurately measure DKIM support, it is necessary to send an email returning a bounced response from the mail servers with the associated message headers and cryptographic signatures. This could be considered abusive and result in the scanner being banned by the mail server being probed. There are other methods to test DKIM whose accuracy we might examine in future work.

Email Security Scoring Overview

Email security scores are dependent on several measurable factors related to TLS usage, as well as DMARC, SPF, TLSA and MTA-STS. We use the same algorithm for TLS in mail as we do for the web.

Other Email Factors (e.g. SPF, DMARC, MTA-STS)

In addition to the same scoring mechanism used for TLS supporting mail servers, mail servers have additional factors used for measurement as described below:

1. Number of accessible, secure mail ports (See Issue #22 for future simplification)
2. Support for STARTTLS capability among mail servers

3. SPF Records
    a. Recursive length of the SPF lookups for mail servers to validate
        i. Recursion depth > 10 is non RFC compliant and is penalized
    b. Adherence of the MX IP addresses to SPF Records (Issue #24)
4. Inclusion of TLSA records and their validity[3]
5. Inclusion and usage of MTA-STS policy
6. Inclusion and usage of DMARC policy

Scoring in Detail

The essential factors that contribute to scoring are explained above. This section explains them in more detail.

Key Decisions for Scoring

There are two key decisions that underlie our scoring approach. The first is that we assess roughly 16 aspects of web sites (and more for mail servers), each of which is weighted from -1 to 1. We order countries (TLDs) by the aggregate of these scores. That is the second decision: we focus on countries, not on domains, and so if a particular domain has a lot of hosts, that collection of hosts contributes more heavily to the country weighting than if we measured domains and then aggregated those into a country score.

To understand why the second matters, let's consider two hypothetical domains, example.com and example.org. Let's say each has subdomains a, b and c. (a.example.com, a.example.org, etc). If each of the example.com domains has a single host, they will each contribute equally to the score for example.com. And if a.example.org has 8 hosts, while b and c each have one, then a.example.org will contribute 80% of example.org's score.

This may be unintuitive, but the alternative is to de-prioritize large concentrations of insecurity because they are run by individual operators.

There is a third decision, yet to be made, which is how much to score security as a matter of recommended practice, and how much to accept deviation from those practices when security is maintained. For example, many authorities recommend turning off TLS1.0 because, by default, it supports many insecure cipher suites. Many organizations continue to support TLS1.0 because older devices and browsers require it. It is possible to configure TLS1.0 to support only secure cipher suites. There's a "strict" philosophy which penalizes organizations for TLS1.0, and a more sympathetic philosophy which only penalizes demonstrably insecure configuration.This particular issue is discussed in further detail in the Scoring Challenges section of this document,

[3] A typically invalid certificate due to its self signed nature must be treated as a valid certificate if it uses the DANE-EE mode validation and contains a valid TLSA record accompanied by a DNSSEC signature.

and we encourage readers familiar with the nuanced arguments to focus on the implication for public health scoring.

The entities facing the choice are likely less wealthy and less able to influence a community to update or upgrade; it may even be that neither is available. Our overall Cyber Public Health program is intended to increase the availability of data to allow us to assess if these choices result in harms, and we look forward to data-driven decisions in this area.

We anticipate an empirical approach to the question, and hope gathered data helps us understand how impactful these choices will be. (If the entities who face this choice are uniformly worse off on other measures, then the choice has less impact on overall ranking or scores.)

Factors That Contribute to Scoring

As mentioned in the overview, we measure certificates as a subset of both security protocols and email. The algorithm is the same, but sites can and should use separate certificates for mail and web, and so

Security Protocol Scoring

For security protocols we measure:

1. Protocol support (TLS 1.0-1.3)
2. Certificates
3. Support for IPv4, and IPv6
    a. IPv4 certificate serving validity / invalidity
    b. IPv6 certificate serving validity / invalidity
4. TLS Cipher suites
5. CAA records

Email Scoring

For email, we measure:

1. Number of accessible, secure mail ports (See Issue #22 for future simplification)
2. Support for STARTTLS capability
3. SPF Records
    a. Recursive length of the SPF lookups for mail servers to validate
        i. Recursion depth > 10 is non RFC compliant and is penalized
    b. Adherence of the MX IP addresses to SPF Records (Issue #24)
4. Inclusion of TLSA records and their validity
5. Inclusion and usage of MTA-STS policy

6. DMARC DNS records

Scoring Certificates and Supported TLS Protocols

Certificate scoring is used in both security protocols and mail protocols, and consists of subscores for:

1. `ProtocolScore`
2. `PermCertValidityScore`
3. `StrictCertValidityScore`
4. `PermPublicKeyTypeSizeScore`
5. `StrictPublicKeyTypeSizeScore`
6. `PermSigAlgScore`
7. `StrictSigAlgScore`
8. `IPV4CertValidityScore`
9. `IPV6CertValidityScore`
10. `CAAUsageScore`
11. `CSUsageScore_TLS10`
12. `CSUsageScore_TLS11`
13. `CSUsageScore_TLS12`
14. `CSUsageScore_TLS13`

Each of these are averaged over the number of hosts that could score well; so for example, if a host is not communicating on IPv6, it has no impact on the score.

Weighting and Ranking

Security is not a checklist process. Some elements weigh more heavily than others. For example, we weigh use of modern TLS versions lower than older versions, as the longer a standard is in place, the more reasonable we think it is to expect it to be in use. We weigh the use of DNSsec-dependent tools like TLSA lower because of risks involved in deploying DNSSEC . …

These factors are then weighted. The current weights are:

```
1 * ProtocolScore,
 0.7 * PermCertValidityScore,
 1 * StrictCertValidityScore,
 0.7 * PermPublicKeyTypeSizeScore,
 1 * StrictPublicKeyTypeSizeScore,
 0.7 * PermSigAlgScore,
 1 * StrictSigAlgScore,
 1 * IPV4CertValidityScore,
 1 * IPV6CertValidityScore,
 1 * CAAUsageScore,
 ( 1 - 0.1 * CSUsageScore_TLS10),
 ( 1 - 0.1 * CSUsageScore_TLS11),
 0.3 * CSUsageScore_TLS12,
 0.5 * CSUsageScore_TLS13,
1 * DMARCScore,
1 * SPFScore,
.2 * TLSAScore,
.3 * MTASTSScore,
```

Each scoring factor column in the matrix is ordered from highest to lowest and then ranked. There are two possible ranks for the country (a) Strict rank, and (b) Permissive rank, which depend on the strict scores (`StrictCertValidityScore, StrictPublicKeyTypeSizeScore, StrictSigAlgScore`) and permissive scores (PermCertValidityScore, PermPublicKeyTypeSizeScore, PermSigAlgScore) keeping all other columns consistent in both rankings.

Scoring Challenges

As with any scientific program, we are aware of challenges and open questions. This section outlines one: evaluation and how it informs scoring.

Evaluation will be an important part of this project. Some of the things we directly measure require an evaluation step or steps. For example, if we want to look at TLS configuration of websites, we can evaluate the version of TLS and the cipher choices for which the server is configured. We might get a response like "New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256." In order to assess whether that is a reasonable choice, we need to evaluate it against some criteria. Sometimes criteria will disagree. For example, some guidance states "Use TLS 1.3 only" while others allow earlier versions in some circumstances. We will need to define what evaluation criteria we are using and why.

Additionally, the scoring mechanism could get very nuanced and opinionated. For example, within TLS, it might be useful to score hosts which use TLS compared to those which don't. The decision for assigning scores based on this criteria reflects on the security offered by the website or mail server since any usage of TLS is better than none. However, within TLS, the usage of TLS 1.0 and 1.1 according to today's best practice should ideally be penalized, but one might argue that some security is better than none. This argument further extends to hosts which use relatively insecure cipher suites with weaker security protocols (eg. SHA1 as the MAC). While most adversaries may not be able to spend 11000 USD to crack a single SHA1 hash as shown in the SHAttered attack, the adversarial model assumed may need to classify the usage of such algorithms as insecure.

Many websites might continue to support older TLS protocols purely because of outdated client devices which might still need support for these devices. Similarly, for mail servers, servers which advertise AUTH-PLAIN prior to establishing a secure session through STARTTLS put themselves at risk of credential sniffing attacks by network adversaries, but the ability to support TLS connections should be scored towards improved security scores compared to those mail servers that do not support TLS connections. Arguably, servers without TLS support and advertising insecure authentication mechanisms must be penalized in the scoring mechanism but the complexity of the scoring functions might make them difficult to implement and explain as more factors dependent on each other are taken into account.

We have currently defined a scoring algorithm for TLS (security protocols), for certificates, and for email. The TLS algorithm is one of two under discussion, and it differs from IETF "Best Current Practice," because the IETF is balancing concerns of interoperability with security.

That balance may represent a community who cannot update. This may represent an important public health gap. Systems may be held in a less state to enable interoperability with systems that are not being updated.

We have discussed calculating what we informally call an "IETF score" where we assign points for doing what the IETF says you must, and taking them away for things they say you must not do.

We are considering calculating both scores, and seeing what the differences show. We are also planning to evaluate our evaluation criteria as we gather data.

References

High Performance Browsing Network. https://hpbn.co/transport-layer-security-tls/

Pochat, Victor Le, et al. "Tranco: A research-oriented top sites ranking hardened against manipulation." *arXiv preprint arXiv:1806.01156* (2018).

Winslow, C E A. The Untilled Fields of Public Health (PDF), Science 51(1306): 23-33 (Jan. 9, 1920).

Appendix A: Score Algorithm

```
a. For each country C in the Keys(CH_map)
b. For each hostname H, and corresponding result R in the
   HR_map
     i.   CertificateStatus = R[Certificate]
    ii.   ValidityScore = { }
   iii.   If CertificateStatus is Empty:
             1. CU_map[C] = append(CU_map[C], H)
    iv.   For each IP, CertificateInfo in CertificateStatus
             1. ValidityStatus = CertificateInfo.status.isValid
             2. ErrorReason = CertificateInfo.status.error
             3. AssignedScore = Lookup Certificate Invalidity
                Score[(Validity, ErrorReason)]
             4. PublicKey = CertificateInfo.publicKey
             5. ValidityScore[PublicKey] =
                MIN(ValidityScore[PublicKey], AssignedScore)
     v.   ValidityStatus = [ ]
    vi.   For _PublicKey, Validity in ValidityScore:
             1. append(ValidityStatus, Validity)
   vii.   CV_map[C] = append(CV_map[C], AND([True if status == 1
          else False for status in ValidityStatus]))
  viii.   CertScore[C] = append(CertScore[C],
          SUM(ValidityStatus) // LEN(ValidityStatus))
c. For each country C in Keys(CH_map)
     i.   UnresolvedHostCount = LEN(CU_map[C]) || 0
    ii.   ResolvedCertStatus = CV_map[C] || [ ]
   iii.   ResolvedCertScore = CertScore[C] || [ ]
    iv.   Valids = SUM(ResolvedCertStatus)
     v.   Invalids = LEN(ResolvedCertStatus) - Valids
    vi.   ValidityScore = SUM(ResolvedCertScore) /
          (LEN(ResolvedCertScore) + UnresolvedHostCount) *
          100.0
   vii.   ValidPercentage = Valids / (Valids + Invalids +
          UnresolvedHostCount) * 100.0
  viii.   TLS_valid[C] = ValidPercentage
    ix.   TLS_adjusted[C] = ValidityScore
d. Return the following scores:
     i.   Basic score: TLS_valid containing C → Validity %age
    ii.   Adjusted score: TLS_adjusted containing C → Adjusted
          Validity %age including penalty
```

Validity status is scored as follows

Validity Status	**Error Code (P=Penalty)**		**Score**
Valid (1 | True)	No Error		1
Invalid (0 | False)	Error		0
	P1 - Hostname Mismatch		-1
	P2 - Self Signed Certificates		-2
	P3 - Expired Certificates	< 90 days	-1
		Between 91 and 365 days	-2
		> 365 days	-3
	P4 - Missing SAN Names		-1
	P5 - Usage of legacy Common Name		-2
	P6 - Unhandled Critical Extension		-1

Results

1. Initialize an empty map CV_{map} indicating the country mapping to the TLS validity status of various hostnames, this will contain a list of booleans eg. *Singapore → [T, T, F …]*
2. Initialize an empty map CU_{map} indicating the country maps to unresolved hosts
3. Initialize a TLS Validity Result TLS_{valid} = { } mapping country to a percentage indicating the validity of certificates for the respective hostnames.
4. Initialize a Certificate Validity Score CertScore mapping country to a TLS validity score
5. Initialize $TLS_{adjusted}$ to store the adjusted scores mapping to each country.