High-Tech Bridge's Free SSL Server Test

API Developer Documentation

Version v1.1

22nd of December 2016

Table of Contents

General overview

High-Tech Bridge provides you with a free API to test your SSL/TLS servers. To assure high speed of service and availability for everyone, the free API allows 20 requests in 3 minutes, 250 requests in total per 24 hours, from one IP address.

In order to prevent abuse, a protection mechanism has been set up to remove the ability to test IPs that are not related to tested domain name. As a consequence if a domain name is resolved into several IPs, a second request will be mandatory, specifying one of the IPs replied by the server along with the token issued (examples are below). However, if the tested domain name can be resolved into only one IP address, it will be immediately tested.

License notice: The API is provided for free both for private and commercial purposes. If you use the API for publicly available service (commercial or not) a link to High-Tech Bridge's Free SSL Server Test is mandatory.

API Specifications

Field Name	Value
Protocol	HTTPS
Request Type	POST
URL	https://www.htbridge.com/ssl/api/v1/check/[ustamp].html - where "ustamp" is an arbitrary UNIX time-stamp (must be an integer). Such construction is done to prevent caching on client side.

POST Data Specification

Field Name	Value
domain:port	must be a valid domain name, or IP address, followed by a port number. If port is not supplied, 443 is used by default.
show_test_results	"false" means that test results will be hidden, "true" means that test results will be displayed in statistics.
choosen_ip	IP address of tested server (if tested domain resolves to multiple addresses).
recheck	"false" will use results from cache if the server has been tested within the past 24 hours, "true" will perform a new test without looking at the cache.
verbosity	1 means output will be detailed, 0 means output will be short.
token	value of the token sent by the server if the tested domain is resolved into several IP addresses.

Example of Transaction using CURL

```
$ curl -XPOST -d 'domain=twitter.com:443&show_test_results=true&recheck=false&verbosity=1'
'https://www.htbridge.com/ssl/api/v1/check/1451425590.html'

{"multiple_ips":["199.16.156.6","199.16.156.102","199.16.156.70","199.16.156.230"],"token":"68j3OCZLEomtjASxKoObjZXzX
7p2M7L0"}


$ curl -XPOST –d
'domain=twitter.com:443&show_test_results=true&recheck=false&choosen_ip=199.16.156.230&verbosity=1&token=68j3O
CZLEomtjASxKoObjZXzX7p2M7L0' 'https://www.htbridge.com/ssl/api/v1/check/1451425590.html'
```

The output array will be composed of the following main elements that will be detailed later in this document:

- **server_info**: containing basic server info, like IP, port, reverse DNS...
- **certificates**: containing information about certs, graphs…
- **nist**: containing all information about NIST compliance
- **hipaa**: containing all information about HIPAA compliance
- **pci_dss**: containing all information about PCI DSS compliance
- **industry_best_practices**: containing all information about industry best practices
- **third_party_content**: containing all the third party content
- **results**: containing the score and the grade
- **highlights**: containing the highlights

The most common syntax for elements is the following one:

```
{ "industry_best_practices":
      "cert_ev": {
            "value": true,
            "message_id": 45,
            "tag": 1,
            "description_id": 0,
            "title_id": 12,
            "visible": true,
            "message": "All the certificates provided by the server are Extended
            Validation (EV) certificates.",
            "title": "CERTIFICATES PROVIDE EV"
      }
}
```

value contains the value for the configuration element, quite often it is a Boolean, but it can sometimes be a string, an integer or something else.

message_id contains an integer, which is the id of the corresponding text message to display. Messages are defined in the appendix table of messages.

tag contains an integer, which is the id we currently use for colors and tags (information, good configuration…).

description_id, just like message_id contains an integer representing the id of the text that has to be shown when mouse is over the small "?" to provide more information. When it is set to 0 (empty), this means that no description is provided and then the small "?" infobox should not be displayed. Descriptions are defined in appendix table of descriptions.

title_id contains an integer like for message_id. This integer is the index of the title of the UI element that should be displayed.

visible contains a boolean acting like a flag to specify if the element must be displayed in UI or not. If set to true, the item must be displayed on UI, if false, it must not be.

message contains textual representation of corresponding message_id

title contains textual representation of corresponding title_id

Server information

Server information part contains different elements about server itself:

"**ip**":

- Syntax: `{ value: string, tag: int }`
- Always present
- Description: the IP address tested

"**port**":

- Syntax: `{ value: int, tag: int }`
- Always present
- Description: the tcp port tested

"**hostname**":

- Syntax: `{ value: string, tag: int }`
- Always present
- Description: the hostname tested (can be an ip address)

"**reverse_dns**":

- Syntax: `{ value: string, tag: int }`
- Always present
- Description: the reverse DNS for the IP tested

"**http_response**":

- Syntax: `{ value: string, tag: int }`
- Absent if server does not support SSL/TLS
- Description: the HTTP response code to a GET request

"**server_signature**":

- Syntax: `{ value: string, tag: int }`
- Absent if server does not support SSL/TLS
- Description: the content of Server HTTP Header

Certificates

Certificates part include certificate information, chains and graphs the same way it was in the previous version, excepted that attributes names are now in lower_case (snake case). It is composed of 3 main sub parts:

- **information**: a list containing detailed information about server certificates,
- **chain_installation_issues:** list of certificate chain insallation issues,
- **chains**: a list containing ordered certificates trust paths,
- **graphs**: a list containing the information needed to build graphs

Information:

As mentioned, the information part is a list containing all server certificates. One server certificate has the following attributes:

"**key_type**":

- Syntax: string
- Always present
- Description: contains the type of key associated (RSA, ECDSA…)

"**key_size**":

- Syntax: int
- Always present
- Description: contains the size of the key associated in bits

"**signature_algorithm**":

- Syntax: string
- Always present
- Description: contains the signature algorithm used to sign the certificate

"**cn**":

- Syntax: string
- Always present
- Description: contains Common Name of the certificate

"**san**":

- Syntax: string
- Always present
- Description: contains the Subject AltNames for which the certificate is valid

"**transparency**":

- Syntax: bool
- Always present
- Description: set to true if the certificate provides transparency

"**ev**":

- Syntax: bool
- Always present
- Description: set to true if the certificate provides Extended Validation

"**valid_from**":

- Syntax: int (timestamp)
- Always present
- Description: the date from which the certificate is valid

"**valid_to**":

- Syntax: int (timestamp)
- Always present
- Description: the expiration date of the certificate

"**valid_now**":

- Syntax: bool
- Always present
- Description: set to true if the certificate is valid at the time of testing

"**expires_soon**":

- Syntax: bool
- Always present
- Description: set to true if the certificate expires in less than 30 days

"**self_signed**":

- Syntax: bool
- Always present
- Description: set to true if the certificate is self-signed

"**ocsp_must_staple**":

- Syntax: bool
- Always present

- Description: set to true if the certificate has must staple extension

"**supports_ocsp_stapling**":

- Syntax: bool
- Always present
- Description: set to true if the certificate supports OCSP stapling

"**valid_for_host**":

- Syntax: bool
- Always present
- Description: set to true if the certificate is valid for the domain tested

"**revoked**":

- Syntax: bool
- Always present
- Description: set to true if the certificate has been revoked

"**known_issuer**":

- Syntax: bool
- Always present
- Description: set to true if the CA that signed the certificate is trusted

"**revocation_information**":

- Syntax: Array comprised of two elements. Elements can be either an array or boolean false depending on presence of matching revocation method. Arrays have the following synthax `{ url: string, revoked: bool, error: bool }`

```
{ "revocation_information":
  { "ocsp": [
      {"url": "http://ti.symcd.com",
       "revoked": false,
       "error": false
      }
  ] },
  { "crl": false }
}
```

- If revocation method is not present, i.e. no CRL, value of matching element is set to boolean **false**

"**trusted**":

- Syntax: bool
- Always present
- Description: set to true if the certificate can be trusted

Chain installation issues:

This section is a list of several installation issue checkups for server certificates. There's a boolean field '**value**' which is set to '**True**' if there is any issue detected, along with it is an array '**results**' annotating the results of separate checks. The structure is as follows:

"**chain_installation_issues**":

- Syntax: `{ value: bool, results: array }`
- Description: value set to True if there is any issue detected, results array populated with separate issue check results.

  The structure of the '**results**' array is as follows:

"**is_chain_complete**":

- Syntax: `{ value: bool, message_id: int, tag: int }`
- Description: set to true if server sends intermediate certs for at least one chain

"**has_sent_root_ca**":

- Syntax: `{ value: bool, message_id: int, tag: int }`
- Description: set to true if the server sends root CA in the cert chain

"**is_order_correct**":

- Syntax: `{ value: bool, message_id: int, tag: int }`
- Description: set to true if certificate chain was provided in correct order

"**has_sent_extra_certs**":

- Syntax: `{ value: bool, message_id: int, tag: int }`
- Description: set to true if server has sent certs that were not expected

Chains:

The chain part is a list of certificate chains that have been reconstructed from the server certificates. A certificate chain is an ordered list of certificates from the server certificate (leaf certificate) to the root CA certificate.

Information provided about the certificates are different as we need at the same time less detailed information and some specific information for instance about HPKP. The attributes of a certificate are the following:

"**sha256**":

- Syntax: string
- Always present
- Description: contains the sha256 sum of the certificate

"**cn**":

- Syntax: string
- Always present
- Description: contains Common Name of the certificate

"**key_type**":

- Syntax: string
- Always present
- Description: contains the type of key associated (RSA, ECDSA…)

"**key_size**":

- Syntax: int
- Always present
- Description: contains the size of the key associated in bits

"**signature_algorithm**":

- Syntax: string
- Always present
- Description: contains the signature algorithm used to sign the certificate

"**valid_to**":

- Syntax: int (timestamp)
- Always present
- Description: the expiration date of the certificate

"**pin**":

- Syntax: string
- Always present
- Description: the pin of the corresponding public key, used in HPKP

"**matches_hpkp**":

- Syntax: bool
- Always present
- Description: set to true if this certificate is pinned

"**cert_type**":

- Syntax: string
- Always present

- Description: contains the type of the cert among:
  - Server certificate
  - Intermediate CA
  - Root CA

"**comment**":

- Syntax: string
- Always present, can be empty
- Description: contains a comment about the certificate among:
  - Self-signed
  - Extended Validation

"**weak_key_size**":

- Syntax: bool
- Always present
- Description: set to true if key size is small for specifig signature algorithm

"**weak_signature_algorithm**":

- Syntax: bool
- Always present
- Description: set to true if signature algorithm is weak

Graphs:

The graphs part contains mainly the same information as the chains part, but with level and children information and trying to remove duplicates from chains. This way, it is possible to draw relationships between certificates.

This section is a list of certificate graphs, each graph being a list of certificates having the following attributes:

"**sha256**":

- Syntax: string
- Always present
- Description: contains the sha256 sum of the certificate

"**cn**":

- Syntax: string
- Always present
- Description: contains Common Name of the certificate

"**key_type**":

- Syntax: string
- Always present
- Description: contains the type of key associated (RSA, ECDSA…)

"**key_size**":

- Syntax: int
- Always present
- Description: contains the size of the key associated in bits

"**signature**":

- Syntax: string
- Always present
- Description: contains the signature algorithm used to sign the certificate

"**valid_to**":

- Syntax: int (timestamp)
- Always present
- Description: the expiration date of the certificate

"**pin**":

- Syntax: string
- Always present
- Description: the pin of the corresponding public key, used in HPKP

"**matches_hpkp**":

- Syntax: bool
- Always present
- Description: set to true if this certificate is pinned

"**cert_type**":

- Syntax: string
- Always present
- Description: contains the type of the cert among:
    - Server certificate
    - Intermediate CA
    - Root CA

"**comment**":

- Syntax: string
- Always present, can be empty
- Description: contains a comment about the certificate among:
    - Self-signed
    - Extended Validation

"**children_hashes**":

- Syntax: list of strings
- Always present, can be empty
- Description: every string contained in this list is the sha256 sum of certificates that have been signed with the current one

"**tree_levels**":

- Syntax: list of integers
- Always present
- Description: every integer contained in this list corresponds to the level of the certificate in the chain, starting from the server certificate. Server certificates have level=0.

NIST

NIST part contains all information related to NIST compliance:

"**cert_x509_v3**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server certificate is an X509 certificate in version 3

"**cert_self_signed**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server cert is self-signed

"**cert_provides_revocation_information**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if certificate provides revocation information

"**cert_small_key**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the private key is too small

"**cert_signature_algorithm_mismatch**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the cert has been signed with a wrong algorithm

"**cert_weak_signature**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the cert has not been signed using SHA2

"**supported_protocols**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int }`. Only supported versions are present:

```
{ "nist":
      { "supported_protocols": [
            { "value": "SSLv3", "tag": 2 },
            { "value": "TLSv1.1", "tag": 1 }
      ] },
}
```

- Absent if server does not support SSL/TLS
- Description: lists supported protocols

"**supports_invalid_protocols**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports protocols that are not approved by NIST

"**supported_cipher_suites**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int, protocols: array }`. Only supported cipher suites are present:

```
{ "nist":  { "supported_cipher_suites": [
```

```
            {"value": "TLS_RSA_WITH_CAMELLIA_128_CBC_SHA",
             "tag": 2,
             "protocols": ["TLSv1.0", "TLSv1.1"]
            },
            {"value": "TLS_DHE_RSA_WITH_AES_256_GCM_SHA384",
             "tag": 1,
             "protocols": ["TLSv1.2"]
            }
    ] } }
```

- Absent if server does not support SSL/TLS
- Description: lists supported cipher suites

"**supports_invalid_cipher_suites**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports cipher suites that are not approved by NIST

"**dh_parameter_size**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support DH suites
- Description: gives the size of the Diffie-Hellman parameter in bits

"**dh_parameter_weak**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support DH suites
- Description: set to true if the Diffie-Hellman parameter size is below NIST requirements (2048)

"**supported_elliptic_curves**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, size: int, tag: int }`. Only supported curves are present:

```
{ "nist": {"supported_protocols": [
            { "value": "P-521 (secp521r1)", "size": 521, "tag": 1 },
            { "value": "P-384 (secp384r1)", "size": 384, "tag": 1 },
      ] }
}
```

- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: lists supported elliptic curves

"**supports_invalid_curves**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: set to true if server supports elliptic curves that are not approved by NIST

"**supports_mandatory_curves**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves

"**supports_tlsv1.1**":

- Description: set to true if server supports at least one of the mandatory curves
- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports TLSv1.1

"**missing_mandatory_ciphers**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int }`. Only missing ciphers that are mandatory are present:

```
{ "nist": {"missing_mandatory_ciphers": [
            { "value": "TLS_RSA_WITH_AES_128_CBC_SHA",
              "tag": 2,
              "protocols": ["TLSv1.0". "TLSv1.1", "TLSv1.2"],
            },
            { "value": "TLS_RSA_WITH_3DES_EDE_CBC_SHA",
              "tag": 2,
              "protocols": ["TLSv1.0". "TLSv1.1"],
            }
      ] }
}
```

- Absent if server does not support SSL/TLS, empty if no mandatory cipher is missing
- Description: lists missing ciphers

"**has_all_mandatory_ciphers**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to false if server is missing mandatory ciphers

"**supports_ocsp_stapling**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if no certificate has OCSP URI set
- Description: set to true if server supports OCSP stapling

"**provides_reneg_information**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server say if it supports or not secure renegotiation

"**ec_point_format**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: set to true if server supports ec point format TLS extension

"**compliant**":

- Syntax: `{ value: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server is compliant with NIST guidelines

HIPAA

HIPAA part contains all information related to HIPAA compliance:

"**cert_x509_v3**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server certificate is an X509 certificate in version 3

"**cert_self_signed**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server cert is self-signed

"**cert_provides_revocation_information**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if certificate provides revocation information

"**cert_small_key**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the private key is too small

"**cert_signature_algorithm_mismatch**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the cert has been signed with a wrong algorithm

"**cert_weak_signature**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the cert has not been signed using SHA2

"**supported_protocols**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int }`. Only supported versions are present:

```
{ "nist": {"supported_protocols": [
            { "value": "SSLv3", "tag": 2 },
            { "value": "TLSv1.1", "tag": 1 }
      ] }
}
```

- Absent if server does not support SSL/TLS
- Description: lists supported protocols

"**supports_invalid_protocols**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports protocols that are not approved by HIPAA

"**supported_cipher_suites**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int, protocols: array }`. Only supported cipher suites are present:

```
{ "hipaa": {"supported_cipher_suites": [
            { "value": "TLS_RSA_WITH_AES_128_CBC_SHA",
              "tag": 2,
              "protocols": ["TLSv1.1", "TLSv1.2"]
            },
            { "value": "TLS_RSA_WITH_3DES_EDE_CBC_SHA",
              "tag": 1,
              "protocols": ["TLSv1.1"]
            }
      ] }
}
```

- Absent if server does not support SSL/TLS
- Description: lists supported cipher suites

"**supports_invalid_cipher_suites**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports cipher suites that are not approved by HIPAA

"**dh_parameter_size**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support DH suites
- Description: gives the size of the Diffie-Hellman parameter in bits

"**dh_parameter_weak**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support DH suites
- Description: set to true if the Diffie-Hellman parameter size is below HIPAA requirements (2048)

"**supported_elliptic_curves**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, size: int, tag: int }`. Only supported curves are present:

```
{ "hipaa":
      {"supported_elliptic_curves": [
            {
```

```
                "value": "P-521 (secp521r1)",
                "size": 521,
                "tag": 1
        },

        {
                "value": "P-384 (secp384r1)",
                "size": 384,
                "tag": 1
        }

    ] }
}
```


- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: lists supported elliptic curves

"**supports_invalid_curves**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: set to true if server supports elliptic curves that are not approved by HIPAA

"**supports_mandatory_curves**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: set to true if server supports at least one of the mandatory curves

"**supports_tlsv1.1**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports TLSv1.1

"**missing_mandatory_ciphers**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int }`. Only missing ciphers that are mandatory are present:


```
{ "nist":
    { "missing_mandatory_ciphers": [
        { "value": "TLS_RSA_WITH_AES_128_CBC_SHA", "tag": 2 },
        { "value": "TLS_RSA_WITH_3DES_EDE_CBC_SHA", "tag": 2 }
    ] }
```

```
}
```

- Absent if server does not support SSL/TLS, empty if no mandatory cipher is missing
- Description: lists missing ciphers

"**has_all_mandatory_ciphers**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to false if server is missing mandatory ciphers

"**supports_ocsp_stapling**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if no certificate has OCSP URI set
- Description: set to true if server supports OCSP stapling

"**provides_reneg_information**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server say if it supports or not secure renegotiation

"**ec_point_format**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: set to true if server supports ec point format TLS extension

"**compliant**":

- Syntax: `{ value: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server is compliant with HIPAA

PCI DSS

"**cert_small_key**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the private key is too small

"**cert_weak_signature**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the cert has not been signed using SHA2

"**cert_trusted**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the cert could be trusted

"**supported_protocols**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int }`. Only supported versions are present:

```
{ "pci_dss":
      { "supported_protocols": [
            { "value": "SSLv3", "tag": 2 },
            { "value": "TLSv1.1", "tag": 1 }
      ] }
}
```

- Absent if server does not support SSL/TLS
- Description: lists supported protocols

"**supports_invalid_protocols**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports protocols that are not approved by PCI DSS

"**supported_cipher_suites**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, tag: int, protocols: array }`. Only supported cipher suites are present:

```
{ "pci_dss":
    {"supported_cipher_suites": [
            { "value": "TLS_RSA_WITH_AES_128_CBC_SHA",
              "tag": 2,
              "protocols": ["TLSv1.1", "TLSv1.2"]
            },
            { "value": "TLS_RSA_WITH_3DES_EDE_CBC_SHA",
              "tag": 1,
              "protocols": ["TLSv1.1"]
            }
      ] }
}
```

- Absent if server does not support SSL/TLS
- Description: lists supported cipher suites

"**supports_invalid_cipher_suites**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports cipher suites that are not approved by PCI DSS

"**dh_parameter_size**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support DH suites
- Description: gives the size of the Diffie-Hellman parameter in bits

"**dh_parameter_weak**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support DH suites
- Description: set to true if the Diffie-Hellman parameter size is below PCI DSS requirements (2048)

"**supported_elliptic_curves**":

- Syntax: the element is a list of arrays having the following syntax: `{ value: string, size: int, tag: int }`. Only supported curves are present:

```
{ "pci_dss":
      { "supported_elliptic_curves": [
            {
                  "value": "P-521 (secp521r1)",
```

```
                "size": 521,
                "tag": 1
            },

            {
                "value": "P-384 (secp384r1)",
                "size": 384,
                "tag": 1
            }

        ] }
    }
```


- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: lists supported elliptic curves

"**supports_invalid_curves**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not support elliptic curves
- Description: set to true if server supports elliptic curves that are not approved by PCI DSS

"**poodle_tls**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: result of the poodle over TLS test
- Values:
    - -1 error
    - 0 not vulnerable
    - 1 vulnerable
    - 2 maybe vulnerable

"**cve_2016_2107**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: result of the CVE-2016-2107 test
- Values:
    - -1 error
    - 0 not vulnerable
    - 1 vulnerable
    - 2 maybe vulnerable

"**supports_insecure_reneg**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the server supports client-initiated insecure renegotiation

"**heartbleed**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the server is vulnerable to heartbleed

"**cve_2014_0224**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if test requires STARTTLS
- Description: result of the poodle over TLS test
- Values:
  - -1 error
  - 0 not vulnerable
  - 1 vulnerable
  - 2 maybe vulnerable

"**drown**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the server is vulnerable to drown

"**poodle_ssl**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the server is vulnerable to poodle over SSL

"**compliant**":

- Syntax: `{ value: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server is compliant with PCI DSS requirements

Industry Best Practices

This part is about best practices.

"**version_intolerance**":

- Syntax: the element is a list of arrays having the following syntax: `{ hex: string, value: string, tag: int } :`

```
{ "industry_best_practices":
      { "version_intolerance": [
            {
                  "hex": "0x0304",
                  "value": "TLSv1.3",
                  "tag": 3
            },
            {
                  "hex": "0x03FF",
                  "value": "TLSv1.FF",
                  "tag": 3
            }
      ] }
}
```

- Absent if server implementation is working fine and is not version intolerant.
- Description: lists existent or non existent TLS versions for which server raises an alert or shuts down the connection, in other words, if it does not tolerate them.

"**cert_valid_too_long**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if cert has been signed for more than 3 years

"**cert_ev**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if cert provides with Extended Validation

"**http_to_https_redirect**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if an error occurs with this test
- Description: set to true if server redirects from HTTP to HTTPS

"**https_to_http_redirect**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if there is no redirection from HTTPS to HTTP
- Description: set to true if server redirects from HTTPS to HTTP

"**mixed_content**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if an error occurs with this test
- Description: set to true if HTTP content is included into HTTPS

"**supports_tlsv1.2**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the server supports TLSv1.2

"**has_preference**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server cipher suite preference is enabled

"**cipher_preference**":

- Syntax: the element is a list of arrays having the following syntax: `{ protocol: string, value: string, size: int, tag: int } :`

```
{ "industry_best_practices":
      { "cipher_preference": [
            {
                  "protocol": "TLSv1.0",
                  "value": "TLS_DHE_RSA_WITH_AES_256_CBC_SHA",
                  "tag": 1
            },

            {
                  "protocol": "TLSv1.2",
                  "value": "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
                  "tag": 1
            }
      ] }
}
```

- Absent if server does not support SSL/TLS, empty if server has no preference
- Description: lists cipher suites preferred per protocol

"**prefers_weak_ciphers**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if the server prefers cipher suites that have not been approved by PCI DSS

"**prefers_pfs**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server prefers cipher suites providing Perfect Forward Secrecy

"**supports_fallback_scsv**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if TLS Fallback SCSV is not needed (server supports only 1 protocol)
- Description: set to true if server supports TLS Fallback SCSV

"**supports_client_initiated_reneg**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports client-initiated renegotiation

"**supports_secure_reneg**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or if server does not provide information regarding support
- Description: set to true if server supports secure renegotiation

"**tls_compression**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS
- Description: set to true if server supports TLS compression

"**has_hsts**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or is not a web server

- Description: set to true if server enforces HSTS

"**hsts_duration**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or does not support HSTS
- Description: this is the duration of HSTS max-age in seconds

"**hsts_long**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or does not support HSTS
- Description: set to true if server's HSTS max-age is above 180 days

"**has_hpkp**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or is not a web server
- Description: set to true if server has HPKP header

"**hpkp_valid**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or does not support HPKP
- Description: set to false if HPKP syntax is invalid or if no pin match

"**hpkp_duration**":

- Syntax: `{ value: int, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or HPKP is invalid
- Description: this is the duration of HPKP max-age in seconds

"**hpkp_long**":

- Syntax: `{ value: bool, message_id: int, tag: int, description_id: int, title_id: int, visible: bool }`
- Absent if server does not support SSL/TLS or HPKP is invalid
- Description: set to true if server's HPKP max-age is above 60 days

Third-party content

This part lists all third-party content found on index page. Tag values are 1 when content is fetched using HTTPS or 3 otherwise.

```
{ "third_party_content": [
      { "value": string, "tag": int },
      ...
] }
```

Results

Results part contains the main results. The attributes are the following:

"**has_ssl_tls**":

- Syntax: boolean
- Always present
- Description: set to true if server supports SSL/TLS

"**score**":

- Syntax: int
- Always present
- Description: server's score

"**grade**":

- Syntax: string
- Always present
- Description: server's grade

Highlights

This part lists highlights in an ordered way. The syntax is the following:

```
{ "highlights": [
      { "highlight_id": int, "tag": int, "highlight": string },
      ...
] }
```

Just like “message_id” or “description_id”, “highlight_id” contains the index where to find the appropriate text in appendix table of messages, descriptions and higlights.

Error handling

If an error occurs, only basic information and an error message will be returned the following way:

```
{ "error": string }
```

or

```
{ "error": string, "server_info": {
      "ip": string,
      "port": int,
      "hostname": string,
      "reverse_dns": string
} }
```

Possible error messages which system can return are:

- `You have performed N tests in the last 3 minutes. The system is currently busy, please try again later.`
- `Domain name was resolved to an invalid IP address.`
- `Invalid IP address.`
- `The domain name does not exist.`
- `An error occured while testing server configuration, server became unreachable during the test.`

Appendix 1: Tag values list

Below is a reminder of the list of tags that are used:

Tag value	Description
0	Nothing, empty
1	Good configuration
2	Non-compliant with NIST guidelines
3	Misconfiguration or weakness
4	Information
5	Non-compliant with PCI DSS requirements
6	Not compliant with NIST and PCI DSS
7	Not vulnerable
8	Deprecated. Dropped in Jun 2018
9	Non-compliant with HIPAA guidance
10	Non-compliant with NIST and HIPAA
11	Non-compliant with HIPAA and PCI DSS
12	Non-compliant with NIST, HIPAA and PCI DSS
13	No Encryption

Appendix 2: List of Messages values

ID	Value
1	The version of the RSA X509 certificate provided by the server is prior to version 3 (the latest one).
2	The version of the ECDSA X509 certificate provided by the server is prior to version 3 (the latest one).
3	The version of the following X509 certificates provided by the server is prior to version 3 (the latest one): RSA, ECDSA.
4	Some of the X509 certificates provided by the server are prior to version 3 (the latest one).
5	All the X509 certificates provided by the server are in version 3.
6	The RSA certificate provided by the server is self-signed.
7	The ECDSA certificate provided by the server is self-signed.
8	The following certificates are self-signed: RSA, ECDSA.
9	Some of the certificates provided by the server are self-signed.

10	All the certificates provided by the server have been signed by a CA.
11	The RSA certificate provided is missing OCSP URI and crlDistributionPoints extension, making impossible to verify if it has been revoked.
12	The ECDSA certificate provided is missing OCSP URI and crlDistributionPoints extension, making impossible to verify if it has been revoked.
13	The following certificates are missing OCSP URI and crlDistributionPoints extension, making impossible to verify if they have been revoked: RSA, ECDSA.
14	Some of the certificates provided are missing OCSP URI and crlDistributionPoints extension, making impossible to verify if they have been revoked.
15	All the certificates sent by the server provide ways to check their revocation status.
16	The RSA certificate's key length is too small.
17	The ECDSA certificate's key length is too small.
18	The following certificates' key lengths are too small: RSA, ECDSA.
19	Some of the certificates have a public key that is too small.
20	All the certificates provided have public keys that are long enough.
21	The RSA certificate provided has not been signed using the proper algorithm according to NIST guidelines.
22	The ECDSA certificate provided has not been signed using the proper algorithm according to NIST guidelines.
23	The following certificates have not been signed using the proper algorithm according to NIST guidelines: RSA, ECDSA.
24	Some of the certificates provided have not been signed using the proper algorithm according to NIST guidelines.
25	All the certificates provided have been signed using the proper algorithm.
26	The RSA certificate provided has been signed using a weak algorithm.
27	The ECDSA certificate provided has been signed using a weak algorithm.
28	The following certificates have been signed using a weak algorithm: RSA, ECDSA.
29	Some of the certificates provided have been signed using a weak algorithm.
30	All the certificates provided have been signed using a strong algorithm.

31	The RSA certificate provided has been validated for more than 3 years. This means that the private key of the server will remain the same for more than 3 years. NIST guidelines suggest limiting certificate validity to 3 years maximum.
32	The ECDSA certificate provided has been validated for more than 3 years. This means that the private key of the server will remain the same for more than 3 years. NIST guidelines suggest limiting certificate validity to 3 years maximum.
33	The following certificates have been validated for more than 3 years: RSA, ECDSA. This means that the private keys of the server will remain the same for more than 3 years. NIST guidelines suggest limiting certificate validity to 3 years maximum.
34	Some of the certificates provided have been validated for more than 3 years. This means that the private keys of the server will remain the same for more than 3 years. NIST guidelines suggest limiting certificate validity to 3 years maximum.
35	All the certificates provided have been validated for less than 3 years.
36	The RSA certificate provided by the server could not be trusted.
37	The ECDSA certificate provided by the server could not be trusted.
38	The following certificates provided by the server could not be trusted: RSA, ECDSA.
39	Some of the certificates provided by the server could not be trusted.
40	All the certificates provided by the server are trusted.
41	The RSA certificate provided is NOT an Extended Validation (EV) certificate.
42	The ECDSA certificate provided is NOT an Extended Validation (EV) certificate.
43	The following certificates are NOT Extended Validation (EV) certificates: RSA, ECDSA.
44	Some of the certificates provided are NOT Extended Validation (EV) certificates.
45	All the certificates provided by the server are Extended Validation (EV) certificates.
46	The HTTP version of the website does not redirect to the HTTPS version. We advise to enable redirection.
47	The HTTP version of the website redirects to the HTTPS version.
48	The website includes HTTP content in HTTPS.
49	The Diffie-Hellman parameter's size is only $value bits. A longer one must be generated to prevent Logjam vulnerability.

50	The server does not support P-256 or P-384 curves which are required by NIST guidelines.
51	The support of TLSv1.1 is mandatory according to NIST guidelines.
52	The server supports TLSv1.1 which is mandatory to comply with NIST guidelines.
53	The server supports TLSv1.2 which is the only SSL/TLS protocol that currently has no known flaws or exploitable weaknesses.
54	The server does not support TLSv1.2 which is the only SSL/TLS protocol that currently has no known flaws or exploitable weaknesses.
55	The server does not prefer cipher suites. We advise to enable this feature in order to enforce usage of the best cipher suites selected.
56	The server enforces cipher suites preference.
57	The server prefers cipher suite that has not been approved by PCI DSS requirements for at least one of the supported protocols.
58	For TLS family of protocols, the server prefers cipher suite(s) providing Perfect Forward Secrecy (PFS).
59	The server does not prefer cipher suites providing strong Perfect Forward Secrecy (PFS). We advise to configure your server to prefer cipher suites with ECDHE or DHE key exchange.
60	The server provides HTTP Strict Transport Security for more than 6 months: $value seconds
61	The server provides HTTP Strict Transport Security for less than 6 months: $value seconds
62	The server does not enforce HTTP Strict Transport Security. We advise to enable it to enforce the user to browse the website in HTTPS.
63	The server provides HTTP Public Key Pinning for more than 2 months: $value seconds
64	The server provides HTTP Public Key Pinning for less than 2 months: $value seconds
65	The server sends an invalid HPKP header: the certificate chain does not match the signature sent, or the syntax is invalid. We advise to review your configuration.
66	The server does not enforce HTTP Public Key Pinning which helps preventing man-in-the-middle attacks.
67	The server supports TLS_FALLBACK_SCSV extension for protocol downgrade attack prevention.
68	TLS_FALLBACK_SCSV extension prevents protocol downgrade attacks. We advise to update your TLS engine to support it.
69	The server is vulnerable to POODLE over TLS.

70	The server's response to invalid TLS packet is not compliant with RFC4346 (section 6.2.3.2) and may be an indicator that the server is vulnerable to POODLE over TLS.
71	The server is not vulnerable to POODLE over TLS.
72	The server is vulnerable to OpenSSL padding-oracle flaw (CVE-2016-2107), consider upgrading OpenSSL.
73	The server is not vulnerable to OpenSSL padding-oracle flaw (CVE-2016-2107).
74	The server may be vulnerable to OpenSSL padding-oracle flaw (CVE-2016-2107), make sure that your OpenSSL version is up to date.
75	The server supports client-initiated secure renegotiation which may be unsafe and allow Denial of Service attacks.
76	The server does not support client-initiated secure renegotiation.
77	The server supports client-initiated insecure renegotiation which is unsafe and may allow Man-In-The-Middle attacks.
78	The server does not support client-initiated insecure renegotiation.
79	The server supports OCSP stapling, which allows better verification of the certificate validation status.
80	The server does not support OCSP stapling. Its support allows better verification of the certificate validation status.
81	The server supports secure server-initiated renegotiation.
82	The server does not support secure server-initiated renegotiation.
83	The server does not provide information if the client should accept secure server-initiated renegotiation requests.
84	TLS compression is supported by the server which may allow CRIME attack. We advise to disable this feature.
85	TLS compression is not supported by the server.
86	The server supports elliptic curves but not the EC_POINT_FORMAT TLS extension.
87	The server supports the EC_POINT_FORMAT TLS extension.
88	The server version of OpenSSL is vulnerable to Heartbleed attack allowing remote compromise of your server. Update your OpenSSL to the latest version urgently!
89	The server version of OpenSSL is not vulnerable to Heartbleed attack.

90	The server is vulnerable to CVE-2014-0224 (OpenSSL CCS flaw), consider upgrading your OpenSSL to the latest version urgently.
91	The server may be vulnerable to CVE-2014-0224 (OpenSSL CCS flaw), make sure that your OpenSSL version is up to date.
92	The server is not vulnerable to CVE-2014-0224 (OpenSSL CCS flaw).
93	Diffie-Hellman parameter size: $value bits
94	The server is not vulnerable to the DROWN attack.
95	The server is vulnerable to the DROWN attack. SSLv2 must be disabled urgently!
96	The server is not vulnerable to POODLE over SSL.
97	The server is vulnerable to POODLE over SSL. SSLv3 should be disabled.
98	The RSA certificate provided has not been signed using the proper algorithm according to HIPAA guidance.
99	The ECDSA certificate provided has not been signed using the proper algorithm according to HIPAA guidance.
100	The following certificates have not been signed using the proper algorithm according to HIPAA guidance: RSA, ECDSA.
101	Some of the certificates provided have not been signed using the proper algorithm according to HIPAA guidance.
102	The server does not support P-256 or P-384 curves which are required by HIPAA guidance.
103	The support of TLSv1.1 is mandatory according to HIPAA guidance.
104	The server supports TLSv1.1 which is mandatory to comply with HIPAA guidance.
105	Intermediate certificate is provided by the server.
106	Intermediate certificate is not provided by the server.
107	An unnecessary root certificate is provided by the server.
108	No unnecessary root certificate sent by the server.
109	The chain provided is in correct order.
110	The chain provided is not in correct order.
111	Server sends useless certificates.

112	Server does not send useless certificates.
113	The server does not support OCSP stapling for its RSA certificate. Its support allows for better verification of the certificate validation status.
114	The server does not support OCSP stapling for its ECDSA certificate. Its support allows for better verification of the certificate validation status.
115	The server does not support OCSP stapling for its RSA and ECDSA certificates. Its support allows for better verification of the certificate validation status.
116	The server does not support OCSP for some of the provided certificates. Its support allows for better verification of the certificate validation status.
117	The server supports OCSP stapling, which allows better verification of the certificate validation status.
118	HTTPS version of the website redirects to HTTP. This is a bad practice since visitors are being redirected from a secure version of the site to an insecure one.

Appendix 3: List of Descriptions values

ID	Value
1	For compatibility reasons, NIST requires to provide X509 certificates in version 3.
2	The trust model of PKI certificates currently resides on the fact they are signed by known Certificate Authority (CA), or a CA that we choose to trust. Self-signed certificates cannot be trusted.
3	PKI certificate contains the server's public key, enabling users to encrypt messages sent to server that on its side will decrypt them using its private key. In case of the loss or compromise of the server's private key, the certificate cannot be trusted anymore and must be revoked and markes as untrusted. However, if a certificate does not contain revocation information, it is impossible to check if it has been revoked or not.
4	Assymetric cryptography uses a public key to encrypt messages, or verify, signatures and a private key to decrypt or sign messages. If the key size is too short there is a risk that an attacker can forge the private key and potentially decrypt all traffic between the client and the server.
5	To be trusted, a certificate is hashed using a specific algorithm in order to get a statistically unique fingerprint to sign it. However, the fingerprint is not mathematically unique and an attacker may forge false certificate with the same hash value to impersonate the server if the hash algorithm used to sign it is too weak.
6	NIST guidelines specify that certificate should not be signed for more than 3 years. In general it is a good practice to renew private key of the server every 1 to 3 years, in order to prevent attacker forging it from the public key.

7	In order to be trusted, a certificate must be signed by a trusted Certificate Authority (CA), the DNS name of the server must match either the Common Name of the certificate or its Subject Alternative Names, it must be valid at the current date (not expired) and it must not have been revoked.
8	Redirectng the users from the HTTP to the HTTPS version is a good practice to enforce secure browsing.
9	When the HTTPS version of a website contains insecure elements, it cannot be totally trusted. Attackers can still intercept these elements which can contain personnal data, or tamper with them to include malicious content in.
10	It is a common best practice to configure TLS servers to have a cipher suite preference, in order to allow enforcing the best compromise between security and performance.
11	Enforcing server preference needs to carefully order supported cipher suites. Preferring a weak cipher suite will cause every browser supporting it to use it instead of a secure one.
12	Perfect-Forward-Secrecy (PFS), based on Diffie-Hellman Ephemeral key exchange, improves global security of TLS. With RSA, an attacker can intercept encrypted communications and record them in order to decrypt them later if he manages to obtain one private key. However, with PFS, it is not possible to use the private key to decrypt messages intercepted in the past.
13	HTTP-Strict-Transport-Security basically allows a server to make users' browsers remember to browse it using HTTPS for a specified duration.
14	Public-Key-Pinning basically allows a server to make users' browsers remember of a list of signatures of certificates to trust for a specified duration. These certificates can either be server certificate or CA certificate.
15	When using CBC cipher suites, TLS imposes padding to be filled with its own length, as the opposite of SSLv3 for which padding could be anything allowing POODLE attack. POODLE over TLS is a vulnerability that appears when a server does not check padding value when using CBC cipher suites.
16	OpenSSL padding-oracle flaw (CVE-2016-2107, CVSSv3 5.9/10) has been introduced because of an incorrect fix for Lucky13 vulnerability and allows attacker revealing encrypted data. It only affects servers supporting hardware acceleration for AES encryption.
17	Client-initiated secure renegotiation (CVE-2011-1473, CVSSv2: 5.0/10) is a vulnerability that may allow Denial of Service (DoS) attacks on servers supporting it.
18	Client-initiated insecure renegotiation (CVE-2009-3555, CVSSv2: 5.8/10) is a vulnerability that may allow attacker to successfully perform Man-in-The-Middle attacks.
19	Heartbleed (CVE-2014-0160, CVSSv2: 5.0/10) is an OpenSSL vulnerability allowing attackers to access random portions of data stored in server's memory. It could include user or admin passwords, private keys and other sensitive data.

20	OpenSSL Change-Cipher-Specs flaw (CVE-2014-0224, CVSSv2: 5.8/10) is a vulnerability affecting OpenSSL and allowing attacker performing Man-in-The-Middle attacks to downgrade cipher suite used between client and server.
21	DROWN vulnerabilty (CVE-2016-0800, CVSSv3 5.9/10) allows attackers to send specially crafted SSLv2 transactions to decrypt TLS connections on servers that use the same RSA private key.
22	POODLE vulnerability (CVE-2014-3566, CVSSv2 4.3/10) is a flaw present in the definition of the SSLv3 protocol. It may allow attackers to decrypt traffic between a browser and a server that use SSLv3 with cipher suties using CBC operation mode.
23	For compatibility reasons, HIPAA guidance requires to provide X509 certificates in version 3.
24	Redirecting the users from HTTPS to HTTP is a major security risk.

Appendix 4: List of Highlights values

ID	Value
1	The server's private key is weak.
2	The server's certificate has been signed using a weak algorithm.
3	The certificate is untrusted.
4	The server's Diffie-Hellman parameter is too small.
5	The server supports elliptic curves that are considered weak.
6	The TLS engine does not support newer version than TLSv1.0 and seems outdated.
7	The server supports protocols that are known as weak.
8	The server supports cipher suites that are not approved by PCI DSS requirements, NIST guidelines and HIPAA guidance.
9	The server supports cipher suites that are not approved by NIST guidelines and HIPAA guidance.
10	The server prefers cipher suites supporting Perfect-Forward-Secrecy.
11	The server provides HTTP Strict Transport Security.
12	The server provides HTTP Public Key Pinning.
13	The server sends an invalid HPKP header.

14	The tested service does not seem to be an HTTPS service.
15	The server is vulnerable to POODLE over TLS, consider upgrading your TLS engine.
16	The server is vulnerable to OpenSSL padding-oracle flaw (CVE-2016-2107), consider upgrading OpenSSL.
17	The server supports client-initiated insecure renegotiation, consider upgrading your TLS engine.
18	The server is vulnerable to Heartbleed, consider upgrading OpenSSL.
19	The server is vulnerable to CVE-2014-0224 (OpenSSL CCS flaw), consider upgrading OpenSSL.
20	The server seems to require certificate-based authentication.
21	The server configuration seems to be good, but is not entirely compliant with NIST guidelines, HIPAA guidance and PCI DSS requirements.
22	The server configuration seems to be good, but is not entirely compliant with NIST guidelines and HIPAA guidance.
23	The server configuration seems to be good, but is not entirely compliant with PCI DSS requirements.
24	The server is vulnerable to the DROWN attack.
25	The server is vulnerable to POODLE over SSL.
26	Server supports HTTPS but it is configured to redirect to HTTP. This is a major security and privacy risk.
27	This server does not have SSL/TLS encryption on port %d. Data exchange with end-users can be intercepted.
28	The server does not tolerate certain TLS versions. This is a sign of faulty implementation.

Appendix 5: List of Titles values

ID	Value
1	X509 CERTIFICATES ARE NOT IN VERSION 3
2	X509 CERTIFICATES ARE IN VERSION 3
3	CERTIFICATES ARE SELF-SIGNED
4	CERTIFICATES DO NOT PROVIDE REVOCATION INFORMATION
5	CERTIFICATES' KEY ARE WEAK

6	SERVER CERTIFICATES ARE SIGNED WITH A WRONG ALGORITHM
7	CERTIFICATES HAVE A WEAK SIGNATURE
8	CERTIFICATES HAVE BEEN SIGNED FOR MORE THAN 3 YEARS
9	CERTIFICATES ARE UNTRUSTED
10	CERTIFICATES ARE TRUSTED
11	CERTIFICATES DO NOT PROVIDE EV
12	CERTIFICATES PROVIDE EV
13	HTTP SITE DOES NOT REDIRECT
14	HTTP SITE DOES REDIRECT
15	MIXED CONTENT
16	DIFFIE-HELLMAN PARAMETER WEAK
17	DIFFIE-HELLMAN PARAMETER SIZE
18	NO SUPPORT FOR COMMON CURVES
19	NO SUPPORT OF TLSV1.1
20	TLSV1.1 SUPPORTED
21	TLSV1.2 SUPPORTED
22	NO SUPPORT OF TLSV1.2
23	SERVER DOES NOT HAVE CIPHER PREFERENCE
24	SERVER HAS CIPHER PREFERENCE
25	SERVER PREFERS WEAK CIPHER SUITES
26	SERVER PREFERS CIPHER SUITES PROVIDING PFS
27	SERVER DOES NOT PREFER CIPHER SUITES PROVIDING PFS
28	SERVER PROVIDES HSTS WITH LONG DURATION
29	SERVER PROVIDES HSTS WITH SHORT DURATION

30	SERVER DOES NOT PROVIDE HSTS
31	SERVER PROVIDES HPKP WITH LONG DURATION
32	SERVER PROVIDES HPKP WITH SHORT DURATION
33	SERVER PROVIDES INVALID HPKP
34	SERVER DOES NOT PROVIDE HPKP
35	TLS_FALLBACK_SCSV
36	POODLE OVER TLS
37	CVE-2016-2107
38	SERVER SUPPORTS CLIENT-INITIATED SECURE RENEGOTIATION
39	SERVER DOES NOT SUPPORT CLIENT-INITIATED SECURE RENEGOTIATION
40	SERVER SUPPORTS CLIENT-INITIATED INSECURE RENEGOTIATION
41	SERVER DOES NOT SUPPORT CLIENT-INITIATED INSECURE RENEGOTIATION
42	SERVER SUPPORTS OCSP STAPLING
43	SERVER DOES NOT SUPPORT OCSP STAPLING
44	SERVER-INITIATED SECURE RENEGOTIATION
45	SERVER SUPPORTS TLS COMPRESSION
46	SERVER DOES NOT SUPPORT TLS COMPRESSION
47	EC_POINT_FORMAT EXTENSION
48	HEARTBLEED
49	CVE-2014-0224
50	DROWN
51	POODLE OVER SSL
52	HTTPS SITE REDIRECTS TO HTTP