Secure Sockets Layer Health Assessment

Mick Pouw, Eric van den Haak

March 14, 2014

Abstract

Tilburg University needs a tool which can assess their SSL services automatically. The research conducted in this paper is performed to determine a health value for an SSL service, which can be implemented in an SSL assessment tool. To determine this value various research has been done.

First, it has been determined what a “good” SSL implementation should look like. A good implementation has for example a valid certificate and has implemented the full chain of trust, but also prohibits the use of SSLv2. After determining what a good SSL service requires, the findings have been compared to common services. Amongst common services are online banking environments, social networks, etc. As it turns out, lots of common services do not implement SSL as it should be.

The findings of this research have been used to develop a series of SSL “tests”. Each of these tests is given a weight based upon the research or it is called *required*. A required test should never fail. If a weighed test fails, the eventual SSL “health” will simply become less. Combining these tests with a proof of concept implementation, based upon existing software, makes it possible to assess an SSL service in a way that covers the demands of Tilburg University. This is giving a single grade to a service which relates to the health status of the service.

Contents

1 Introduction

Secure Socket Layer (SSL), also known as Transport Layer Security (TLS) is a transport layer security protocol[1]. The protocol is based upon the X.509 Public Key Infrastructure Certificate Standard[2], and therefore relies on a chain of trust. Public-key cryptography is used to set up a connection, whereafter a symmetric key is negotiated. Further communication is then encrypted with a symmetric key algorithm. SSL is integrated within most web services and is therefore one of the most important security protocols used today.

Tilburg University also has a lot of SSL services. Maintaining all of these is a lot of work. Therefore, they have requested research for an automated SSL "health" assessment tool, so they can easily check their services. Tools already exist that grade a server's SSL service implementation, but these either cannot be automated or take a long time[21].

This research will be conducted to determine the best SSL practices for a server side environment. Herefore, research will be done to SSL server settings, not to the SSL protocol itself.

Background

The assignment for this research is given by Tilburg University. This research is conducted by Mick Pouw and Eric van den Haak as a part of the curriculum of the System and Network Engineering master program[3] at the University of Amsterdam. The research is supervised by Thijs Kinkhorst, Unix System Administrator, and Teun Nijssen, Security Officer, at Tilburg University.

[1]rfc246 - The Transport Layer Security (TLS) Protocol Version 1.2

[2]rfc6818 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

[3]http://www.os3.nl/

Research questions

The main goal of the research is to create a proof of concept for a tool which is able to test the “health” of an SSL server. In order to make this tool, research has to be done. The following research questions have been posed;

- Main Question: How can we determine SSL "health" of a server side implementation?
- Sub Question: How can we determine a "bad" SSL implementation?
- Sub Question: What mistakes are commonly made by server administrators regarding implementing SSL?
- Sub Question: How can we classify these mistakes?
- Sub Question: How can we develop a tool that automates checking the SSL "health" of a server side implementation?

Related work

Due to the common usage of SSL it is very important that it is as secure as it can be. There are a lot of tools available which can be used to test an SSL service. One of these is SSLlabs[21] which provides a very comprehensive report. These tools form a very good base to test your SSL public service. Another related tool is SSLyze[11]. This is an off-line SSL service scanner which provides an XML document regarding information of the service. This tool does not rate the service but does give a lot of information out of the box which can be very useful to build upon.

2 Approach

In order to give a grade to an SSL server implementation it has to be known what a server implementation should look like. Therefore a literature study has been done. Then common services have been inspected upon their implementation. By combining these results, a weighed value is given to each setting. The final result has been used to create a proof of concept tool that grades an SSL service.

3 Research

The research is subdivided in sections according to our Sub Questions.

3.1 Implementing SSL, the right way

For determining bad SSL implementations, we have researched how SSL should be implemented. This subsection covers the most important aspects of a healthy SSL service, according to our own research.

3.1.1 Certificates

One important aspect of SSL is the certificate. The SSL certificate “guarantees” that the SSL service you connect to is indeed the service you want. An SSL certificate is an implementation of the X.509[5][28] standard. This section describes an SSL certificate briefly and also mentions what is necessary to make it a valid one.

An SSL certificate has the following structure

- Certificate
  - Version
  - Serial Number
  - Issuer
  - Certificate Signature Algorithm
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Subject Public Key Algorithm
    - Subject’s Public Key
  - Extensions
    - Extension items...

- Certificate Signature Algorithm
- Certificate Signature Value

The version field defines the used certificate version. This is mostly Version 3 as of that version extensions are introduced. The Serial Number is a unique number for a certificate signed by a specific Certificate Authority (CA). This is necessary to be able to uniquely distinguish certificate and to revoke them. The issuer is the entity that has signed the certificate. The validity describes between which dates the certificate is valid. The subject describes the service for which the certificate is issued. The subjects public key info contains information about the public key of the subject. The extensions contain optional information, either issued by the certificate authority or inserted by the requester. With these, the CA can force explicit use of the certificate. The Certificate Signature Algorithm describes which algorithm was used to generate the signature. The Certificate Signature Value is the signature itself.

For trusting an SSL service, all attributes of its certificate should be correct.

Trusting the issuer
The certificate is signed by an issuer so the client knows for sure the issuer has approved the certificate. It is however necessary to trust the issuer as well, otherwise there is no point in signing. Therefore a client has to check whether the issuer is trusted. This checking works like a chain, and is called a chain of trust. The issuer that signed the subject, is a subject itself of its own issuer. This goes on until the root certificate is reached, which is signed by itself. These root certificates have to be trusted by the client. If so, the client can then trust the certificate of the service because it trusts the root certificate.

To be able to do this, the SSL service should provide the certificates in the chain of trust as well, along its own certificate. If not, the client does not know where to look. This is called certificate chaining.

Validity check
The certificate might be trusted, but it also has to be valid. This is checked by comparing the current date to the values in the Validity field. If the current date is between the Not Before and Not After field, the certificate is valid. This might sound silly, but if an administrator forgets to update the certificates on a service when they are expired, the service will not be trusted.

Subject check
Checking whether the subject in the certificate matches the subject of the service you are using is another very important check. Within HTTPS the subject should match the host name of the service. If this would not be tested, it would be very easy to spoof a service with a “valid” certificate from another service (having a

different subject).

Hashing Algorithm check
Some hashing algorithms should be avoided at all times. These algorithms are MD5 and less strong algorithms. Certificate Authorities these days won't sign MD5 hashes anymore, however it is possible that some long-term certificates still have the MD5 algorithm. These certificates are vulnerable to collision attacks[13]. MD5 hashes might even be presented in the certificate chain, endangering many certificates at once.

Random number weakness
Generating an SSL certificate requires a good random number. Having a weak random number generator can mean that there is someone out there who can predict the outcome of it and is thus able to create an identical key pair. Research has been done on all public SSL and SSH services to find weak public keys which should thus be avoided[14]. While they do not give away all known vulnerable public keys, they do provide a check feature for public accessible SSL/SSH services.

Revocation
It can occur that a certificate has to be revoked before the Not After date. For example if a private key has been compromised. For this, the Certificate Revocation List (CRL) mechanism has been introduced. This list lists the serial numbers of revoked certificates of a certain CA. This means that when full trust of an SSL service is necessary, the CRL should be consulted. CA's publish their lists publicly. A newer form of revocation checking is the Online Certificate Status Protocol(OCSP)[16]. This is a protocol which can be use to request the status of a single certificate, and thus omit downloading a CRL. The objective remains that an SSL service uses a non-revoked certificate.

3.1.2 Protocol

The SSL protocol is divided into several versions. The oldest SSL version that is still required for some browsers, is SSLv3. The list of the available and used SSL protocols can be found below:

- SSLv2
- SSLv3
- TLS1.0[8]
- TLS1.1[9]
- TLS1.2[10]

SSLv2 / SSLv3 protocol
SSLv2 is considered unsafe and should no longer be used as a protocol. SSLv3 is a significant improvement to the SSLv2 protocol and adds more security and resistance to known attacks on SSLv2 to the protocol[27]. Since SSLv3 is required for a safe connection for Internet Explorer 6, this could still be enabled to not block any legitimate users.

Since the SSLv2 protocol is very insecure[27], we are going to rule out SSLv2 as a potential SSL protocol. However, there are also vulnerabillties in the SSLv3 protocol. There are known plaintext attacks on the SSLv3 protocol[7] for example. Another weakness is how the key is derived. This is done based on a MD5 hashing. MD5 hashing have collissions and is conciderete broken. This might be the main reason to disable it.

Since SSLv3 implementation is only used to support older browsers e.g. Internet Explorer 6[20]. The use of Internet Explorer 6, embedded in Windows XP, should end when XP is no longer supported[18]. Therefore the lowest possible SSL protocol should be raised to TLS1.0.

TLS protocol
TLSv1.0 is the direct successor of SSLv3. A lot of the specification that is available for TLSv1.0 is also in SSLv3 with some changes in words[27]. TLSv1.0 however adds more ciphers to the cipher suite. They are listed in Appendix A. The TLSv1.1 protocol tries to improve on the security vulnerabillities that SSLv3 and TLSv1.0 have by fixing the possible attacks on TLSv1.0. The known plaintext attack, for example, is mitigated by the use of explicit initialization vectors[9].

TLSv1.2 takes it one step further. There are more differences towards TLSv1.1. One of the primary changes is that the MD5/SHA1 hash is replaced by default to SHA256[10]. Because of the new hashing algorithms, SSL is no longer vulnerable to MD5 collisions [13]. Also the AES encryption has been added to the supported cipher suites. This provides a more secure environment for the SSL connection.

Compression
The SSL protocol supports compression. If the default SSL compression is used, a vulnerability is created. This vulnerability is called the Compression Ratio Info-leak Made Easy (CRIME) attack. This is a clever trick to figure out actual values within the encrypted message. An easy to follow explanation can be found online[24]. It is thus recommended to turn off SSL compression. Apart from the possibility of having a small amount of extra bytes, turning it off should have no disadvantages.

3.1.3 Cipher suites

An SSL connection needs to be able to communicate securely. This is done by choosing a cipher suite. This cipher suite is then used to secure the connection.

The client of an SSL based connection transmits its known cipher suites towards the server. The server then selects the cipher suite it wants to use and sends this back to the client. Once the cipher suite is selected, a secure connection will be established. This connection uses the chosen cipher. Thus client and server both need a common known cipher suite in order to set up a secured connection. A server administrator determines the available ciphers of an SSL service. For the client this is usually the program which they use to connect to the SSL service. In case of HTTPS, the browser vendor adds the supported cipher suites which the client can (optionally) disable or enable.

A full collection of the cipher suites that OpenSSL[4] supports can be found in the appendix 'A' and can also be found in the documentation of OpenSSL[19].

Disabling RC4
Microsoft recommends to disable the use of the RC4 cipher[1]. This is based on recent research[2] towards the cipher suite. While exploiting RC4 is not easy, not using it at all seems to be good choice.

Using strong ciphers
As we now know some background of the protocol and the cipher suites, we can determine what ciphers not to use, and thus also seek what ciphers to use. Since we know that MD5 is broken, we will not use the MD5 hashing algorithm. Also NULL encryption might not be such a great idea, since the traffic will not be encrypted at all.

Also note that RC4, as mentioned in the previous section, is not easily exploited, but should be considered not secure. Disabling the RC4 cipher suites is recommended.

A good level of key material must be used to have a secure connection[23]. Common sense advice is to set the supported bit size for ciphers to a minimum of 128 bits. This eliminates weak ciphers but also the entire DES[5] and 3DES[6] cipher suites.

[4]OpenSSL is one of the most used SSL implementation available.
[5]56 bit keysize and some parity.
[6]Effectively having only 112 bit (2x56) keysize.

3.1.4 Popular known vulnerabilities

Some vulnerabilities were mentioned by Tilburg University.

BEAST Attack
The BEAST attack is a client side attack which is based upon predicting the initialisation vector used in CBC mode encryption by performing a Man in the Middle attack[12]. The BEAST attack is a client side attack and is fixed by most up-to-date clients now[22]. One way to prevent this attack via the server side is to allow only non-CBC mode cipher suites. This would however result into using RC4, which is not desired either. Because most clients should be fixed by now we will ignore the BEAST attack.

The Beast attack relies on the ability of predicting the Initialisation Vector(IV) which is used in CBC mode cipher suites. The IV “guarantees” that a random pattern in cipher block chaining occurs, regarding the input. Thus having the same raw input twice will result in two different encrypted blocks. A solution to fix a predictable IV is to send the first block of bytes with having only the first byte containing application information[7]. The followed blocks contain the rest of information.

Lucky 13
Another mentioned vulnerability is the Lucky 13 attack[3]. This attack exploits a server error response to an SSL message. Depending on the response time, the attacker can determine the type of error that has occurred and use this to make the server decrypt a message without having the encrypt key. According to the creators of the Lucky 13 attack, updating to the latest versions of SSL implemented software will fix the vulnerabilities[25].

[7]Solution is proposed by Xuelei Fan (https://bugzilla.mozilla.org/show_bug.cgi?id=665814#c59)

3.2 Common mistakes

The second step is to look for common mistakes. For this, we will determine our scope and the services we want to test. We also have to define “common” mistakes.

3.2.1 Scope

Services

To determine the services we are going to test, we have to know what services are available and on which port they run. The Internet Assigned Number Authority provides a list of all registered SSL services along with their port numbers[6]. We have picked the following services as we thought they are the most commonly used. This selection is based upon our own experience.

Protocol	Port	Description
https	443	http over SSL/TLS
imaps	993	imap over SSL/TLS
pop3s	995	pop3 over SSL/TLS

Table 1: Common mistakes service scope

Hosts

We wanted to have a wide range of different SSL hosts and services to research for their certificate and settings. Because we are on a limited time we don’t want to have too many hosts. Our examined hosts can be found in the appendices (Appendix B).

Definition

Based upon our first research (Section 3.1), we can now determine what is a good SSL implementation. We can therefore also determine when an SSL service contains a mistake. The following table contains our demands for a “healthy” SSL service.

Signature hash algorithm	The hashing algorithm used for signing the certificate should be strong. (No MD5)
Certificate (chain) trusted	The certificate chain has to be implemented in the right way and has to be trusted.
Certificate is valid	The certificate should not be expired, neither should it be valid from the future.
Subject name matches	The subject of the certificate should match the hostname of the service.
No Debian weak keys	The certificate should have strong a key.
Compression disabled	SSL Compression should be disabled (Crime vulnerability).
Key length at least 128bits	At least 128bit ciphersuites should be used.
Cipher suites do not contain MD5	MD5 should not be used anymore.
Cipher suites do not contain RC4	RC4 should not be used anymore.
Perfect forward secrecy available	Perfect Forward Secrecy(PFS) should be enabled.
SSLv2 Disabled	SSLv2 should be disabled.
SSLv3 Disabled	SSLv3 should be disabled.
TLSv1.0 Enabled	TLSv1.0 should be enabled.
TLSv1.1 Enabled	TLSv1.1 should be enabled.
TLSv1.2 Enabled	TLSv1.2 should be enabled.

Table 2: Determined demands of an SSL service

3.2.2 Results

As mentioned above, we had several tests to check the certificate validity. The percentage of passing these tests can be read in table 3. The enabling of SSLv3 resulted in the fact that most clients did not pass our SSLv3 test, since it was enabled. Most server administrators keep this enabled because there are still backwards compatibillity issues with Miscrosoft Windows XP and Internet Explorer 6.

We however recommend to turn of SSLv3 because the protocol support for the better TLSv1.0 is already in all the browsers available[8]. Also Windows XP is no longer supported from the 8th of April 2014.

Another test that has a low passing rate is the usage of RC4. This encryption method is weaker then expected and should thus be turned off. However the majority of the web servers still use RC4.

Testname	Percentage passed
Signature hash algorithm	100%
Certificate (chain) trusted	100%
Certificate is valid	100%
No Debian weak keys	100%
Subject name matches	91%
Compression disabled	100%
Cipher suites do not contain MD5	57%
Perfect forward secrecy available	46%
Cipher suites do not contain RC4	17%
Key length at least 128bits	89%
SSLv2 disabled	94%
SSLv3 disabled	3%
TLSv1.0 enabled	97%
TLSv1.1 enabled	63%
TLSv1.2 enabled	63%

Table 3: Percentage of test passing for hosts

[8]Except Internet Explorer 6

3.3 Classifying mistakes

Based upon our research we have developed our tests. We have included tests that determine whether the certificates are implemented securely, but also tests that conclude whether the certificate is implemented correctly. For example whether the chaining is correct and if the date is still valid.

The tests we have done are explained in section 3.3.1. We need to give a value to a test in order to weigh the test. The final grade will be determined by adding the weight of all the tests and then dividing by the total possible weight. This will result in a percentage that can be interpreted as the health of a SSL based service.

However, there is one exception to this calculations. Whenever a so called show-stopper[9] test does not pass, the final percentage will be set to 0%. In practice, a service using a certificate which is does not pass all of the show-stopper tests should not be trusted by any client. Most browsers will display a warning.

[9]A test that must be passed to get a correct implementation, for example the validity of the SSL certificate

3.3.1 Classification

To be able to set up the tool, we have designed an example test (table 4).

Example Test

Name	Example
Proposition	Requirement in order to pass the test
Weight	50
Required	No

Table 4: Test example

A test can either fail or pass. If a tests fails which is required (and thus vital for the SSL service), the service fails and the health is then 0. For the other tests we have determined that we will divide the sum of the total weight of the passed tests divided by the sum of the total weight of all executed tests. This results in a percentage of the passed tests, with the ability to make more weighted tests having more influence on the final grade. If all the tests are positive the health of the tested service is 100%.

Formulas

$$\{requiredtests\} \subset \{passedtests\} \tag{1}$$

Equation 1: The set of all required tests has to be a subset of all passed tests.

$$100 * \frac{\sum\limits_{i=1}^{N} p_i}{\sum\limits_{j=1}^{M} t_j} \tag{2}$$

Equation 2: Where p is a set of all weights of the passed tests and t is a set of all weights of all performed tests.

3.3.2 Implementation

In this section we describe what weight and the required flag we have given to each test and with what motivation. The weight values have been chosen so that $0 <= weight <= 100$ having 0 is no weight at all and 100 is the biggest weight. To make decisions more easily, we have made up the following table to make rating ourselves relatively easy.

Rating legend

Rating	Urgency to repair
0-19	Very low
20-39	Low
40-59	Medium
60-79	High
80-100	Very high

Hashing Signature

Proposition	The hashing algorithm used to sign the certificate is not MD5
Weight	80
Required	No

Having an MD5 signed certificate is considered bad because they are vulnerable to collision attacks[13]. In our findings we have not found any service using an SSL certificate signed with an MD5 hash. Because this can result a real issue we have determined this as a high risk.

Trust Test

Proposition	The certificate and chain has to be trusted
Weight	0
Required	Yes

Having an untrusted certificate (chain) will cause the service to fail. Therefore we have determined that this test is required to pass. This test will validate the certificate of the service by the certificate chain supplied by the service.

Validity Test

Proposition	The certificate has to be valid. (Date check)
Weight	0
Required	Yes

Having an expired (or future certificate) will cause the service to fail. Therefore we have determined that this test is required to pass.

Weak keys

Proposition	The certificate does not have a weak key
Weight	100
Required	No

Having a weak key means that finding the private key of the certificate is very easy. Therefore we have determined that it is very important to fix this as soon as possible and gave it the highest possible weight.

Subject check

Proposition	The certificate’s subject matches the host name of the service
Weight	0
Required	Yes

Having an SSL service with a certificate of which the subject is not matching the host name will cause the service to fail. Therefore we have determined that this test is required to pass.

Compression check

Proposition	SSL Compression is disabled
Weight	50
Required	No

Having SSL compression enabled will enable the possibility for a CRIME attack[24]. Disabling SSL compression is therefore desirable. We have determined the weight at medium because most tested services had already disabled SSL compression.

MD5 Usage

Proposition	Cipher suites using MD5 are not used.
Weight	50
Required	No

As stated before, MD5 is vulnerable to collision attacks and cipher-suites containing it should therefore be disabled. Newer clients probably won't propose these hashing algorithms towards a SSL based service. We therefore have decided that this is a medium weighted test.

RC4 Usage

Proposition	Cipher suites using RC4 are not used.
Weight	80
Required	No

It is recommended to turn off RC4[2]. Because clients still propose this cipher, we considered turning RC4 off as a high demand.

PFS Usage

Proposition	Perfect forward secrecy cipher suites are enabled.
Weight	50
Required	No

Having perfect forward secrecy is higly recommended by us. However, only a small amount of tested hosts has this feature enabled. It is also not necessary for a service to implement this to be working in good order. Because we do want services to adopt this feature, we have put this test on medium.

Weak Chipers

Proposition	Cipher suites don't use less than 128bit keys.
Weight	80
Required	No

It is desirable to have strong keys because this reduces the possibility of a successful brute-force attack. A few of our test subjects allowed for encryption with a 40bit key. This is very weak and therefore considered dangerous. Therefore we have decided to put this on high weight.

SSLv2

Proposition	SSLv2 is disabled.
Weight	100
Required	No

SSLv2 is considered broken. Therefore we have determined that disabling SSLv2 should be done as soon as possible, resulting in a very high issue.

SSLv3

Proposition	SSLv3 is disabled.
Weight	30
Required	No

SSLv3 has some known vulnerabilities. It is however still used because of backwards compatibility to older clients. For example Microsoft Internet Explorer on Windows XP. While we would really like to see that SSLv3 is disabled, lot's of companies can not do so because their customers will fail to connect to their services. For this reason we have determined this as a relatively low issue.

TLSv1.0

Proposition	TLSv1.0 is enabled.
Weight	75
Required	No

TLSv1.0 is enabled on every host we have tested. Most clients support TLSv1.0 and we have therefore determined that enabling this protocol is desired. Turning this version off and turn newer versions on is theoretically better but the world is not yet ready for this, as lots of clients do not support newer versions yet. We have therefore determined that turning this version on is of a high demand.

TLSv1.1

Proposition	TLSv1.1 is enabled.
Weight	100
Required	No

TLSv1.1 is more secure than TLSv1.0. Turning it on does not break TLSv1.0 so we

do recommend to turn this version on. We have therefore set this test as a very high demand.

TLSv1.2

Proposition	TLSv1.2 is enabled.
Weight	100
Required	No

TLSv1.2 is the latest stable TLS version. We therefore also recommend to turn this on as it won’t break support for older versions of TLS. Therefore TLSv1.2 also has a high weight. One drawback is that not all server software supports this version yet.

3.4 Implement findings

Based upon our findings we are able to create an health assessment tool. The tool is going to be a proof of concept and is able to run on a Linux pc and on Mac OS X. The tool will be using SSLyze[11][10] as an underlying basis. SSLyze generates an XML document containing information regarding the SSL service and it's implementation. Therefore we have found SSLyze a great basis to build our tool upon. To implement the Debian weak key vulnerability check we will use an online service[15]. The certificate chain itself will be tested via openSSL.

The proof of concept tool will be written in Python and has to be modular. This means that the tool has to be written in such a way that adding new tests can be done without changing the application itself and is very easy. The same concept will be used to generate output. Output will be handled by so called printers, which can be added and modified without making changes to the application itself. With this feature it is thus possible to create an output like XML or JSON, and integrate the results in existing monitoring tools. Various terminal output printers for easy and fast reading will be available as well.

The application is divided into several classes. We have created a class diagram to give an overview of the application. This class diagram can be found in figure 1.

To implement a new test, all one has to do is inherit the Test class and put the file containing this test into the tests folder. The application will then dynamically load the test into the application. Adding a different printer works the same, this is done by adding a class which inherits the Printer class to the printer folder. To select a printer, you can pass the selected printer as an argument in the command line interface.

3.4.1 The running application

When the application starts it tries to scan the provided hosts and corresponding port numbers (host:port). These port numbers are not mandatory and the tool will use 443[11] if not provided. For example, services like IMAPS can be scanned by appending :993 to the host name.

For each of the provided hosts, a service object will be created. The application will try to scan the services by using the SSLyze application as a library. The XML

[10]A tool used to gather information about an SSL service (`https://github.com/iSECPartners/SSLyze`)

[11]HTTPS socket

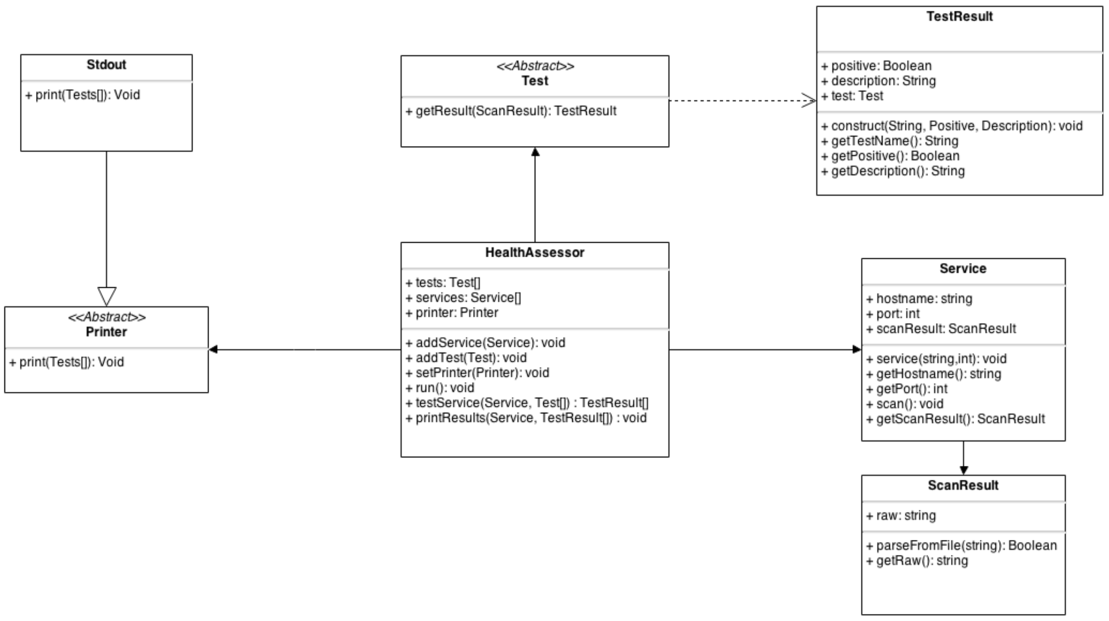


Figure 1: Class diagram

that is provided by the SSLyze application will be interpreted by an instance of the Scanresult class. Once the scan is completed, we will move towards the tests.

For each service, the corresponding scanresult object will be passed towards the available tests. The test will then determine whether the test passes or fails and returns a testresult object. For each service, an array with testresult objects will be passed on to the printer.

The printer then formats the results of the tests in a desired format. Once every result is printed, the application will terminate and the user can look at the results.

3.4.2 Testing Tilburg University IP space

Now that we have a working SSL health assessment tool, we have ran it against the entire Tilburg University IP space. To keep the scope within limits, we have decided to test all publicly accessible HTTPS servers (accessible from outside of the Tilburg University network). With nmap[17] we have first scanned the entire IP space for hosts with an open 443 port.

```
$ nmap 137.56.0.0/16 -p 443 --open --verbose
```

This resulted in 176 hosts with an open 443 port. To determine which port was actually listening we have used Curl[4].

```
 $ curl https://hostname:443 --connect-timeout 10; echo $?
```

For each host, a Curl response code was given. Response code 28 means a connection time-out, so we have omitted the hosts from our tests having this code. This resulted in a remainder of 159 hosts. Because among these hosts are also servers which can have a different host name than the SSL service it services, we have modified our subject test. The subject test is still executed, but does not affect the outcome of the test. Normally, having a wrong subject would result in a broken connection. The result of our health assessment tool can be found in Appendix D. It took our tool about 30 minutes to test all the services. Because not all the hosts responded immediately it took the test quite some time to run.

Results

Among our results is a wide range of different outcomes, varying between 0 and 97 percent. Remarkably, 97 is the highest score and the reason for this is the fact that SSLv3 is enabled on all hosts. This is due to the fact that the University has to have a lot of backwards compatibility due to old clients and thus can not omit this version yet. Apart from this, a score of 97 is in our opinion very healthy. The lowest score in the list is 0. This means that normal clients would break connection upon connecting to these hosts. What concerns us most are the hosts with a score around and below 40. Most of these hosts have SSLv2 enabled which is really insecure by now.

3.4.3 Testing IMAPS vs HTTPS

The proof of concept tool we created also has the possibillity to scan other services like IMAPS[12]. For this reason we also tried to scan IMAPS services. IMAPS would make a great test environment to check whether HTTPS was implemented better then IMAPS. There is however one problem with IMAPS servers, there are way more HTTPS servers available than IMAPS servers. For this reason there are only a few selected IMAPS servers in appendix C.

Results

The selected IMAPS services did not score well. Most of them scored 60%. This is mostly because of the enabling of SSLv3 and keeping TLSv1.1 and TLSv1.2 off. Also the useage of MD5 and RC4 ciphersuites have lead to these low values. The recommended settings are thus not followed in the IMAPS servers.

[12]Imap is a protocol to retreive mail

3.4.4 Testing SURFconext Identity Providers

SURFcontext is the federated authentication platform that is provided by SURFnet, the national research and education network in The Netherlands [13]. Almost all of the institutions for higher education and research in The Netherlands use this platform to use a single place of logging in. The organizations use their Identity Provider (IdP) to provide an interface to log into. Because this is the place where account usernames and passwords are exchanged, it is very important that the SSL health is of good quality. We scanned all of the SURFconext IdP's to test their services. We did this through a list that we received from the SURFnet organization. The result of this scan can be found in appendix E.

Results

In these tests, we found out that several services running the HTTPS server were insecure. In table 5 we show that the most of the hosts are considerate insecure by our tool. The outliers however score only 15%. This is mostly because SSLv2 and SSLv3 are enabled but all the TLS protocols are disabled. This results in a low grade. The SSLv2 must definetly be disabled to enhance the security of these domains.

Score	Number of hosts
$<$ 40%	5
40-50%	8
50-60%	82
60-70%	9
70-80 %	13
$>$ 80 %	20

Table 5: SURFconext summary table

The cipher suites are also not very well choosen for these hosts, almost all of them still use MD5 and RC4. To improve overal security, it is better to disable these aswell. Last but not least, the TSLv1.1 and TLSv1.2 must be enabled to improve the security to current standards.

[13]http://www.surf.nl/diensten-en-producten/surfconext/index.html

Responsible disclosure

Although our scans are based on publically available data and services reachable from the internet, we've practiced so called 'responsible disclosure' where we announced the results of the tests ahead of publication of this report to the respective service administrators to give them a chance to address any issues. The results of the scan on the Tilburg University network have been coordinated with the university's incident response team UvT-CERT, and the SURFconext administrators have been informed about the scores of their IdP's.

4 Conclusion

We set out to find a way to test SSL server side implementations for their health in a way that it can easily be implemented in software. Although software that tests the health of an SSL service already exist, this research shows a different way in testing. The main difference between existing software and the developed method is the way of grading an SSL service and the ability to do this in bulk, making it easy to monitor a whole server cluster for its status. The software has been tested in the large environments and had positive results. The following sections describe the results of our research.

How can we determine a bad SSL implementation?
Based upon literature studies on how SSL services should be implemented, the criteria for good SSL implementations are determined. For example, support for TLSv1.2 is mandatory and earlier versions are less secure or support less strong cipher suites. However, for backward compatibility it is just not possible to enable TLSv1.2 only. Therefore is concluded that TLSv1.0 and 1.1 should still be enabled. SSLv3 is still used widely but should be disabled. SSLv2 is really broken so this should be disabled at all times.

What mistakes are commonly made by server administrators regarding implementing SSL?
The performed research into real world services indicate that the most common mistakes are that 97% still uses SSLv3 and 37% of the services have not enabled TLSv1.1 or TLSv1.2. Another recurring theme is the usage of RC4 which is proven vulnerable. A lot of common SSL services are not implemented the correct way.

How can we classify these mistakes?
In order to classify a server setting, we have given each setting a certain weight based upon the level of vulnerability and put them into a test. For example, this results in granting a very high weight for enabling SSLv2, which should definitely not be done. The health value of an SSL service is based upon the classification of the tests and is represented by a formula that uses the weight of all passed tests combined with the total weight of all tests.

5 Future work

While doing this research we discovered several interesting fields that have yet to be explored. Some of these fields are listed below.

5.1 Starttls

Starttls is one of the fields that still need to be examined. This is basically starting an unsecure connection and then upgrade the connection to use SSL. This has not been researched in this paper and could be researched in the future.

5.2 HTTPS protocol

There are several aspects of the HTTPS protocol specifically. This is for example the headers that can be sent with an HTTPS page. The research done in this paper was about the general SSL TLS connection, therefore going deeper in one of its uses was beyond the scope of this project.

An example of such a header is the https strict header. Whenever a browser receives this header, the browser knows for future visits to go to the HTTPS version of the web page. When a man in the middle attack is performed in such a way that it downgrades the HTTPS to plain HTTP, the browser knows that it must not serve the web page to the end user.

5.3 Proof of Concept

The proof of concept does not validate SSL services that request an (optional) client certificate in a right way. This is however desirable since this is less common but is a valid feature of an SSL service.

It also does not handle server name indication, which is also very desirable if you want to test a host that serves multiple domains.

As last, the tool is based upon SSLyze[11], which is not providing all necessary data for all tests. These tests are implemented in a different way using either openSSL or an online service[15] for detecting Debian weak keys. As of Python 3.4(which is currently in beta), a lot of SSL features have been added. Amongst these features is for example a test for the chain of trust. Rewriting the tool as a standalone tool might be a nice next step.

References

[1] Ie11 automatically makes over 40sites continue to work, 2013. http://blogs.msdn.com/b/ie/archive/2013/11/12/ie11-automatically-makes-over-40-of-the-web-more-secure-while-making-sure-sites-continue-to-work.aspx.

[2] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C.N. Schuldt. On the security of rc4 in tls and wpa. 2013.

[3] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. February 2013.

[4] Various Authors. Curl, 2014. http://curl.haxx.se/.

[5] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008. Updated by RFC 6818.

[6] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335 (Best Current Practice), August 2011. http://www.ietf.org/rfc/rfc6335.txt.

[7] Bruce Schneier David Wagner. Analysis of the ssl 3.0 protocol, April 1996. https://www.schneier.com/paper-ssl-revised.pdf.

[8] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, Internet Engineering Task Force, January 1999.

[9] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746, 6176.

[10] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[11] Alban Diquet and Aaron Grattafiori. Sslyze, 2014. https://code.google.com/p/sslyze/.

[12] Thai Duong. Beast, September 2011. http://vnhacker.blogspot.nl/2011/09/beast.html.

[13] Dario Forte. The death of md5. *Network Security*, 2009.

[14] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium*, August 2012.

[15] Nadia Heninger and J. Alex Halderman. Fastgcd, 2012. `https://factorable.net/keycheck.html`.

[16] R. Huang, Q. Wu, H. Asaeda, and G. Zorn. RTP Control Protocol (RTCP) Extended Report (XR) Block for MPEG-2 Transport Stream (TS) Program Specific Information (PSI) Independent Decodability Statistics Metrics Reporting. RFC 6990 (Proposed Standard), August 2013.

[17] Gordon Lyon. Nmap, 2014. `http://nmap.org/`.

[18] Microsoft. Microsoft support lifecycle, January 2014. `http://support.microsoft.com/lifecycle/?ln=en-gb&c2=1173`.

[19] Openssl. Openssl cipher list, 01 2014. `http://www.openssl.org/docs/apps/ciphers.html`.

[20] Qualys. Internet explorer 6 support configuration, January 2014. `https://www.ssllabs.com/ssltest/viewClient.html?name=IE&version=6&platform=XP`.

[21] Qualys. Ssl server test tool, 2014. `https://www.ssllabs.com/ssltest/index.html`.

[22] Ivan Ristic. Is beast still a threat?, 2013. `https://community.qualys.com/blogs/securitylabs/2013/09/10/is-beast-still-a-threat` Qualys SSL labs.

[23] Ivan Ristić. Ssl/tls deployment best practices, September 2013. `https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices_1.3.pdf` Qualys SSL labs.

[24] Tom Ritter. Details on the "crime" attack, 2012. `https://isecpartners.com/blog/2012/september/details-on-the-crime-attack.aspx`.

[25] Pratik Guha Sarkar and Shawn Fitzgerald. Lucky 13, 2013. `http://www.isg.rhul.ac.uk/tls/Lucky13.html`.

[26] Alex Wawro. Researchers crack web encryption. *PC World*, 2011.

[27] Loren Weith. Differences between sslv2, sslv3, and tls, 2006. `http://www.yaksman.org/~lweith/ssl.pdf`.

[28] P. Yee. Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 6818 (Proposed Standard), January 2013.

Appendices

A SSL Ciphers

A.1 SSLv3

Cipher name openssl	Implementation
SSL_RSA_WITH_NULL_MD5	NULL-MD5
SSL_RSA_WITH_NULL_SHA	NULL-SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
SSL_RSA_WITH_RC4_128_MD5	RC4-MD5
SSL_RSA_WITH_RC4_128_SHA	RC4-SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	EXP-RC2-CBC-MD5
SSL_RSA_WITH_IDEA_CBC_SHA	IDEA-CBC-SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DES-CBC-SHA
SSL_RSA_WITH_DES_CBC_SHA	DES-CBC-SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-DH-DSS-DES-CBC-SHA
SSL_DH_DSS_WITH_DES_CBC_SHA	DH-DSS-DES-CBC-SHA
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH-DSS-DES-CBC3-SHA
SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DH-RSA-DES-CBC-SHA
SSL_DH_RSA_WITH_DES_CBC_SHA	DH-RSA-DES-CBC-SHA
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH-RSA-DES-CBC3-SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-DHE-DSS-DES-CBC-SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA	DHE-DSS-CBC-SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE-DSS-DES-CBC3-SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DHE-RSA-DES-CBC-SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA	DHE-RSA-DES-CBC-SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE-RSA-DES-CBC3-SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	EXP-ADH-RC4-MD5
SSL_DH_anon_WITH_RC4_128_MD5	ADH-RC4-MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	EXP-ADH-DES-CBC-SHA
SSL_DH_anon_WITH_DES_CBC_SHA	ADH-DES-CBC-SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	ADH-DES-CBC3-SHA
SSL_FORTEZZA_KEA_WITH_NULL_SHA	Not implemented.
SSL_FORTEZZA_KEA_WITH_FORTEZZA_CBC_SHA	Not implemented.
SSL_FORTEZZA_KEA_WITH_RC4_128_SHA	Not implemented.

A.2 TLSv1.0/TLSv1.1

Cipher name openssl	Implementation
TLS_RSA_WITH_NULL_MD5	NULL-MD5
TLS_RSA_WITH_NULL_SHA	NULL-SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
TLS_RSA_WITH_RC4_128_MD5	RC4-MD5
TLS_RSA_WITH_RC4_128_SHA	RC4-SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	EXP-RC2-CBC-MD5
TLS_RSA_WITH_IDEA_CBC_SHA	IDEA-CBC-SHA
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DES-CBC-SHA
TLS_RSA_WITH_DES_CBC_SHA	DES-CBC-SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	Not implemented.
TLS_DH_DSS_WITH_DES_CBC_SHA	Not implemented.
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	Not implemented.
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_DES_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	Not implemented.
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-DHE-DSS-DES-CBC-SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE-DSS-CBC-SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE-DSS-DES-CBC3-SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DHE-RSA-DES-CBC-SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE-RSA-DES-CBC-SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE-RSA-DES-CBC3-SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5	EXP-ADH-RC4-MD5
TLS_DH_anon_WITH_RC4_128_MD5	ADH-RC4-MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	EXP-ADH-DES-CBC-SHA
TLS_DH_anon_WITH_DES_CBC_SHA	ADH-DES-CBC-SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	ADH-DES-CBC3-SHA

A.3 TLSv1.2

Cipher name openssl	Implementation
TLS_RSA_WITH_NULL_SHA256	NULL-SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256	AES128-SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256	AES256-SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256	AES128-GCM-SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384	AES256-GCM-SHA384
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	DH-RSA-AES128-SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	DH-RSA-AES256-SHA256
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	DH-RSA-AES128-GCM-SHA256
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	DH-RSA-AES256-GCM-SHA384
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	DH-DSS-AES128-SHA256
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	DH-DSS-AES256-SHA256
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	DH-DSS-AES128-GCM-SHA256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	DH-DSS-AES256-GCM-SHA384
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	DHE-RSA-AES128-SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	DHE-RSA-AES256-SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	DHE-RSA-AES128-GCM-SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	DHE-RSA-AES256-GCM-SHA384
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	DHE-DSS-AES128-SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	DHE-DSS-AES256-SHA256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	DHE-DSS-AES128-GCM-SHA256
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	DHE-DSS-AES256-GCM-SHA384
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	ECDH-RSA-AES128-SHA256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	ECDH-RSA-AES256-SHA384
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	ECDH-RSA-AES128-GCM-SHA256
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	ECDH-RSA-AES256-GCM-SHA384
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	ECDH-ECDSA-AES128-SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	ECDH-ECDSA-AES256-SHA384
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	ECDH-ECDSA-AES128-GCM-SHA256
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	ECDH-ECDSA-AES256-GCM-SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE-RSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	ECDHE-RSA-AES256-SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE-ECDSA-AES128-SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	ECDHE-ECDSA-AES256-SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE-ECDSA-AES128-GCM-SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE-ECDSA-AES256-GCM-SHA384
TLS_DH_anon_WITH_AES_128_CBC_SHA256	ADH-AES128-SHA256
TLS_DH_anon_WITH_AES_256_CBC_SHA256	ADH-AES256-SHA256
TLS_DH_anon_WITH_AES_128_GCM_SHA256	ADH-AES128-GCM-SHA256
TLS_DH_anon_WITH_AES_256_GCM_SHA384	ADH-AES256-GCM-SHA384

B Examined Hosts

The hosts and their corresponding services we used in our research to look for common mistakes are listed below.

Tag	Description	Weight	Required
A	Signature hash algorithm	80	No
B	Certificate (chain) trusted	0	Yes
C	Certificate is valid	0	Yes
D	No Debian weak keys	100	No
E	Subject name matches	0	Yes
F	Compression disabled	50	No
G	Cipher suites do not contain MD5	50	No
H	Perfect forward secrecy available	50	No
I	Cipher suites do not contain RC4	80	No
J	Key length at least 128bits	80	No
K	SSLv2 disabled	100	No
L	SSLv3 disabled	30	No
M	TLSv1.0 enabled	75	No
N	TLSv1.1 enabled	100	No
O	TLSv1.2 enabled	100	No

Table 6: Common mistakes service legend

Host	Port	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
www.os3.nl	443	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
www.hva.nl	443	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
www.uvt.nl	443	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
www.hostnet.nl	443	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
www.transip.nl	443	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
www.versio.nl	443	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
www.pcextreme.nl	443	√	√	√	√	√	√	×	×	×	×	×	×	√	×	×
www.mijndomein.nl	443	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
www.ing.nl	443	√	√	√	√	√	√	√	×	√	√	√	×	√	√	√
www.abnamro.nl	443	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
www.rabobank.nl	443	√	√	√	√	√	√	×	×	×	√	√	×	√	√	√
www.snsbank.nl	443	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
www.bol.com	443	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
www.wehkamp.nl	443	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
www.conrad.nl	443	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
www.alternate.nl	443	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
www.4launch.nl	443	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
www.google.nl	443	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
www.microsoft.com	443	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
www.yahoo.com	443	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
www.duckduckgo.com	443	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
www.ask.com	443	√	√	√	√	×	√	×	×	×	×	×	×	√	√	√
www.facebook.com	443	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
www.linkedin.com	443	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
www.twitter.com	443	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
www.tumblr.com	443	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
www.instagram.com	443	√	√	√	√	×	√	×	×	×	√	√	×	√	×	×
www.edarling.nl	443	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
www.relatieplanet.nl	443	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
www.e-matching.nl	443	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
www.studentdating.eu	443	√	√	√	√	×	√	×	√	×	√	√	×	√	×	×
www.tweakers.net	443	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
www.ov-chipkaart.nl	443	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
www.9292.nl	443	√	√	√	√	√	√	√	×	√	√	√	×	×	×	×
www.ns.nl	443	√	√	√	√	√	√	×	×	×	×	√	×	√	√	√

Table 7: Common mistakes service scope. ($\surd$ = in order, $\times$ is not in order.)

C Testresults IMAPS

These are the results for the tests on the production IMAPS servers

Tag	Description	Weight	Required
A	Signature hash algorithm	80	No
B	Certificate (chain) trusted	0	Yes
C	Certificate is valid	0	Yes
D	No Debian weak keys	100	No
E	Subject name matches	0	Yes
F	Compression disabled	50	No
G	Cipher suites do not contain MD5	50	No
H	Perfect forward secrecy available	50	No
I	Cipher suites do not contain RC4	80	No
J	Key length at least 128bits	80	No
K	SSLv2 disabled	100	No
L	SSLv3 disabled	30	No
M	TLSv1.0 enabled	75	No
N	TLSv1.1 enabled	100	No
O	TLSv1.2 enabled	100	No

Table 8: Imaps service legend

Host	Port	Score	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
mailhost.uvt.nl	465	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
imap.hostnet.nl	993	51.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	×	×
imap.os3.nl	993	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
imap-mail.outlook.com	993	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
imap.gmail.com	993	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
imap.xs4all.nl	993	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
mailbox.uvt.nl	993	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
webmail.uva.nl	993	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
mail.axc.nl	993	74.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	×	×

Table 9: Imaps service results. ($\surd$ = in order, $\times$ is not in order.)

D Testresults Tilburg University IP Space

This results are based upon our health assessment tool with one change in the settings. The Subject Name check is performed but does not affect the outcome of the test. This is done due the fact that a single server can host multiple services and thus listens to multiple host names. The result of the check is however listed in our result table because for some hosts it might be desirable to have this test passed.

Tag	Description	Weight	Required
A	Signature hash algorithm	80	No
B	Certificate (chain) trusted	0	Yes
C	Certificate is valid	0	Yes
D	No Debian weak keys	100	No
E	Subject name matches	0	Yes
F	Compression disabled	50	No
G	Cipher suites do not contain MD5	50	No
H	Perfect forward secrecy available	50	No
I	Cipher suites do not contain RC4	80	No
J	Key length at least 128bits	80	No
K	SSLv2 disabled	100	No
L	SSLv3 disabled	30	No
M	TLSv1.0 enabled	75	No
N	TLSv1.1 enabled	100	No
O	TLSv1.2 enabled	100	No

Table 10: Tilburg University IP Space legend

Host	Port	Score	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
vpn.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
137.56.127.11	443	60.0%	√	√	√	√	o	√	√	×	×	√	√	×	√	×	×
freitag-1.uvt.nl	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
freitag.uvt.nl	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
ef0171.uvt.nl	443	88.0%	√	√	√	√	o	√	√	√	×	√	√	×	√	√	√
ef0174.uvt.nl	443	82.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	√	√
maximus.uvt.nl	443	0.0%	√	×	×	√	o	√	×	√	×	√	√	×	√	×	×
flowsel.uvt.nl	443	88.0%	√	√	√	√	o	√	√	√	×	√	√	×	√	√	√
caesar.ilo.uvt.nl	443	0.0%	√	×	√	√	o	√	×	×	×	×	√	×	√	√	√
stuwww.uvt.nl	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
africa.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
zermelo.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
dolus.uvt.nl	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
buizel.uvt.nl	443	82.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
homsar.uvt.nl	443	88.0%	×	√	√	√	o	√	√	√	√	√	√	×	√	√	√
pichu.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
acceptatie.meresco.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
xe0123.uvt.nl	443	0.0%	√	×	√	√	o	√	×	√	×	√	√	×	√	×	×
lunatone.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
aplan001.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
lw0159.uvt.nl	443	0.0%	×	×	×	√	o	√	√	√	×	√	√	×	√	√	√
lw0198.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
jupiler.uvt.nl	443	54.0%	√	√	√	√	o	√	×	×	×	√	√	×	√	×	×
cdata3.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata4.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata4-1.uvt.nl	443	0.0%	√	×	×	√	o	√	×	√	×	√	√	×	√	×	×
cdata8.uvt.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
cdata9.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
cs0218.uvt.nl	443	0.0%	√	×	√	√	o	√	×	×	×	√	√	×	√	×	×

Table 11: Tilburg University IP Space results ($\surd$ = in order, $\times$ is not in order, o is ignored.)

Host	Port	Score	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
cdata21.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata23.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata24.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata26.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cs0225.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
cs0229.uvt.nl	443	0.0%	√	×	×	√	o	√	×	√	×	√	√	×	√	×	×
cs0230.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
cs0231.uvt.nl	443	0.0%	√	×	×	√	o	√	×	√	×	√	√	×	√	×	×
cdata27.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cs0234.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
www.profilesregistry.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
cs0240.uvt.nl	443	0.0%	√	×	√	√	o	√	×	√	×	√	√	×	√	×	×
cs0241.uvt.nl	443	0.0%	√	×	√	√	o	√	×	√	×	√	√	×	√	×	×
cdata29.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata30.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
cdata31.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
ls0022.uvt.nl	443	65.0%	√	√	√	√	o	√	√	√	×	√	√	×	√	×	×
ls0023.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
ls0026.uvt.nl	443	65.0%	√	√	√	√	o	√	√	√	×	√	√	×	√	×	×
wronski.uvt.nl	443	60.0%	√	√	√	√	o	√	√	×	×	√	√	×	√	×	×
nash.uvt.nl	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
praalder.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
graves.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
ha-1.redirect.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
ha-2.redirect.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
azumaya.uvt.nl	443	0.0%	√	×	√	√	o	√	√	√	√	√	√	×	√	√	√
cataldi.uvt.nl	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
xv0032.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
drec.cert.uvt.nl	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
mayall.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
tnas012.tiasnimbas.edu	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
tnli002.tiasnimbas.edu	443	60.0%	√	√	√	√	o	√	√	×	×	√	√	×	√	×	×
ts0024.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
neumann.tiasnimbas.edu	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
tnli005.tiasnimbas.edu	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
siegel.tiasnimbas.edu	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
prjsf101.campus.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
kahina.uvt.nl	443	54.0%	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
remote.feb.uvt.nl	443	34.0%	√	√	√	√	√	√	×	×	×	×	×	×	√	×	×
es0061.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
es0063.uvt.nl	443	69.0%	√	√	√	√	o	√	√	×	√	√	√	×	√	×	×
monitor.feb.uvt.nl	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
es0091.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
ls0136.uvt.nl	443	0.0%	√	×	√	√	o	√	√	√	×	√	√	×	√	×	×
ls0152.uvt.nl	443	0.0%	√	×	√	√	o	√	√	√	√	√	√	×	√	√	√
ls0158.uvt.nl	443	65.0%	√	√	√	√	o	√	√	√	×	√	√	×	√	×	×
gregorius.uvt.nl	443	0.0%	√	×	√	√	√	√	√	√	√	√	√	×	√	×	×
portalis.uvt.nl	443	45.0%	√	√	√	√	√	√	×	×	×	×	√	×	√	×	×
rs0226.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
dbicluster.oder.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
galois.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
sherwood-ha8.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
lovelace.uvt.nl	443	60.0%	√	√	√	√	o	√	√	×	×	√	√	×	√	×	×
qurra-ha.uvt.nl	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
yunus-ha.uvt.nl	443	97.0%	√	√	√	√	o	√	√	√	√	√	√	×	√	√	√
pplan001.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
descartes.uvt.nl	443	35.0%	√	√	√	×	o	√	×	×	×	√	√	×	×	×	×
cooper.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
cr0072.uvt.nl	443	82.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	√	√
davinci-ha0.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
dbicluster.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
davinci-ha1.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
sherwood-ha3.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
sherwood-ha4.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
sherwood-ha5.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
sherwood-ha6.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
sherwood-ha7.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
norma.uvt.nl	443	54.0%	√	√	√	√	o	√	×	×	×	√	√	×	√	×	×
curry.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
fibonacci.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
hermite.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
productie.gx.uvt.nl	443	60.0%	√	√	√	√	o	√	√	×	×	√	√	×	√	×	×

Table 12: Tilburg University IP Space results ($\surd$ = in order, $\times$ is not in order, o is ignored.)

Host	Port	Score	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
cr0136.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
dloprd.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
cardano.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
azimuth.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
tolstoj.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
cr0191.uvt.nl	443	82.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	√	√
meresco.uvt.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
cr0197.uvt.nl	443	82.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	√	√
sherwood-ha1.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
davinci-ha4.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
davinci-ha6.uvt.nl	443	15.0%	√	√	√	×	o	√	×	×	×	×	×	×	×	×	×
webap001.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
an0024.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
tsgateway.campus.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
spitswww.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
an0032.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
an0033.uvt.nl	443	54.0%	√	√	√	√	o	√	×	×	×	√	√	×	√	×	×
an0034.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
an0036.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
an0037.uvt.nl	443	54.0%	√	√	√	√	o	√	×	×	×	√	√	×	√	×	×
an0038.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
an0039.uvt.nl	443	60.0%	√	√	√	√	o	√	×	√	×	√	√	×	√	×	×
feweb.uvt.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
autodiscover.campus.uvt.nl	443	58.0%	√	√	√	√	o	√	×	×	×	√	√	√	√	×	×
sip.tilburguniversity.edu	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
lynccon.tilburguniversity.edu	443	64.0%	√	√	√	√	√	√	√	×	√	√	√	√	×	×	×
lynccon.tilburguniversity.edu	443	64.0%	√	√	√	√	√	√	√	×	√	√	√	√	×	×	×
lync.tilburguniversity.edu	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
officewebapps.tilburguniversity.edu	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
an0057.uvt.nl	443	54.0%	√	√	√	√	o	√	×	×	×	√	√	×	√	×	×

Table 13: Tilburg University IP Space results. ($\surd$ = in order, $\times$ is not in order, o is ignored.)

E Testresults SURFconext IdP’s

These are the results for the tests on the production SURFconext IdP’s.

Tag	Description	Weight	Required
A	Signature hash algorithm	80	No
B	Certificate (chain) trusted	0	Yes
C	Certificate is valid	0	Yes
D	No Debian weak keys	100	No
E	Subject name matches	0	Yes
F	Compression disabled	50	No
G	Cipher suites do not contain MD5	50	No
H	Perfect forward secrecy available	50	No
I	Cipher suites do not contain RC4	80	No
J	Key length at least 128bits	80	No
K	SSLv2 disabled	100	No
L	SSLv3 disabled	30	No
M	TLSv1.0 enabled	75	No
N	TLSv1.1 enabled	100	No
O	TLSv1.2 enabled	100	No

Table 14: SURFnet SSO result legend

Host	Port	Score	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
access.hro.nl	443	65.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	×	×
adfs.amolf.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.artez.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.che.nl	443	45.0%	√	√	√	√	√	√	×	×	×	×	√	×	√	×	×
adfs.cibap.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.differ.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.esciencecenter.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.fryske-akademy.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.gildeopleidingen.nl	443	45.0%	√	√	√	√	√	√	×	×	×	×	√	×	√	×	×
adfs.glu.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.grac.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.hsleiden.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.hsmarnix.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.iknl.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.ipabo.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.kb.nl	443	65.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	×	×
adfs.knmi.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.kpz.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
adfs.maastrichtuniversity.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.myhz.nl	443	54.0%	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
adfs.ncoi.com	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.netwerkopleidingartsen.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.nioz.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.nwo.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.nyenrode.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.somt.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
adfs.stenden.com	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.studielink.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
adfs.tue.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs.uvh.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs01.kempel.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs1.oba.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs2.cbs.knaw.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs2.cito.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs2.fontys.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs2.hubrecht.eu	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs2.politieacademie.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs20.hva.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
adfs2host.tmg.rocmn.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfs2prod.aventus.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
adfsv2.inholland.nl	443	54.0%	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
auth.onderwijsgroeptilburg.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
authentigate.surfmarket.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
bureau.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
conext.authenticatie.ru.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
dans.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
ehumanities.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
engine.surfconext.nl	443	82.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
engine.surfconext.nl	443	82.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
espee.SURFnet.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
fed.rijnijssel.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federaasje.tresoar.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.amc.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.cbg.nl	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
federatie.clusius.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.driestar-educatief.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.gh.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.graafschapcollege.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.hku.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.lumc.nl	443	60.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	×	×
federatie.nivel.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.trimbos.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.umcn.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federatie.zadkine.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federation.hanze.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federation.hdh.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
federation.nioo.knaw.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×

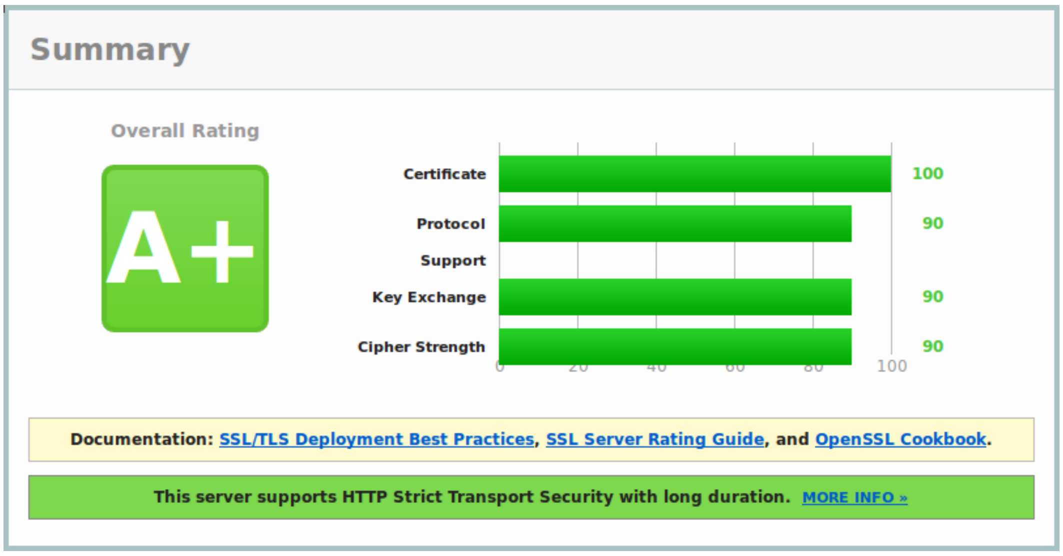
Table 15: SURFconext IdP’s. ($\surd$ = in order, $\times$ is not in order.)

Host	Port	Score	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
federation.novay.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
fedlogin.noh-i.nl	443	37.0%	√	√	√	√	√	√	×	×	×	×	×	×	×	×	×
fedlogin.ojh.nl	443	37.0%	√	√	√	√	√	√	×	×	×	×	×	×	×	×	×
fedlogin.schakelzone.nl	443	37.0%	√	√	√	√	√	√	×	×	×	×	×	×	×	×	×
fedlogin.studienet.ou.nl	443	37.0%	√	√	√	√	√	√	×	×	×	×	×	×	×	×	×
fedlogin.utwente.nl	443	37.0%	√	√	√	√	√	√	×	×	×	×	×	×	×	×	×
fs.multrix.com	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
fs.rocmondriaan.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
fshub.umcutrecht.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
gatekeeper2.tudelft.nl	443	51.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	×	×
glr-adfs.glr.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
huygens.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
icin.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
idp.cwi.nl	443	82.0%	√	√	√	√	√	√	√	×	×	√	√	×	√	√	√
idp.dir.garr.it	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
idp.ihe.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
idp.mijnhelicon.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
idp.mpi.nl	443	40.0%	√	√	√	√	√	√	×	√	×	×	×	×	√	×	×
idp.SURFnet.nl	443	64.0%	√	√	√	√	√	√	√	×	√	√	√	√	×	×	×
idp01.nhl.nl	443	40.0%	√	√	√	√	√	√	×	√	×	×	×	×	√	×	×
idservice.zuyd.nl	443	45.0%	√	√	√	√	√	√	×	×	×	×	√	×	√	×	×
iisg.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
imogen.SURFnet.nl	443	65.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	×	×
inloggen.hogeschoolutrecht.nl	443	74.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	×	×
login.ahk.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
login.avans.nl	443	51.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	×	×
login.has.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
login.services.uu.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
login.stc-r.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
login.terena.org	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
login.uaccess.leidenuniv.nl	443	51.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	×	×
logon.erasmusmc.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
mdb-vw-adfs.zebi.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
meertens.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
nam-id.astron.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
namidp.rocvantwente.nl	443	65.0%	√	√	√	√	√	√	√	√	√	×	√	×	√	×	×
nias.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
nidi.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
niod.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
nsab4050.nlda.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
openidp.feide.no	443	88.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	√	√
rathenau.idp.knaw.nl	443	73.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	√	√
remote.bplusc.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
saml.hhs.nl	443	40.0%	√	√	√	√	√	√	×	√	×	×	×	×	√	×	×
saml.uvt.nl	443	97.0%	√	√	√	√	√	√	√	√	√	√	√	×	√	√	√
secure.uva.nl	443	54.0%	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
senldogo0803.springer-sbm.com	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
signon.rug.nl	443	51.0%	√	√	√	√	√	√	×	√	×	×	√	×	√	×	×
sso.eur.nl	443	54.0%	√	√	√	√	√	√	×	×	×	√	√	×	√	×	×
sso.han.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
sso.nikhef.nl	443	69.0%	√	√	√	√	√	√	√	×	√	√	√	×	√	×	×
sso.sara.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
sso.saxion.nl	443	65.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	×	×
sts.deltion.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
sts.mumc.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
sts.nuffic.nl	443	82.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	√	√
sts.roc-nijmegen.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
sts.talnet.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
sts.windesheim.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
sts.wur.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
student.asre.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
surf-sso.ubvu.vu.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
surfspot.biblionetwerken.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
tcs01.pbl.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
teamwerk.fed.vumc.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
wayf-test.SURFnet.nl	443	40.0%	√	√	√	√	√	√	×	√	×	×	×	×	√	×	×
wayf.SURFnet.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
wayf.SURFnet.nl	443	60.0%	√	√	√	√	√	√	×	√	×	√	√	×	√	×	×
wayf.wayf.dk	443	65.0%	√	√	√	√	√	√	√	√	×	√	√	×	√	×	×
www.onegini.me	443	69.0%	√	√	√	√	√	√	√	×	√	√	√	×	√	×	×

Table 16: SURFconext IdP's. (√ = in order, × is not in order.)

F SSLlabs Comparison

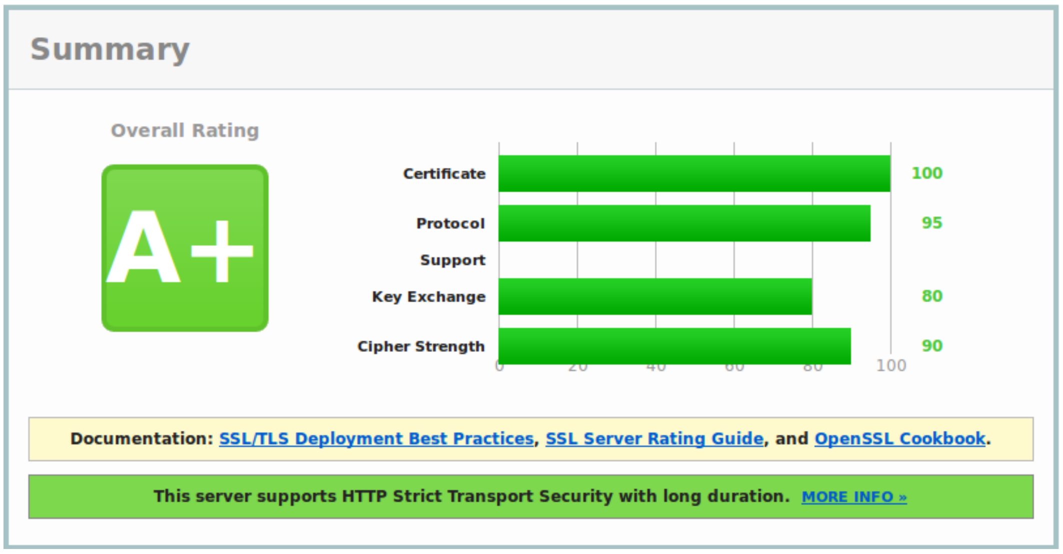
SSLlabs is used to test SSL hosts as well. However, it is found that is it possible to get the best result (A+) with different settings. Below is the outcome of two different hosts getting the same overall rating while having different settings.



Documentation: SSL/TLS Deployment Best Practices, SSL Server Rating Guide, and OpenSSL Cookbook.

This server supports HTTP Strict Transport Security with long duration. MORE INFO »

Figure 2: Tilburg University test server



Documentation: SSL/TLS Deployment Best Practices, SSL Server Rating Guide, and OpenSSL Cookbook.

This server supports HTTP Strict Transport Security with long duration. MORE INFO »

Figure 3: Own test server