Samen aanjagen van vernieuwing

Handreiking beveiligde communicatie

TLS en andere beveiligingsstandaarden voor webverkeer

Auteur(s): SURF, MBO Digitaal
Versie: 1.4
Datum: 1 juli 2025
Kenmerk: Handreiking instellen beveiligde communicatie

Inhoudsopgave

1 Inleiding

Cryptografie, ook wel crypto genoemd, is het omzetten van informatie in een code zodat een ander het niet kan lezen. Dit doet men als men gevoelige informatie veilig wil bewaren of versturen. Meestal bestaat cryptografie uit een algoritme voor versleutelen (encrypt) en ontsleutelen (decrypt) en worden daarbij één (symmetrisch) of meerdere (asymmetrisch) sleutels toegepast. Om risico's te verkleinen is het van belang dat websites ondersteuning bieden aan moderne internetstandaarden en dat deze juist geïmplementeerd zijn. Uit de IV-metingen[1] van SURF is gebleken dat veel instellingen laag scoren op het thema wat onder andere ondersteuning van encryptie meet: 'TLS', oftewel Transport Layer Security. In deze handreiking leggen we daarom uit hoe je deze internetstandaarden voor het beveiligen van communicatie kunt instellen.

Noot: Deze handreiking beperkt zich tot websites. Hoe je DNSSEC en DNS-based Authentication of Named Entities (DANE) voor Microsoft implementeert is opgenomen in de handreiking beveiliging e-mail[2] omdat Microsoft beperkt ondersteuning biedt voor DNSSEC en daarmee ook DANE.

1.1 HTTPS (Hyper Text Transport Protocol Secure)

HTTPS is een protocol om webverkeer te versleutelen via een certificaat. Websitebezoekers herkennen zo'n verbinding aan 'https://' aan het begin van de link. Daarbovenop is er HTTP Strict Transport Security (HSTS). Dat is een extra maatregel zodat er tijdens een bezoek niet (stiekem) teruggeschakeld kan worden naar een niet versleutelde (lees: onveilige) HTTP-verbinding. Beide standaarden zijn basisbeveiliging tegen ongewenste omleidingen en het onderscheppen van webverkeer.

1.2 TLS (Transport Layer Security)

TLS zorgt voor beveiligde internetverbindingen, met als doel de veilige uitwisseling van gegevens tussen internetsystemen (zoals websites of mailservers). Dit maakt het voor cybercriminelen moeilijker om internetverkeer te onderscheppen of te manipuleren.

1.3 Certificaten

Via TLS kan een server zijn identiteit aantonen met behulp van een X.509-certificaat. De client kan uitsluitend via een certificaat vaststellen dat hij met de server communiceert en niet met een derde die van plan is om de communicatie af te luisteren of te manipuleren. In zo'n certificaat staat ook welke sleutels je kan gebruiken om het webverkeer te versleutelen. Het verwerven en beheren van certificaten is essentieel voor de bereikbaarheid en beveiliging van websites en diensten.

1.4 Beveiligingsopties

Met beveiligingsopties worden onder andere de HTTP security headers, security.txt en cookies bedoeld. De beveiligingsopties dragen bij aan het verlagen van bepaalde aanvallen op websites, zoals cross-site-scripting (XSS) of downgrade aanvallen. Met een security.txt bestand maak je duidelijk waar derden kwetsbaarheden kunnen melden en welke spelregels er gelden

[1] https://wiki.surfnet.nl/display/SCIPR/Hoofddomeinen+IV-metingen

[2] https://sec.surf.nl/asset/handreiking-beveiligen-e-mail/

('coordinated vulnerability disclosure')[3]. Cookies kun je attributen toewijzen om de beveiliging te verbeteren. In deze handreiking leggen we uit hoe je een aantal van deze beveiligingsopties kunt implementeren en welke afwegingen je daarin kunt maken.

[3] https://sec.surf.nl/handreiking-cvd-en-security-txt/

2 TLS in de praktijk: sessieopbouw en veilige algoritmes

2.1 Werking van een TLS-sessie

Een verbinding tussen een client en server die met TLS (Transport Layer Security) is beveiligd, wordt een TLS-sessie genoemd. Zo'n sessie bestaat uit twee fasen:

1. **De handschakefase**
   Tijdens de handshake onderhandelen de client en server over de wijze waarop de sessie wordt opgezet. Hierbij worden cryptografische algoritmes overeengekomen, afhankelijk van de ondersteunde TLS-versie aan beide zijden. De handshake wordt geïnitieerd door de client.

2. **De applicatiefase**
   Na afronding van de handshake wordt de sessie gebruikt als een beveiligde tunnel. In deze tunnel vindt versleutelde gegevensoverdracht plaats tussen de applicaties van de client en server. De TLS-sessie waarborgt de integriteit en vertrouwelijkheid van de communicatie.

TLS versleutelt alleen de inhoud van de communicatie (payload). Metadata over het dataverkeer – zoals IP-adressen, tijdstippen en sommige headers – blijft zichtbaar. Hierdoor kan bijvoorbeeld een SOC of aanvaller nog wel zien wanneer, tussen wie en hoe vaak er wordt gecommuniceerd, maar niet wát er wordt verstuurd.

2.2 Cryprografische algoritmes en configuraties

TLS maakt gebruik van meerdere cryptografische bouwstenen, waaronder:

- protocolversie (bijv. TLS 1.3)
- certificaatverificatie
- hashfuncties
- sleuteluitwisseling
- bulkversleuteling
- sleutellengtes
- gebruikte elliptic curves

Aanbevolen algoritmes

De tabel hieronder toont per categorie welke algoritmes als veilig of uit te faseren worden beschouwd. Deze indeling is gebaseerd op actuele richtlijnen van o.a. het NCSC[4], Mozilla[5], OWASP[6] en SSL Labs[7].

Je gebruikt dit overzicht bij het configureren van je webserver of applicatie. Houd daarbij rekening met je doelgroep: voor sommige toepassingen is brede compatibiliteit vereist, terwijl andere juist maximale veiligheid vragen.

[4] https://www.ncsc.nl/documenten/publicaties/2025/juni/01/ict-beveiligingsrichtlijnen-voor-transport-layer-security-2025-05

[5] https://wiki.mozilla.org/Security/Server_Side_TLS

[6] https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html

[7] https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices

Versie / Algoritme	Goed / Voldoende	Uit te faseren / Onvoldoende
TLS	TLS 1.3, TLS 1.2	TLS 1.0, SSL 3.0, SSL 2.0, SSL 1.0
Certificaatverificatie	ECDSA, RSA	DSS, EXPORT-varianten, PSK, Anon, NULL
Hashfuncties	SHA-512, SHA-384, SHA-256	SHA-224, SHA-1, MD5
Sleuteluitwisseling	X25519MLKEM768, SecP256r1MLKEM768, SecP384r1MLKEM1024, ECDHE	DHE, RSA, DH, ECDH, KRB5, NULL, PSK, SRP
Bulkversleuteling	AES-256-GCM, ChaCha20-Poly1305, AES-256-CCM, AES-128-GCM, AES-256-CBC, AES-128-CBC	3DES-CBC, AES-256-CCM_8, AES-128-CCM_8, IDEA, DES, RC4, NULL
Sleutellengte RSA	Minimaal 3072 bit	2048 – 3071 bit, < 2048 bit
Elliptic curves	secp521r1, Secp384r1, secp256r1, Curve448, Curve25519, brainpoolP512r1, brainpoolP384r1, brainpoolP256r1	Secp224r1, Andere curves

Let op: de minimale veilige sleutellengtes verschillen per algoritme. Op de website Keylength vind je per gebruikstoepassing de aanbevolen sleutellengtes.

2.3 TLS configuraties testen

Na het toepassen of wijzigen van een TLS-configuratie is het belangrijk om de instellingen zorgvuldig te testen. Dit geldt zowel voor webservers als voor clients. Testen helpt om misconfiguraties te voorkomen en garandeert dat je voldoet aan de vereisten voor veilige communicatie.

Testtool voor clients

Voor het testen hoe clients TLS ondersteunen kun je gebruik maken van o.a. de volgende tools:

- **SSL Labs Client Test**: https://clienttest.ssllabs.com:8443/ssltest/viewMyClient.html
- **How's My SSL**: https://www.howsmyssl.com/

Testtools voorwebservers

Controleer of je server correct is geconfigureerd, welke cipher suites en protocollen worden ondersteund, en of certificaten goed zijn geïnstalleerd met o.a. deze tools:

- **SSL Labs Server Test:** https://www.ssllabs.com/ssltest/
- **SSL Shopper SSL Checker:** https://www.sslshopper.com/ssl-checker.html
- **internet.nl:** https://internet.nl/

Aanvullende tips

- Gebruik geautomatiseerde scans via vulnerability management tooling om periodiek te controleren op verouderde instellingen of versleuteling.
- Test niet alleen nieuwe implementaties, maar ook na software-updates of wijzigingen in infrastructuur (bijv. load balancers of reverse proxies).
- Documenteer testresultaten en vertaal deze waar nodig naar aanpassingen in het risicodossier of configuratiestandaarden.

3 Veilige en toekomstbestendige TLS-configuratie

Een juiste configuratie van TLS (Transport Layer Security) is cruciaal voor het beveiligen van netwerkverkeer. TLS beschermt de vertrouwelijkheid en integriteit van gegevens die tussen systemen worden uitgewisseld, zoals bij webapplicaties, e-mailservers en API-koppelingen. Onjuiste of verouderde instellingen maken diensten kwetsbaar voor afluisteren, manipulatie of downgrade-aanvallen.

In dit hoofdstuk beschrijven we een efficiënte en toekomstbestendige manier om TLS goed in te stellen, op basis van de Mozilla SSL Configuration Generator[8].

3.1 Actuele configuratievoorbeelden via Mozilla SSL Configuration Generator

Het handmatig configureren van TLS kan complex zijn en is afhankelijk van factoren zoals het type server, gebruikte softwareversies en specifieke compatibiliteitseisen. Om dit proces te vereenvoudigen, het gebruik aan van de Mozilla SSL Configuration Generator een mogelijkheid.

Deze SSL Configuration Generator biedt:

- Actuele configuratievoorbeelden voor populaire webservers en andere toepassingen (zoals Apache, Nginx, HAProxy, Postfix, Exim, etc.
- Instelbare beveiligingsniveaus: Modern, Intermediate of Old.
- Adviezen die aansluiten bij best practices van onder andere OWASP en NCSC.
- Regelmatige updates, zodat de configuraties meebewegen met technologische en dreigingsontwikkelingen.

Werkwijze

Gebruik de SSL Configuration Generator in de volgende stappen:

1. Selecteer de gewenste server of toepassing.
2. Kies het bijpassende beveiligingsniveau (zie 3.2).
3. Pas optioneel aanvullende instellingen aan (zoals HTTP/2 of HSTS).
4. Kopieer de gegenereerde configuratie en pas deze toe op je systeem.
5. Test de configuratie met tools als SSL Labs of lokale validatietools.

De gegenereerde instellingen kunnen meestal direct worden toegepast of dienen als startpunt voor je eigen configuratie, afgestemd op je lokale situatie en beleidskaders.

3.2 Beveiligingsniveaus: maak een bewuste keuze

In de Mozilla SSL Configuration Generator kun je kiezen uit drie configuratieprofielen. Elk profiel maakt een andere afweging tussen beveiliging en compatibiliteit. Kies het profiel dat het beste past bij de toepassing en gebruikersgroep van de dienst.

Modern profiel

- **Doelgroep**: moderne browsers en systemen
- **Ondersteunt**: uitsluitend TLS 1.3 en de sterkste cipher suites
- **Voordeel**: maximale veiligheid
- **Beperking**: oudere clients kunnen geen verbinding maken

[8] https://ssl-config.mozilla.org/

Aanbevolen bij:

- Interne toepassingen waarbij clients beheerd worden
- Nieuwe infrastructuur zonder legacy
- Diensten met een hoog beveiligingsniveau (bijv. adminportalen)

Intermediate profiel

- **Doelgroep**: breed scala aan browsers en systemen vanaf ca. 2014
- **Ondersteunt**: TLS 1.2 en TLS 1.3 met veilige cipher suites
- **Voordeel**: goede balans tussen veiligheid en compatibiliteit
- **Beperking**: iets minder strikt dan “Modern”

Aanbevolen bij:

- Publiek toegankelijke webdiensten of portals
- Diensten met gebruikers op uiteenlopende apparaten (bijv. studenten)
- Algemene productieomgevingen

Old profiel

- **Doelgroep**: legacy-systemen en sterk verouderde clients
- **Ondersteunt**: ook oudere TLS-versies (TLS 1.0/1.1) en zwakkere algoritmes
- **Voordeel**: maximale compatibiliteit
- **Beperking**: verminderde veiligheid, verhoogd risico op aanvallen

Alleen gebruiken bij:

- Onvervangbare verouderde systemen
- Tijdelijke situaties waarin migratie nog niet mogelijk is

Let op: Gebruik dit profiel uitsluitend met een expliciete risicoafweging. Documenteer afwijkingen, isoleer systemen indien mogelijk en plan een migratietraject.

Microsoft IIS

Voor Microsoft IIS kan de tool IIS Crypto gebruikt worden. Zorg ervoor dat via deze tool de juiste cipher suites geselecteerd worden. Gebruik hiervoor de tabel in 2.2.

4 Certificaten

Goed certificaatbeheer is essentieel voor het voorkomen van diverse beveiligingsrisico's. Het helpt bij het beschermen tegen subdomein overnames, waarbij onbevoegden toegang kunnen krijgen tot gevoelige informatie door zich voor te doen als jouw domein. Ook voorkomt het de onbereikbaarheid van websites door verlopen certificaten, wat essentieel is voor het behouden van vertrouwen en betrouwbaarheid. Daarnaast draagt het bij aan het voorkomen van onbetrouwbare verbindingen en data-interceptie, waardoor de integriteit en vertrouwelijkheid van informatie gewaarborgd blijft. Echter, de vereisten rondom certificatenbeheer veranderen in een rap tempo:

1. **Verkorte geldigheidsduur**. Vroeger hadden certificaten een geldigheidsduur van meerdere jaren. Tegenwoordig is dat teruggebracht tot één jaar en in de nabije toekomst zal dit richting één maand gaan.
2. **Handmatig beheer is inefficiënt.** Met de verkorte geldigheidsduur betekent dit dat ICT-beheerders potentieel elke maand honderden certificaten handmatig moeten bijwerken. Dit vertaalt zich naar vele dagen werk, wat niet alleen inefficiënt is, maar ook tijdrovend en duur.
3. **Externe kosten**: Wanneer jullie instelling besluit om certificatenbeheer uit te besteden, of dit al heeft gedaan, zullen de frequente updates leiden tot aanzienlijk hogere kosten vanwege de benodigde werkuren.

Om het beheren van certificaten te vereenvoudigen wordt sterk aanbevolen om gebruik te maken van SURFcertificaten en het ACME-protocol, zie §4.3.
Let op: het gebruik van zelf ondertekende certificaten dient vermeden te worden.

4.1 Certificaat transparantie

Alle SSL/TLS certificaten die uitgegeven worden door publieke CA's moeten sinds het Diginotar incident gepubliceerd worden in Certificate Transparency (CT) Logs. Dat wordt afgedwongen door het CA/Browser Forum. Sectigo en Google stellen de beste CT-Logs gratis beschikbaar. Je kunt deze per domein opvragen via bijvoorbeeld: https://crt.sh/. Instellingen worden geadviseerd om hun certificaten te monitoren via SURFcertificaten of eigen montior-tooling. Omdat de logs publiekelijk toegankelijk zijn is monitoring belangrijk. De transparency logs kunnen gebruikt worden voor bijvoorbeeld detectie van phishing sites, het traceren kwaadaardige infrastructuur en het monitoren gebruik van van de instellingsnaam.

4.2 CAA-record

DNS vertaalt namen naar IP-adressen en met een DNS CAA-record kunnen instellingen duidelijk maken welke leverancier(s) (Certificate Authority (CA)) geautoriseerd zijn om certificaten voor het (sub)domein uit te geven. Om te testen of voor een (sub)domein een CAA-record in de DNS-zone aanwezig is kun je gebruik maken van caatest. Je kunt een CAA-record eenvoudig genereren via de handige tool van sslmate. Geadviseerd wordt om voor ieder domein een CAA-record in de DNS-zone te plaatsen. Een CA is verplicht te controleren op de aanwezigheid van een CAA-record voor de Fully Qualified Domain Name (FQDN).

Voorbeeld SURF:

surf.nl.	600	IN	CAA	0 issue "harica.gr"

4.3 ACME (Automated Certificate Management Environment)

ACME (RFC 8555) automatiseert de uitgifte en vernieuwing van X.509-certificaten. Dit proces vermindert arbeidsintensiviteit en foutgevoeligheid, essentieel bij kortere levensduur van certificaten. ACME is leverancier-onafhankelijk, biedt schaalbaarheid voor individuele certificaten per server en verbetert de beveiliging door doordat een kleinere scope en kortere geldigheid per certificaat haalbaar wordt. Ook voor het werken met externe leveranciers is ACME een uitkomst, aangezien je niet langer handmatig nieuwe certificaten naar ze hoeft te sturen, maar ze eenmalig een ACME-account geeft om zelf certificaten te kunnen vernieuwen. Een bekende implementatie van ACME is certbot, maar er zijn meer dan 100 verschillende clients, die integreren met een veelvoud aan systemen, die je kan gebruiken i.c.m. ACME.

4.4 API (Application Programming Interface)

Naast de open standaard ACME bieden veel Certifiate Authorities (CA's), zo ook Harica, via SURFcertificaten een API, die een aanvullende methode biedt voor het beheer van certificaten. Deze API stelt gebruikers in staat om direct met de CA te communiceren voor het beheren van certificaatlevenscycli. Dit omvat het aanvragen, vernieuwen, intrekken en beheren van certificaten, gebruikers en domeinen. De API is bijzonder nuttig voor organisaties die een meer geavanceerde, op maat gemaakte benadering van certificaatbeheer nodig hebben, vooral bij grootschalige of complexe implementaties.

5 Beveiligingsopties

Er zijn verschillende beveiligingsopties (security headers) mogelijk die ieder op hun eigen manier de kans en impact van specifieke cyberaanvallen verminderen. Het gaat om de volgende security headers:

- **X-Content-Type-Options**
- **X-Frame-Options**
- **Content-Security-Policy (CSP)**
- **Referrer-Policy**
- **Strict-Transport-Security (HSTS)**
- **X-XSS-Protection**
- **Permissions-Policy**
- **Clear-Site-Data**
- **Security.txt**
- **Cookies**

Via Security Headers of Mozilla Observatory kun je controleren of de security headers voor een (sub)domein (correct) geïmplementeerd zijn. Er zijn aanvullend op de genoemde security headers meer security headers beschikbaar, zie voor meer informatie en handelingsperspectief de richtlijnen van OWASP.[9]

Technologie	Waar implementeren	Voorbeeld
PHP	In de broncode van de pagina	header("X-Frame-Options: DENY");
Apache	.htaccess bestand	<IfModule mod_headers.c> Header set X-Frame-Options "DENY" </IfModule>
IIS	web.config bestand	<system.webServer> ... <httpProtocol> <customHeaders> <add name="X-Frame-Options" value="DENY" /> </customHeaders> </httpProtocol> ... </system.webServer>
Nginx	Voeg de regel toe aan de sites-enabled configuratie bestanden	`add_header "X-Frame-Options" "DENY";`
HAProxy	Voeg de regel(s) toe aan je front-end, listen of backend configuratie	`http-response set-header X-Frame-Options DENY`

Express	Je kunt helmet gebruiken om HTTP Security headers in te stellen	const helmet = require('helmet');
		const app = express();
		// Sets "X-Frame-Options: SAMEORIGIN"
		app.use(
		helmet.frameguard({
		action: "sameorigin",
		})
		);

[9] https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html

5.1 X-Content-Type-Options

Deze header voorkomt dat de browser bestanden interpreteert als een ander MIME type dan wat gespecificeerd is in de Content-Type header (bijvoorbeeld *tekst/plain* als *text/css*). Stel daarom het volgende in:

```
X-Content-Type-Options: nosniff
```

5.2 X-Frame-Options

Deze header verbetert de bescherming tegen clickjacking aanvallen. De header geeft instructies aan de browser of bepaalde content in frames weergeven mag worden. Er zijn drie mogelijke opties in te stellen:

Deny	no rendering within a frame
Sameorigin	no rendering if origin mismatch
Allow-from:	[Domein]

Voorbeeld:

```
X-Frame-Options: deny
```

5.3 Content Security Policy

Geadviseerd wordt om met een basis te beginnen waarbij afbeeldingen, scripts, AJAX, formulier acties en CSS van dezelfde oorsprong toegestaan worden:

```
Content-Security-Policy: default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';base-uri 'self';form-action 'self'
```

5.4 Referrer Policy

Deze header bepaald welke informatie over doorverwijzingen in de Referrer header opgenomen moeten worden wanneer verzoeken worden gedaan. Er zijn verschillende opties mogelijk, maar geadviseerd wordt om alleen de oorsprong van verzoeken vast te leggen in verband met de privacy van bezoekers:

```
Referrer-Policy: strict-origin-when-cross-origin
```

5.5 Strict-Transport-Security (HSTS)

Deze header beschermt websites tegen downgrade aanvallen en cookie hijacking. Geadviseerd wordt om het volgende in te stellen:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

5.6 X-XSS-Protection

De X-XSS-Protection-header is verouderd door moderne browsers en het gebruik ervan kan extra beveiligingsproblemen aan de clientzijde veroorzaken. Daarom wordt aanbevolen om de header in te stellen op: `X-XSS-Protection: 0` om de XSS-auditor uit te schakelen en niet toe te staan dat deze het standaardgedrag overneemt van de browser die het antwoord afhandelt. Gebruik in plaats daarvan de Content-Security-Policy.

5.7 Permissions-Policy

De header Permissions-Policy vervangt de bestaande Feature-Policy header voor het beheren van het delegeren van machtigingen en krachtige functies. De header maakt gebruik van een gestructureerde syntaxis en stelt sites in staat om strakker te beperken welke oorsprongen toegang kunnen krijgen tot functies. Geadviseerd wordt om de Permissions-Policy te genereren via de tool op https://www.permissionspolicy.com/

5.8 Clear-Site-Data

De Clear-Site-Data-header wist browsegegevens (cookies, opslag, cache) die zijn gekoppeld aan de aanvragende website. Het stelt ontwikkelaars in staat om meer controle te hebben over de gegevens die lokaal door een browser zijn opgeslagen voor hun oorsprong. Deze header is bijvoorbeeld handig tijdens een uitlogproces, om ervoor te zorgen dat alle opgeslagen inhoud aan de clientzijde, zoals cookies, opslag en cache, wordt verwijderd. Met de opmars van bijvoorbeeld infostealers is dit een hulpmiddel om misbruik te voorkomen. Opties:

“cache”
“cookies”
“storage”
“executionContexts”
“*”

Voorbeeld:

```
Clear-Site-Data: “cache”, “cookies”, “storage”
```

5.9 Security.txt

Met een security.txt maak je aan de buitenwereld duidelijk waar kwetsbaarheden gemeld kunnen worden en welke spelregels er gelden. Hoe je dit in kunt stellen en wat je vooraf moet organiseren kun je lezen in het stappenplan van SURF.

Voorbeeld Coordinated Vulnerability Disclosure (CVD) beleid van SURF
Voorbeeld centraal CVD-beleid mbo sector
Voorbeeld security.txt van SURF

5.10 Cookies

Alle cookies moeten het Secure attribuut toegewezen hebben om te voorkomen dat gevoelige informatie onbeveiligd (HTTP) getransporteerd wordt. Ook wordt geadviseerd om het HttpOnly attribuut toe te voegen om de kans op succes van XSS-aanvallen te verminderen.

Voorbeeld:

```
Set-Cookie: <cookie-name>=<cookie-value>; Secure; HttpOnly
```

Controleren of cookies correct geïmplementeerd zijn kun je via de Storage inspector in je browser.

6 Verwijzingen

6.1 Relatie met SURFaudit Toetsingskader IB

Deze handreiking heeft raakvlak met de volgende elementen van het SURFaudit Toetsingskader IB:

- GO.02 Beleid
- RM.03 Plan voor risicobehandeling en beperking van risico's
- HR.05 Kennisdeling
- CO.02 Configuratiedatabase en baseline
- SM.01 Security Baselines
- SM.05 Testen van, inspectie van en toezicht op beveiliging
- SM.07 Beheer van bedreigingen en kwetsbaarheden
- SM.10 Beheer van cryptografische sleutels
- SM.12 Beheersing van malware-aanvallen

6.2 SURF dienstverlening

Een aantal diensten van SURF kunnen bijdragen aan het verbeteren van de communicatiebeveiliging en daarmee de 'TLS' score uit de IV-metingen.

Dienst	Relevantie	Contactpersoon
SURFcertificaten	Certificaatbeheer	certificaten-beheer@surf.nl
SURFdomeinen	Domeinbeheer	dns-beheer@surf.nl
Security Expertise Centrum	Achtergrondinformatie & kennisdeling	sec@surf.nl