Dresden, den 22.06.2021

# Handreichung zur sicheren Konfiguration von TLS

Die folgenden Konfigurationsvorlagen können zur Konfiguration verbreiteter Webserver nach den Empfehlungen des AKIF verwendet werden. Als Alternative wird für Apache und nginx das etwas striktere “intermediate” Profil aus dem Mozilla SSL Configuration Generator [1] gezeigt.

Vor dem Aktivieren dieser Konfiguration sollte unbedingt geprüft werden, ob die Clients die Verwendung von TLS1.2 unterstützen. Dies kann z.B. durch Protokollierung der TLS Version und CipherSuite in den Access-Logs des Servers geschehen. Falls ein nennenswerter Anteil der Clients mit älteren TLS Versionen zugreift, ist abzuwägen ob die Verfügbarkeit des Dienstes für alle Nutzer oder der potentielle Sicherheitsgewinn wichtiger ist.

OCSP Stapling sollte nur aktiviert werden, wenn der Server Zugriff auf die Webseiten/OCSP URLs der beteiligten CAs hat.

Die Konfiguration von HSTS (HTTP Strict Transport Security) wurde bewusst ausgespart. Vor dem Deployment entsprechender Parameter muss die Funktionisfähigkeit des TLS Setup unbedingt getestet werden. Andernfalls muss man mit der Nicht-Erreichbarkeit der Seite rechnen.

Update 22.06.2021: TLS1.3 für nginx ergänzt

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

*Adresse:*
Technische Universität Dresden
Sachgebiet Informationssicherheit, TUD-CERT
01062 Dresden

*Internet*:
https://tu-dresden.de/cert
*E-Mail:*
cert@tu-dresden.de

TUD CERT Computer Emergency Response Team

# 1. Apache

Konfiguration nach AKIF Empfehlung:

```
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite          ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-
SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-DSS-AES128-
GCM-SHA256:DHE-DSS-AES256-GCM-SHA384:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-
SHA256:DHE-DSS-AES128-SHA256:DHE-DSS-AES256-SHA256
SSLHonorCipherOrder     on
SSLCompression          off
SSLSessionTickets       off

# OCSP Stapling,
# activate only if server has access to CA's OCSP urls
# only in httpd 2.3.3 and later
SSLUseStapling          on
SSLStaplingResponderTimeout 5
SSLStaplingReturnResponderErrors off
SSLStaplingCache        shmcb:/var/run/ocsp(128000)
```

Alternativ, das “Intermediate” Profile:

```
SSLProtocol             all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite          ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-
SHA256:DHE-RSA-AES256-GCM-SHA384
SSLHonorCipherOrder     off
SSLSessionTickets       off

# OCSP Stapling,
# activate only if server has access to CA's OCSP urls
# only in httpd 2.3.3 and later
SSLUseStapling          on
SSLStaplingResponderTimeout 5
SSLStaplingReturnResponderErrors off
SSLStaplingCache        shmcb:/var/run/ocsp(128000)
```

## 2. Nginx

Konfiguration nach AKIF Empfehlung:

```
   ssl_session_timeout 1d;
   ssl_session_cache shared:SSL:50m;
   ssl_session_tickets off;


   # modern configuration. tweak to your needs.
   ssl_protocols TLSv1.2 TLSv1.3;
   ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-
SHA256:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-DSS-AES128-
GCM-SHA256:DHE-DSS-AES256-GCM-SHA384:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-
SHA256:DHE-DSS-AES128-SHA256:DHE-DSS-AES256-SHA256';
   ssl_prefer_server_ciphers on;

   # OCSP Stapling ---
   # fetch OCSP records from URL in ssl_certificate and cache them
   # enable only if server has access to CA's OCSP urls
   ssl_stapling on;
   ssl_stapling_verify on;

   ## verify chain of trust of OCSP response using Root CA and Intermediate
certs
   ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

   resolver <IP DNS resolver>;
```

“Intermediate” Profil:

```
   ssl_session_timeout 1d;
   ssl_session_cache shared:SSL:50m;
   ssl_session_tickets off;


   # modern configuration. tweak to your needs.
   ssl_protocols TLSv1.2 TLSv1.3;
   ssl_ciphers  ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-
SHA256:DHE-RSA-AES256-GCM-SHA384;
   ssl_prefer_server_ciphers off;
```

```
   # HSTS (ngx_http_headers_module is required) (15768000 seconds = 6
months)
   #add_header Strict-Transport-Security max-age=15768000;

   # OCSP Stapling ---
   # fetch OCSP records from URL in ssl_certificate and cache them
   # enable only if server has access to CA's OCSP urls
   ssl_stapling on;
   ssl_stapling_verify on;

   ## verify chain of trust of OCSP response using Root CA and Intermediate
certs
   ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

   resolver <IP DNS resolver>;
```

## 3. IIS

Microsoft IIS wird über Registry Settings und Group Policies konfiguriert. Die SSL/TLS Protokollversionen können über Registry Einträge konfiguriert werden. Die unterstützten Cipher-Suites werden über eine Group Policy konfiguriert.

Registry Einträge zum Deaktivieren von SSL2, SSL3, TLS 1.0, TLS1.1 auf der Serverseite:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\
SCHANNEL\Protocols\SSL 2.0\Server]
"DisabledByDefault"=dword:00000001
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\
SCHANNEL\Protocols\SSL 3.0\Server]
"DisabledByDefault"=dword:00000001
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\
SCHANNEL\Protocols\TLS 1.0\Server]
"DisabledByDefault"=dword:00000001
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\
SCHANNEL\Protocols\TLS 1.1\Server]
"DisabledByDefault"=dword:00000001
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\
SCHANNEL\Protocols\TLS 1.2\Server]
"DisabledByDefault"=dword:00000000
"Enabled"=dword:00000001
```

Zur Konfiguration der Cipher Suites kommt die Group Policy “SSL Cipher Suite Order” unter Computer Configuration > Administrative Templates > Network > SSL Configuration Settings zum Einsatz. Siehe auch https://docs.microsoft.com/en-us/windows-server/security/tls/manage-tls#configuring-tls-cipher-suite-order

Dort wird eine komma-separierte Liste der Cipher Suites eingetragen. Die folgende Liste entspricht der Konfigurationsempfehlung des AKIF.

**Achtung:** Unter Windows 10 muss der Name der Elliptic Curve (“_P256” oder “_P384”) weggelassen werden.

```
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384
```

```
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384_P384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256_P256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
```

Alternativ kann die Konfiguration der CipherSuites über PowerShell erfolgen:

```
$akif_suites = @(
    "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
    "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
    "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
    "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
    "TLS_DHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS_DHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS_DHE_DSS_WITH_AES_256_CBC_SHA256",
    "TLS_DHE_DSS_WITH_AES_128_CBC_SHA256")

$suites = Get-TlsCipherSuite

foreach ( $s in $suites ) {
    Disable-TlsCipherSuite -name $s.Name
}

$index = 0
foreach ( $s in $akif_suites ) {
    Enable-TlsCipherSuite -name $s -Position $index
    $index = $index + 1
}
```