



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)



راه اندازی و پیکربندی امن پروتکل SSL/TLS بر روی سرویس دهنده وب Weblogic

شماره مستند APA-AMIRKABIR-13960610-1
تاریخ نگارش ۱۰ شهریور ۱۳۹۶
شماره نگارش ۱/۰
نگارش آپای امیرکبیر
طبقه بندی عادی

فهرست مطالب

۱	مقدمه	۱
۲	ارزیابی وضعیت فعلی سرویس دهنده	۲
۳	فعال سازی ارتباطات HTTPS	۳
۸	موارد پیشنهادی برای ارتقای امنیت	۴
۸	تنظیم الگوریتم های قدرتمند و Forward secrecy	۴-۱
۹	به روز رسانی نرم افزارها و نسخه ها	۴-۲
۱۰	فعال کردن OCSP Stapling	۴-۳
۱۰	فعال کردن HSTS	۴-۴
۱۲	مراجع	۵

۱ مقدمه

پروتکل‌های SSL و TLS جهت امن کردن ارتباط میان کاربر و سرور از طریق تصدیق هویت، رمزنگاری و صحت، طراحی و پیاده‌سازی شده است. جهت امن کردن داده‌ها این پروتکل‌ها از cipher suite هایی استفاده می‌کنند. هر cipher suite ترکیبی از الگوریتم‌های اصالت‌سنجی، رمزنگاری و کد تصدیق هویت پیغام (MAC) است. در زمان پیکربندی SSL/TLS باید تنظیمات به‌درستی انجام شده و cipher suite های امن مورد استفاده قرار گیرد. علاوه بر آن، باید تنظیمات دیگری جهت امن‌سازی SSL انجام شود که برخی از مهم‌ترین این تنظیمات شامل غیرفعال کردن SSL 2.0 و SSL 3.0، غیرفعال کردن TLS 1.0 Compression و cipher suite های نا امن است. پیکربندی ارائه شده بر روی سروری با مشخصات زیر انجام شده است.

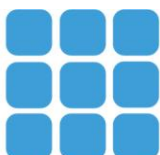
نام نرم‌افزار	نسخه‌ی مورد استفاده
سیستم‌عامل	Linux
وب سرور	WebLogic 10.3.6

۲ ارزیابی وضعیت فعلی سرویس دهنده

برای ارزیابی وضعیت امنیتی SSL/TLS در سرویس دهنده خود از سرویس زیر استفاده نمایید:

<https://sslcheck.certcc.ir/>

پس از انجام موارد امنیتی زیر مجدداً با استفاده از آدرس‌های فوق سرویس خود را پویش کنید تا از برطرف شدن مشکلات موجود مطمئن شوید.



۳ فعال سازی ارتباطات HTTPS

برای پیکربندی سرویس دهنده HTTPS و استفاده از این پروتکل ابتدا باید گواهی نامه دیجیتال مربوطه را از مراکز صدور گواهی (CA^۱) معتبر دریافت کرد (یا گواهی Self-sign را تولید کرد). گرفتن گواهی دارای مراحل است که برای اطلاعات بیشتر در این زمینه می توانید به گزارش ارائه شده توسط پژوهشکده آپای دانشگاه صنعتی امیرکبیر که در آدرس زیر قرار دارد مراجعه کنید:

<http://apa.aut.ac.ir/?p=971>

مزایای گواهی نامه های صادر شده از یک مرکز صدور گواهی مورد اعتماد این است که گواهی نامه، توسط یک مرکز ریشه مورد اعتماد امضا شده و مورد اعتماد همه است. با این حال، در اختیار داشتن گواهی نامه های صادر شده از یک مرکز صدور گواهی، نیازمند هزینه بیشتری است. بنابراین می توان در شبکه ی داخلی از یک گواهی Self-sign استفاده کرد. در غیر این صورت باید گواهی از یک مرکز ریشه معتبر دریافت شده و مورد استفاده قرار گیرد.

در این گزارش، در ابتدا چگونگی ایجاد و استفاده از گواهی نامه های Self-sign در WebLogic شرح داده خواهد شد. سپس نحوه ی امن سازی پروتکل SSL بر روی WebLogic آورده خواهد شد.

مرحله اول: کلیدهای identity میزبان و Keystore را تولید کنید.

در این بخش چگونگی استفاده از یک Keytool جهت تولید گواهی نامه های قدرتمند و همچنین JCEKS keystoreها برای یک دامنه، شرح داده خواهد شد.

دستوری که در ادامه می آید هر دو گواهی نامه دامنه و keystore را توسط keytool می سازد. گواهی نامه یا کلید توسط SHA256 با تاریخ انقضای ۳ سال امضا می شود. برای تولید کلید از دستور `hostname -f` استفاده شده که تولید کلید و keystore را بر اساس نام سرور ایجاد می کند. بخش هایی که با رنگ قرمز مشخص شده را باید بر اساس مقادیر مناسب، مقداردهی کرد.

```
keytool -genkey -alias `hostname -f` -keyalg RSA -keysize 2048 -sigalg SHA256withRSA -  
validity 1095 -keypass mykeypass -storetype jceks -keystore `hostname -f`_identity.jck -storepass  
mystorepass -dname "CN=`hostname -f`, OU=MyOU,O=MyORG, L=MyCity, S=MyState,  
C=MyCountryCode"
```

مرحله ۲: identity مربوط به گواهی root را استخراج کنید.

در این مرحله باید identity گواهی نامه را از شناسه keystore استخراج کنیم. این شناسه گواهی نامه در مرحله بعد در یک keystore مورد اعتماد وارد می شود. مقدار مربوط به پارامتر storepass، مقداری است که در دستور قبل به پارامتر داده شده است.

```
keytool -export -alias `hostname` -f -file `hostname` -f.cer -keystore `hostname` -f_identity.jck -storepass password
```

مرحله ۳: شناسه گواهی نامه را در Keystore مورد اعتماد وارد کنید.

در این مرحله نیاز است تا یک keystore مورد اعتماد JCEKS ایجاد شود. Keystroe مورد اعتماد شامل زنجیره های cert یا گواهی نامه هایی می شود که برای ایجاد اعتماد در طی SSL handshake استفاده شده اند.

```
keytool -import -alias `hostname` -f -file `hostname` -f.cer -keystore `hostname` -f_trust.jck -storetype jceks -storepass mystorepass -noprompt
```

هنگامی که گواهی نامه ها و keystore ها ساخته شدند، باید در یک فضای امن (یک دایرکتوری با دسترسی محدود) ذخیره گردند. باید owner مربوط Weblogic server دسترسی Read_Only به cert ها و keystore ها داشته باشد.

مرحله ۴: تنظیمات SSL را بررسی کنید.

در ابتدا باید SSL listener و پورت مربوط به آن را تنظیم کرد.

۱. در کنسول مدیریتی، به بخش تنظیمات سرور بروید:

Environment > Servers > Admin Server

۲. گزینه SSL Listen Port Enabled را فعال کنید.

۳. به پورت وارد شده برای SSL توجه کنید.

۴. تغییرات را ذخیره کنید.

Name:	AdminServer
Machine:	new_UnixMachine_1
Cluster:	(Standalone)
Listen Address:	<input type="text"/>
<input checked="" type="checkbox"/> Listen Port Enabled	
Listen Port:	<input type="text" value="7001"/>
<input checked="" type="checkbox"/> SSL Listen Port Enabled	
SSL Listen Port:	<input type="text" value="7002"/>
<input type="checkbox"/> Client Cert Proxy Enabled	
Java Compiler:	<input type="text" value="javac"/>
Diagnostic Volume:	<input type="text" value="Low"/>

شکل ۱: فعال کردن SSL listener

مرحله ۵: تنظیم کردن Keystore های Weblogic

حالا که keystore ها و گواهی نامه های که برای شناسایی و اعتماد تولید شدند، باید بر روی WebLogic تنظیم گردد تا از آن ها استفاده کند.

۱. admin server را start کنید.
۲. به کنسول admin وارد شوید.
۳. مسیر Environment > Servers > Admin Server را پیدا کنید.
۴. بر روی تب Keystore کلیک کنید.
۵. بر روی دکمه change در بخش Keystore کلیک کرده و “Custom Identity and Trust” را انتخاب کنید.
۶. در زیرمجموعه Custom Identity Keystore مسیر کامل شناسه keystore (که با نام hostname -f _identity.jck تولید شد) را مشخص کنید.
۷. نوع Custom Identity Keystore را در زیرمجموعه JCEKS مشخص کنید.

۸. کلمه‌ی عبور (که در هنگام تولید کلید مورد استفاده قرار گرفت) را برای شناسه keystore در زیرمجموعه Custom Identity Keystore Passphrase مشخص کنید.

۹. همان‌گونه که شناسه keystore را تنظیم کردید، Custom Trust Keystore (که با نام hostname -f_trust.jck تولید شد) و نوع Custom Trust Keystore و عبارت عبور را نیز تنظیم کنید.

۱۰. تغییرات را ذخیره کنید.

شکل زیر، محیط مربوطه را نشان می‌دهد.

Keystores ensure the secure storage and management of private keys and trusted certificate authorities (CAs). This page lets you view and define various keystore configurations.

Keystores:	Custom Identity and Custom Trust Change
Identity	
Custom Identity Keystore:	<input type="text" value="/u01/oracle/wlsdomains/base"/>
Custom Identity Keystore Type:	<input type="text" value="JCEKS"/>
Custom Identity Keystore Passphrase:	<input type="password" value="....."/>
Confirm Custom Identity Keystore Passphrase:	<input type="password" value="....."/>
Trust	
Custom Trust Keystore:	<input type="text" value="/u01/oracle/wlsdomains/base"/>
Custom Trust Keystore Type:	<input type="text" value="JCEKS"/>
Custom Trust Keystore Passphrase:	<input type="password" value="....."/>
Confirm Custom Trust Keystore Passphrase:	<input type="password" value="....."/>
<input type="button" value="Save"/>	

شکل ۲: تنظیم keystoreها

مرحله ۶: WebLogic SSL را تنظیم کنید.

بعد از پیکربندی مربوطه به keystore، ما می‌توانیم SSL را برای WebLogic تنظیم کنیم.

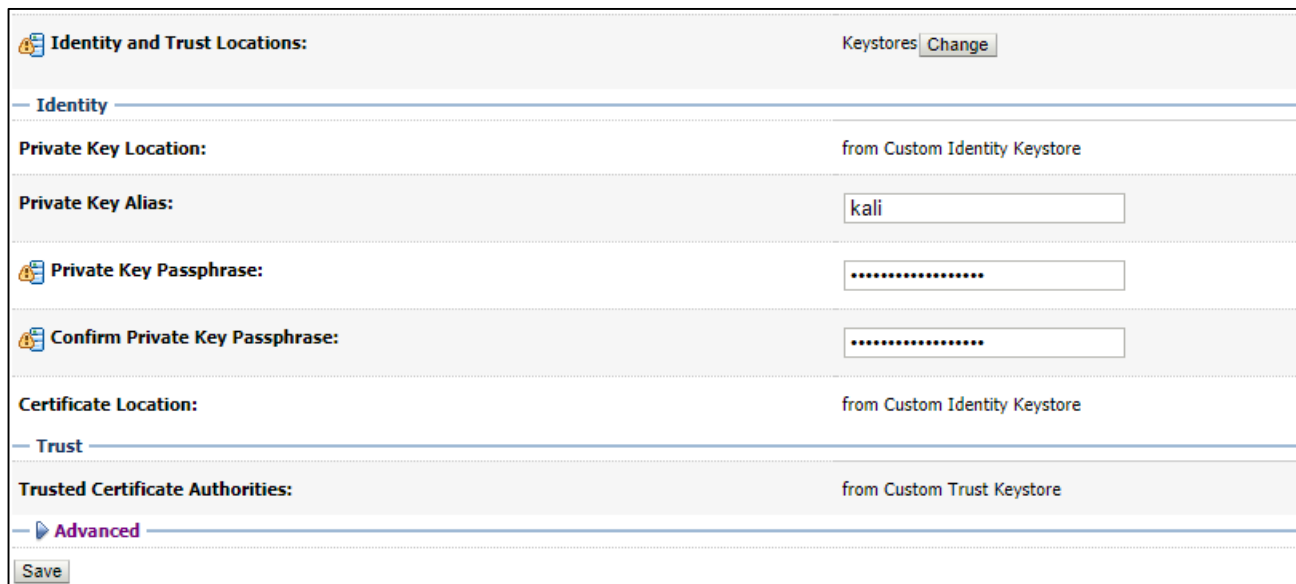
۱. بر روی تب SSL کلیک کنید.

۲. Private Key Alias را مشخص کنید. از نام مستعار (قرار گرفته بعد از پارامتر -alias) استفاده کنید که هنگام ساخت کلید خصوصی و شناسه keystore، تعیین کرده‌اید.

۳. کلمه‌ی عبور کلید خصوصی را مشخص کنید.

۴. بر روی Save کلیک کنید.

عکس زیر پی‌کربندی‌های مربوطه را نشان می‌دهد.



شکل ۳: تنظیمات SSL

۵. سپس بر روی گزینه advance کلیک کنید.

۶. گزینه مربوط به Use JSSE SSL را فعال کنید (برای نسخه‌های 10g و 11g).

۷. در انتها گزینه save را بزنید.

مرحله ۷: فعال کردن debugging بر روی SSL (اختیاری)

بعد از فعال کردن SSL بر روی Weblogic ممکن است به دلایل مختلف مشکلاتی ایجاد شود، که فعال کردن debugging می‌تواند به شناسایی مشکلات کمک کند. جهت فعال کردن آن باید مراحل زیر را انجام داد:

۱. در ابتدا باید با استفاده از یک ویرایشگر اسکریپت setDomainEnv.sh قرار گرفته در مسیر

\$DOMAIN_HOME/bin/ باز شود.

۲. سپس خط زیر باید برای پارامتر JAVA_OPTIONS اضافه گردد.

```
-Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true
```

```
JAVA_OPTIONS="${JAVA_OPTIONS} ${JAVA_PROPERTIES} -Djava.net.debug=all -Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true -Djava.net.debug=all -Dssl.debug=true -Dweblogic.StdoutDebugEnabled=true -Dwlw.iterativeDev=${iterativeDevFlag} -Dwlw.testConsole=${testConsoleFlag} -Dwlw.logErrorsToConsole=${logErrorsToConsoleFlag}"
export JAVA_OPTIONS
```

شکل ۴: نحوه فعال سازی debugging

۴ موارد پیشنهادی برای ارتقای امنیت

در این بخش چگونگی پیکربندی امن SSL/TLS را در سرویس‌دهنده وب Weblogic بیان می‌کنیم. مواردی همچون استثنا کردن برخی الگوریتم‌های رمز به منظور کاهش حملاتی شبیه به CRIME، FREAK و LogJAM، غیرفعال‌سازی نسخه‌های ناامن SSL، برقرار کردن رمزنگاری‌های قوی که از Forward (FS) Secrecy پشتیبانی می‌کنند را بیان می‌کنیم.

نکته: ذکر این نکته لازم است که بسیاری از تنظیمات مانند cipher order و HPKP و ... توسط WebLogic پشتیبانی نمی‌شود.

۴-۱ تنظیم الگوریتم‌های قدرتمند و Forward secrecy

SSLv2 و SSLv3 (به خاطر حمله POODLE) ناامن هستند و باید غیرفعال شوند. برای غیرفعال‌سازی SSL، خط زیر را در فایل setDomainEnv.sh قرار گرفته در مسیر \$DOMAIN_HOME/bin/ اضافه کنید. پارامتر JAVA_OPTIONS را پیدا کرده و خط زیر را به آن اضافه کنید.

```
-Dweblogic.security.SSL.protocolVersion=TLSv1
```

خط بالا به Weblogic می‌گوید که SSL را غیرفعال کرده و فقط از TLS 1.0، TLS 1.1 یا TLS 1.2 استفاده کند. شما همچنین می‌توانید با اضافه کردن دستور زیر به خط JAVA_OPTIONS، حداقل نسخه قابل پشتیبانی را مشخص کنید:

```
weblogic.security.SSL.minimumProtocolVersion=[protocol]
```

که [protocol] می‌تواند شامل موارد زیر باشد:

- TLSv1
- TLSv1.1
- TLSv1.2

همچنین با اضافه کردن پارامترهای زیر به JAVA_OPTIONS، نیز می‌توان امکان استفاده از cipherهای ضعیف را غیرفعال کرد.

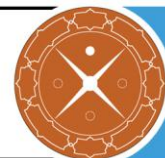
```
-Dweblogic.security.SSL.allowUnencryptedNullCipher=false  
-Dweblogic.security.disableNullCipher=true
```

علاوه بر موارد ذکر شده، باید Forward Secrecy نیز فعال شود. Forward Secrecy اطمینان می‌دهد که صحت^۱ یک کلید جلسه^۲ حتی وقتی که کلیدهای زیادی مورد مخاطره قرار گرفتند، حفظ می‌شود. FS کامل^۳

^۱ Integrity

^۲ Session Key

^۳ Perfect Forward Secrecy



این مورد را با استخراج یک کلید جدید برای هر جلسه، به انجام می‌رساند. این بدان معناست که زمانی که کلید خصوصی به مخاطره افتاد، نمی‌تواند برای رمزگشایی ترافیک SSL مورد استفاده قرار گیرد. جهت تنظیم cipher Suiteهای قدرتمند و Forward secrecy می‌توان از دستورات زیر استفاده کرد. خط‌های زیر را در \$DOMAIN_HOME/config/config.xml وارد کنید. برای پشتیبانی از AES 256، نیاز دارید تا بسته JCE را برای JDK نصب کنید.

```
<server>
<name>AdminServer</name>
<ssl>
<enabled>true</enabled>

<ciphersuite>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_DHE_DSS_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_DHE_DSS_WITH_AES_256_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_DHE_DSS_WITH_AES_128_CBC_SHA</ciphersuite>
<ciphersuite>TLS_DHE_DSS_WITH_AES_128_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_RSA_WITH_AES_256_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_RSA_WITH_AES_128_CBC_SHA</ciphersuite>
<ciphersuite>TLS_RSA_WITH_AES_128_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384</ciphersuite>
<ciphersuite>TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384</ciphersuite>
<ciphersuite>TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDH_RSA_WITH_AES_256_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDH_RSA_WITH_AES_128_CBC_SHA</ciphersuite>
<ciphersuite>TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256</ciphersuite>
<ciphersuite>TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256</ciphersuite>
</hostname-verifier xsi:nil="true"></hostname-verifier>
```

بعد از پیکربندی با استفاده از اسکریپت در دسترس در آدرس زیر، می‌توان بررسی کرد که چه مرورگرهایی امکان ارتباط با سرور را خواهند داشت.

<https://testssl.sh/testssl.sh>

۲-۴ به روزرسانی نرم‌افزارها و نسخه‌ها

یکی از توصیه‌های مهم در زمینه‌ی پیکربندی امن SSL/TLS به روز بودن نسخه‌ی وب سرور و نصب آخرین وصله‌های امنیتی بر روی آن است.

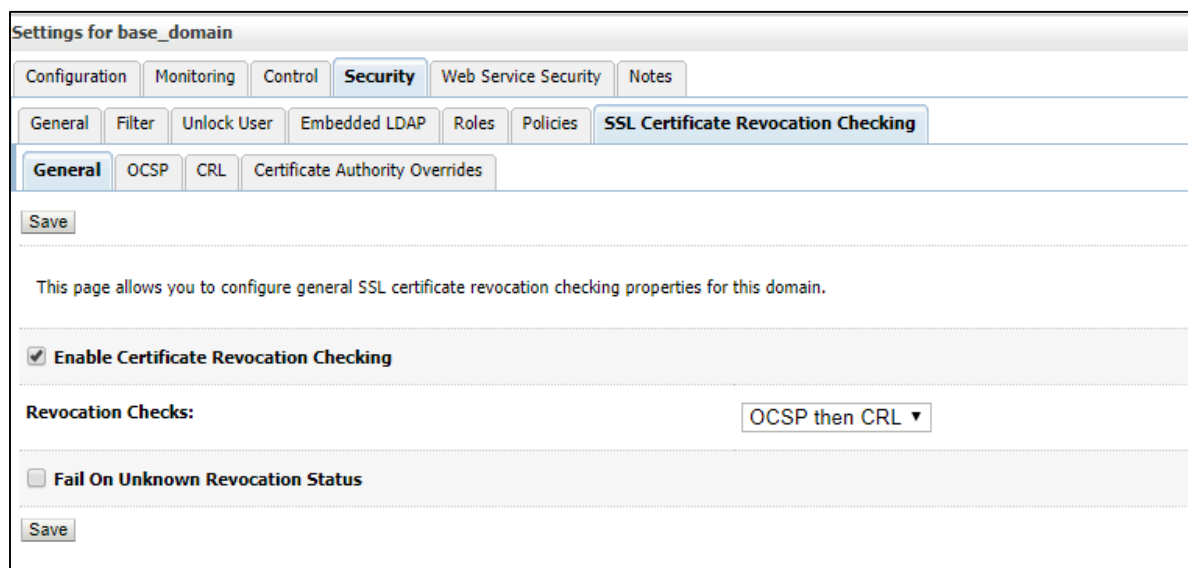


فعال کردن OCSP Stapling ۳-۴

OCSP Stapling روشی برای بالا بردن سرعت در چک کردن لیست ابطال کلید برای گواهی است. با استفاده از OCSP Stapling نیاز نیست که سرویس گیرنده درخواستی را به سرور OCSP بدهد و با استفاده از اطلاعات مهیا شده همراه گواهی، می تواند از باطل نبودن گواهی اطمینان حاصل کند.

برای فعال کردن آن بر روی وب سرور می توان مراحل زیر را انجام داد:

- در سمت چپ پنل مدیریتی بر در بخش Domain Structure بر روی نام دامنه کلیک کنید.
- سپس از مسیر Security > SSL Certificate Revocation Checking > General وارد شده و گزینه Enable Certificate Revocation Checking را فعال کنید.
- در بخش Revocation Checks نیز می تواند روش certificate revocation checking را انتخاب کنید.
- سپس بر روی گزینه save کلیک کنید.



Settings for base_domain

Configuration Monitoring Control **Security** Web Service Security Notes

General Filter Unlock User Embedded LDAP Roles Policies **SSL Certificate Revocation Checking**

General OCSP CRL Certificate Authority Overrides

Save

This page allows you to configure general SSL certificate revocation checking properties for this domain.

Enable Certificate Revocation Checking

Revocation Checks: OCSP then CRL ▼

Fail On Unknown Revocation Status

Save

شکل ۵: تنظیم OCSP

فعال کردن HSTS ۴-۴

HTTP Strict Transport Security یک بهبود امنیتی برای برنامه های تحت وبی است که از پروتکل HTTPS استفاده می کنند. وجود این مکانیسم باعث جلوگیری از Downgrade Attack و Cookie Hijacking می شود. این قابلیت همچنین مرورگر را ملزم می کند که حتماً از پروتکل HTTPS برای ارتباط با سرور استفاده کند.

پیکربندی مشخصی برای فعال سازی HSTS در weblogic وجود ندارد (مانند فعال کردن یک گزینه در کنسول مدیریتی) و برای این کار نیاز است که یک فیلتر ایجاد و به web application مورد نظر اضافه گردد. در ابتدا

باید خطوط زیر به فایل web.xml قرار گرفته در مسیر \$MW_HOME/wlserver/server/lib/consoleapp/webapp/WEB-INF/web.xml (البته مسیر ممکن است در برخی از موارد به علت انتخاب نام متفاوت تفاوت جزئی داشته باشد) اضافه گردد:

```
<filter>
  <filter-name>HSTSFilter</filter-name>
  <filter-class>security.HSTSFilter</filter-class>
</filter>
```

برای ایجاد فیلتر بالا (security.HSTSFilter) باید به صورت زیر عمل کرده و سپس آن را به web application اضافه کرد:

```
package security;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletResponse;

public class HSTSFilter implements Filter {

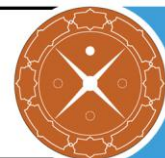
    public void doFilter(ServletRequest req, ServletResponse res,
        FilterChain chain) throws IOException, ServletException {
        HttpServletResponse resp = (HttpServletResponse) res;

        if (req.isSecure())
            resp.setHeader("Strict-Transport-Security", "max-age=31622400; includeSubDomains");

        chain.doFilter(req, resp);
    }
}
```

همچنین برای هدایت خودکار ارتباطات HTTP به HTTP می‌توانید مقدار 'transport-guarantee' را به صورت CONFIDENTIAL یا INTEGRAL در web.xml قرار دهید که weblogic، کلاینت‌هایی که از HTTP استفاده می‌کنند را به صورت خودکار به سمت HTTPS مسيردهی می‌کند. خطوط زیر باید در فایل web.xml درون تگ security-constraint قرار داده شود.

```
<user-data-constraint>
  <description>
    This is how the user data must be transmitted.
  </description>
  <transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
```



۵ مراجع

- [1]. <https://docs.oracle.com/middleware/1221/wls/LOCKD/practices.htm#LOCKD124>
- [2]. https://docs.oracle.com/cd/E13222_01/wls/docs81/webserv/security.html
- [3]. <http://www.oracle.com/technetwork/articles/soa/patil-certrevoc-1873528.html>
- [4]. https://docs.oracle.com/cd/E24329_01/web.1211/e24422/ssl.htm#SECMG384
- [5]. <https://stackoverflow.com/questions/38971541/how-can-we-enable-hstshttp-strict-transport-security-in-weblogic-server>
- [6]. <https://serverfault.com/questions/693876/how-can-the-hsts-header-be-added-to-weblogic>
- [7]. <http://docs.oracle.com/javase/6/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider>
- [8]. <http://weblogic-wonders.com/weblogic/2014/06/24/recommended-best-practices-securing-weblogic-server/>
- [9]. <http://remotepsadmins.com/2015/01/24/ssl-weblogic/>

