راه‌اندازی و پیکربندی امن پروتکل SSL/TLS بر روی سرویس‌دهنده وب Tomcat نسخه ۶ و ۷

شماره مستند APA-AMIRKABIR-13950704-1

تاریخ نگارش ..... ۴ مهر ۱۳۹۵

شماره نگارش ۵/۰

نگارش ..... آپای امیرکبیر

طبقه‌بندی ..... عادی

فهرست مطالب

۱ مقدمه

برای تأمین محرمانگی و جامعیت داده‌های مبادله شده می‌توان از پروتکل‌های استانداردی که بدین منظور طراحی شده استفاده کرد. در حال حاضر مهم‌ترین پروتکل رمزنگاری که در سطح اینترنت برای رمزنگاری داده‌های لایه کاربرد و تأمین امنیت ارتباطات استفاده می‌شود، پروتکل SSL/TLS است. در این گزارش مراحل نصب و ایمن‌سازی پروتکل SSL/TLS بر روی سرویس‌دهنده وب Tomcat نسخه ۶,۰,۴۵ و ۷,۰,۷۲ ارائه شده است.

۲ پیکربندی Tomcat برای ارتباطات HTTPS

۲–۱ آماده سازی keystore

Tomcat در حال حاضر تنها روی فرمت‌های JKS، PKCS12 یا PKCS12 از keystore عمل می‌کند. JKS فرمت "Java KeyStore" استاندارد جاوا است و توسط خط فرمان keytool ایجاد شده است. فرمت PKCS12 بر مبنای استاندارد اینترنت است و می‌تواند از طریق OpenSSL و Key-Manager مایکروسافت دستکاری شود.

کلیدهایی که برای تراکنش‌های SSL در Tomcat استفاده خواهند شد در داخل فایلی به نام "keystore" که توسط رمز عبور محافظت خواهند شد، ذخیره می‌شوند. گام اول برای فعال‌سازی SSL روی سرور این است که این فایل را بسازید و ویرایش کنید. شما می‌توانید این فایل را در دو راه مختلف ایجاد کنید:

- وارد کردن کلید موجود به داخل keystore
- یا ایجاد کردن کلیدهای جدید

برای وارد کردن یک گواهی موجود به داخل keystore با فرمت JKS، لطفا مستندات مربوطه (در مستندات JDK شما) درباره keytool را بخوانید. توجه کنید که OpenSSL اغلب توضیحات قابل خواندنی را قبل از کلید اضافه می‌کند اما keytool آن را پشتیبانی نمی‌کند. همچنین اگر گواهی شما دارای توضیحاتی قبل از داده کلید است، قبل از اینکه گواهی را به keytool وارد کنید، آن توضیحات را پاک کنید.

برای وارد کردن گواهی موجود که توسط CA شما امضا شده است به داخل keystore با فرمت PKCS12 با استفاده از OpenSSL، شما باید دستوری مانند زیر را وارد کنید:

Tomcat 6.0.45&7.0.72:

```
openssl pkcs12 -export -in mycert.crt -inkey mykey.key
                       -out mycert.p12 -name tomcat -CAfile myCA.crt
                       -caname root -chain
```

برای ایجاد یک JKS جدید حاوی گواهی خود-امضا[1]، فرمان‌های زیر را در خط فرمان اجرا کنید:

Windows-Tomcat 6.0.45&7.0.72:

```
"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA
```

Unix-Tomcat 6.0.45&7.0.72:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
```

این فرمان یک فایل جدید را با نام ".keystore" در شاخه اصلی کاربر ایجاد خواهد کرد. برای معین کردن مکان یا نام دلخواه، پارامتر keystore– را به همراه مسیر کامل در ادامه آن اضافه کنید. شما همچنین باید این مکان جدید را در فایل پیکربندی server.xml وارد کنید. برای مثال:

Windows-Tomcat 6.0.45&7.0.72:

[1] self-signed

```
"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA
  -keystore \path\to\my\keystore
```

Unix-Tomcat 6.0.45&7.0.72:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
  -keystore /path/to/my/keystore
```

بعد از اجرای این دستور، از شما خواسته می‌شود تا رمز عبور مورد نظر را وارد کنید. با وجود اینکه رمز عبور پیش فرضی که در Tomcat استفاده می‌شود، "changeit" است، شما می‌توانید رمز عبور مورد نظر خود را قرار دهید. شما همچنین باید یک رمز عبور برای فایل پیکربندی server.xml قرار دهید.

در ادامه اطلاعات عمومی درباره این گواهی از شما خواسته خواهد شد. این اطلاعات به کاربرانی که صفحه (امن) شما را مشاهده می‌کنند نشان داده خواهد شد تا اطلاعات بیان شده در اینجا با چیزی که آنها انتظار دارند تطابق داشته باشد.

در نهایت از شما خواسته خواهد شد تا رمز عبور مخصوص این گواهی را وارد کنید (مخالف با دیگر گواهی‌هایی که در keystore یکسان قرار دارند). شما باید از رمز عبور یکسانی که در خود keystore مورد استفاده قرار گرفت، استفاده کنید. این یک محدودیت در پیاده‌سازی Tomcat است.

اگر همه چیز موفقیت‌آمیز باشد، شما حالا یک فایل keystore با یک گواهی دارید که می‌تواند به وسیله سرور شما مورد استفاده قرار بگیرد.

توجه: رمز عبور کلید خصوصی و keystore باید یکسان باشد. اگر آنها با هم فرق داشته باشند شما یک پیغام خطا در امتداد خط java.io.IOException: Cannot recover key دریافت خواهید کرد.

۲–۲ ویرایش فایل پیکربندی Tomcat

Tomcat می‌تواند از دو پیاده‌سازی مختلف SSL استفاده کند:

- پیاده‌سازی JSSE، که به عنوان قسمتی از Java runtime (از ۱,۴) ارائه شده است.
- پیاده‌سازی APR، که از موتور OpenSSL به صورت پیش‌فرض استفاده می‌کند.

جزییات دقیق پیکربندی، وابسته به نوع پیاده‌سازی مورد استفاده است. اگر شما پیکربندی اتصال دهنده[۱] را توسط protocol="HTTP/1.1" انجام داده‌اید، پس پیاده‌سازی مورد استفاده به وسیله Tomcat، به صورت خودکار انتخاب می‌شود.

برای تعریف یک اتصال دهنده جاوا (JSSE) ، بدون در نظر گرفتن اینکه کتابخانه APR بارگزاری شده یا نه، از یکی از موارد زیر استفاده می‌کنیم.

Tomcat 6.0.45:

[۱] Connector

مرکز پژوهشی آپا (آگاهی‌رسانی، پشتیبانی، امداد برای آسیب‌پذیری‌ها و حوادث امنیتی سایبری)
تهران - بالاتر از چهارراه ولیعصر - نبش کوچه بالاور – ساختمان معاونت پژوهشی دانشگاه صنعتی امیرکبیر - طبقه سوم
کد پستی: ۱۵۹۱۶۳۴۳۱۱ تلفکس: ۶۶۴۶۰۳۰۸ Web: https://apa.aut.ac.ir - Email: autcert@aut.ac.ir

```
<!-- Define a HTTP/1.1 Connector on port 8443, JSSE BIO implementation
-->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
           port="8443" .../>

<!-- Define a HTTP/1.1 Connector on port 8443, JSSE NIO implementation
-->
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" .../>
```

Tomcat 7.0.72:

```
<!-- Define a HTTP/1.1 Connector on port 8443, JSSE NIO implementation
-->
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" .../>

<!-- Define a HTTP/1.1 Connector on port 8443, JSSE BIO implementation
-->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
           port="8443" .../>
```

متناوبا برای مشخص کردن یک اتصال دهنده APR (کتابخانه APR باید در دسترس باشد) از دستور زیر استفاده می‌کنیم:

Tomcat 6.0.45&7.0.72:

```
<!-- Define a HTTP/1.1 Connector on port 8443, APR implementation
-->
<Connector protocol="org.apache.coyote.http11.Http11AprProtocol"
           port="8443" .../>
```

اگر شما در حال استفاده از APR هستید، در واقع شما گزینه پیکربندی یک موتور جایگزین برای OpenSSL را دارید.

Tomcat 6.0.45&7.0.72:

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
          SSLEngine="someengine" SSLRandomSeed="somedevice" />
```

مقدار پیش فرض به صورت زیر است:

Tomcat 6.0.45&7.0.72:

```
<Listener
className="org.apache.catalina.core.AprLifecycleListener"
          SSLEngine="on" SSLRandomSeed="builtin" />
```

همچنین برای استفاده از SSL تحت APR، اطمینان حاصل کنید که خصیصه SSLEngine برابر off نباشد. مقدار پیش‌فرض برای این خصیصه on است و اگر شما مقدار دیگری را برای آن در نظر گرفتید باید از اعتبار آن اطمینان حاصل کنید.

مرکز پژوهشی آپا (آگاهی‌رسانی، پشتیبانی، امداد برای آسیب‌پذیری‌ها و حوادث امنیتی سایبری)
تهران - بالاتر از چهارراه ولیعصر - نبش کوچه بالاور – ساختمان معاونت پژوهشی دانشگاه صنعتی امیرکبیر - طبقه سوم
کد پستی: ۱۵۹۱۶۳۴۳۱۱ تلفکس: ۶۶۴۶۰۳۰۸ Web: https://apa.aut.ac.ir - Email: autcert@aut.ac.ir

گام آخر، پیکربندی اتصال دهنده در فایل $CATALINA_BASE/conf/server.xml است که $CATALINA_BASE بیان کننده لغت‌نامه پایه برای Tomcat 6 است. یک مثال برای عنصر <Connector> برای یک SSL connector، در فایل پیش فرض server.xml که همراه با Tomcat نصب شده است، قرار دارد.

Tomcat 6.0.45:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
           protocol="org.apache.coyote.http11.Http11Protocol"
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="${user.home}/.keystore"
keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS"/>
```

Tomcat 7.0.72:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="${user.home}/.keystore"
keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS"/>
```

APR connector از صفت‌های متفاوتی برای تنظیمات SSL (مخصوصا کلیدها و گواهی‌ها) استفاده می‌کند. مثالی از پیکربندی یک APR در زیر آمده است.

Tomcat 6.0.45&7.0.72:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
           protocol="org.apache.coyote.http11.Http11AprProtocol"
           port="8443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           SSLCertificateFile="/usr/local/ssl/server.crt"
           SSLCertificateKeyFile="/usr/local/ssl/server.pem"
           SSLVerifyClient="optional"
SSLProtocol="TLSv1+TLSv1.1+TLSv1.2"/>
```

یک پورت TCP/IP روی Tomcat برای ارتباطات از منابع گوش فرا می‌دهد. شما می‌توانید شماره پورت مورد نظر خودتان را قرار دهید (مانند پورت ۴۴۳ که پورت پیش فرض برای ارتباطات https است). اگرچه، برای اینکه Tomcat بتواند روی پورت‌های کمتر از ۱۰۲۴ کار کند، تنظیمات خاطی نیاز است که خارج از مباحث این گزارش است. اگر شماره پورت را در اینجا تغییر داده‌اید، باید مقدار صفت redirectPort را هم روی non-SSL connector تغییر دهید و بعد از تکمیل این تغییرات، شما باید Tomcat را راه‌اندازی مجدد کنید.

مرکز پژوهشی آپا (آگاهی‌رسانی، پشتیبانی، امداد برای آسیب‌پذیری‌ها و حوادث امنیتی سایبری)
تهران - بالاتر از چهارراه ولیعصر - نبش کوچه بالاور – ساختمان معاونت پژوهشی دانشگاه صنعتی امیرکبیر - طبقه سوم
کد پستی: ۱۵۹۱۶۳۴۳۱۱ تلفکس: ۶۶۴۶۰۳۰۸ Web: https://apa.aut.ac.ir - Email: autcert@aut.ac.ir

۲–۳ نصب یک گواهی از مرکز صدور گواهی

برای بدست آوردن و نصب گواهی از یک مرکز صدور گواهی(CA)[۱] باید مراحل زیر را انجام دهید و برای اطلاعات بیشتر در این باره می‌توانید به گزارش ارائه شده توسط پژوهشکده آپای دانشگاه صنعتی امیرکبیر که در آدرس زیر موجود است مراجعه کنید:

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

۲–۴ تولید یک درخواست امضا گواهی

به منظور بدست آوردن یک گواهی از مرکز صدور گواهی، شما باید یک درخواست امضا گواهی(CSR)[۲] بسازید که برای تولید گواهی در CA استفاده خواهد شد. برای ساخت یک CSR گام‌های زیر را انجام دهید:

- ساخت یک گواهی خود-امضا[۳]

Tomcat 6.0.45&7.0.72:

```
keytool -genkey -alias tomcat -keyalg RSA
   -keystore <your_keystore_filename>
```

- ساخت CSR

Tomcat 6.0.45&7.0.72:

```
keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr
    -keystore <your_keystore_filename>
```

هم اکنون شما یک فایل به نام certreq.csr خواهید داشت که می‌توانید آن را به مرکز صدور گواهی ارسال و گواهی خود را دریافت کنید.

۲–۵ نصب گواهی

بعد از اینکه گواهی خود را از CA دریافت کردید، شما می‌توانید آن را در keystore محلی قرار دهید. اول از همه، شما باید یک زنجیره گواهی یا گواهی ریشه را در keystore قرار دهید و سپس گواهی خود را قرار دهید.

- زنجیره گواهی را از CA خود دریافت کنید.
- زنجیره گواهی را در keystore خود قرار دهید.

Tomcat 6.0.45&7.0.72:

```
keytool -import -alias root -keystore <your_keystore_filename>
    -trustcacerts -file <filename_of_the_chain_certificate>
```

- در نهایت گواهی جدید خود را وارد کنید.

---

[۱] Certificate Authority
[۲] Certificate Signing Request
[۳] Self-signed Certificate

مرکز پژوهشی آپا (آگاهی‌رسانی، پشتیبانی، امداد برای آسیب‌پذیری‌ها و حوادث امنیتی سایبری)
تهران - بالاتر از چهارراه ولیعصر - نبش کوچه بالاور – ساختمان معاونت پژوهشی دانشگاه صنعتی امیرکبیر - طبقه سوم
کد پستی: ۱۵۹۱۶۳۴۳۱۱ تلفکس: ۶۶۴۶۰۳۰۸ Web: https://apa.aut.ac.ir - Email: autcert@aut.ac.ir

Tomcat 6.0.45&7.0.72:

```
keytool -import -alias tomcat -keystore <your_keystore_filename>
    -file <your_certificate_filename>
```

۳ امن سازی پروتکل SSL/TLS

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

برای بررسی وضعیت امنیتی پروتکل SSL/TLS سرویس‌دهنده خود، می‌توانید به ابزاری که بدین منظور توسط پژوهشکده آپای دانشگاه صنعتی امیرکبیر طراحی شده و در آدرس زیر قرار دارد، مراجعه کنید.

https://sslcheck.certcc.ir

۳–۱ غیر فعال کردن الگوریتم‌های رمزنگاری ضعیف

اگر شما از Tomcat 5.5 یا Tomcat 6 روی JDK1.6 استفاده می‌کنید، الگوریتم‌های رمزنگاری زیر به صورت پیش‌فرض فعال هستند:

```
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
```

الگوریتم‌های زیر از لیست بالا ضعیف هستند:

```
SSL_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
```

برای غیر فعال کردن الگوریتم‌های رمزنگاری ضعیف باید خط زیر را در قسمت Connector از فایل server.xml قرار دهید (توصیه می‌شود قبل از تغییرات، از این فابل نسخه پشتیبان تهیه کنید) و سپس Tomcat را راه‌اندازی مجدد کنید:

Tomcat 6:

```
ciphers="SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA"
```

توجه: فاصله خالی بین نام الگوریتم‌ها باید حذف شود.

Tomcat 7 with Java7:

```
ciphers="TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384,
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA,
TLS_ECDH_ECDSA_WITH_RC4_128_SHA,
TLS_ECDH_RSA_WITH_RC4_128_SHA,
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_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_EMPTY_RENEGOTIATION_INFO_SCSVF
```

۳-۲ غیر فعال کردن TLSv2 و TLSv3

زمانی که از Tomcat با JSSE connectors استفاده می‌کنید، پروتکل SSL می‌تواند از طریق فایل زیر پیکربندی شود:

$TOMCAT_HOME/conf/server.xml

باید تنظیمات مربوطه را به صورت زیر را در Tomcat 5 و Tomcat 6 (قبل از 6.0.38) انجام دهید:

Tomcat 5 and 6 (prior to 6.0.38)

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
               maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
               clientAuth="false" sslProtocols = "TLSv1,TLSv1.1,TLSv1.2" />
```

Tomcat 6 (6.0.38 and later)&7

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
               maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
               clientAuth="false" sslEnabledProtocols = "TLSv1,TLSv1.1,TLSv1.2"
/>
```

توجه کنید که TLSv1.1 و TLSv1.2 در جاوا ۷ پشتیبانی می‌شوند؛ اگر چه اضافه کردن این دستورات در جاوا۶ مشکلی ایجاد نمی‌کند ولی نمی‌توانید TLSv1.1 و TLSv1.2 را فعال کنید.

مرکز پژوهشی آپا (آگاهی‌رسانی، پشتیبانی، امداد برای آسیب‌پذیری‌ها و حوادث امنیتی سایبری)
تهران - بالاتر از چهارراه ولیعصر - نبش کوچه بالاور – ساختمان معاونت پژوهشی دانشگاه صنعتی امیرکبیر - طبقه سوم
کد پستی: ۱۵۹۱۶۳۴۳۱۱ تلفکس: ۶۶۴۶۰۳۰۸ Web: https://apa.aut.ac.ir - Email: autcert@aut.ac.ir

۴ منابع

1 http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html

2 https://www.mulesoft.com/tcat/tomcat-ssl

3 http://tomcat.apache.org/tomcat-6.0-doc/apr.html

4 https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html

5 https://access.redhat.com/solutions/1232233

6 http://www.fromdev.com/2009/02/tomcat-best-practices-securing-ssl-by.html

7 https://ssl.comodo.com/support/ssl-technical-faqs/how-to-disable-weak-ciphers-in-tomcat-7-8.php

مرکز پژوهشی آپا (آگاهی‌رسانی، پشتیبانی، امداد برای آسیب‌پذیری‌ها و حوادث امنیتی سایبری)
تهران - بالاتر از چهارراه ولیعصر - نبش کوچه بالاور – ساختمان معاونت پژوهشی دانشگاه صنعتی امیرکبیر - طبقه سوم
کد پستی: ۱۵۹۱۶۳۴۳۱۱ تلفکس: ۶۶۴۶۰۳۰۸ Web: https://apa.aut.ac.ir - Email: autcert@aut.ac.ir