
# มาตรฐานการพัฒนาระบบงาน เพื่อความมั่นคงปลอดภัยของ กทพ. เวอร์ชั่น ๑.๑

ฝ่ายสารสนเทศ

ธันวาคม ๒๕๖๓

# สารบัญ

# คำนำ

โครงการพัฒนากระบวนการด้านความมั่นคงปลอดภัยในการพัฒนาระบบงาน ได้กำหนดไว้ในแผนพัฒนาดิจิทัล เพื่อระบบเทคโนโลยีสารสนเทศของ กทพ. มีงบประมาณ ๒๕๖๐ - ๒๕๖๔ EXAT-ICT 4.0 อยู่ในยุทธศาสตร์ที่ ๖ การสร้างความเชื่อมั่นในการใช้งานเทคโนโลยีดิจิทัล กลยุทธ์ (๑) พัฒนาระบบเทคโนโลยีสารสนเทศและการสื่อสารให้มีประสิทธิภาพ เพียงพอ และมีความมั่นคงปลอดภัย โดยมีวัตถุประสงค์เพื่อรองรับการให้บริการและเพื่อใช้ในการดำเนินงานของ กทพ. และในปัจจุบันภัยคุกคามด้านความมั่นคงปลอดภัยสารสนเทศเกิดจากสาเหตุหลักคือการขาดการพัฒนาความมั่นคงปลอดภัยในการพัฒนาระบบสารสนเทศ ซึ่งเป็นสิ่งสำคัญที่จะทำให้เกิดกิจกรรมด้านความมั่นคงปลอดภัยขึ้นในกระบวนการพัฒนาระบบสารสนเทศ และจะทำให้ลดจำนวนช่องโหว่ และความเสี่ยงจากการนำเทคโนโลยีสารสนเทศไปใช้งานจริง ทั้งนี้ยังทำให้บุคลากรของ กทพ. ในส่วนงานพัฒนาระบบสารสนเทศมีองค์ความรู้ และความเข้าใจในกระบวนการพัฒนาระบบงานให้มีความมั่นคงปลอดภัยอย่างเป็นมาตรฐาน

การจัดทำมาตรฐานการพัฒนาระบบงานเพื่อความมั่นคงปลอดภัยของ กทพ. เพื่อเป็นมาตรฐานในการพัฒนาระบบและใช้เป็นแนวทางการตรวจสอบประสิทธิภาพของโปรแกรม (Source Code) โดยอ้างอิงจากมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ (Web Application Security Standard) เวอร์ชั่น ๑.๐ ของสำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) กระทรวงดิจิทัลเพื่อเศรษฐกิจและสังคม

คณะผู้จัดทำ กองระบบงานคอมพิวเตอร์

กันยายน ๒๕๖๒

## ๑. ขอบข่ายและการนำไปใช้

ในปัจจุบันหน่วยงานต่าง ๆ ใช้เว็บไซต์เป็นเครื่องมือในการเผยแพร่ประชาสัมพันธ์ข่าวสารและติดต่อกับลูกค้าเพื่อดำเนินกิจกรรมทางธุรกิจ ด้วยเหตุนี้เว็บไซต์จึงเป็นเป้าหมายหนึ่งที่ถูกคุกคามด้านสารสนเทศมากที่สุด โดยเฉพาะเว็บไซต์ที่มีชื่อเสียง หรือเว็บไซต์ที่มีช่องโหว่ มักมีความเสี่ยงที่จะตกเป็นเป้าหมายการโจมตีจากผู้ประสงค์ร้ายอยู่เสมอ โดยอาจเป็นการโจมตีเพื่อเปลี่ยนแปลงข้อมูลหน้าเว็บ (Web Defacement) การโจมตีเพื่อขโมยข้อมูลสำคัญการโจมตีเพื่อใช้เป็นฐานในการเผยแพร่มัลแวร์ (Malware URL) หรือใช้เป็นฐานในการฉ้อโกงทางการเงินผ่านหน้าเว็บไซต์หลอกลวง (Phishing Website) ภัยคุกคามเหล่านี้ล้วนก่อให้เกิดผลกระทบต่อความน่าเชื่อถือของเว็บไซต์การโจมตีเว็บไซต์ดังที่กล่าวมาแล้ว ล้วนอาศัยช่องโหว่ของโปรแกรมประยุกต์บนเว็บ (Web applications) เป็นส่วนมาก

มาตรฐานการพัฒนาระบบงานเพื่อความมั่นคงปลอดภัยของ กทพ. ฉบับนี้ เป็นแนวทางสำหรับการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บเพื่อให้มีความมั่นคงปลอดภัย รวมถึงการเสนอแนะแนวทางเพื่อป้องกันการโจมตีและแก้ไขช่องโหว่ที่เกี่ยวข้อง โดยขอบเขตของข้อเสนอแนะฉบับนี้มุ่งเน้นไปที่การรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บด้วยการพัฒนา และทดสอบโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัยจากช่องโหว่ดังต่อไปนี้

(๑) SQL Injection

(๒) OS Command Injection

(๓) Unchecked Path Parameter หรือ Directory Traversal

(๔) Improper Session Management

(๕) Cross-site Scripting

(๖) Cross-site Script Request Forgery

(๗) HTTP Header Injection

(๘) Mail Header Injection

(๙) Lack of Authentication and authorization

## ๒. มาตรฐานการพัฒนาระบบงานเพื่อความมั่นคงปลอดภัยของ กทพ.

### ๒.๑ SQL Injection

การแทรกคำสั่ง SQL ผ่านพารามิเตอร์ต่างๆ ของเว็บไซต์ เช่น GET POST เป็นต้น ทำให้สามารถดำเนินการใดๆ ก็ตามกับฐานข้อมูลได้

ปัจจุบันมีฐานข้อมูลประเภท NoSQL ซึ่งไม่ได้ใช้คำสั่ง SQL ในการเข้าถึงข้อมูลในฐานข้อมูล ทำให้ปลอดภัยจากการโจมตีด้วยเทคนิคนี้มากขึ้น

**การป้องกันการโจมตี**

- มีการจัดทำ Prepared Statement และ/หรือ Stored Procedure
- ไม่เขียนคำสั่ง SQL โดยตรงในตัวแปร (Parameter) ที่ส่งโดยตรงไปยังโปรแกรมประยุกต์บนเว็บ

**การลดความเสียหายที่เกิดจากการถูกโจมตี**

- ควบคุมการแสดงผลข้อมูล Error Message
- กำหนดสิทธิขั้นต่ำให้กับบัญชีผู้ใช้ฐานข้อมูล

## ตัวอย่างการทำ Prepared Statement

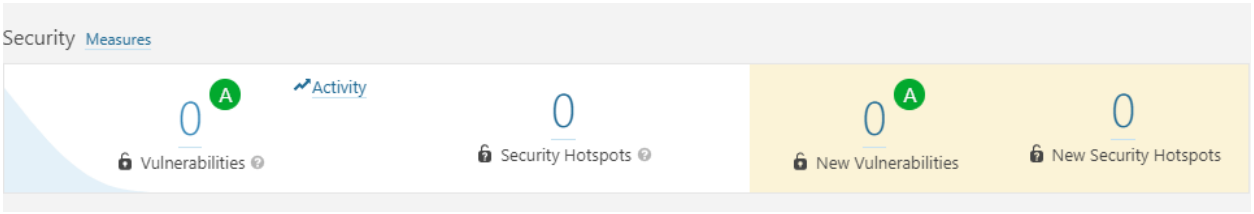
```
//-- Open transaction
        ojConn.openTran();

        //-- New Object StringBuilder
        Oj_sb = new StringBuilder();

        //-- SQL Command
        Oj_sb.append(""INSERT INTO TBL_EMP("");
        Oj_sb.append("" EMP_ID"");
        Oj_sb.append("", EMP_FNAME"");
        Oj_sb.append("", EMP_LNAME"");
        Oj_sb.append("", CREATE_BY"");
        Oj_sb.append("", CREATE_DATE"");
        Oj_sb.append("")VALUES("");
        Oj_sb.append("" ?"");
        Oj_sb.append("", ?"");
        Oj_sb.append("", ?"");
        Oj_sb.append("", ?"");
        Oj_sb.append("", ?"");
        Oj_sb.append(")"");

        //-- prepareCall Statement
        Oj_cstmt = (OracleCallableStatement)ojConn.getConnection().prepareCall(Oj_sb.toString());

        //-- Loop ArrayList<Model>
        for(cls_Emp model:ojList){
           //-- Start Index
           li_index = 1;

           //-- put value
           Oj_cstmt.setString(li_index++, model.getEmp_id());
           Oj_cstmt.setString(li_index++, model.getEmp_fname());
           Oj_cstmt.setString(li_index++, model.getEmp_lname());
           Oj_cstmt.setString(li_index++, model.getCreate_by());
           Oj_cstmt.setTimestamp(li_index++, new Timestamp(today.getTime()));

           //-- Add batch statement
           Oj_cstmt.addBatch();
        }

        //-- Execute
        Oj_cstmt.executeBatch();

        //-- Commit transaction
        if(!ojConn.commitTran()) {

           //-- Get Error message
           gs_message = doGetMessage(""(insert)::"" + ojConn.getErrorMessage());
        }

        //-- Close Statement
        Oj_cstmt.closeOnCompletion();"
```

ตัวอย่างการทำ Stored Procedure

```
create or replace PROCEDURE PROC_INSERT_TBL_ACTIVITY
(
  WORK_ID in CHAR
  , ACTIVITY_NAME in CLOB
  , ACTIVITY_INPUT_DEPCOD in CHAR
  , ACTIVITY_INPUT_EMPID in CHAR
)AS
BEGIN
  INSERT INTO TBL_ACTIVITY(ACTIVITY_ID, WORK_ID, ACTIVITY_NAME, ACTIVITY_INPUT_DEPCOD,
ACTIVITY_INPUT_EMPID)
  VALUES (FUNCTION_GENKEY(SEQ_TBL_ACTIVITY.NEXTVAL, '3'), WORK_ID, ACTIVITY_NAME, ACTIVITY_INPUT_DEPCOD,
ACTIVITY_INPUT_EMPID);
END PROC_INSERT_TBL_ACTIVITY;
```

**การทดสอบความมั่นคงปลอดภัย**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยตัวเอง เช่น ทดลองเพิ่ม “or 1=1” ที่ประโยค where เช่น “Select * from users where id=10 or 1=1”;
- ทดสอบด้วยโปรแกรม SQLmap

## ๒.๒ OS Command Injection

การโจมตีผ่านคำสั่งระดับระบบปฏิบัติการ (OS Command) หรือผ่านพารามิเตอร์ต่างๆ ของเว็บไซต์ เช่น GET POST เป็นต้น เพื่อสั่งดำเนินการใดๆ ผ่านโปรแกรม ซึ่งจะนำไปสู่การรั่วไหลของข้อมูลสำคัญ ทำให้สามารถเข้าควบคุมเครื่องบริการเว็บ หรือใช้เป็นฐานเพื่อโจมตีเครื่องบริการเว็บอื่นๆ

**การป้องกันการโจมตี**

- ปิดการใช้งานคำสั่งต่าง ๆ เพื่อป้องกันการเรียกใช้ที่ไม่พึงประสงค์ เช่น exec(), passthru(), shell_exec(), system(), popen() ในภาษา PHP

**การลดความเสียหายที่เกิดจากการถูกโจมตี**

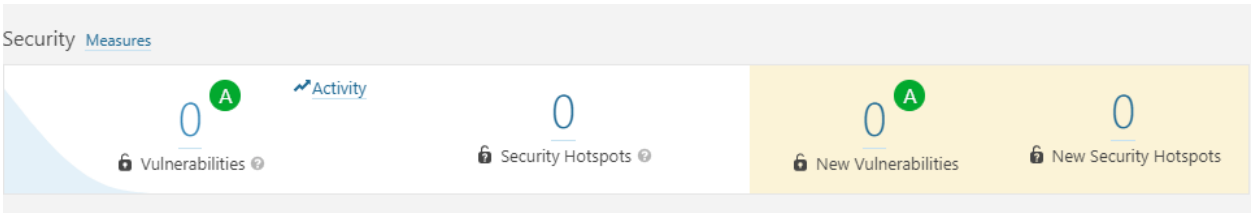
- (ในกรณีที่มีการเรียกใช้คำสั่ง OS Command) ต้องตรวจสอบ Variables ที่จะใช้กับตัวแปรของ OS Command ก่อนนำไปประมวลผล

ตัวอย่างการจัดทำ Validation

```
public String checkTagHTML(String fns_input){
      String s_result = """";
      try{
         s_result = fns_input.replace(""";"", ""&#59;"");
         s_result = s_result.replace(""<"", ""&lt;"");
         s_result = s_result.replace(""">"", ""&gt;"");
         s_result = s_result.replace(""\""""", ""&#92;"");
         s_result = s_result.replace(""'"", ""&#39;"");
         s_result = s_result.replace(""("", ""&#40;"");
         s_result = s_result.replace("")"", ""&#41;"");
      }
      catch(Exception ex)
      {
         this.gs_message = ex.getMessage();
      }
      return s_result;
}
```

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยตัวเอง เช่นในภาษา PHP สามารถใช้ Semicolon (;) เพื่อระบุว่าจบส่วนที่เป็น URL และตามด้วยคำสั่ง OS Command ได้ โดยคำสั่ง ‘%3B’ คือรหัสของ Semicolon ที่เข้ารหัสเรียบร้อยแล้ว
- ทดสอบด้วยโปรแกรม Nikto หรือ SQLmap

## ๒.๓ Unchecked Path Parameter / Directory Traversal

การเรียกชื่อไฟล์หรือข้อมูลที่เก็บบนเครื่องบริการเว็บได้โดยตรง เช่น ใส่คำสั่ง ?file=../../secret.txt ต่อท้าย URL ของเว็บไซต์ ทำให้สามารถเข้าถึงไฟล์ที่อยู่บนเครื่องบริการเว็บได้โดยไม่ได้รับอนุญาต

**<u>การป้องกันการโจมตี</u>**

- ไม่อนุญาตให้ใส่ Filename ที่สร้างจากระบบ ควรมาจากการสุ่ม
- กำหนดที่อยู่ในการเก็บชื่อไฟล์ให้ชัดเจน

**<u>การลดความเสียหายที่เกิดจากการถูกโจมตี</u>**

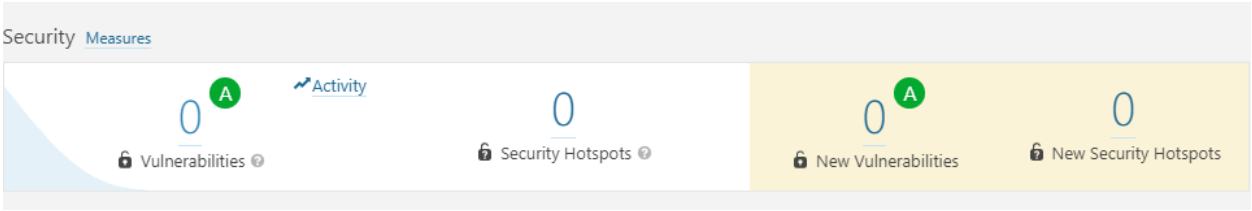
- กำหนด Permission การเข้าถึงไฟล์บนเครื่องบริการเว็บให้เหมาะสม
- ตรวจสอบ Filename เช่น มีการแปลง String ที่ระบุ Directory ได้ เช่น / -> % 2F

ตัวอย่างการตัวอย่างการเปลี่ยนชื่อไฟล์

```
for(FileItem item : multiparts){
		if(!item.isFormField()){
			//-- Fet file name
			file_name = new File(item.getName()).getName();

			if(!file_name.equals("""")){
				//-- Get extension file
				String ext = FilenameUtils.getExtension(file_name);

				//-- Get Max DocNo
				ls_docNo = dao_TRDoc.do_getMaxDocNo(Oj_conn, ls_doc_id);

				//-- Change file name
				//-- 255710001_001.xls
				file_input = ls_doc_id + ""_"" + ls_docNo + "".""  + ext;

				//-- Write file
				item.write( new File(ls_doc_path + ""/"" + file_input));

				//-- New model
				model = new cls_TRDoc(file_name, ext, ls_doc_id, ls_docNo, ls_doc_path, ls_user_id, null);

				//-- Insert to DB
				lb_result = dao_TRDoc.insert(Oj_conn, model);
			}
			if(lb_result) {
				ls_message = ""File Uploaded Successfully"";
			}
			else{
				ls_message = ""File Uploaded Failed"";
			}
		}
}
```

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยตัวเอง เช่น ใส่คำสั่ง ?file=../../secret.txt ต่อท้าย URL ของเว็บไซต์
- ทดสอบด้วยโปรแกรม DirBuster

## ๒.๔ Improper Session Management

การดักขโมย Session ID ของผู้ใช้บริการหรือนำเอา Session ID ไปใช้ในการเข้าเว็บไซต์ด้วยสิทธิของเจ้าของ Session

**<u>การป้องกันการโจมตี</u>**

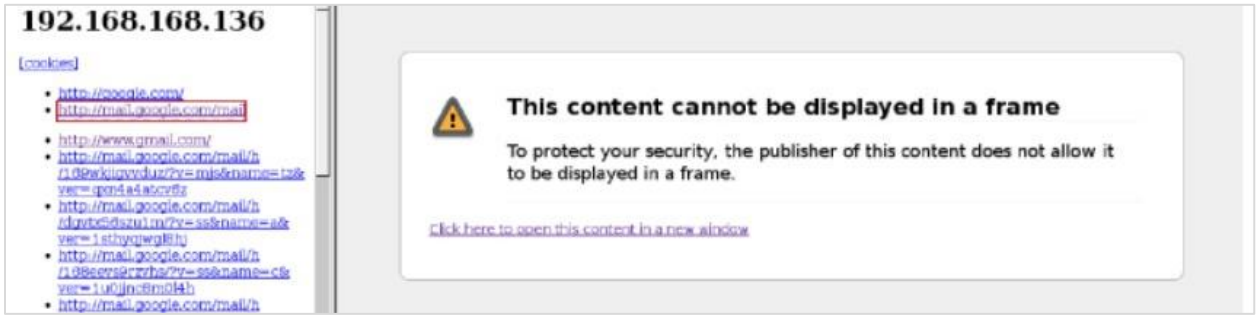
- สร้าง Session ID เป็นที่ยากต่อการคาดเดา (ไม่ใช้ Algorithm ที่ง่ายเกินไป เช่น Pseudo Random Number)
- ไม่ใช้ URL Parameter ในการเก็บ Session ID
- เมื่อมีการใช้งาน HTTPS Protocol ใช้ Secure Attribute ของ Cookies

**<u>การลดความเสียหายที่เกิดจากการถูกโจมตี</u>**

- กำหนดให้ Session ID เป็นค่าสุ่ม
- กำหนดวันหมดอายุการใช้งานของ Cookies ที่เก็บ Session ID

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากโปรแกรม Nikto





- ทดสอบด้วยโปรแกรม Cain & Abel

## ๒.๕ Cross-Site Scripting

การแทรกคำสั่งต่างๆ เช่น JavaScript เข้าไปในเว็บเพจ เมื่อผู้ใช้บริการเรียกหน้าเว็บเพจนั้น ทำให้สามารถแสดงหน้าเว็บเพจปลอมหรือข้อมูลปลอมบนเว็บไซต์ หรือขโมย Cookie และปลอมแปลง Session ID

**<u>การป้องกันการโจมตี</u>**

- ทำ Output Validation เพื่อป้องกันการแสดงผลหรือการประมวลผลที่ไม่เหมาะสมและไม่ตรงตามวัตถุประสงค์
- ทำ HTML Entity Encoding หรือ URL Encoding กับข้อมูลที่จะแสดงผล
- ตรวจสอบ Input Validation ไม่ให้ใช้ HTML Tag ใด ๆ เช่นไม่ Generate Content จาก Tag <script></script>
- ไม่อนุญาตให้มีการเรียก Stylesheets จากเว็บไซต์ที่ไม่ได้ตรวจสอบก่อน
- ตั้งค่า Charset Parameter ของ HTTP Content-Type Header
- โปรแกรมประยุกต์บนเว็บต้องมีการตรวจสอบข้อมูลชุดคำสั่งในเว็บไซต์

**<u>การลดความเสียหายที่เกิดจากการถูกโจมตี</u>**

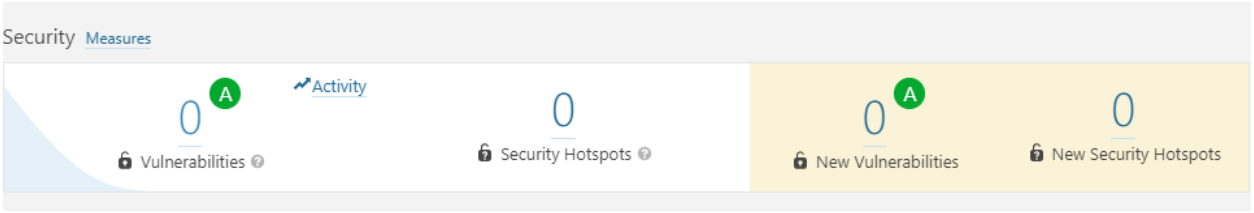
- มีการใช้งาน HTTPOnly Cookie Flag

ตัวอย่างการจัดทำ Validation

```
String emp_id = Oj_util.checkTagHTML(Oj_util.checkNull(request.getParameter("emp_id")));
String emp_fname = Oj_util.checkTagHTML(Oj_util.checkNull(request.getParameter("emp_fname")));
public String checkTagHTML(String fns_input){
    String s_result = "";
    try{
       s_result = fns_input.replace(";", "&#59;");
       s_result = s_result.replace("<", "&lt;");
       s_result = s_result.replace(">", "&gt;");
       s_result = s_result.replace("\"", "&#92;");
       s_result = s_result.replace("'", "&#39;");
       s_result = s_result.replace("(", "&#40;");
       s_result = s_result.replace(")", "&#41;");
    }
    catch(Exception ex)
    {
       this.gs_message = ex.getMessage();
    }
    return s_result;
 }
```

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยตัวเอง โดยทำการตรวจสอบข้อมูลที่รับเข้ามาจากฝั่งผู้ใช้บริการก่อนที่จะนำข้อมูลนั้นไปประมวลผล
- ทดสอบด้วยโปรแกรม OWASP

## ๒.๖ Cross-Site Script Forgery

การปลอมแปลงคำสั่งข้อมูลให้เสมือนเป็นคำสั่งจากผู้ใช้บริการ ทำให้สามารถเข้าถึงและดำเนินการกับบริการต่าง ๆ ที่เข้าถึงได้เฉพาะผู้ใช้บริการที่ Login แล้ว เช่น ผู้ประสงค์ร้ายอาศัยช่องโหว่ของเว็บเพจปลอมแปลงคำสั่งข้อมูลให้เสมือนเป็นคำสั่งจากเจ้าของบัญชีจริงเพื่อติดต่อกับระบบธนาคารทางอินเทอร์เน็ต ทำให้ระบบเชื่อและเข้าใจว่าเจ้าของบัญชีต้องการทำธุรกรรมการเงินนั้นๆ

**<u>การป้องกันการโจมตี</u>**

- ฟังก์ชันต่าง ๆ ควรดำเนินการผ่าน POST Method และตรวจสอบความถูกต้องกับค่าที่ซ่อนอยู่ภายใน POST Method หรือมีการใช้งาน Unique Token ร่วมกับการส่งข้อมูลหรือคำสั่งผ่านแบบฟอร์มก่อนจะดำเนินการต่อ
- มีฟังก์ชันการยืนยันตัวตนของผู้ใช้งานอีกครั้งและให้กรอก Captcha เมื่อมีการเปลี่ยนแปลงสถานะการทำงานในฟังก์ชันที่สำคัญ ๆ

**<u>การลดความเสียหายที่เกิดจากการถูกโจมตี</u>**

- มีการส่งอีเมลอัตโนมัติ แจ้งผู้ใช้บริการทุกครั้งเมื่อการดำเนินการที่สำคัญทำสำเร็จ

ตัวอย่างการสร้างรหัสยืนยันตนเองหลังจากการล็อคอิน

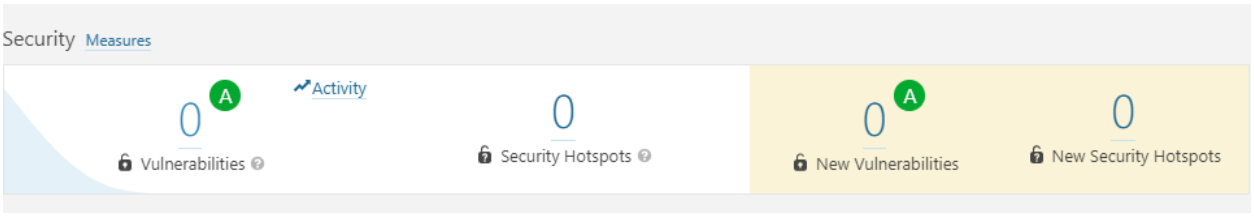
```
//-- Process
Oj_sec = new callSecurityServices_EXAT();
Oj_sec.call_VerifyLogin (ls_u_id, ls_u_pass, ""STD"", ls_clientIP);

//-- Check Result
if (Oj_sec.ResultCode ()==0){
      SimpleDateFormat Oj_sdf = new SimpleDateFormat(""yyyyMMddHHmmss"", Locale.ENGLISH);
      Date date_now = new Date();
      double ld_random = Math.random();
      session.setAttribute (""SES_SELF_"", ls_u_id + Oj_sdf.format(date_now) + Double.toString(ld_random));
}
else{
      if(session.getAttribute (""SES_SELF_"") != null){
         session.removeAttribute(""SES_SELF_"");
      }
      ls_message = Oj_sec.ResultText();
}
```

```
<form role=""form"" id=""form_manage"" name=""form_manage"" method=""post"" action=""javascript:submit()"">
            <table>
                        <input type=""hidden"" id=""hidValidate"" name=""hidValidate""
value=""FM1_<%=(String)session.getAttribute(""SES_SELF_"")%>"" />
```

```
String validate = ""FM1_"" + Oj_util.checkNull(request.getParameter(""validate""));

//-- Check access
if(session.getAttribute (""SES_SELF_"") != null){
      String ls_compare = (String)session.getAttribute (""SES_SELF_"");

      if(!validate.equals(ls_compare)){
         out.println(""Session time out!!"");
         return;
      }
}
else{
      out.println(""Session time out!!"");
      return;
}
```

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยตัวเอง โดยทำการตรวจสอบข้อมูลที่รับเข้ามาจากฝั่งผู้ใช้บริการก่อนที่จะนำข้อมูลนั้นไปประมวลผล

## ๒.๗ HTTP Header Injection

ผู้ไม่ประสงค์ดีสามารถที่จะปลอมแปลง HTTP Header และ Body ทั้งแบบ Request Header, Response Head, Request Body และ Response Body เพื่อฝัง script อันตรายและสร้าง cookie ปลอม เพื่อโจมตีเครื่องให้บริการและสร้างความเสียหายให้กับผู้ใช้บริการ

**<u>การป้องกันการโจมตี</u>**

- ไม่ให้แสดงข้อมูล HTTP Header โดยตรง
- เพิ่มการป้องกัน Unexpected Line Feeds ด้วยตนเอง (กรณีมีการใช้งาน HTTP Header API และไม่มีฟังก์ชัน Line Feed Neutralization)
- มีการทำ Input Validate ที่เครื่องแม่ข่าย

**<u>การลดความเสียหายที่เกิดจากการถูกโจมตี</u>**

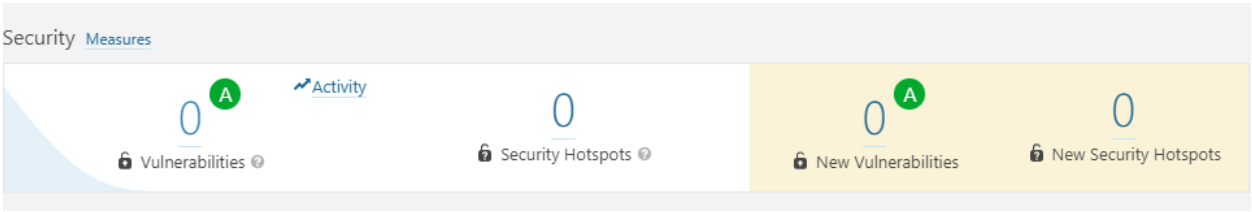
- ลบอักขระพิเศษ (Input Line Feed Character) ทั้งหมดที่ ปรากฏใน External Text Input

ตัวอย่างการทำ Validation File Upload

```
if (FileUpload1.HasFile)
 {
        // Get the name of the file to upload.
        string fileName = Server.HtmlEncode(FileUpload1.FileName);
        // Get the extension of the uploaded file.
        string extension = System.IO.Path.GetExtension(fileName);

        if (extension == "".pdf"")
        {
                //upload accordingly
        }
}
```

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยโปรแกรม Burp Suite

## ๒.๘ Mail Header Injection

ผู้ไม่ประสงค์ดีสามารถตั้งค่า เปลี่ยนแปลง หรือเพิ่มอีเมลอื่นๆได้ตามต้องการ โดยการปลอม Mail Header เพื่อกระจาย Spam Mail หรือสร้างความเสียหายให้กับผู้ได้รับอื่นๆต่อไป

**การป้องกันการโจมตี**

- กำหนดค่าคงที่ (Fixed Values) สำหรับองค์ประกอบของ Header เช่น from, to, cc, bcc, subject ซึ่งแนะนำไม่ให้ใช้ External Parameter เพื่อกำหนดค่าของ Email Header Element
- ในกรณีที่ไม่สามารถใช้การกำหนดค่าคงที่ (Fixed Header) ใน Header ให้ใช้ Email-sending API ที่สามารถใช้งานร่วมกับโปรแกรมประยุกต์บนเว็บหรือภาษาที่ใช้ในการพัฒนาได้
- กระบวนการส่งอีเมลควรกระทำในลักษณะ Background Process ทีละรายการ

**การลดความเสียหายที่เกิดจากการถูกโจมตี**

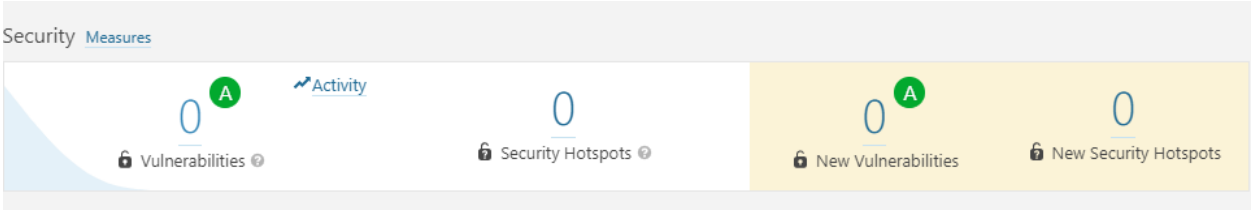
- ลบอักขระพิเศษ (Input Line Feed Character) ทั้งหมดที่รับข้อมูลจากผู้ใช้บริการ

ตัวอย่างการ Fixed Values Header (ไม่ใช้ External Parameter)

```
MimeMessage msg = new MimeMessage(session);
//set message headers
msg.addHeader(""Content-type"", ""text/HTML; charset=UTF-8"");
msg.addHeader(""format"", ""flowed"");
msg.addHeader(""Content-Transfer-Encoding"", ""8bit"");
```

**การทดสอบความมั่นคงปลอดภัย**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยโปรแกรม Burp Suite โดยทดลองปลอม mail header ทั้งในแบบ fixed values และ email sending api

## ๒.๙ Lack of Authentication and Authorization

การปลอมตัวเป็นผู้ใช้บริการและเข้าถึงข้อมูลที่ไม่ได้รับอนุญาต เช่นการแอบอ้างเป็นเจ้าของบัญชีเพื่อขอเปลี่ยนรหัสการเข้าใช้งาน

**<u>การป้องกันการโจมตี</u>**

- มีระบบยืนยันตัวตน ที่มีการระบุข้อมูลส่วนตัวร่วมด้วย เช่นรหัสผ่าน เป็นต้น
- เก็บรหัสผ่านที่มีการเข้ารหัสลับตามที่มาตรฐานด้านความมั่นคงปลอดภัยกำหนด เช่น AES หรือ Triple DES เป็นต้น
- มีกระบวนการที่ชัดเจน เพื่อให้มั่นใจได้ว่าผู้ใช้บริการที่เข้าสู่ระบบ ไม่สามารถเข้าถึงบัญชีผู้ใช้และข้อมูลของผู้ใช้คนอื่น ๆ ได้ ผู้ใช้ระบบเป็นตัวตนจริง เช่น Two Steps Verification

**<u>การลดความเสียหายที่เกิดจากการถูกโจมตี</u>**

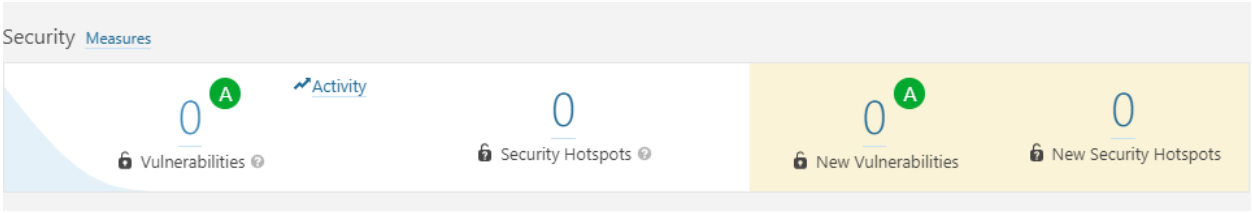
- รหัสผ่านที่ใช้ ต้องประกอบด้วยอักษรตัวเล็ก ตัวใหญ่ ตัวเลขและตัวอักขระพิเศษ ความยาวไม่น้อยกว่า 8 หลัก

ตัวอย่างการกำหนดรหัสผ่านที่ประกอบด้วยอักษรตัวเล็ก ตัวใหญ่ ตัวเลข และอักขระพิเศษ

```
private static final String PASSWORD_PATTERN = ""((?=.*[a-z])(?=.*\\d)(?=.*[A-Z])(?=.*[@#$%!]).{8,40})"";

public PasswordValidator() {
      pattern = Pattern.compile(PASSWORD_PATTERN);
}
```

**<u>การทดสอบความมั่นคงปลอดภัย</u>**

- ผลการทดสอบจากการสแกนโปรแกรม SonarQube




- ทดสอบด้วยโปรแกรม Brute force เช่น Medusa และ Burp Suite

## ๓. ตรวจสอบคุณภาพรหัสต้นฉบับ (Source Code)

ผู้พัฒนาโปรแกรมจำนวนมากทั้งภายในและภายนอกองค์กรพัฒนาโปรแกรมด้วยความเร่งรีบ เพื่อให้มีมาตรฐานเดียวกันจำเป็นต้องมีเครื่องมือช่วยตรวจสอบรหัสต้นฉบับเพื่อตรวจสอบคุณภาพและความปลอดภัยให้เป็นไปตามมาตรฐาน

## ๓.๑. เครื่องมือที่ใช้ตรวจสอบรหัสต้นฉบับ

SonarQube เป็นเครื่องมือ Free & Open Source ที่ช่วยวิเคราะห์ตรวจสอบคุณภาพและความปลอดภัยของรหัสต้นฉบับตามเกณฑ์มาตรฐาน CWE รวมถึง SANS Top 25 และ OWASP Top 10 เพื่อนักพัฒนาใช้ตรวจสอบและปรับปรุงคุณภาพรหัสต้นฉบับให้ดียิ่งขึ้น

## ๓.๒. ขั้นตอนการติดตั้ง SonarQube

### ๓.๒.๑. ติดตั้ง SonarQube server บน Ubuntu

1) หา docker sonarqube server

   ***Exat: ~$*** sudo docker search sonarqube

2) นำ sonarqube มาใช้งาน

   ***Exat: ~$*** sudo docker pull sonarqube

3) ตรวจสอบว่า sonarqube มาหรือไม่

   ***Exat: ~$*** sudo docker images

4) นำ image file มาใส่ลง container ของ docker

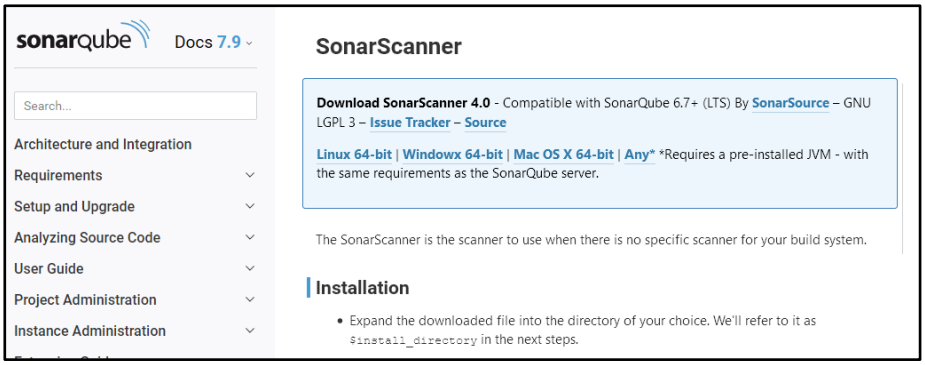
   ***Exat: ~$*** sudo docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube

5) ตรวจสอบว่าพร้อมใช้งานโดยเข้าไปที่ http://localhost:9000/

### ๓.๒.๒. ติดตั้ง SonarQube cilent

1) ดาวน์โหลด sonarqube client จาก

   https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/




2) แตกไฟล์

   ***Exat: ~$*** unzip sonar-scanner-cli-4.0.0.1744-linux.zip

## ๓.๓. ขั้นตอนการตรวจสอบคุณภาพโค้ด

### ๓.๓.๑. จัดเตรียมโปรเจคที่ต้องการตรวจสอบคุณภาพ

1) เตรียมโครงสร้างของไฟล์ที่จะใช้ตรวจสอบให้ตรงกับรูปแบบที่ SonarQube ต้องการ โดยสร้าง โฟลเดอร์ ProjectTest เพื่อใส่ไฟล์รหัสต้นฉบับสำหรับแต่ละโปรเจค
2) สร้างโฟลเดอร์ภายใน ProjectTest ชื่อว่า src และคัดลอกไฟล์รหัสต้นฉบับทั้งหมดภายในโฟลเดอร์ src/java_file (รวม package โฟลเดอร์ด้วย)
3) สร้างโฟลเดอร์ภายใน ProjectTest ชื่อว่า classes และคัดลอกไฟล์รหัสต้นฉบับทั้งหมดภายในโฟลเดอร์ classes/java_class (รวม package โฟลเดอร์ด้วย)
4) สร้างไฟล์ sonar-project.properties แยกให้แต่ละโปรเจคที่ต้องการทดสอบโดยภายใน ProjectTest ต้องมีการตั้งค่าดังนี้

```
#-- server
sonar.host.url = http://localhost:9000

#-- project identity
sonar.projectKey = ProjectTest
sonar.projectName = ProjectTest
sonar.projectVersion = 1.0
sonar.language = java

#-- define source code
sonar.source = src
sonar.java.binaries = classes
```

### ๓.๓.๒. เริ่มการตรวจสอบคุณภาพ

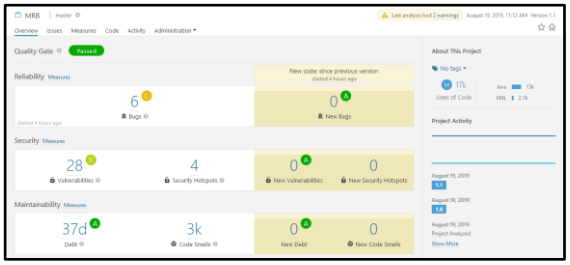
ใช้คำสั่ง sonar-scanner ภายใน ProjectTest ที่เดียวกับที่ไฟล์ .properties อยู่ เช่น

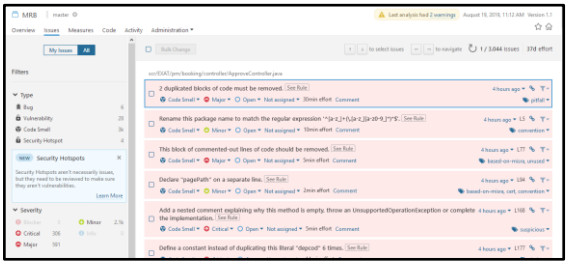
```
#-- ubuntu
:~/ProjectSonarqube/ProjectTest$ sonar-scanner

#-- windows
C:\Users\Pm\ ProjectSonarqube\ProjectTest> sonar-scanner.bat
```

* เพื่อให้เรียกใช้โปรแกรม sonar-scanner ได้สะดวกขึ้นควร set path เพื่อให้ terminal/cmd เห็นโปรแกรม sonar-scanner ก่อนทำการตรวจสอบคุณภาพ

## ๓.๔. ผลลัพธ์ที่ได้จากการตรวจสอบคุณภาพ

ตรวจสอบผลลัพธ์ที่ SonarQube server http://localhost:9000 จะแสดงข้อมูลแต่ละโปรเจคที่เราเคยตรวจสอบ เมื่อกดดูรายละเอียดของแต่ละโปรเจคจะเห็นรายการข้อผิดพลาดในรหัสต้นฉบับในแต่ละไฟล์และบรรทัด รวมถึงข้อแนะนำที่จะทำให้ข้อผิดพลาดน้อยลง เช่น การประเมินระดับความร้ายแรง โค้ดซ้ำซ้อนหรือโค้ดที่อาจไม่เกี่ยวข้อง วิธีการปรับปรุงรหัสต้นฉบับให้ดีขึ้น เป็นต้น





**๓.๔.๑. หมวดหมู่ของข้อผิดพลาด** ข้อผิดพลาดแบ่งออกได้เป็น 3 กลุ่ม

๑) **Bug** คือจุดบกพร่องที่ทำให้โปรแกรมทำงานผิดพลาดจำเป็นต้องแก้ไขทันที

๒) **Vulnerability** คือจุดที่สามารถใช้เป็นช่องโหว่หรือปล่อยให้เกิดการโจมตีได้

๓) **Code Smell** คือจุดที่ทำให้เกิดความสับสนและยากต่อการบำรุงรักษา

**๓.๔.๒. หมวดหมู่ของความรุนแรง**

ซึ่งในแต่ละจุดบกพร่องที่ SonarQube ตรวจพบจะถูกตั้งระดับความรุนแรง เพื่อให้เห็นภาพรวมของจุดบกพร่องและเลือกพิจารณาแก้ไขตามความเร่งด่วนและรุ่นแรง ซึ่งแบ่งออกเป็น ๕ ระดับดังนี้

๑) **BLOCKER** คือจุดที่มีความเป็นไปได้สูงที่จะส่งผลต่อการทำงานของโปรแกรมและต้องแก้ไขด่วน เช่น ทำให้เกิด Memory Leak หรือการไม่มีการปิดการเชื่อมต่อกับฐานข้อมูล

๒) **CRITICAL** คือจุดบกพร่องที่มีความเป็นไปได้ว่าจะส่งผลต่อการทำงานของโปรแกรม หรือเป็นจุดให้เกิดปัญหาด้านความปลอดภัย เช่น ตรวจเจอจุดบกพร่องแต่ไม่แจ้งเตือนปัญหาใน catch หรือก่อให้เกิด SQL Injection ได้ จำเป็นต้องทบทวนวิธีการเขียนโดยด่วน

๓) **MAJOR** คือข้อบกพร่องด้านคุณภาพของซอสโค้ด เช่น เขียนซ้ำซ้อน หรือตัวแปรที่ไม่มีการเรียกไม่ใช้งาน

๔) **MINOR** คือข้อบกพร่องด้านคุณภาพของซอสโค้ด เช่น จำนวนบรรทัดมากไป หรือการใช้ switch ขั้นต่ำต้องมี ๓ กรณี

๕) **INFO** ไม่ใช่ข้อบกพร่องที่ต้องรีบแก้ไขหรือด้านคุณภาพซอสโค้ดแต่ตรวจพบ

### ๓.๔.๓. การประเมินด้านความน่าเชื่อถือและความปลอดภัย

การประเมินแบ่งออกเป็น ๕ ระดับในแต่ละด้านทั้งความน่าเชื่อถือและความปลอดภัย ซึ่งแสดงความรุนแรงแบ่งตามสี ดังนี้

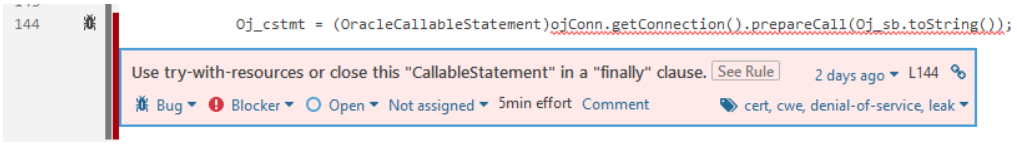
A B C D E

| ความน่าเชื่อถือ | ความปลอดภัย |
|---|---|
| A คือ ไม่มีข้อผิดพลาดเลย | A คือ ไม่มีจุดบกพร่องเลย |
| B คือมี Minor Bug อยู่ | B คือมี Minor Vulnerability อยู่ |
| C คือมี Major Bug อยู่ | C คือมี Major Vulnerability อยู่ |
| D คือมี Critical Bug อยู่ | D คือมี Critical Vulnerability อยู่ |
| E คือมี Blocker Bug อยู่ | E คือมี Blocker Vulnerability อยู่ |

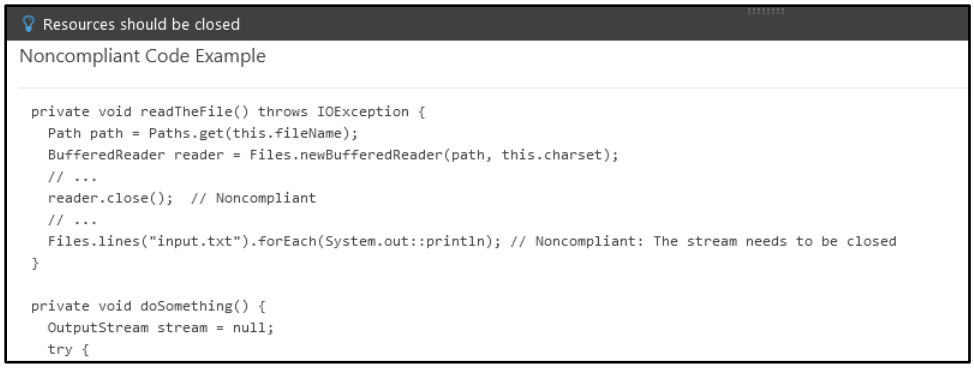
## ๓.๕. การแก้ไขและปรับแต่งซอสโค้ด

๑) ประเมินความน่าเชื่อถือ ความปลอดภัย ประเภทข้อผิดพลาด ความรุนแรง เพื่อหาจุดที่ต้องเร่งรีบแก้ไขเป็นอันดับแรก เลือกตามความรุนแรงของข้อผิดพลาด จากมากไปน้อย

๒) ดูสาเหตุของข้อบกพร่องพร้อมคำแนะนำจากประเภทข้อผิดพลาดที่อยู่หน้าบรรทัด




๓) เลือก “See Rule” เพื่อดูสาเหตุที่เกิดข้อผิดพลาดโดยละเอียด รวมถึงวิธีการแก้ไขซอสโค้ดให้ถูกต้อง



# บรรณานุกรม

สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) กระทรวงดิจิทัลเพื่อเศรษฐกิจและสังคม, “ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จาเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสาหรับโปรแกรมประยุกต์บนเว็บ (Web Application Security Standard) เวอร์ชั่น ๑.๐,” [ออนไลน์]. Available: https://standard.etda.or.th/wp-content/uploads/2018/09/20150405-ER-WAS-V07-33-R1.pdf