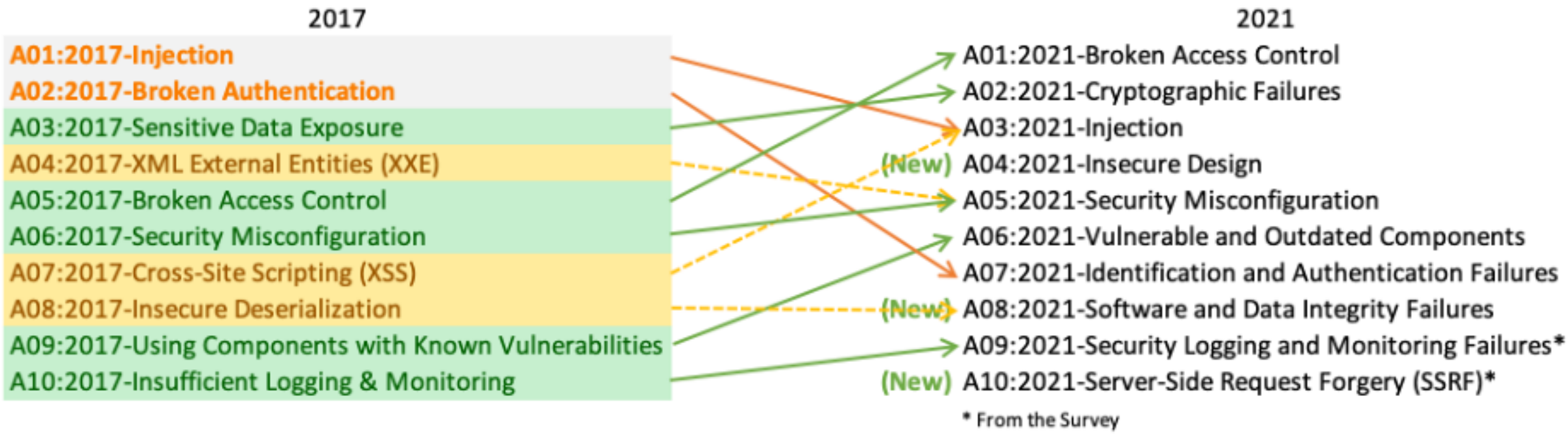

e-Cop®
Managed Security Service Provider

Cross-site-Scripting
&
Session Hijack

OWASP Top-10 2017

OWASP Top 10 - 2017
A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE)
A5:2017-Broken Access Control
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring



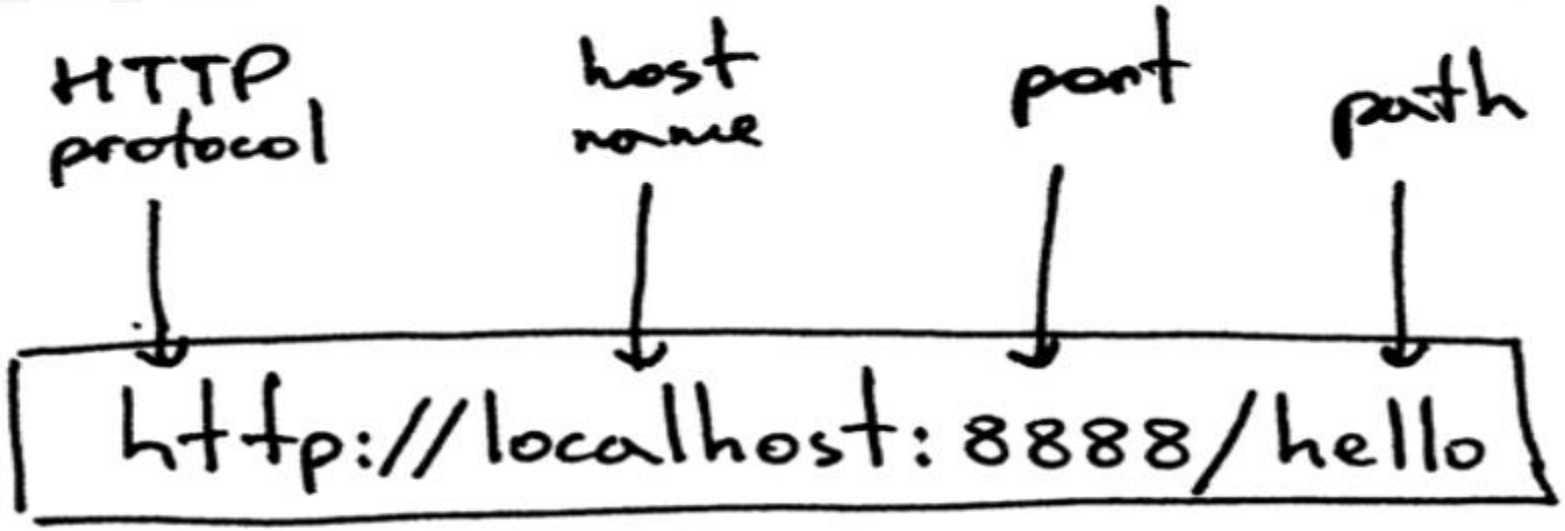
A3 2021
&
A7 2021

Intro

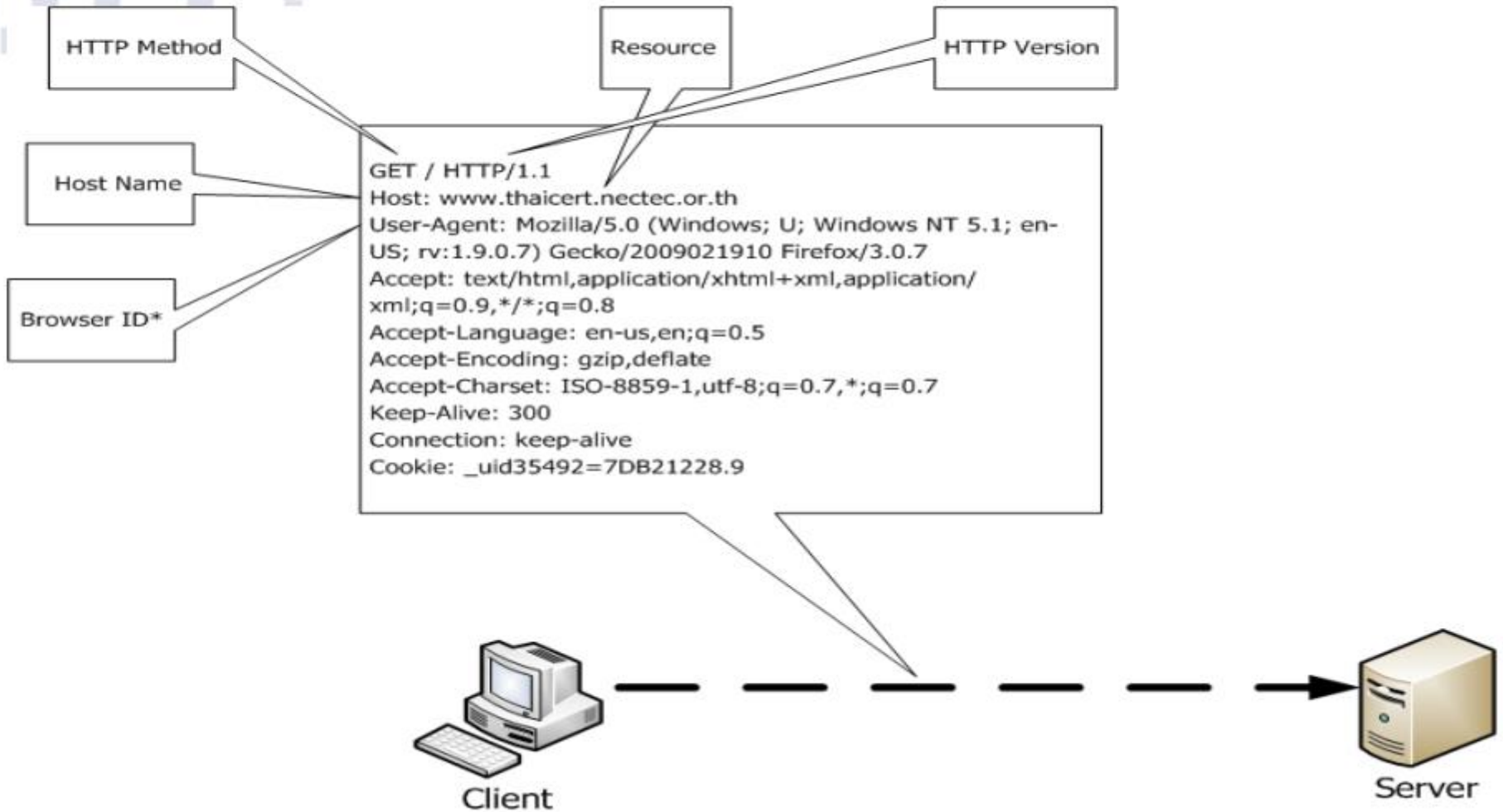
ทำความรู้จัก HTTP Protocol

- HTTP ย่อมาจาก Hypertext Transfer Protocol เป็นโปรโตคอล (Protocol) สื่อสารที่ทำงานอยู่ในระดับ Application Layer บนโปรโตคอล TCP/IP
- เป็นโปรโตคอลหลักที่ใช้ในการแลกเปลี่ยนข้อมูล (HTML) กันระหว่าง Web Server และ Web Client (Browser)
- ใช้ URL (Uniform Resoure Locator) ในการเข้าถึงเว็ปไซต์ (Web Site) ซึ่งจะขึ้นต้นด้วย http:// ตามด้วยชื่อของเว็ปไซต์
- ทำงานที่พอร์ต (port) 80 (มาตรฐาน)
- ส่งข้อมูลเป็นแบบ Clear text คือ ไม่มีการเข้ารหัสลับข้อมูลในระหว่างการส่ง(None-Encryption)

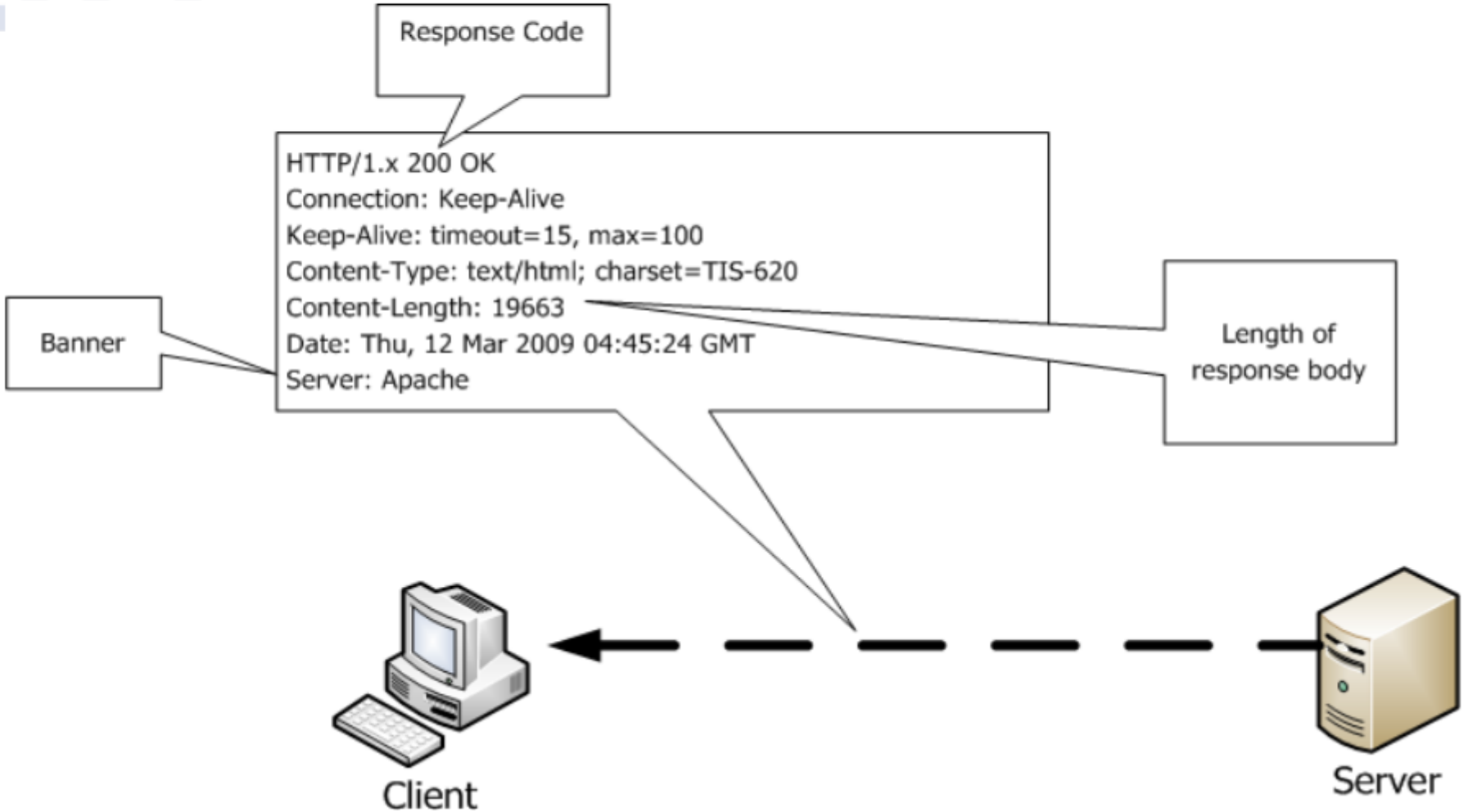
ทำความรู้จัก HTTP Protocol



HTTP Request



HTTP Response



HTTP Method

- GET

The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

- HEAD

Same as GET, but transfers the status line and header section only.

- POST

A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

HTTP Header

Client request

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Server response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
<head>
  <title>An Example Page</title>
</head>
<body>
  Hello World, this is a very simple HTML document.
</body>
</html>
```

HTTP Status Code

- 1xx Informational
- 2xx The request was successfully
- 3xx The client redirection to a different resource
- 4xx The request contains and error of some kind
- 5xx The server encountered an error fulfilling the request

HTTP Status Code Example

- 100 Continue
- 200 OK
- 201 Created
- 301 Moved Permanently
- 302 Found
- 400 Bad Request
- 401 Unauthorized
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error
- 502 Bad Gateway

ทำความรู้จัก URL Encode

- URL encoding converts characters into a format that can be transmitted over the Internet.
- URLs can only be sent over the Internet using the ASCII character-set.
- Since URLs often contain characters outside the ASCII set, the URL has to be converted into a valid ASCII format.
- URL encoding replaces unsafe ASCII characters with a "%" followed by two hexadecimal digits.
- URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign or with %20.

URL Encode: Example

- http://www.permadi.com/tutorial/urlEncoding/example.html?var=This+is+a+simple+%26+short+test.
  - The <space> character has been URL encoded as "+".
  - The & character has been URL encoded as "%26“
- var=%24+%26+%3C+%3E+%3F+%3B+%23+%3A+%3D+%2C+%22+%27+%7E+%2B+%25

Character	URL Encoded
;	%3B
?	%3F
/	%2F
:	%3A
#	%23
&	%26
=	%3D
+	%2B
$	%24

,	%2C
<space>	%20 or +
%	%25
<	%3C
>	%3E
~	%7E
%	%25

ทำความรู้จัก User Agent

- In computing, a user agent is software (a software agent) that is acting on behalf of a user. One common use of the term refers to a web browser telling a website information about the browser and operating system. This allows the website to customize content for the capabilities of a particular device, but also raises privacy issues.




ทำความรู้จัก User Agent

- The User-Agent request header contains a characteristic string that allows the network protocol peers to identify the application type, operating system, software vendor or software version of the requesting software user agent.

- **Syntax**
  - User-Agent: **<product> / <product-version> <comment>**
  - Common format for web browsers:
  - User-Agent: Mozilla/<version> (<system-information>) <platform> (<platform-details>) <extensions>

User Agent: Examples

- Mozilla/5.0 (platform; rv:geckoversion) Gecko/geckotrail Firefox/firefoxversion
  - ***Mozilla/5.0*** is the general token that says the browser is Mozilla compatible, and is common to almost every browser today.
  - ***platform*** describes the native platform the browser is running on (e.g. Windows, Mac, Linux or Android), and whether or not it's a mobile phone. Firefox OS phones simply say "Mobile"; the web is the platform. Note that ***platform*** can consist of multiple "; "-separated tokens. See below for further details and examples.
  - **rv:*geckoversion*** indicates the release version of Gecko (such as *"17.0"*). In recent browsers, ***geckoversion*** is the same as ***firefoxversion***.
  - ***Gecko/geckotrail*** indicates that the browser is based on Gecko.
  - On Desktop, ***geckotrail*** is the fixed string "20100101"
  - ***Firefox/firefoxversion*** indicates the browser is Firefox, and provides the version (such as "*17.0*").

User Agent: Examples

- Firefox
  - Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0
  - Mozilla/5.0 (Macintosh; Intel Mac OS X x.y; rv:42.0) Gecko/20100101 Firefox/42.0

- Chrome
  - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36

ทำความรู้จัก Session&Cookie

- HTTP is a stateless protocol
- Need session mechanism to track subsequent communications
- Session == identity
- Should be random, unlikely to be predictable
- Session vs Cookie
- Session
  - Information stores on server
  - SessionID stores on client
- Cookie
  - Information stores on server and client

Cookie

- The Set-Cookie HTTP response header sends cookies from the server to the user agent. A simple cookie is set like this
  - Set-Cookie: <cookie-name>=<cookie-value>

- HTTP/1.0 200 OK
- Content-type: text/html
- Set-Cookie: yummy_cookie=choco
- Set-Cookie: tasty_cookie=strawberry
- [page content]

GET /sample_page.html HTTP/1.1

Host: www.example.org

Cookie: yummy_cookie=choco;

tasty_cookie=strawberry

XSS

- Cross-Site Scripting (XSS) คือ ช่องโหว่ที่ผู้ไม่ประสงค์ดีสามารถแทรกคำสั่งอันตรายเข้าสู่เว็บแอปพลิเคชัน เช่น คำสั่งของ Javascript หรือ HTML ทำให้ผู้ไม่ประสงค์ดีสามารถขโมยเซสชั่น (session) หรือ เปลี่ยนหน้าเว็บไซต์ไปยังเว็บไซต์ของผู้ไม่ประสงค์ดีได้สำเร็จ

http://www.bluekaizen.org/xss-attack-through-metasploit/

What can an attacker do with this Vulnerability?

- Stealing the Identity and Confidential Data (credit card details)
- Bypassing restriction in websites.
- Session Hijacking (Stealing session)
- Malware Attack
- Website Defacement
- Denial of Service attacks (Dos)

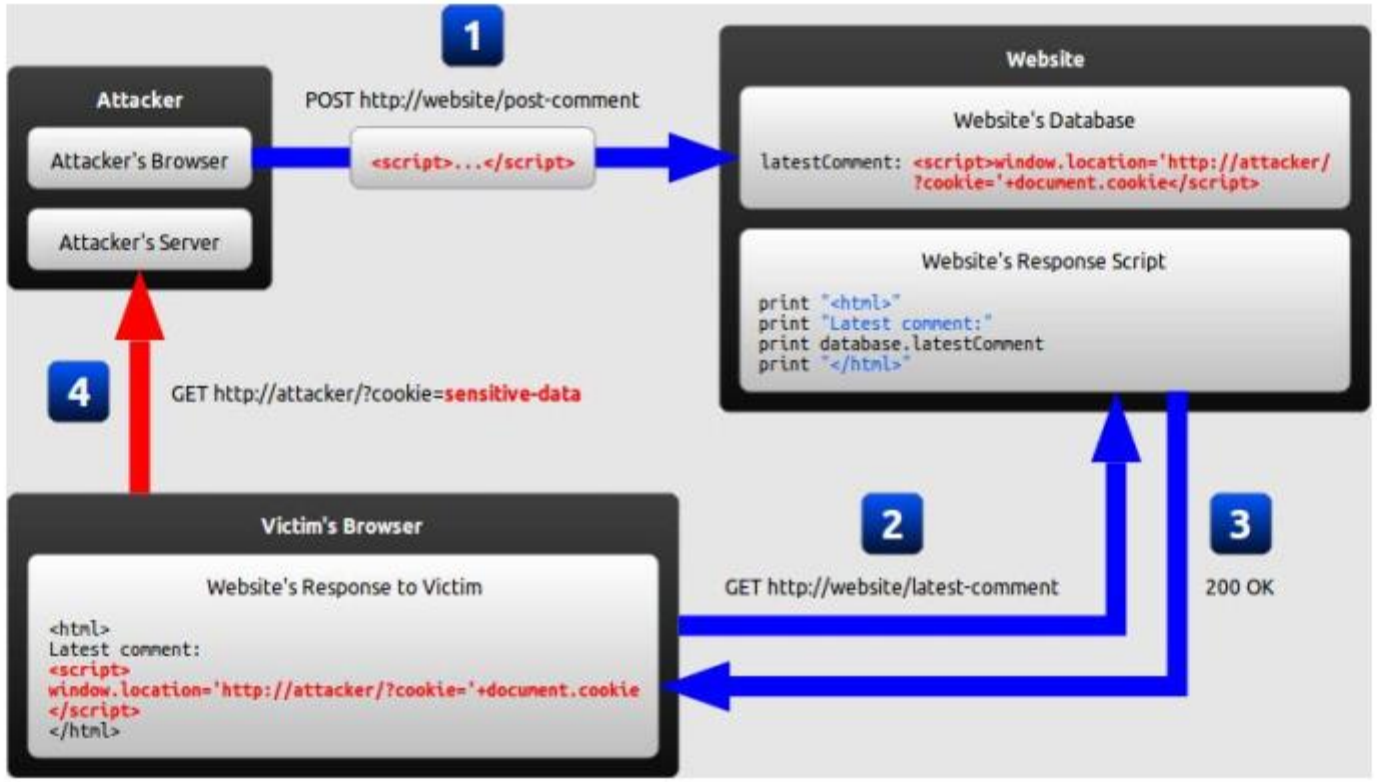
Type of XSS

- **Stored XSS**

- The most damaging type of XSS is Stored (Persistent) XSS. Stored XSS attacks involves an attacker injecting a script (referred to as the payload) that is permanently stored (persisted) on the target application (for instance within a database). The classic example of stored XSS is a malicious script inserted by an attacker in a comment field on a blog or in a forum post.
- When a victim navigates to the affected web page in a browser, the XSS payload will be served as part of the web page (just like a legitimate comment would). This means that victims will inadvertently end-up executing the malicious script once the page is viewed in a browser.

Type of XSS

- Persistent XSS, where the malicious string originates from the website's database



https://excess-xss.com

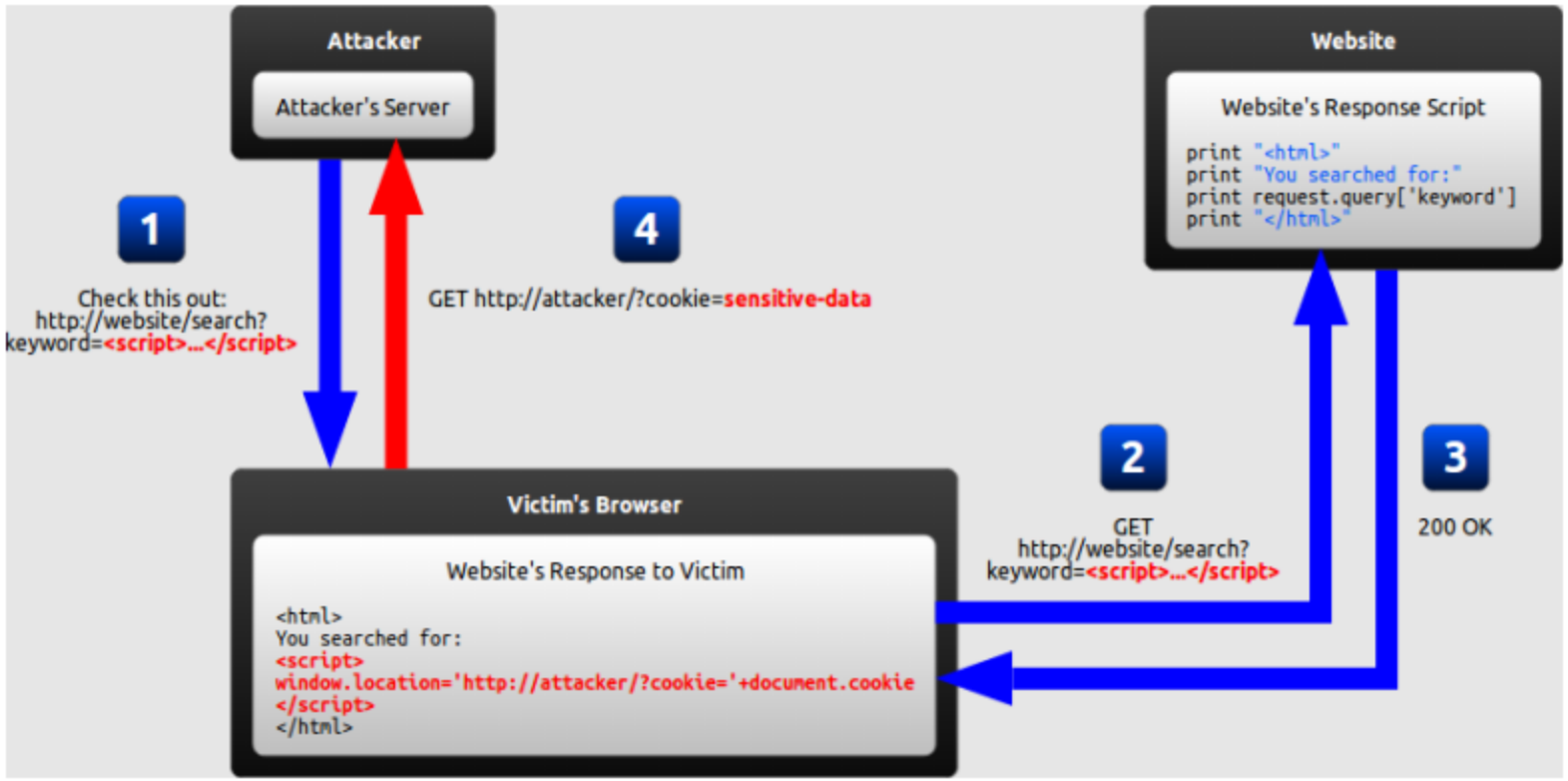
Type of XSS

- **Reflected XSS**

- The second, and by far most common type of XSS is Reflected XSS. In Reflected XSS, the attacker's payload script has to be part of the request which is sent to the web server and reflected back in such a way that the HTTP response includes the payload from the HTTP request. Using Phishing emails and other social engineering techniques, the attacker lures the victim to inadvertently make a request to the server which contains the XSS payload and ends-up executing the script that gets reflected and executed inside the browser. Since Reflected XSS isn't a persistent attack, the attacker needs to deliver the payload to each victim – social networks are often conveniently used for the dissemination of Reflected XSS attacks.

Type of XSS

- Reflected XSS, where the malicious string originates from the victim's request



https://excess-xss.com

Workshop 1

แนวทางป้องกัน

- ดำเนินการตรวจสอบข้อมูล (input validation) ทุกๆ ช่อง input ที่มีการรับค่าจากผู้ใช้งาน และไม่นำไปแสดงผลบน tag ที่สามารถเรียกใช้คำสั่ง Javascript โดยการทำ Data Sanitizing คือ การตัด keyword หรือ tag ออกและ HTML Escaping คือ การ encode อักษระพิเศษในอยู่ใน format ของ HTML encode

```
<form method='get' action='index.php'>
<input name="search" value="<?php echo $_GET['search'];?>" />
<input type=submit name='getdata' value='Search' /></form>
```

http://servername/index.php?search="><script>alert(0)</script>

```
<?php
$search = filter_input(INPUT_POST | INPUT_GET, 'search', FILTER_SANITIZE_SPECIAL_CHARS);
?>
<form method='get' action='index.php'>
<input name="search" value="<?php echo $search;?>" />
<input type=submit name='getdata' value='Search' /></form>
```

filter.default="special_chars"

ตรวจสอบอักษระ (&, <, >, ", ')

แนวทางป้องกัน Cross-Site Scripting (XSS)

- ใช้ฟังก์ชัน stripslashes เพื่อตัดตัวอักษร “\” ออกจากข้อมูลภายในพารามิเตอร์
- ใช้ฟังก์ชัน mysql_real_escape_string เพื่อเพิ่ม “\” ไปยังหน้าข้อความอักขระพิเศษทำให้ข้อความอักขระพิเศษนั้น เป็น string
- ใช้ฟังก์ชัน htmlspecialchar เพื่อแปลงข้อความอักขระพิเศษให้เป็น Entity reference เช่น หากผู้บุกรุกใส่ข้อความ <script> คำสั่งนี้จะแปลงค่าให้เป็น <script>

แนวทางป้องกัน Cross-Site Scripting (XSS)

- Encoding is the act of escaping user input so that the browser interprets it only as data, not as code. The most recognizable type of encoding in web development is HTML escaping, which converts characters like < and > into < and >, respectively.
- The following pseudocode is an example of how user input could be encoded using HTML escaping and then inserted into a page by a server-side script:

```
print "<html>"
print "Latest comment: "
print encodeHtml(userInput)
print "</html>"
```

```
<html>
Latest comment:
&lt;script&gt;...&lt;/script&gt;
</html>
```

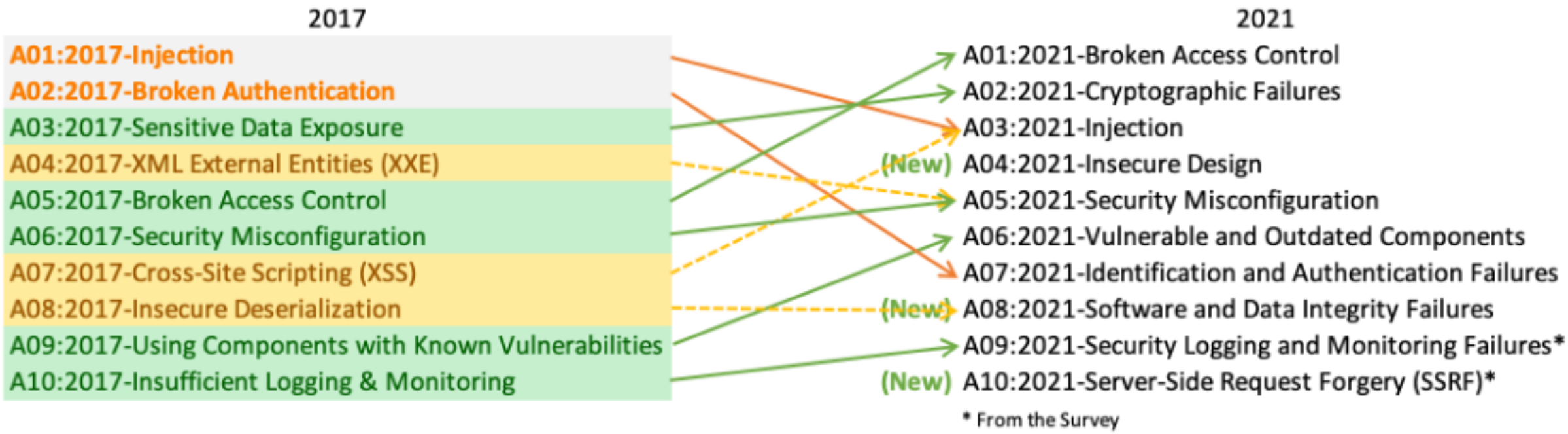
แนวทางป้องกัน Cross-Site Scripting (XSS)

- HttpOnly Cookie คือ การ flag ลงใน Set-Cookie HTTP response header เพื่อทำการป้องกัน client side script เข้าถึง cookie โดยไม่ได้รับอนุญาต
- ไฟล์ httpd.conf บน apache server
  - Header set X-XSS-Protection "1; mode=block“
  - Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
- ไฟล์ php.ini
- session.cookie_httponly = True

Session Hijack

OWASP Top-10 2017

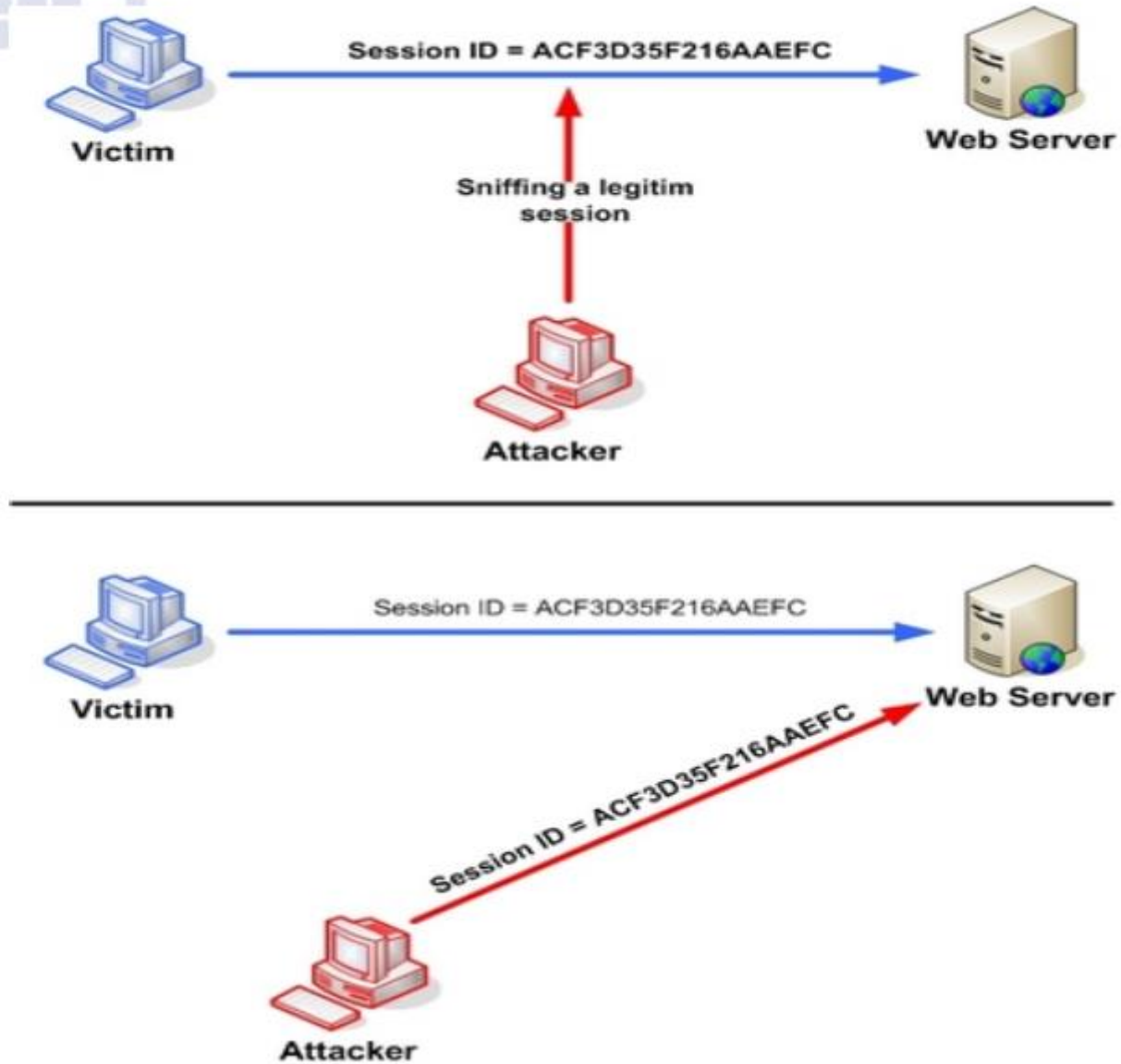
OWASP Top 10 - 2017
A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE)
A5:2017-Broken Access Control
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring



A3 2021
&
A7 2021

- Broken Authentication and Session Management คือ รูปแบบการโจมตีที่อาศัยช่องโหว่จากฟังก์ชันการพิสูจน์ตัวตน (Authentication) ที่ไม่ปลอดภัยเพียงพอ เช่น การตั้งชื่อผู้ใช้งานและรหัสผ่านที่ง่ายต่อการคาดเดา, ฟังก์ชันการ Reset รหัสผ่าน , ฟังก์ชันการป้องกัน Brute Force เป็นต้น รวมไปถึงการบริหารจัดการสิทธิในการเข้าถึงที่อ่อนแอ (Session Management) ทำให้ผู้บุกรุกสามารถเข้าสู่ระบบได้โดยง่าย

Session hijacking attack Example



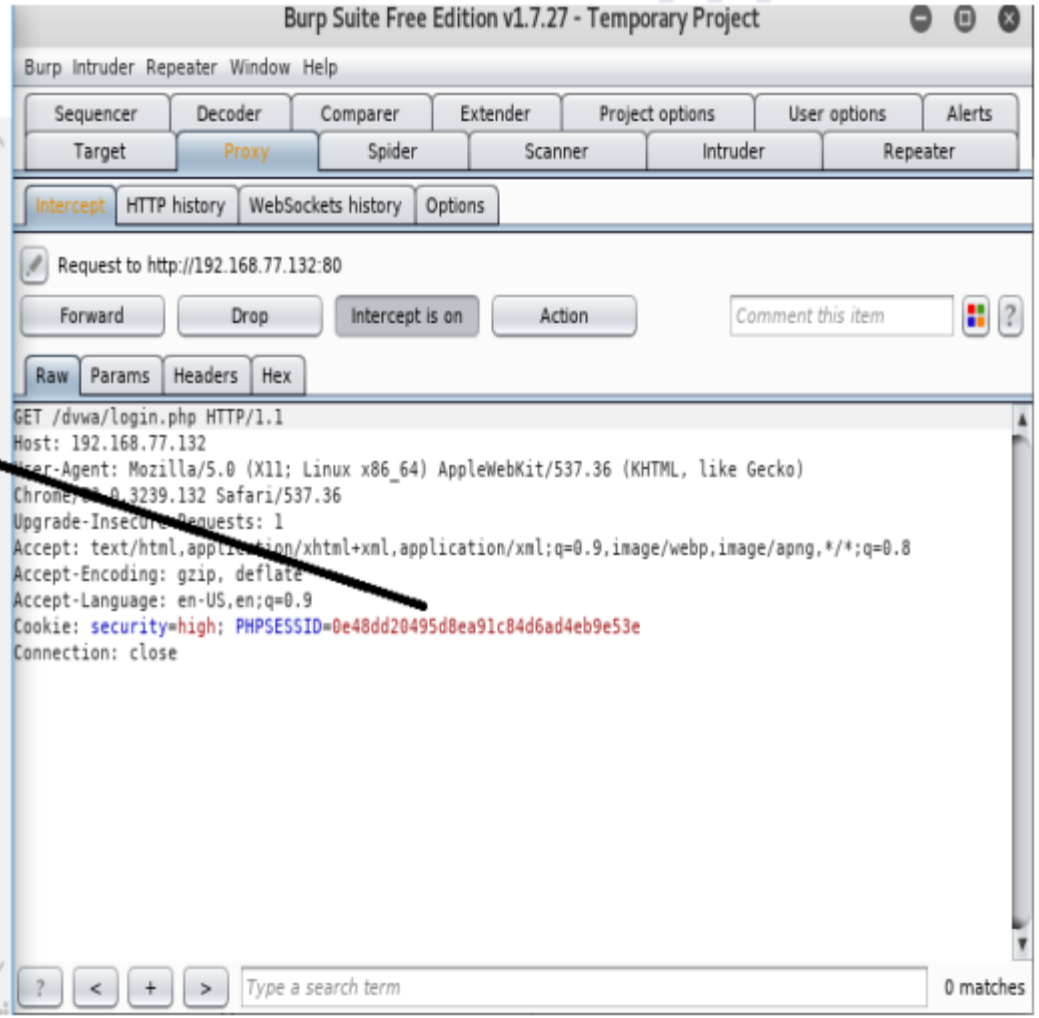
Session Management Example

Untitled - Notepad

File Edit Format View Help

```
POST /dvwa/login.php HTTP/1.1
Host: 192.168.77.132
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 F
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.77.132/dvwa/login.php
Cookie: security=high; PHPSESSID=2ee85c4d241857b2095f5e561645b84e
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

username=admin&password=password&Login=Login
```



Workshop 2

แนวทางป้องกัน

- Password Strength คือ การตั้งรหัสผ่านที่มีความยาวไม่น้อยกว่า 8 ตัวอักษร ให้ยากต่อการคาดเดาที่มีทั้งตัวอักษรเล็กและใหญ่, ตัวเลข, อักษระพิเศษ เป็นต้น
- Password Use คือ การจำกัดหรือกำหนดจำนวนครั้งที่สามารถทำการ Login ผิดพลาดเข้าสู่เว็บแอพพลิเคชันและแจ้งถึงวันที่ / เวลา, หมายเลขไอพี, ประเทศจากการเข้าสู่ระบบที่ประสบความสำเร็จครั้งล่าสุด
- Password Change Controls คือ การควบคุมการเปลี่ยนรหัสผ่านควรให้ผู้ใช้งานระบุทั้งรหัสผ่านเก่าและรหัสผ่านใหม่ ในกรณีที่ผู้ใช้งานไม่ทราบรหัสเก่าให้ทำการตอบคำถามที่ตั้งไว้หรือส่ง SMS เข้าโทรศัพท์
- Password Storage คือ การจัดเก็บรหัสผ่านให้มีความมั่นคงปลอดภัยโดยใช้รูปแบบ hash หรือ Encryption เพื่อป้องการเข้าถึงจากผู้ไม่ประสงค์ดี
- Protecting Credentials in Transit คือ การป้องกันการเปิดเผยข้อมูลสำคัญระหว่างการรับ/ส่งข้อมูลบนระบบเครือข่ายอินเทอร์เน็ต โดยใช้การใช้งาน Protocol SSL บนเครื่องให้บริการเว็บเซิร์ฟเวอร์

- Session ID Protection คือ การป้องกันเซสชันของผู้ใช้งานในการระหว่างการรับ/ส่งข้อมูลบนระบบเครือข่ายอินเทอร์เน็ตบนเครื่องให้บริการเว็บเซิร์ฟเวอร์ มีดังต่อไปนี้
  - การใช้งาน Protocol SSL
  - ไม่แสดงข้อมูลเซสชันของผู้ใช้งานบนช่อง URL
  - รหัสเซสชันควรมีความยาวที่ซับซ้อนไม่สามารถคาดเดาได้ง่าย
  - รหัสเซสชันมีการกำหนดไว้หมดอายุ (กรณีผู้ใช้งานไม่ได้ทำการ Logout ออกจากระบบ)
  - รหัสเซสชันเปลี่ยนแปลงใหม่ทุกครั้งที่มีการ Login เข้าสู่ระบบ
  - รหัสเซสชันมีระยะเวลาการอัพเดตข้อมูลตามเวลาที่กำหนด
  - รหัสเซสชันควรมีมากกว่า 1 ตัวแปรในการ Login เข้าสู่ระบบ

Q&A

Thank You

Nattawut Opasieamlikit (Head of CSIRT)
0902405079 Nattawut@bcg-ecop.net