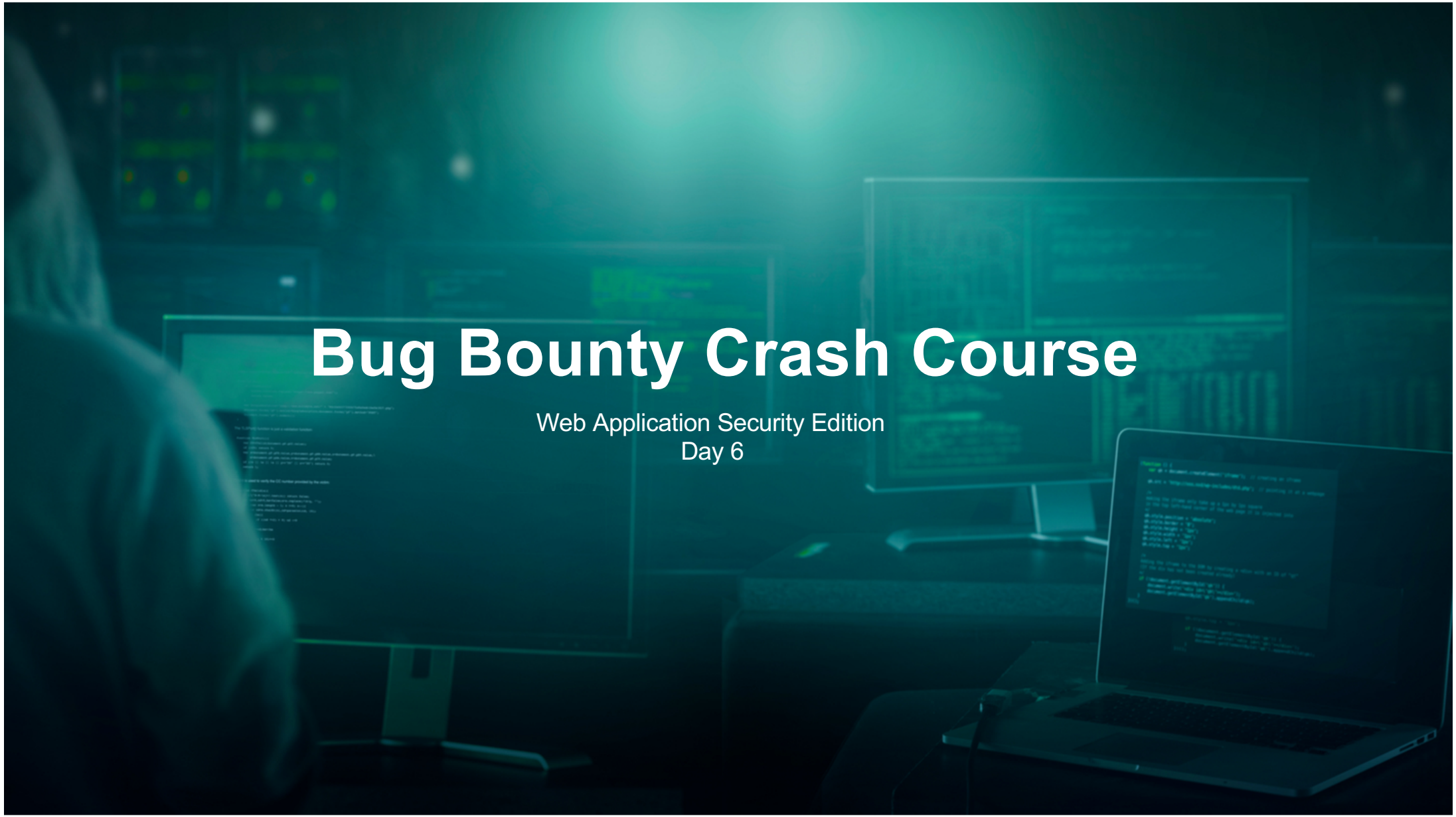


Bug Bounty Crash Course

Web Application Security Edition
Day 6

OWASP Top 10 : A4 XML External Entity

A4 :2017

XML External Entities (XXE)

10

App. Specific	Exploitability: 2	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Attackers can exploit vulnerable XML processors if they can upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies or integrations.		By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. SAST tools can discover this issue by inspecting dependencies and configuration. DAST tools require additional manual steps to detect and exploit this issue. Manual testers need to be trained in how to test for XXE, as it not commonly tested as of 2017.		These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks. The business impact depends on the protection needs of all affected application and data.	

Source: OWASP

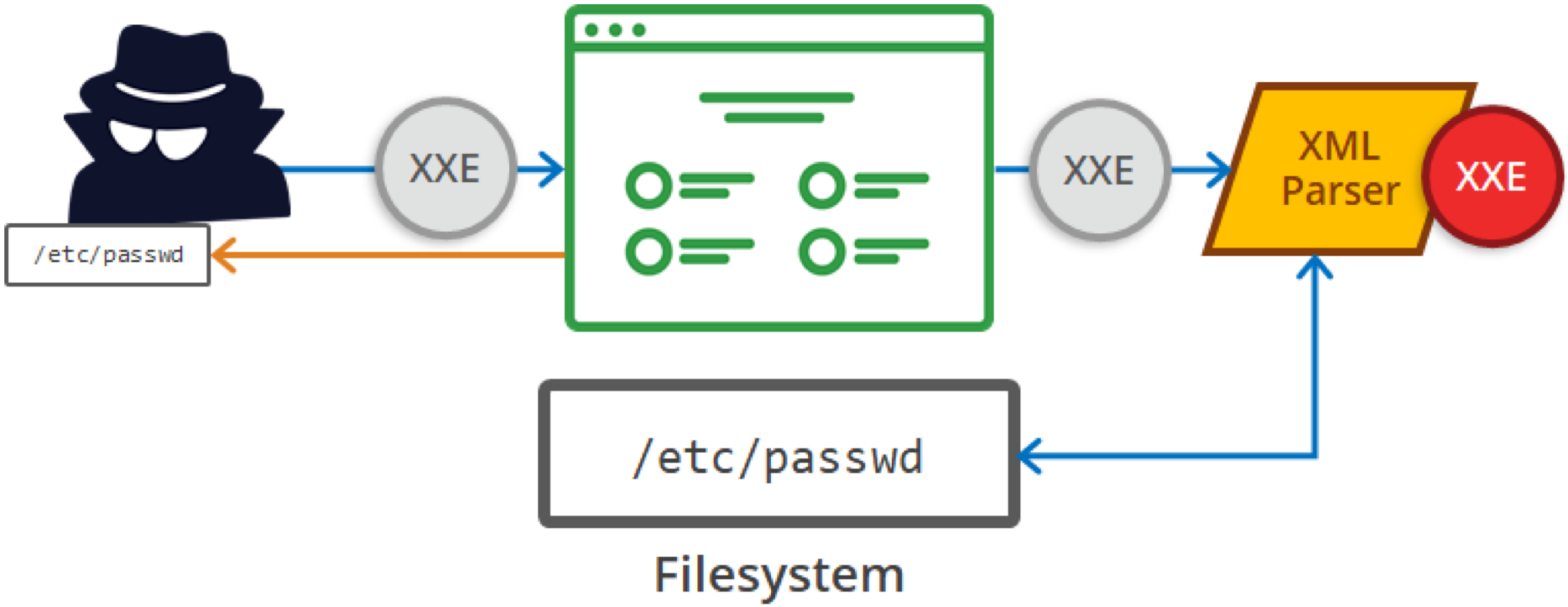
When is XXE Injection possible?

- XML (Extensible Markup Language)
  - Services - XML-RPC, SOAP, REST
  - Documents - XML, HTML, DOCX
  - Images - SVG, EXIF data
- XML Parser (XML Processor)

XXE Injection : Steps

- Identifying weak XML Parser and sending Request
- XML Processor retrieving malicious external entity within the Document Type Declaration
- XML Processor validates DTD resolves external entity
- Response is sent back.

When is XXE Injection?



Source: https://www.acunetix.com/wp-content/uploads/2017/07/image1.png

XXE Injection can lead to

- Reading Files
- Server Side Request Forgery
- Data Exfiltration via Blind XXE attack
- Retrieving Data from error messages via Blind XXE attack

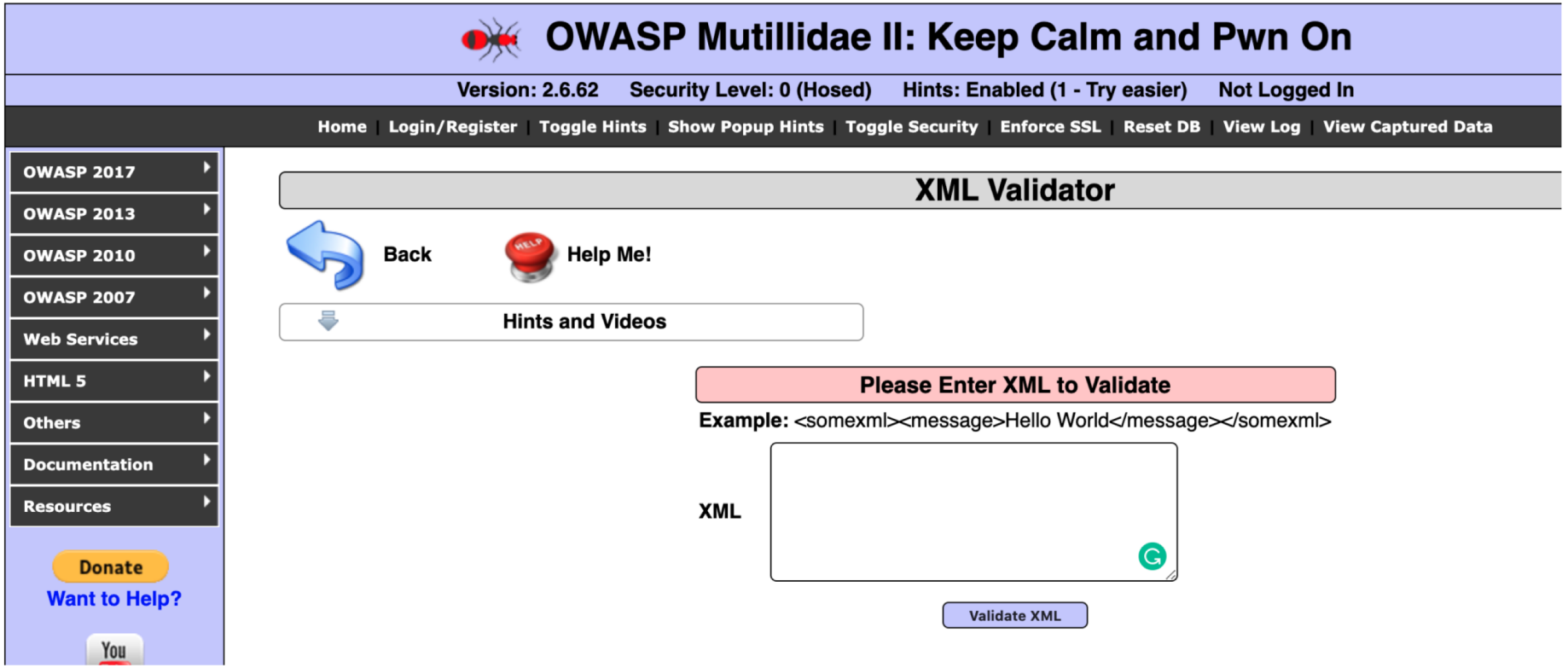
When is the application vulnerable?

- The application accepts XML directly or XML uploads, especially from untrusted sources, or inserts untrusted data into XML documents, which is then parsed by an XML processor.
- Any of the XML processors in the application or SOAP based web services has document type definitions (DTDs) enabled
- If your application uses Security Assertion Markup Language (SAML) for identity processing within federated security or single sign on (SSO) purposes. SAML uses XML for identity assertions, and may be vulnerable

Source: OWASP

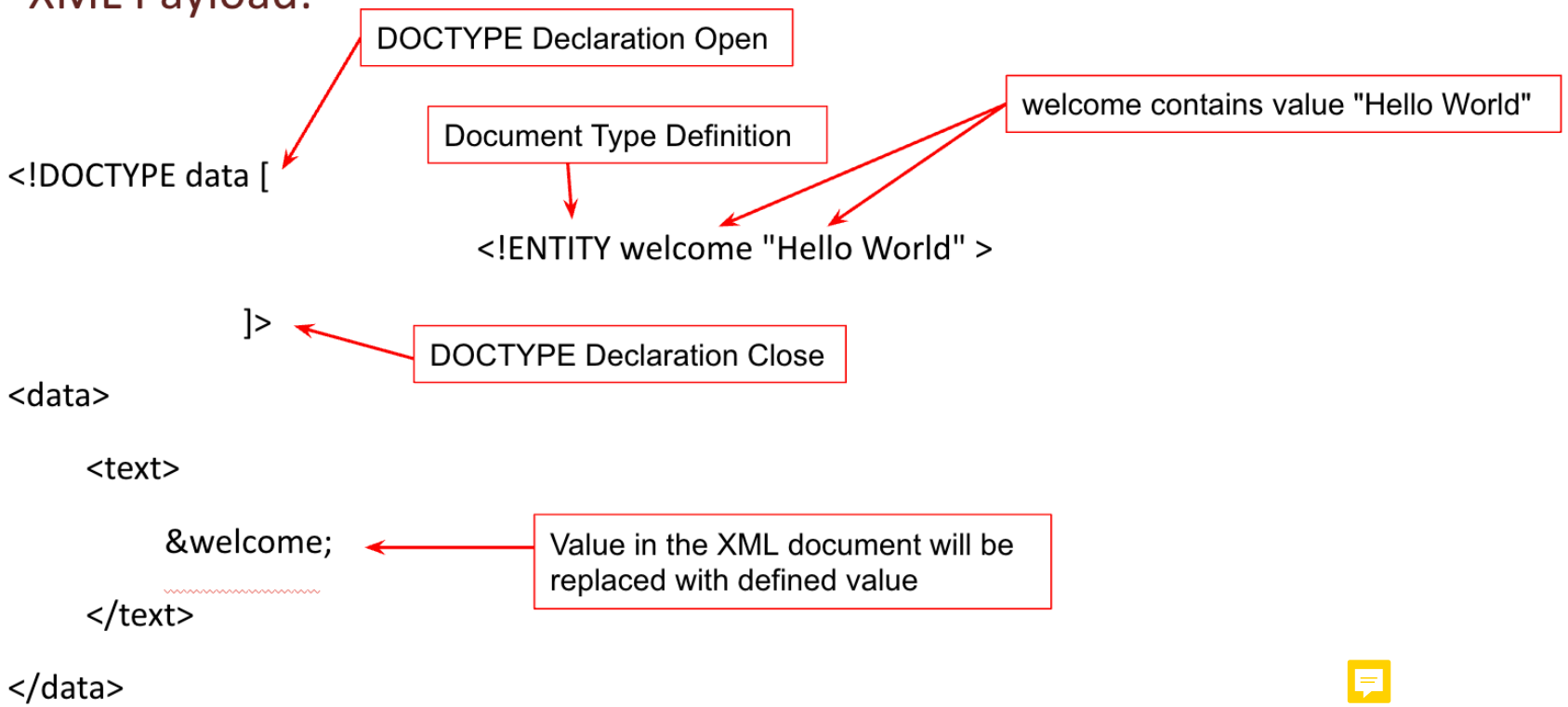
XML External Entity Attack

- Mutillidae- XML External Entity



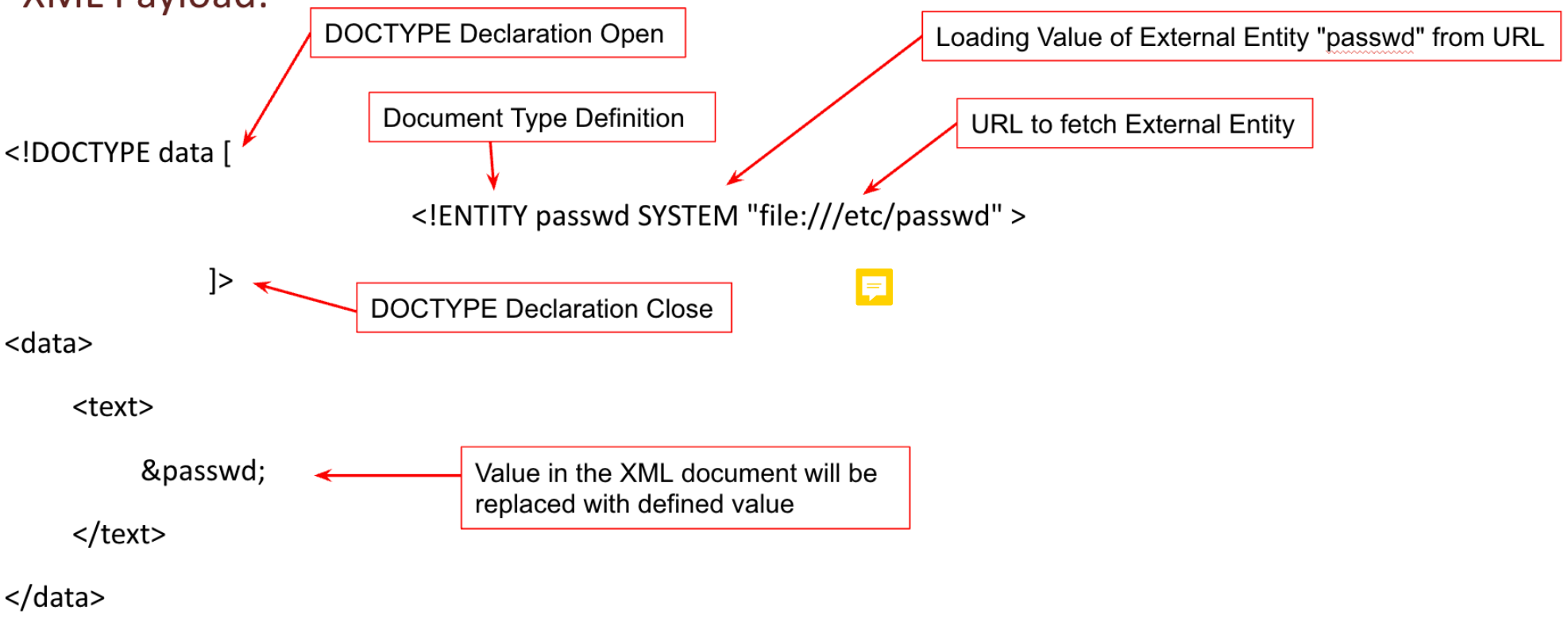
XML External Entity Attack

- XML Payload:



XML External Entity Attack

- XML Payload:



Lab: XML External Entity

Lab URL: https://attackdefense.com/challengedetails?cid=1889

Video URL: https://youtu.be/EYHrwOageNY

Homework Lab: Apache Solr 8.1.1

Lab URL: https://attackdefense.com/challengedetails?cid=1531

Homework Lab: Apache Solr

Lab URL: https://attackdefense.com/challengedetails?cid=1530

Prevention

- Whenever possible, use less complex data formats such as JSON, and avoiding serialization of sensitive data.
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system. Use dependency checkers. Update SOAP to SOAP 1.2 or higher.
- Disable XML external entity and DTD processing in all XML parsers in the application, as per the OWASP Cheat Sheet 'XXE Prevention'.
- Implement positive ("whitelisting") server-side input validation, filtering, or sanitization to prevent hostile data within XML documents, headers, or nodes.

Source: OWASP

OWASP Top 10 : A5 Broken Access Control

Homebrew bug allowed researcher full access to GitHub repos

08 August 2018

The IT crowd

ServiceDesk flaw could give attackers full access to tech support systems

24 January 2020

Flight check-in app gave users access to other travellers' boarding passes

19 July 2019

OWASP Top 10 : A5 Broken Access Control

A5 :2017

Broken Access Control

App. Specific	Exploitability: 2	Prevalence: 2	Detectability: 2	Technical: 3	Business ?
Exploitation of access control is a core skill of attackers. SAST and DAST tools can detect the absence of access control but cannot verify if it is functional when it is present. Access control is detectable using manual means, or possibly through automation for the absence of access controls in certain frameworks.		Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers. Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc.		The technical impact is attackers acting as users or administrators, or users using privileged functions, or creating, accessing, updating or deleting every record. The business impact depends on the protection needs of the application and data.	

Source: OWASP

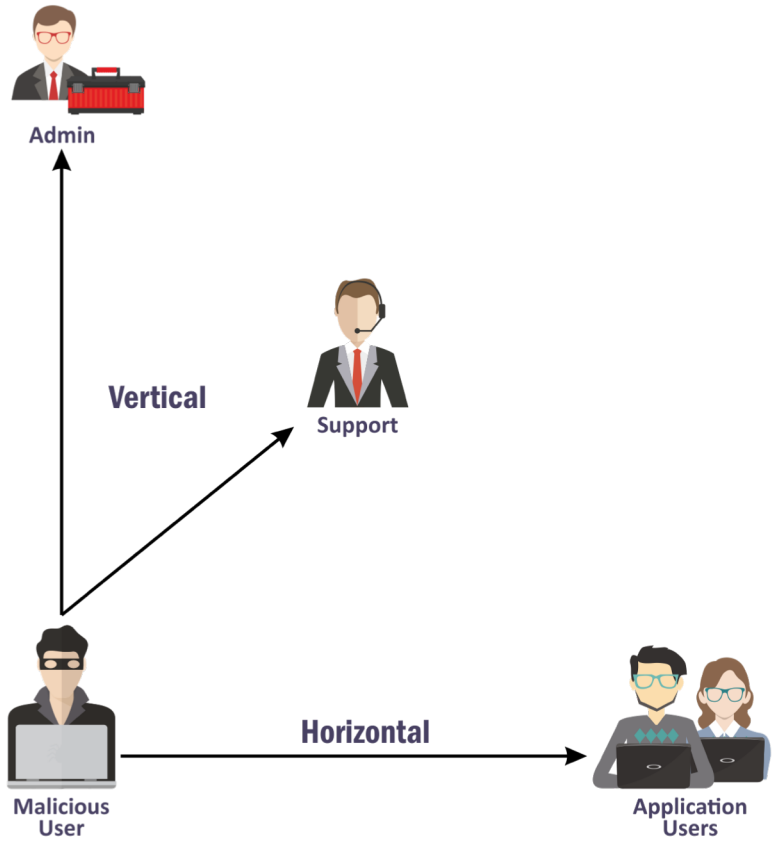
When is the application vulnerable?

- Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or simply using a custom API attack tool.
- Allowing the primary key to be changed to another users record, permitting viewing or editing someone else's account.
- Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.

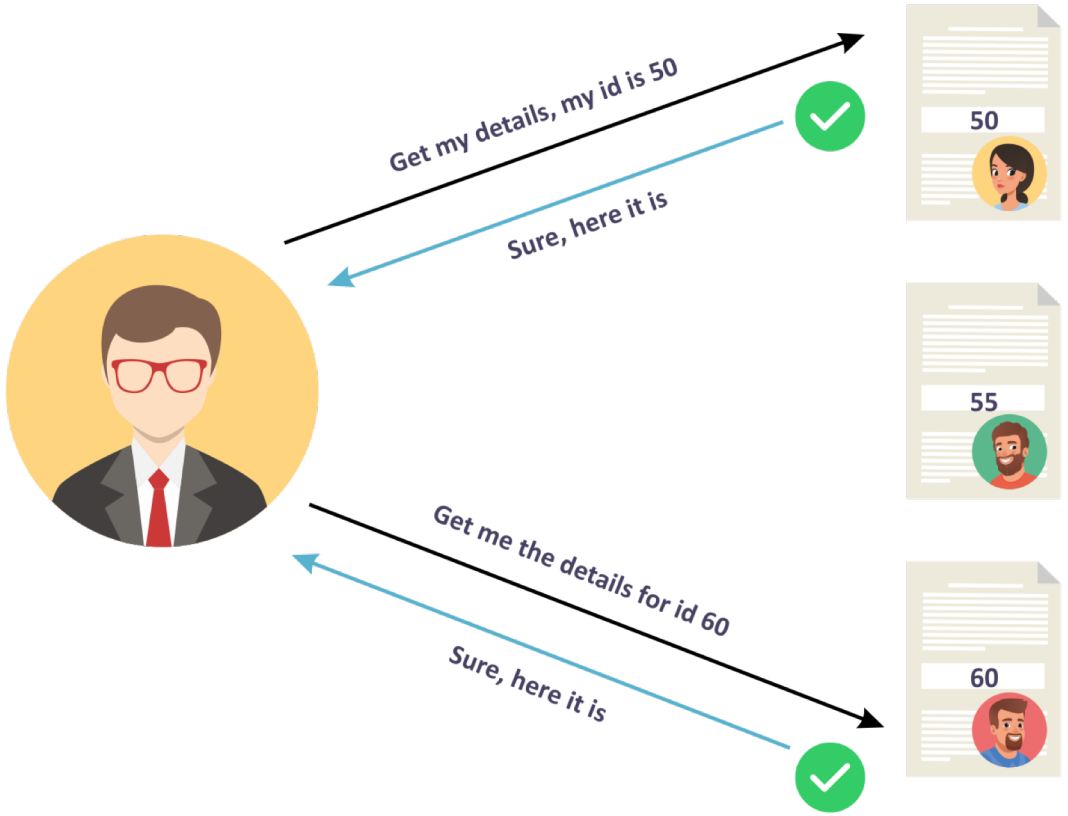
Source: OWASP

Access Control Categories

- Horizontal Access Control
- Vertical Access Control



Broken Access Control : IDOR



Lab: Insecure Direct Object Reference

Lab URL: https://attackdefense.com/challengedetails?cid=1907

Video URL: https://youtu.be/LF5DV_g7QBE

Lab: Insecure Direct Object Reference II

Lab URL: https://attackdefense.com/challengedetails?cid=1899

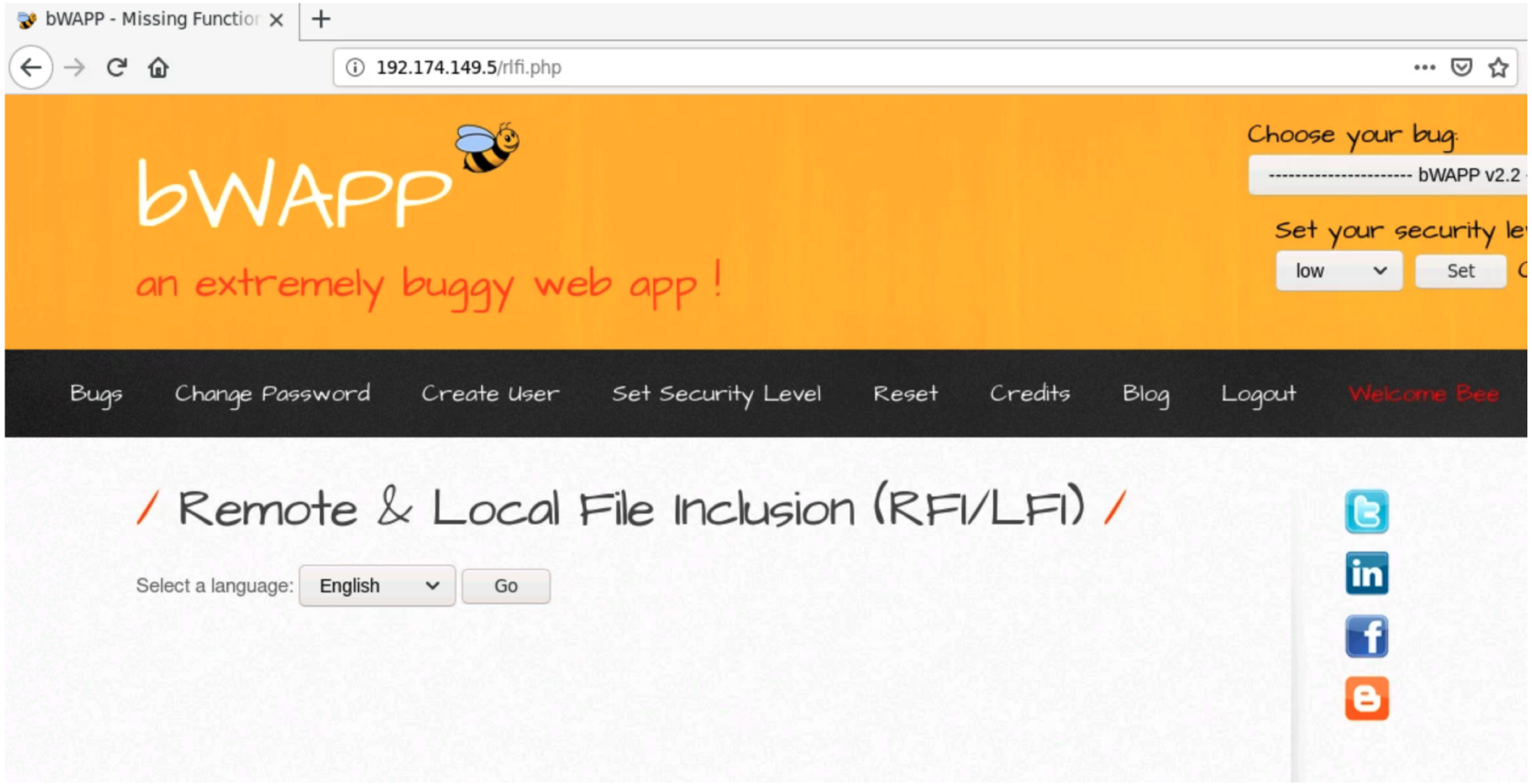
Video URL: https://youtu.be/K20DBHufewQ

Local File Inclusion

- Can lead to:
  - Sensitive Data Exposure
  - Remote Code Execution
  - Cross Site Scripting

Broken Access Control : Local File Inclusion

- bWAPP - Local File Inclusion (RFI/LFI)



Lab: Local File Inclusion

Lab URL: https://attackdefense.com/challengedetails?cid=1899

Video URL: https://youtu.be/IRZ1KKIMffo

BloofoxCMS : Local File Inclusion

```
// show file
if(isset($_POST["fileurl"])) {
    $fileurl = $_POST["fileurl"];
}
if(isset($_GET["fileurl"])) {
    $fileurl = "../".$_GET["fileurl"];
}

if(file_exists($fileurl)) {
    $filelength = filesize($fileurl);
    $readfile = fopen($fileurl,"r");
    $file = fread($readfile,$filelength);
    fclose($readfile);
}
```

Checks if GET or POST parameter is set

fileurl ⇒ /var/www/html/../../../etc/passwd
⇒ /etc/passwd

Read File if it exists

Lab: BloofoxCMS

Lab URL: https://attackdefense.com/challengedetails?cid=279

Video URL: https://youtu.be/R7fBm5YJoIE

PChart : Local File Inclusion

```
<?php

if ( isset($_GET["Action"]))

{

      $Script = $_GET["Script"];

            highlight_file($Script);

      exit();

 }

 ?>
```

Checks if GET or POST parameter is set

Prints out syntax highlighted version of the code

$Script ⇒ /var/www/html/../../../etc/passwd
⇒ /etc/passwd

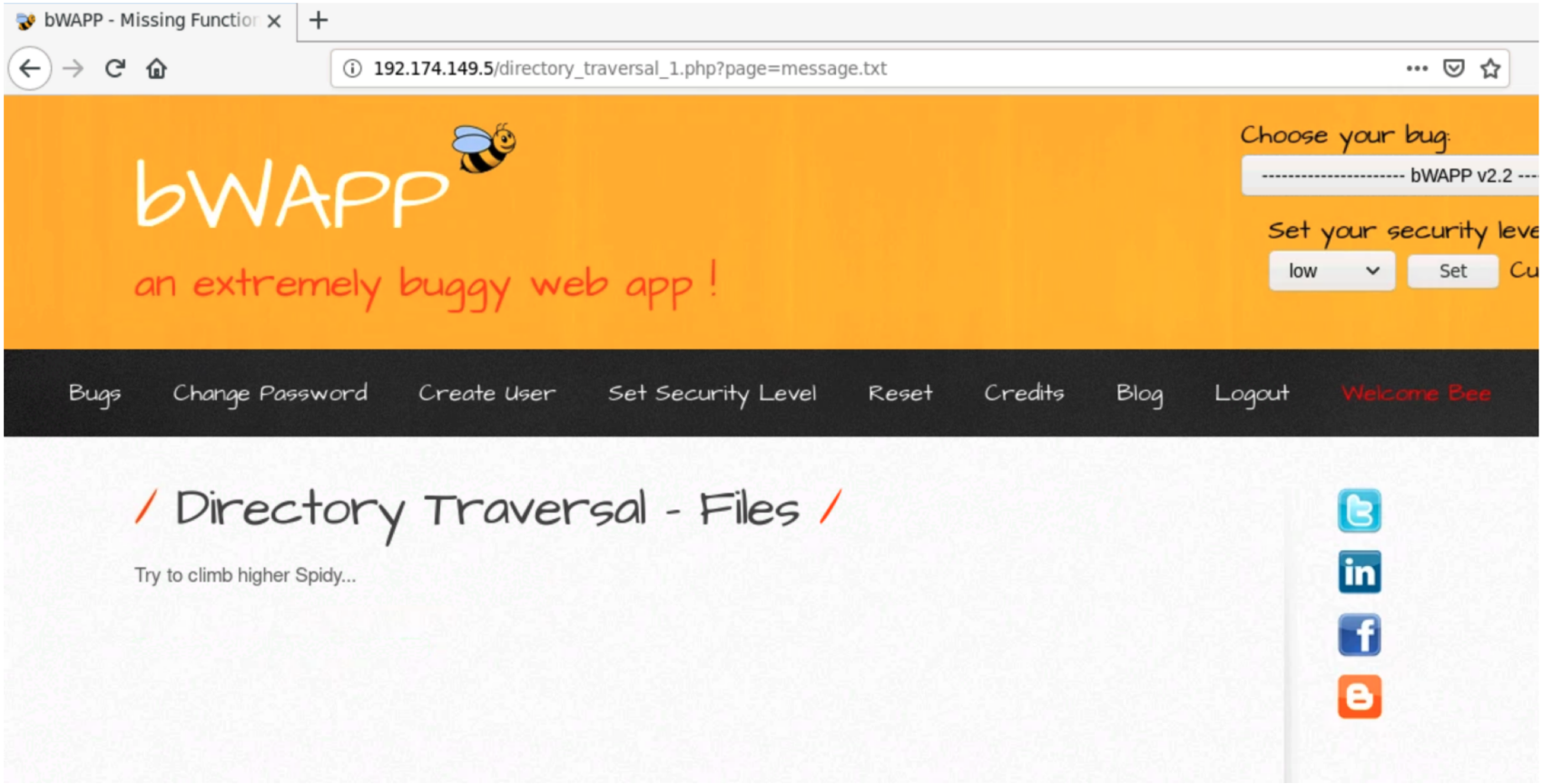
Lab: PChart

Lab URL: https://attackdefense.com/challengedetails?cid=277

Video URL: https://youtu.be/CXYQtkksS7A

Broken Access Control : Directory Traversal

- bWAPP - Directory Traversal - Files



Lab: Directory Traversal

Lab URL: https://attackdefense.com/challengedetails?cid=1899

Video URL: https://youtu.be/Sx26dpb7G6c