UCL CYBER SECURITY SOCIETY

INTRO TO WEB SECURITY

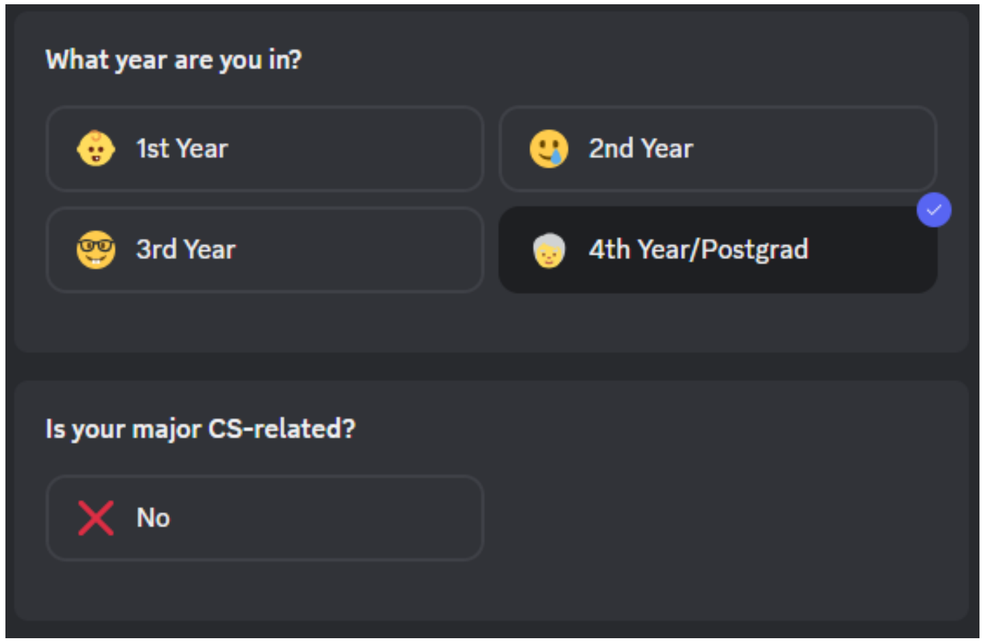
11 December 2024

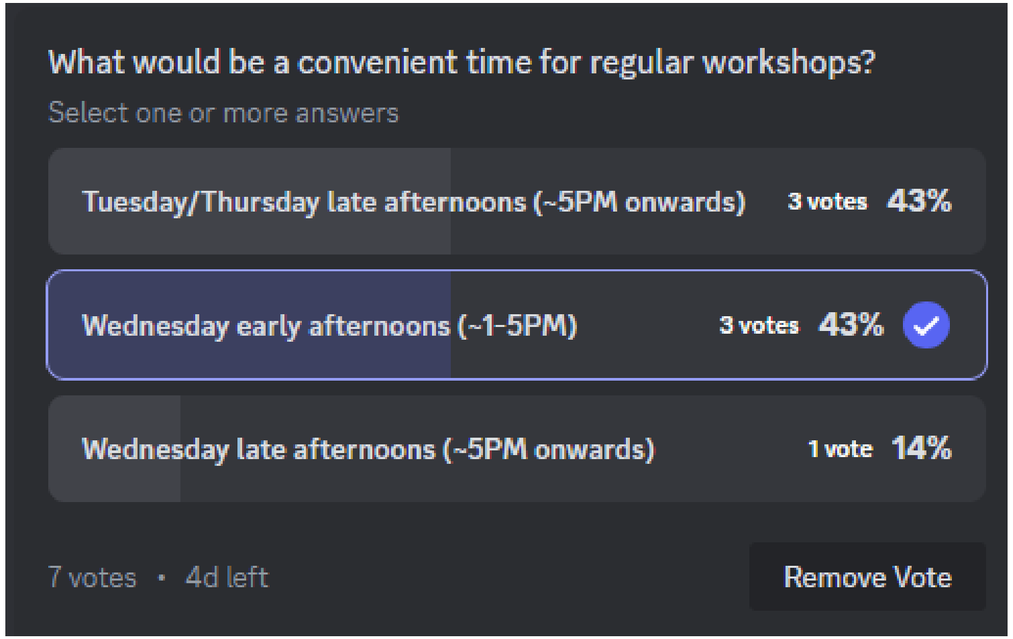
OVERVIEW

1. Intro to Web Security
2. Foundations
3. CTF Web Basics
4. Burp Suite

DISCORD NEWS

- You can get new roles!
- There’s a poll for workshop time next term (also on Whatsapp)





INTRO TO WEB SECURITY

WHAT

- Protect networks, servers, and computer systems
  - Unauthorized access
  - Unauthorized use
  - Unauthorized modification or destruction
- Related areas:
  - Cloud security
  - Network security
  - Application security

WHY

- Ensure smooth operation of business
- Service interruptions can be costly
- Sensitive information could be
  - Shared with competitors
  - Used to disable or hijack services
  - Put customer privacy at risk
- Financial loss, decreased productivity, damage to reputation, customer loss

HOW

- Web application firewall (WAF)
- Secure web gateway (SWG)
- Intrusion prevention system (IPS)
- URL filtering
- DNS controls
- TLS/SSL encryption
- Vulnerability scanners

WHO

- Developers
- Penetration testers
- Ethical hackers
- Security engineers
- Incident response teams
- System administrators
- CTF players!

EQUIFAX BREACH

- Loss of personal data of 147 million consumers
  - SSN, birth date, addresses, credit card details
- Vulnerability in Apache Struts web app
- Remote Code Execution flaw
- Importance of
  - Timely patching of known vulnerabilities
  - Efficient vulnerability management
  - Regular security testing

FOUNDATIONS

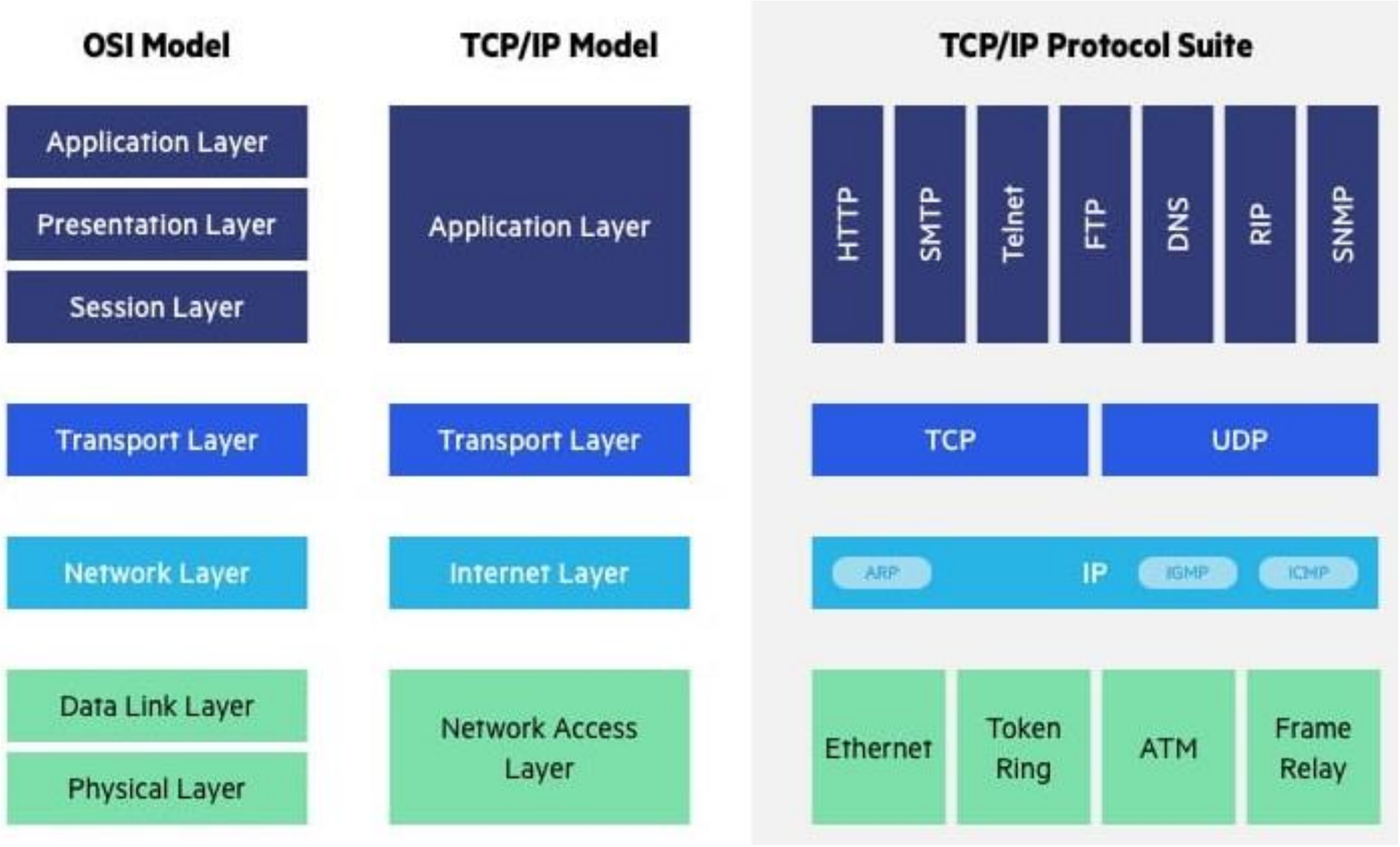
OSI MODEL

- Open Systems Interconnection (OSI) model
- 7 layers
  - Each responsible for a different aspect of communication
- Specifies tasks of protocols
- Standardises network communication development
- Broad, theoretical, protocol-independent framework

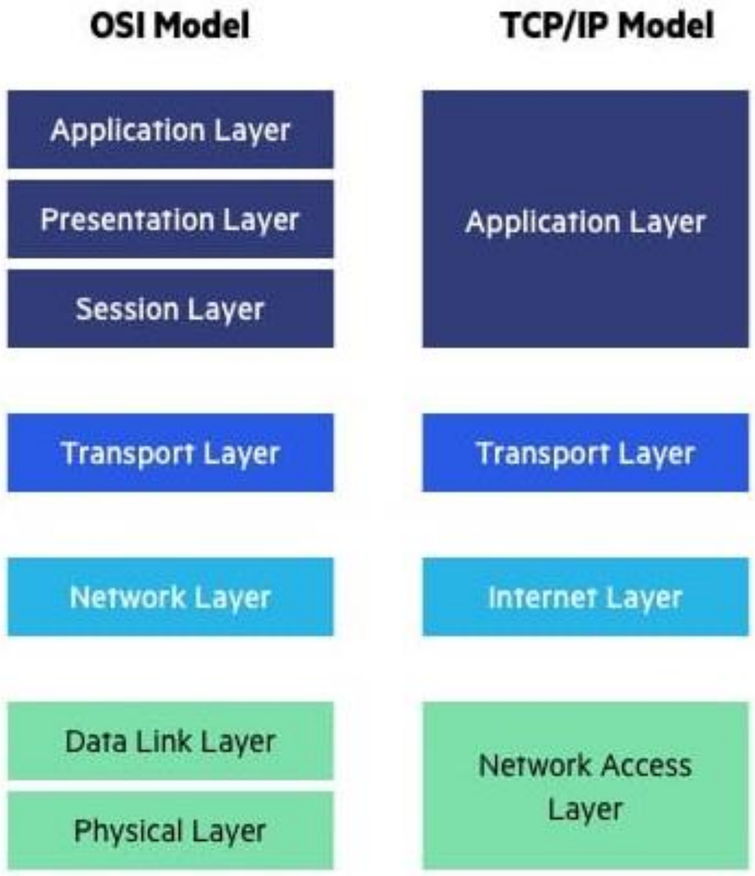
TCP/IP MODEL

- Transmission Control Protocol/Internet Protocol
- 4 layers
- More practical
- Relies on standardised protocols
- Concise version of OSI model

OSI VS TCP/IP

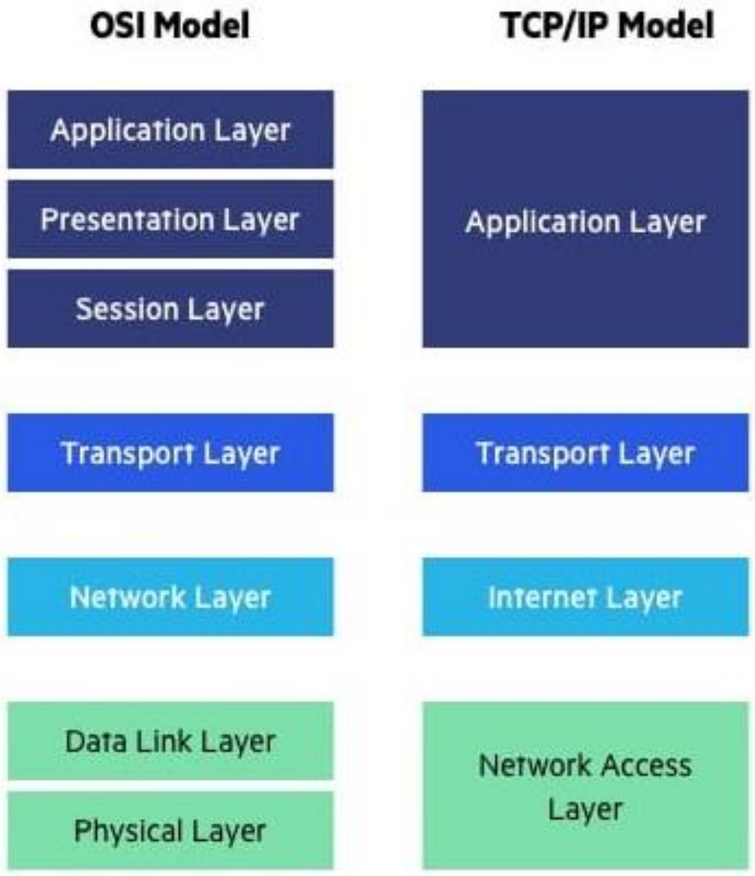

OSI Model
TCP/IP Model
TCP/IP Protocol Suite
Application Layer
Presentation Layer
Session Layer
Application Layer
HTTP
SMTP
Telnet
FTP
DNS
RIP
SNMP
Transport Layer
Transport Layer
TCP
UDP
Network Layer
Internet Layer
ARP
IP
IGMP
ICMP
Data Link Layer
Physical Layer
Network Access Layer
Ethernet
Token Ring
ATM
Frame Relay

OSI VS TCP/IP



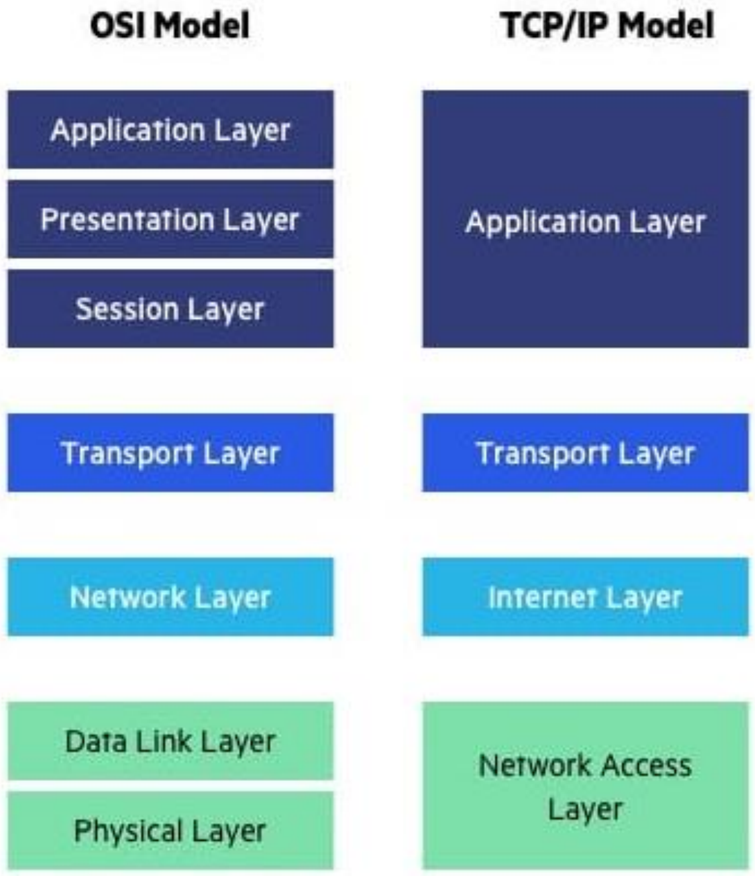
- Application layer
  - User interacts with this layer directly
  - Provides applications with access to network
  - HTTP, FTP, SSH

OSI VS TCP/IP



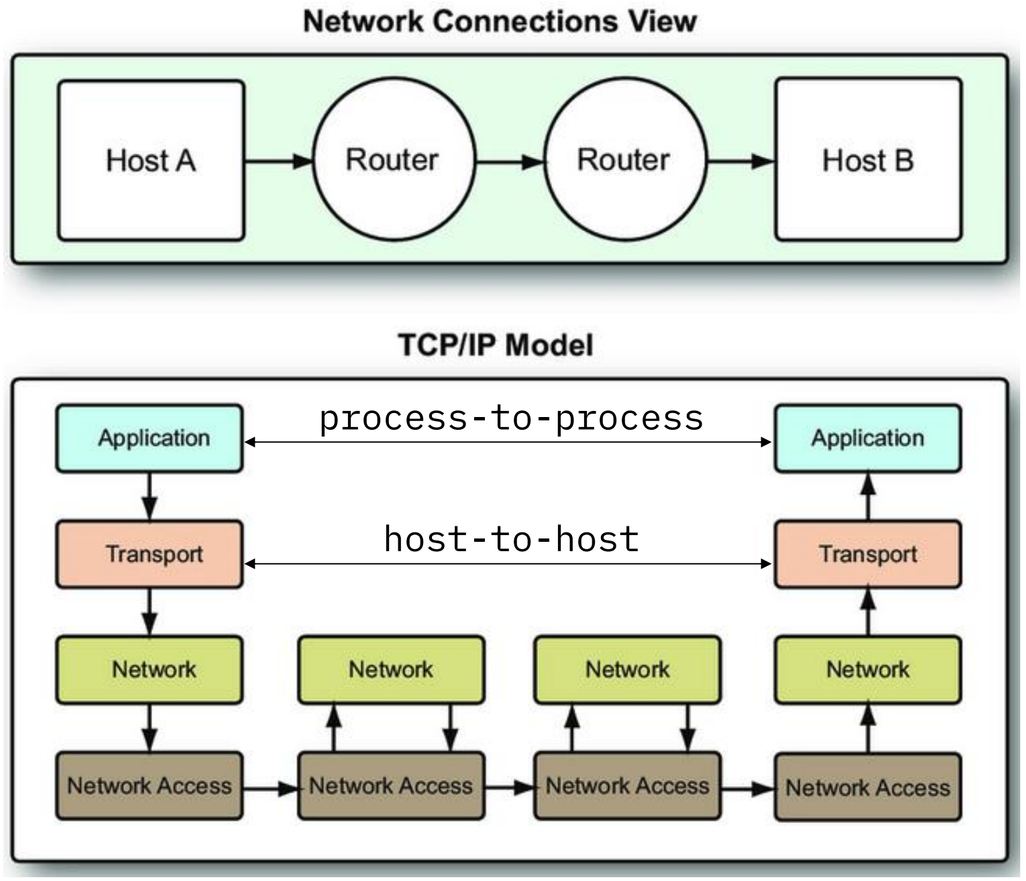
- Transport layer
  - Handles data transmission between hosts
  - TCP, UDP
- Internet layer
  - Responsible for routing data through the web
  - IP protocol

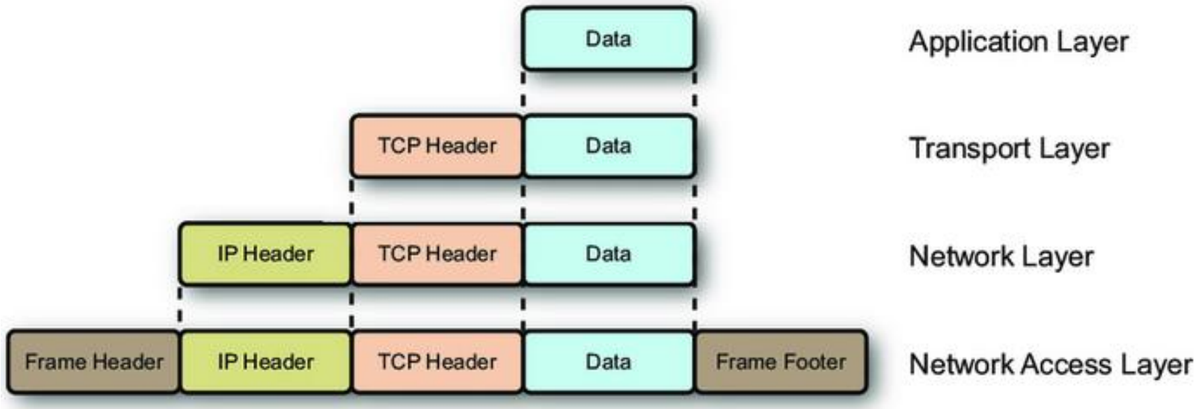
OSI VS TCP/IP



- Link layer
  - Provides reliable data links between two nodes
  - Ethernet, Wifi

TCP/IP


Network Connections View
Host A
Router
Router
Host B
TCP/IP Model
Application
process-to-process
Application
Transport
host-to-host
Transport
Network
Network
Network
Network
Network Access
Network Access
Network Access
Network Access


Data
Application Layer
TCP Header
Data
Transport Layer
IP Header
TCP Header
Data
Network Layer
Frame Header
IP Header
TCP Header
Data
Frame Footer
Network Access Layer

COMMON TERMS

- **Application**: Program that performs a specific task
  - Not related to computer itself
  - Performs a task for the user
- **Process**: Instance of a running program
  - Can create, listen on, and use sockets
  - Communicates through a socket bound to a port

COMMON TERMS

- **Socket**:
  Endpoint for sending/receiving data across a computer network
  - Allows process-to-process communication
  - Processes may be on different machines
  - Interface between application and transport layer
  - Socket address: IP address + port
- **Port**: Identifier used to send data to appropriate process

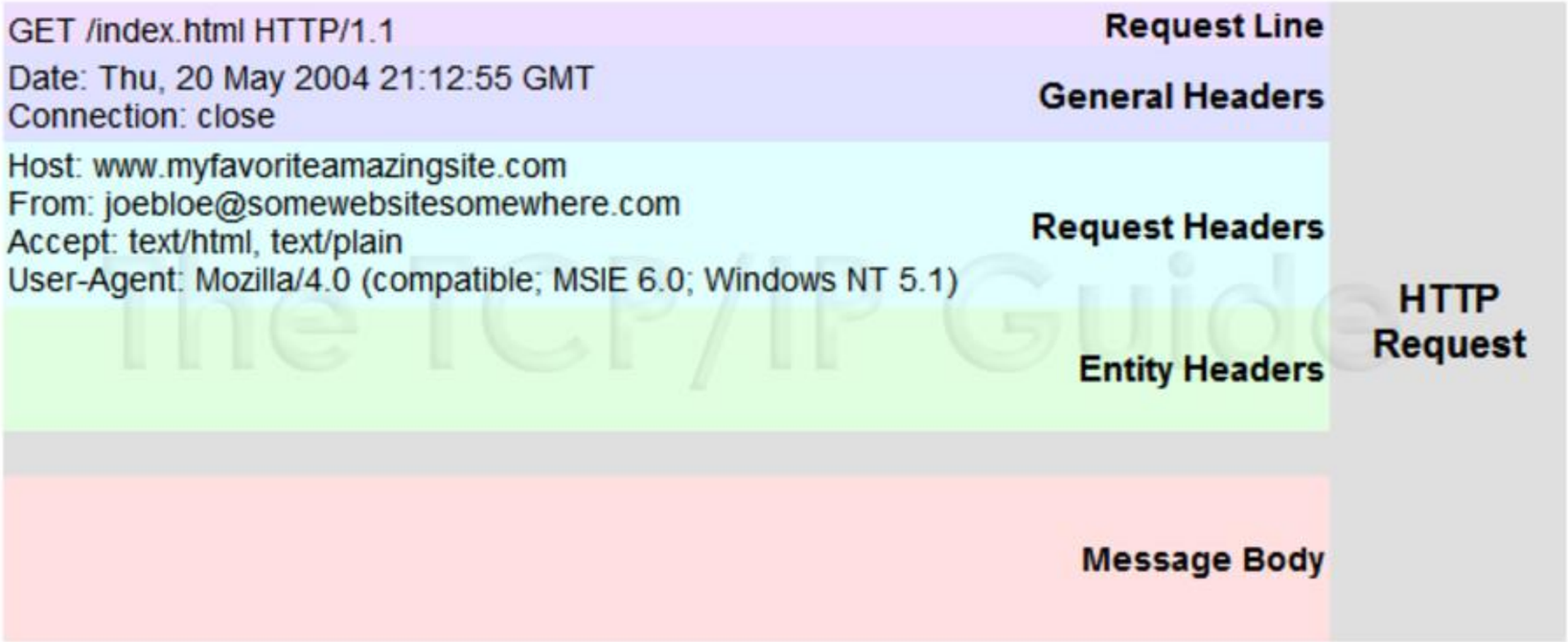
PUTTING IT TOGETHER

- TCP header: source + destination port
- IP header: source + destination address
- Data flow:
  - Application sends data to socket
  - Socket passes it to transport and internet layer
  - Packet is passed to network access layer
  - Packet is framed and handed to Network Interface Card (NIC) for transmission

HTTP BASICS

REQUEST

- HTTP request: ask for resource (e.g. HTML page or image)



HTTP BASICS

RESPONSE

- HTTP response: provides requested resource (if request successful)

HTTP/1.1 200 OK	**Status Line**	**HTTP Response**
Date: Thu, 20 May 2004 21:12:58 GMT Connection: close	**General Headers**	
Server: Apache/1.3.27 Accept-Ranges: bytes	**Response Headers**	
Content-Type: text/html Content-Length: 170 Last-Modified: Tue, 18 May 2004 10:14:49 GMT	**Entity Headers**	
<html> <head> <title>Welcome to the Amazing Site!</title> </head> <body> This site is under construction. Please come back later. Sorry! </body> </html>	**Message Body**	

HTTP BASICS

METHODS

- GET: retrieve resource from server
- POST: create new resource
  - Request body contains attributes of new resource
- PUT: replace existing resource
- HEAD: retrieve only headers of a resource
- DELETE: delete resource

Request body example:

```
{
  "name": "Sneakers",
  "color": "blue",
  "price": 59.95,
  "currency": "USD"
}
```

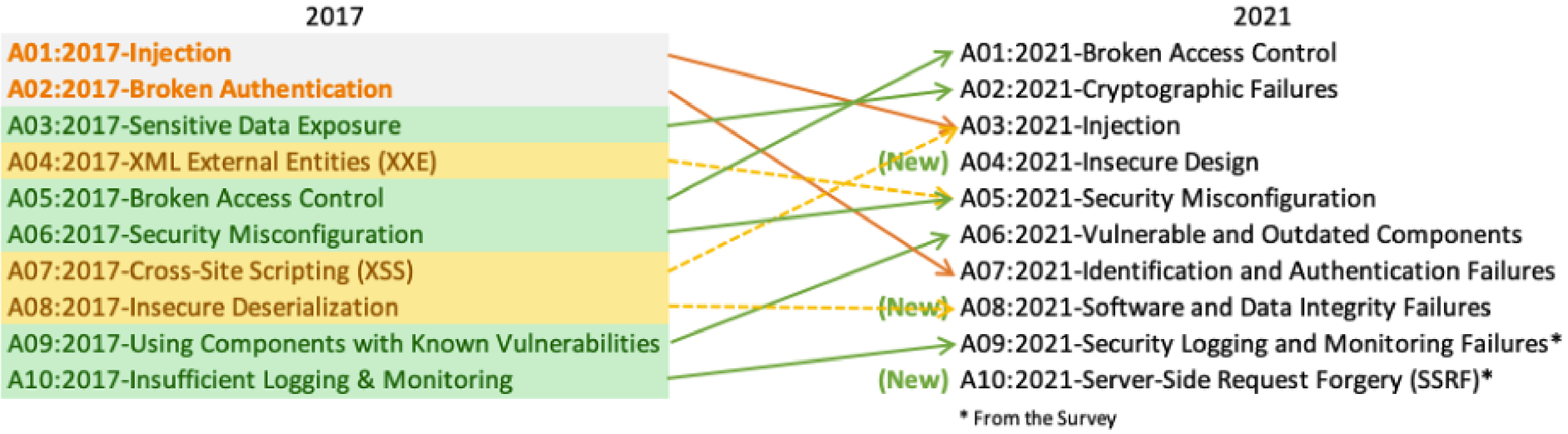
HTTP BASICS

GET VS POST

- GET:
  - query string (name/value pairs) sent in URL
  - Recorded in browser history
  - URL (including query parameters) is logged on the server
- Never use GET for passwords or other sensitive information
- Neither provide confidentiality without HTTPS

OWASP TOP 10

- Standard awareness document
- Most critical security risks to web applications



SQL INJECTION

- Extract, add or modify data
- Manipulate results of queries used for authentication
- SQL: standard language for interacting with databases
- Protections:
  - Sanitize input
  - Use prepared queries
- Routes: user input, cookies, server variables

SQL INJECTION

- Username: bob’ OR user<>’bob
- Password: foo OR pass<>’foo

```
$username = $HTTP_POST_VARS['username'];
$password = $HTTP_POST_VARS['passwd'];

$query = "SELECT * FROM logintable WHERE user = '"
    . $username . "' AND pass = '" . $password . "'";
...
$result = mysql_query($query);

if (!$results)
    die_bad_login();
```

```
SELECT * FROM logintable WHERE user=
'bob' or user<>'bob' AND pass='foo' OR pass<>'foo'
```

SQL INJECTION

- Normal usage: login="john" and pin="1234“
- Malicious usage: login="admin' --" and pin="0

```java
public class Show extends HttpServlet {
  public ResultSet getuserInfo(String login, String pin) {
    Connection conn = DriverManager.getConnection("MyDB"};
    Statement stmt = conn.createStatement();
    String queryString = "";

    queryString = "SELECT accounts FROM users WHERE ";
    if ((! login.equals("")) && (! pin.equals("))) {
        queryString += "login='" + login +
        "' AND pin=" + pin;
    } else {
        queryString+="login='guest'";
    }

    ResultSet tempSet = stmt.execute(queryString);
    return tempSet;
    }
}
```

```sql
SELECT accounts FROM users WHERE login='john' AND pin=1234
```

```sql
SELECT accounts FROM users WHERE login='admin' --' AND pin=0
```

LOCAL FILE INCLUSION

- Trick app into running or exposing files on a web server
- Can expose sensitive information
- Can lead to remote code execution
- App uses path as input to retrieve files
- Simple filters can be bypassed, e.g. using URL encoding

LOCAL FILE INCLUSION

- File store path: root/example/all_files/file_sharing/uploads
- Malicious input for downloading files: example.com/download?file=../../../../etc/passwd
- Malicious input for executing script: example.com/download?file=evil.php

```php
<?php
  //get file name from user
   $file = $_GET['file'];
  //Reteieve file
 include(",/uploads/$file");
?>
```

IDOR

- Indirect direct object references (IDOR)
- Type of access control vulnerability
- App uses user-supplied input to access objects directly
- Can lead to privilege escalation

IDOR

- App might use URL like this to access profile:
  https://example.com/users/123
  → attacker can change number to access another user's profile

- Sensitive files might be located in static files and use incremented file names
  → attacker can retrieve them like this:
  https://example.com/static/12144.txt

- Identifier might be in POST body
  → attacker can modify user_id field

```
<form action="/update_profile" method="post">
  <!-- Other fields for updating name, email, etc. -->
  <input type="hidden" name="user_id" value="12345">
  <button type="submit">Update Profile</button>
</form>
```

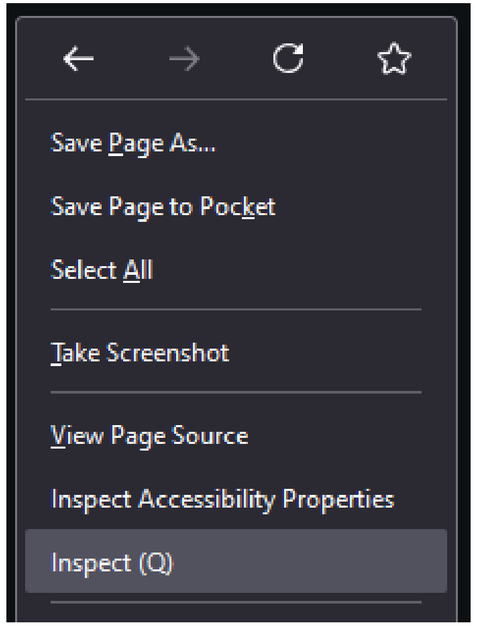
CTF WEB BASICS

APPROACH

- Look for hint in challenge description
- Front-end vulnerability?
  → use browser’s inspect tools
  → look at sources loaded, requests sent, etc.
  → look at Javascript
- Backend vulnerability?
  → use a tool like Burp Suite
  → view, modify, and resend HTTP requests
  → understand how app works

VIEWING SOURCE

- Right click on webpage and select “Inspect” to see the code that the website is running on your computer
- HTML and CSS
- Javascript scripts
- Edit HTML directly and see it affect the website
- See requests made



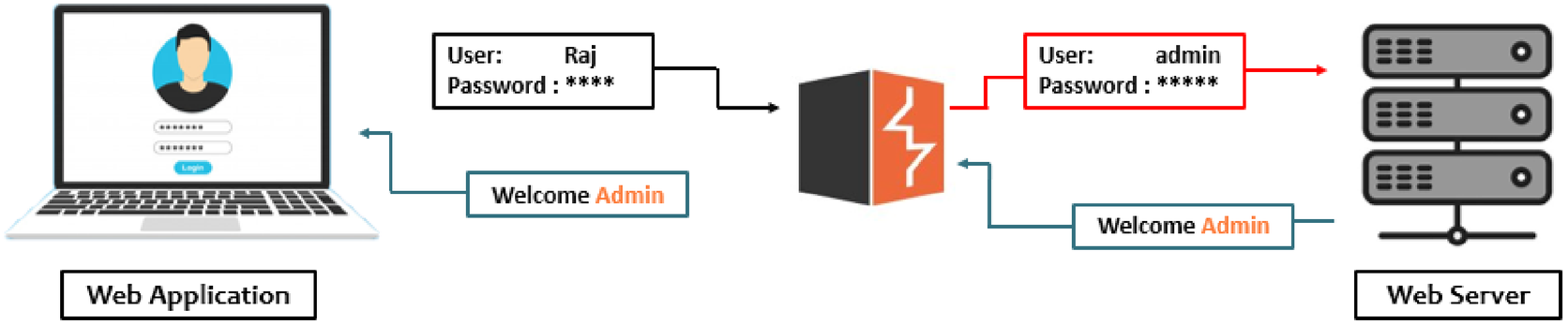
BURP SUITE

INTRO

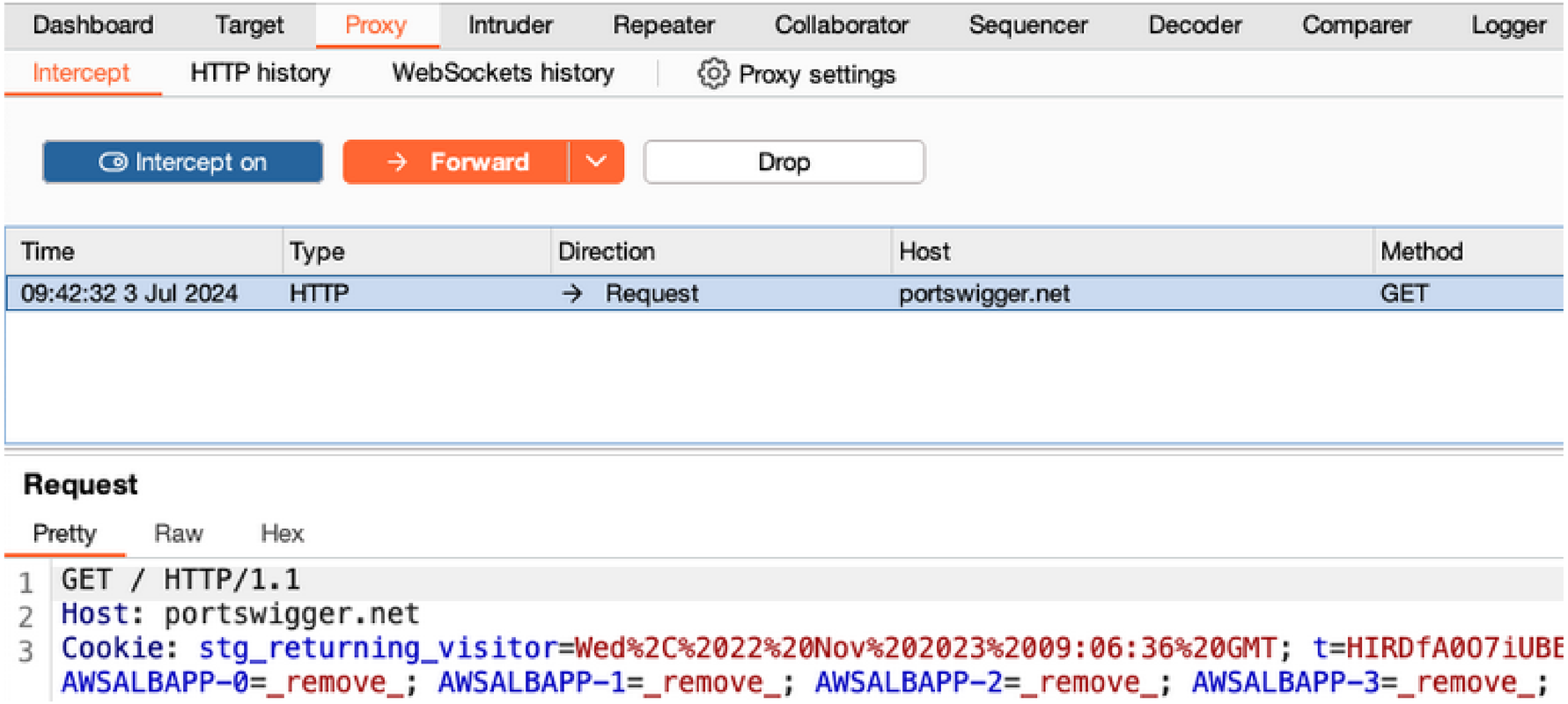
- Collection of tools for testing web application security
- Some features:
  - HTTP proxy
  - Web app security scanner
  - Attack automation
  - Plugin API with lots of third-party addons
- Free version only has essential manual tools

BURP PROXY

- Intercept HTTP traffic for analysis and playback
- HTTP requests can be modified before being forwarded
- Study how website behaves



BURP PROXY


Dashboard
Target
Proxy
Intruder
Repeater
Collaborator
Sequencer
Decoder
Comparer
Logger
Intercept
HTTP history
WebSockets history
Proxy settings
Intercept on
Forward
Drop
Time
Type
Direction
Host
Method
09:42:32 3 Jul 2024
HTTP
Request
portswigger.net
GET
Request
Pretty
Raw
Hex
GET / HTTP/1.1
Host: portswigger.net
Cookie: stg_returning_visitor=Wed%2C%2022%20Nov%202023%2009:06:36%20GMT; t=HIRDfA0O7iUBE
AWSALBAPP-0=_remove_; AWSALBAPP-1=_remove_; AWSALBAPP-2=_remove_; AWSALBAPP-3=_remove_;

DEMO: GET AHEAD

THANK YOU FOR COMING!!

Feedback Form