FORTINET | USA

XPERTS

SUMMIT 2024

Helping you create a digitally secure future.

Introduction to HTTP

Module 01

Agenda

- Introduction to HTTP
- HTTP Requests
- HTTP Response
- HTTP Authentication
- Secure HTTP
- Overview of OWASP Top 10
- Tools for HTTP development and testing

XPERTS SUMMIT 2024

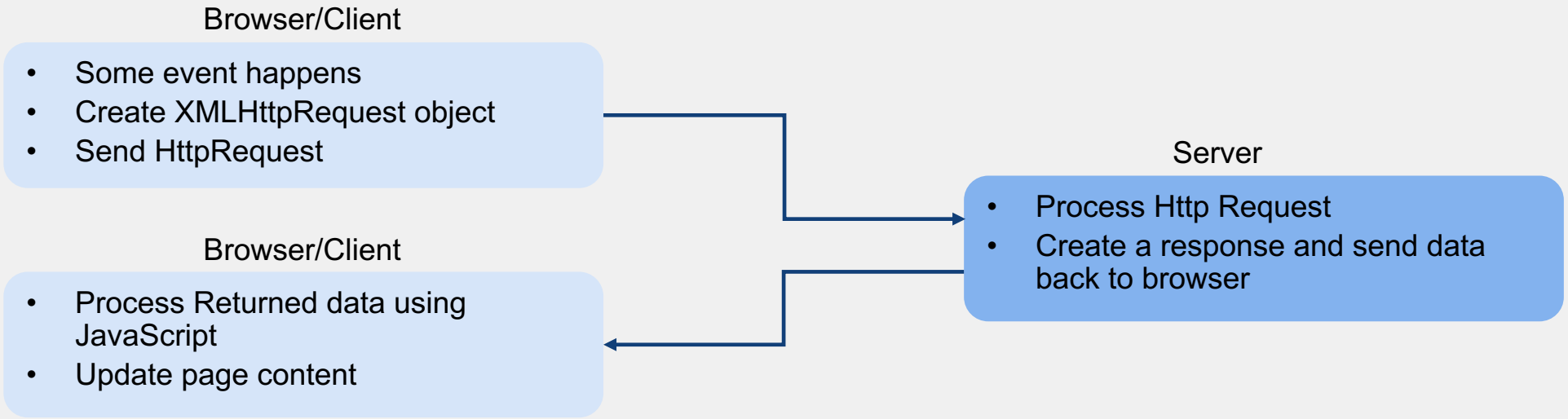
Introduction to HTTP

Sub header

What is HTTP?

Hypertext Transfer Protocol:

is an application layer (OSI layer 7) protocol designed to transfer information between networked devices.
is a client/server protocol running on top of TCP (OSI layer 4)
Communication is based on requests (client) and responses (server)
is the foundation of the World Wide Web, and is used to load webpages using hypertext links.



HTTP Versions

HTTP 1.1: First standardized HTTP protocol, released in 1997

- Underlying TCP connections could be reused for multiple requests
- Pipelining – allowed a second request to be sent before the answer to the first was fully transmitted
- Content negotiation (between client and server), including language, encoding, and type.
- Addition of Host header allowed hosting different domains from the same IP address for server collocation.

HTTP 2: designed for greater performance compared to 1.1

- Binary (not text) protocol which allowed for the implementation of improved optimization techniques
- Multiplexing – enabling parallel requests, made over the same connection
- Compressed Headers – removes duplication and overhead of transmitted data
- Server Push – server can proactively send site assets to the client before they have asked for them*

HTTP 3: designed to further enhance performance and security

- **QUIC (Quick UDP Internet Connections*)**
  - Normally (with TCP) multiple round trips are required: 3-way handshake to establish session, TLS 1.3 handshake to authenticate and negotiate cryptographic parameters
  - QUIC combines these functions
  - Lost data and re-transmits are handled by QUIC (since UDP in connectionless)

XPERTS SUMMIT 2024

HTTP Requests

HTTP Requests

- **HTTP version type**: 1.1, 2 or 3
- **URL**: Uniform Resource Locator is the address of a unique resource on the internet.
- **HTTP request methods (verbs)**: indicate the desired action to be performed on the resource.
- **HTTP request headers**: key-value pairs contained in every request, which are used to communicate information and context to the server
- **HTTP request body (optional)**: generally used in a PUT or POST. Includes data required to fulfill the request, for instance HTML form data.

HTTP Request Methods

- **GET**: requests a representation of the specified resource. Requests with this method should only retrieve data.
- **POST:** submits an entity to the specified resource, usually causing a change in state or other effect on server
- **PUT:** replaces all current representations of the target resource with the request payload
- **HEAD:** requests the headers associated with a resource, but does not return the resource itself
- **DELETE:** deletes the specified resource
- **CONNECT:** establishes a tunnel to the server identified by the target resource. Generally used to encrypt traffic such that the data is unreadable to a proxy
- **OPTIONS:** describes the communication options for the target resource
- **TRACE:** performs a message loop-back test along the path to the target resource
- **PATCH:** applies partial modifications to a resource

XPERTS SUMMIT 2024

HTTP Response

HTTP Response Status Codes

- 1xx: Information
- 2xx: Successful
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

Cookies! – What is cookie?

a.k.a. **“web cookie”** or **“browser cookie”** : A small piece of data sent from server to a user’s web browser.

Delicious Confectionary treat…

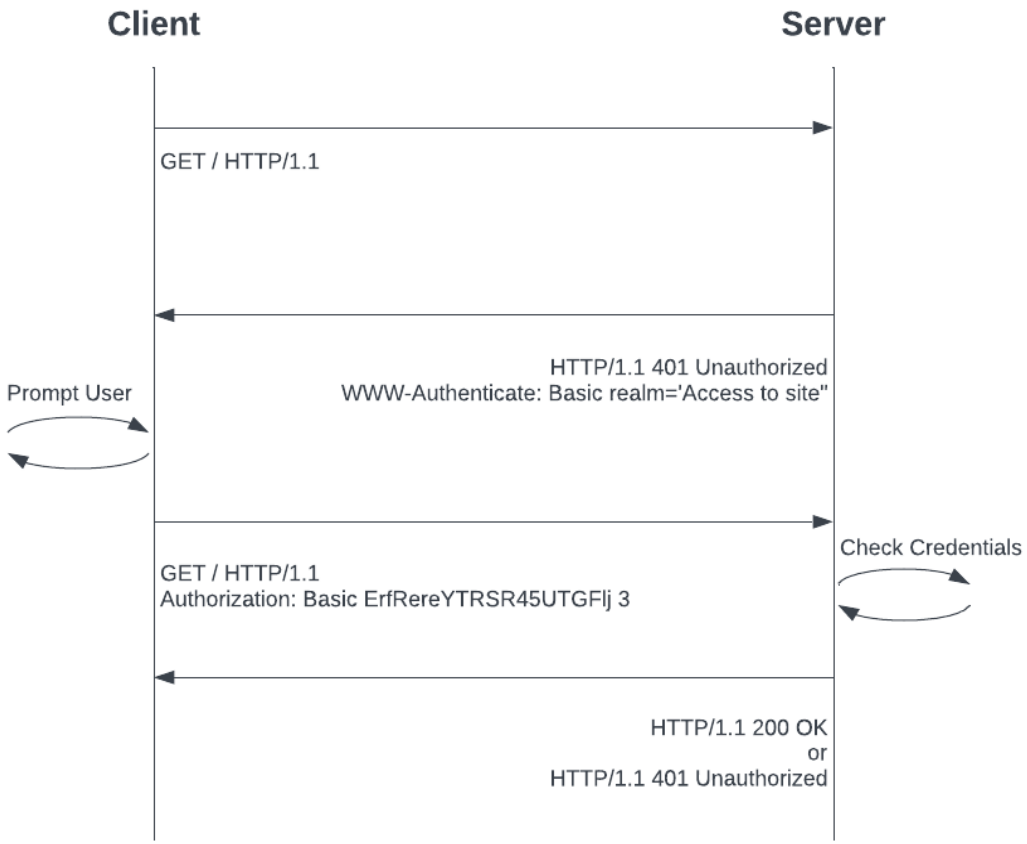
XPERTS SUMMIT 2024

HTTP Authentication

General HTTP authentication framework: RFC 7235

Challenge/Response workflow

1. **Client** makes the initial request
2. **Server** responds with 401 (unauthorized) and provides WWW-Authenticate response header to explain how to authorize
3. **User** is prompted to input credentials
4. **Client** will respond with Authorization request header with the provided credentials
5. **Server** checks creds and responds



Auth Related Response Status Codes

- **401: Unauthorized**

Request has not been completed because it lacks valid auth credentials for the requested resource. Sent with WWW-Authenticate response header sent describing how client can authenticate

- **403: Forbidden**

Server understands the request but refuses to authorize it. No re-authentication attempt will be presented

- **407: Proxy Authentication Required**

Request has not been applied because it lacks valid authentication credentials for the proxy server which is between the browser and the server. Sent with Proxy-Authenticate response header describing how client can authenticate

- **404: Not Found**

This may be returned by the server in the case of inadequate privileges or lack of authentication, in order to hide the existence of the page.

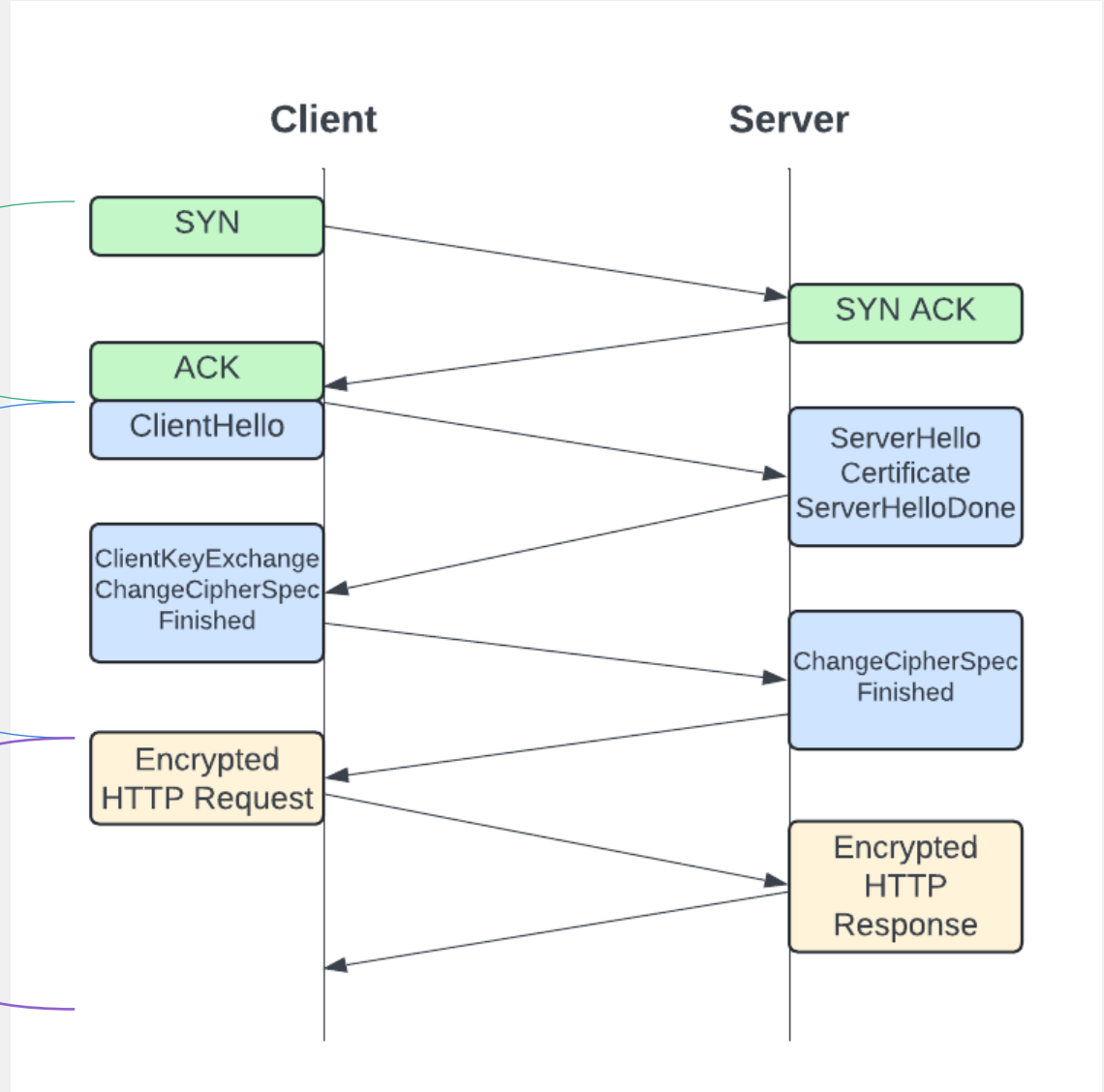
Authentication Schemes

Authenticaton Scheme	Description
Anonymous	No authentication info is presented. Equivalent to granting everyone access to the resource
Basic	Basic authentication uses base64 encoding with username and password
Bearer	(a.k.a. token authentication) client must send a token in the Authorization header when making request. OAUTH and JWT are examples
Digest	Server applies MD5 cryptographic hashing and a nonce value to prevent replay attacks. Hash values are affixed to the person's username and password before they are sent over the network, enabling the provider's server to authenticate the client.
HOBA	HTTP Origin-Bound authentication. Not password based but instead uses public-private keys.
Mutual	Both client and server are authenticated using Digital Certificates over TLS
Negotiate/NTLM	Windows authentication using Kerberos.
VAPID	Voluntary Application Server Identification authentication is designed to allows sites to authenticate with push servers independently

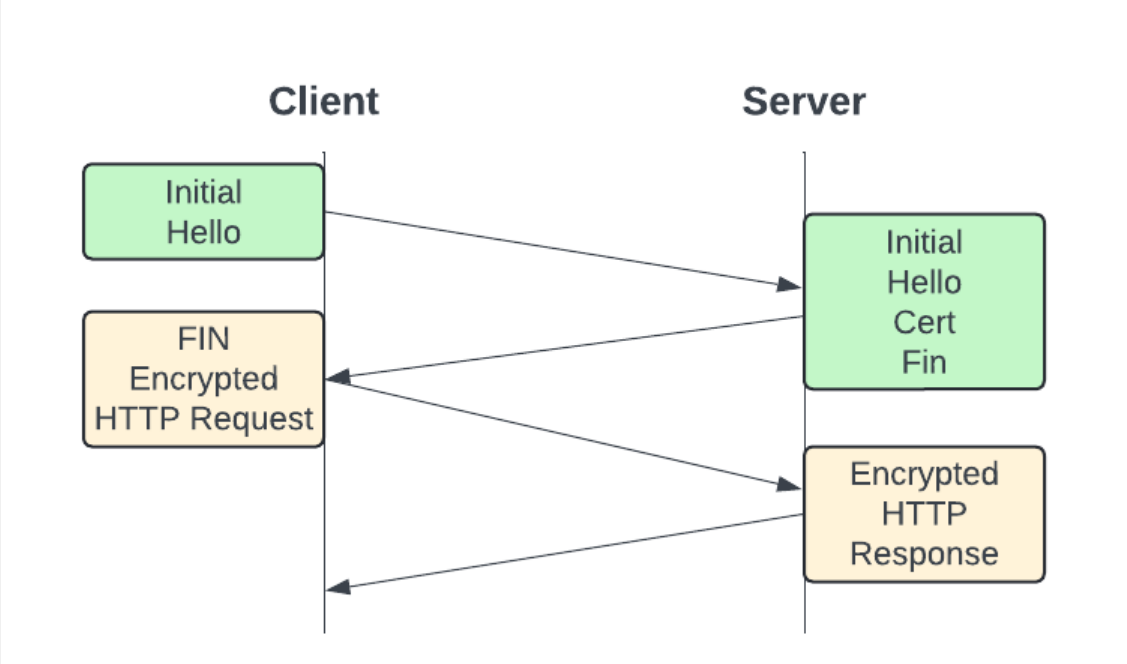
XPERTS SUMMIT 2024

Secure HTTP

HTTP over Transport Layer Security (TLS)



HTTP over QUIC



XPERTS
SUMMIT 2024

Overview of OWASP Top 10

Overview

- A01:2021 – Broken Access Control
- A02:2021 – Cryptographic Failures
- A03:2021 - Injection
- A04:2021 – Insecure Design
- A05:2021 – Security Misconfiguration
- A06:2021 – Vulnerable and Outdated Components
- A07:2021 – Identification and Authentication Failures
- A08:2021 – Software and Data Integrity Failures
- A09:2021 – Security Logging and Monitoring Failures
- A10:2021 – Server-Side Request Forgery

A01: Broken Access Control

- **Risk Description**

Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

- **Attack Example**

An attacker modifies data in a SQL call that is accessing account information

- **Explanation**

Often occurs in systems where functional access controls are not properly implemented using the principles of least privilege

A02: Cryptographic Failures

- **Risk Description**

Failure to encrypt (or properly encrypt) sensitive data in transit.

- **Attack Example**

Application data is encrypted in the database, but Data is automatically decrypted when retrieved, allowing a SQL injection flaw to retrieve credit card numbers in clear text.

- **Explanation**

Assuming data is encrypted in transit, this issue can occur when default or weak crypto keys and algorithms are used.

A03: Injection

- **Risk Description**

Injection attacks occur when attackers exploit vulnerabilities in an application to send malicious code into a system. This can allow them to execute unauthorized commands, access data, or manipulate the system's operations.

- **Attack Example**

An attacker sends "101 OR 1=1" instead of "101" to change SQL queries run against a database.

- **Explanation**

This can occur when proper input validation is not used, and can be exacerbated by improper or lack of SQL controls.

A04: Insecure Design

- **Risk Description**

Insecure design is a broad category of architectural and design flaws in web applications that can be exploited by hackers.

- **Attack Example**

Insecure design can lead to a number of other attacks, by failing to consider them as possibilities.

- **Explanation**

This occurs when software or systems are created without considering security implications from the beginning. This can result in vulnerabilities that attackers can exploit to gain unauthorized access or cause damage.

A05: Security Misconfiguration

- **Risk Description**

Improper configuration of security settings, permissions, and controls that can lead to vulnerabilities and unauthorized access

- **Attack Example**

An attacker could move from a single compromised device within the victim's infrastructure, due to lack of proper segmentation.

- **Explanation**

This is a broad topic, encompassing many different possible issues. This can occur in an environment where security is not a priority, or the team responsible for security is poorly staffed and/or trained.

A06: Vulnerable and Outdated Components

- **Risk Description**

Software components in web applications that have security flaws or are no longer supported by their developers.

- **Attack Example**

Attackers exploit a vulnerability in Apache Struts, an open-source framework ,in order to steal the personal information of customers.

- **Explanation**

This often occurs when developers use third party libraries with known vulnerabilities, or have not been kept up to date.

A07: Identification and Authentication Failures

- **Risk Description**

Security vulnerabilities that occur when a system or application is unable to correctly identify or authenticate a user

- **Attack Example**

An attacker uses credential stuffing lists to access sensitive data or resources.

- **Explanation**

This attack most commonly occurs when applications do not implement Multi-Factor authentication.

A08: Software and Data Integrity Failures

- **Risk Description**

Vulnerabilities in software or infrastructure that allow an attacker to modify or delete data in an unauthorized manner.

- **Attack Example**

An attacker brute forced SolarWinds and was able to move laterally within the environment. They introduced a malicious component into the CI/CD pipeline. This was then downloaded by 18,000 organizations and led to further exploit of those organizations

- **Explanation**

This attack can happen when an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). Organizations should use digital signatures to ensure software is coming from trusted sources.

A09: Security Logging and Monitoring Failures

- **Risk Description**

Security vulnerabilities that are exploited when an application or system doesn't log or monitor security events properly.

- **Attack Example**

A children's health plan provider's website operator couldn't detect a breach due to a lack of monitoring and logging. An external party informed the health plan provider that an attacker had accessed and modified thousands of sensitive health records of more than 3.5 million children.

- **Explanation**

This is likely to occur in organizations where security logs are not properly gathered or retained. In extreme cases where organizations don’t have enough skilled resources, these logs may be present, but are un-noticed.

A10: Server-Side Request Forgery (SSRF)

- **Risk Description**

Allows an attacker to trick a server-side application into making requests to an unintended location

- **Attack Example**

Sensitive data exposure – Attackers can access local files or internal services to gain sensitive information such as file:///etc/passwd and http://localhost:28017/.

- **Explanation**

In an SSRF attack, the attacker can supply or modify a URL that the server's code will read or submit data to. By carefully selecting the URLs, the attacker may be able to: read server configuration, connect to internal services, and perform post requests towards internal services.

XPERTS SUMMIT 2024

Tools for HTTP

Tools

- **Burpsuite**

  A set of tools used for penetration testing of web Applications. Community edition is free. Professional and enterprise editions can be purchased.

  https://portswigger.net/burp

- **ZAP**

  Free open-source proxy tool that helps identify security vulnerabilities in web applications

  https://www.zaproxy.org/

- **Wireshark**

  Free open-source network analyser that captures and displays network traffic in real time.

- **Browser Developer Tools**

  Free with most modern browsers. They allow developers to inspect and debug a website's content and resources.

  https://developer.chrome.com/docs/devtools/open


FORTINET