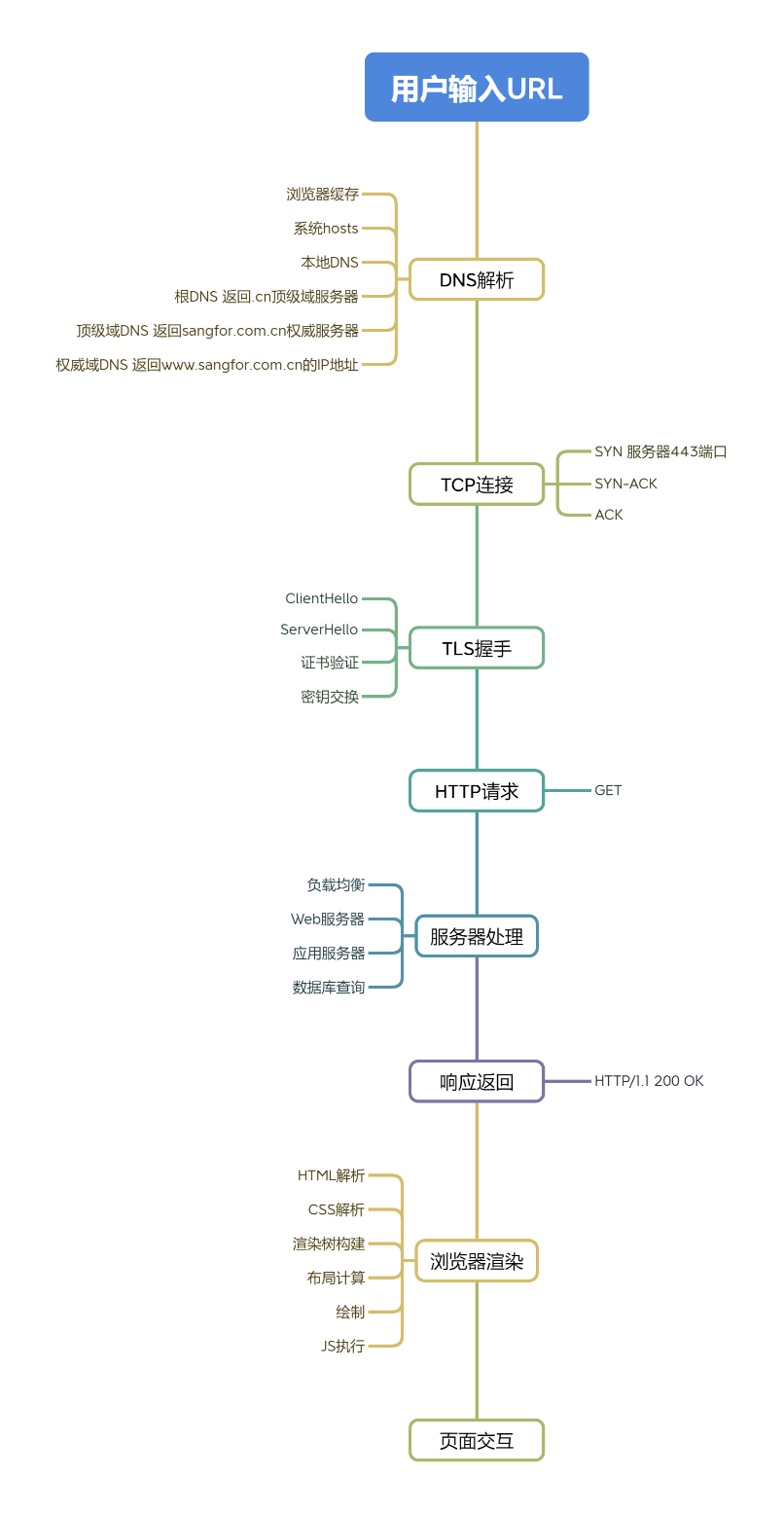
实验报告六

|  |  |  |
| --- | --- | --- |
| **姓名：彭彦杰** | **时间：2025.06.14** | **评分：** |

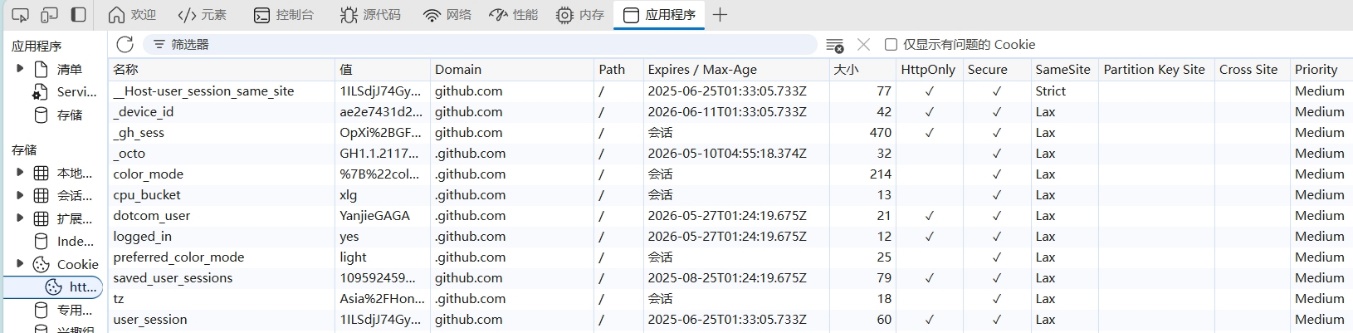
# 深信服安全基础

## 网站工作原理



## 网站Cookie的作用

**1. GitHub的Cookie**



**2. 关键 Cookie 解析**

* user\_session：

作用：登录会话凭证

安全属性：HttpOnly + Secure

特点：加密内容，有效期较长

* logged\_in：

作用：标记登录状态

特点：会话级Cookie（关闭浏览器即失效）

* preferred\_color\_mode：

作用：存储用户界面主题偏好

特点：长期有效

* \_gh\_sess：

作用：服务端会话标识

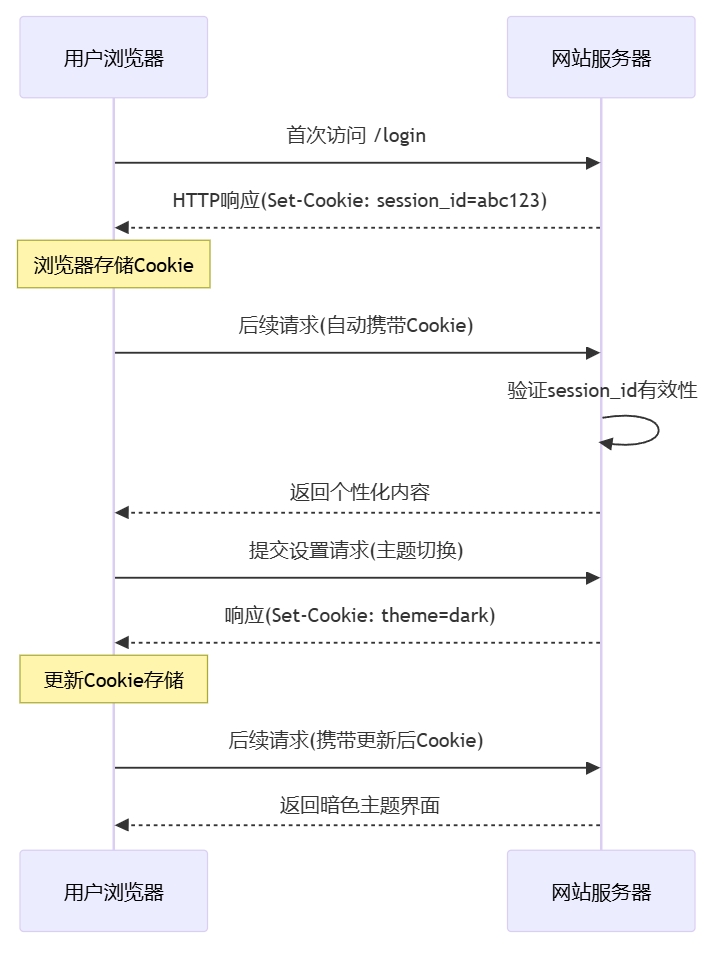
安全属性：HttpOnly + Secure

**3. Cookie 核心****应用场景**

表1.1 Cookie 的应用场景

|  |  |  |
| --- | --- | --- |
| **场景类型** | **具体应用** | **安全要求** |
| **身份认证** | 用户登录状态保持 | 必须HttpOnly+Secure |
| **个性化设置** | 界面主题/语言偏好 | 可非HttpOnly |
| **购物车** | 电商网站暂存选中商品 | 建议Secure |
| **用户行为追踪** | 访问路径分析/广告转化 | 需用户授权 |
| **A/B测试** | 用户分组展示不同版本 | 可非HttpOnly |
| **跨站点单点登录** | 同一公司不同服务共享登录状态 | 必须HttpOnly+Secure+SameSite |

**4. Cookie 工作原理**



## CSRF vs XSS vs SQL注入：核心区别

|  |  |  |  |
| --- | --- | --- | --- |
| **维度** | **CSRF攻击** | **XSS攻击** | **SQL注入攻击** |
| **攻击目标** | 利用用户身份执行操作 | 窃取用户数据/会话 | 窃取/篡改数据库 |
| **利用点** | 用户已登录+访问恶意网站 | 网站存在未过滤的用户输入点 | 用户输入拼接到SQL语句时 |
| **攻击位置** | 跨站请求（不同域名） | 当前网站内执行恶意脚本 | 服务端数据库操作 |
| **核心防御** | CSRF Token + SameSite Cookie | 输出编码 + CSP | 参数化查询 + 输入过滤 |
| **危害特点** | 以用户名义执行操作 | 客户端数据泄露 | 服务端数据泄露 |

关键区别：

* CSRF利用的是浏览器对Cookie的自动携带机制
* XSS利用的是网站对用户输入的信任
* SQL注入利用的是数据库对查询语句的信任

## UTM vs NGFW 本质区别

|  |  |  |
| --- | --- | --- |
| **维度** | **UTM** | **下一代防火墙（NGFW）** |
| **设计理念** | 功能堆砌 | 深度集成 |
| **检测引擎** | 多引擎串联（性能瓶颈） | 单通道并行检测（高性能） |
| **核心能力** | 基础威胁防护 | 应用/用户/内容深度识别 |
| **策略配置** | 独立策略管理 | 统一策略：允许销售组使用微信传文件 |
| **性能影响** | 开启全部功能时性能下降70%+ | 全功能开启性能下降<20% |

**关键技术对比：**

UTM处理流程：流量 → 防火墙 → IPS → AV → 输出（多次拆包检测）

NGFW处理流程：流量 → 单次拆包 → 并行检测所有威胁 → 输出