匿名论坛平台

网络实习中期报告

田晶晶 *　徐晟 *　倪天炜 *　傅智毅 **

* 北京大学计算机系　** 北京大学元培学院

1. 需求分析

论坛是网络上十分常见的一种平台，是网友进行在线交流、讨论、发表言论的重要工具。从功能来看，常用的论坛又分成电影、美食、文化、留学等类型。从北京大学本校学生的需求来看，PKU helper 已经集成了两个不同性质的论坛：树洞和 BBS。

其中，BBS 是典型的在线论坛，按主题分类，用户注册登陆后就可以发帖。而树洞则是所谓的匿名论坛。但是树洞的一个特征是“伪匿名”，虽然呈现给用户的是匿名论坛，但是在服务器端看到的完全是另一个情况。用户所有的注册信息都可以和帖子一一对应，也就是非匿名。目前网络上很少有以匿名为特征的论坛，即使有，大部分也只是呈现给用户的伪匿名。我们的目的是做一个真正的匿名论坛。可以匿名到什么程度，是实验过程中要探究的问题。

(a) 北大树洞　　(b) 未名 BBS

图 1. 校内两大论坛平台

2016 年 4 月，一起针对 pku helper 的安全漏洞被爆出，黑客通过 SQL 注入的攻击手段，获取了 pku helper 端的用户所有信息。让人震惊的是，作为用户与校园身份认证系统中转的 pku helper 服务器，密码都是明文存储的。本该是给用户提供匿名服务，让用户自由发表看法的论坛，随着数据被获取，用户的隐私一览无遗。这也是我们选择匿名论坛作为课程项目的动机之一：实现一个真正、安全的匿名论坛，保护用户的隐私。

另外，树洞目前作为一个“匿名论坛”，实际上缺少很多提升用户体验的功能，例如主题分类，点赞功能等等，这也是我们实现时要考虑的一个问题。

2. 客户端设计

对于 web 服务来说，客户端的作用是在浏览器渲染出合理美观的网页，并与服务端交互数据。具体来说，采用主流的 HTML/CSS/JavaScript 客户端编程语言，设计网页，并向服务端发送某个 url 的 http 请求，然后从服务端获取数据并加载到网页中。

就本项目匿名论坛平台而言，客户端的网页主要分为四个，welcome.html 欢迎页，home.html 论坛首页，user.html 用户主页，thread.html 论坛帖页面。这些网页采用共同的模板，保持一致的风格，给人以匿名论坛的神秘又活泼感；同时，这些网页彼此之间功能独立，程序解耦，通过链接实现跳转。

以下是对四个网页的程序设计的介绍：

2.1. welcome.html（图 2）

作为进入论坛后的首个页面，这个页面提供的功能包括：用户注册、用户登录以及以游客身份登录。在

图 2. welcome.html

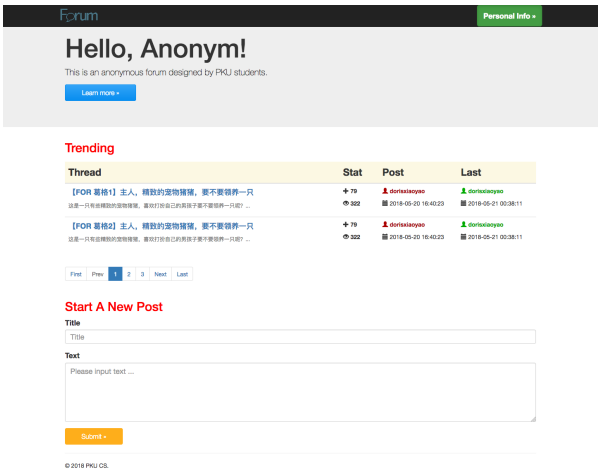

图 3. home.html

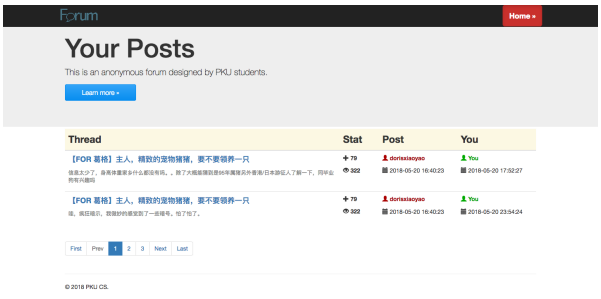

图 4. user.html

这个界面上，我们设计了两个文本框分别输入用户名和密码，三个按钮分别对应注册、登录和游客登录三个功能。用户注册和登录需要在相应的文本框里输入用户名和密码。

与服务端的交互我们采用 json 来实现。在 json 中用一个字段表明是注册、登录还是游客登录。如果是游客登录，则不需要在 json 中附加额外的信息，服务

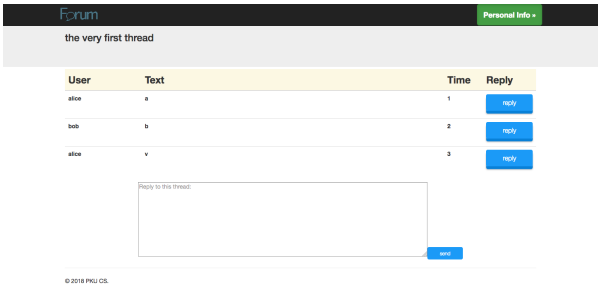

图 5. thread.html

端在接收到请求后会为其生成一个会话并返回一个唯一的 cookie 作为游客登录的凭证。如果是用户注册或用户登录，则需要在 json 中附上用户名和密码这两个字段。服务器会返回一个 json 表示登录是否成功以及（如果失败）失败的原因。如果登录成功，则会自动重定向到 home.html，在那个页面中用户可以浏览主题帖。

另外由于用户名和密码这些信息可能会被中途截获，因此我们需要对其做加密处理之后再发送给服务端。加密处理的这些内容具体在第四部分“加密特色”中讨论。

2.2. home.html（图 3）

论坛主页 home 提供给用户的主要功能有两个：浏览全站的热门帖子，发帖。

关于热门帖子，和未名 BBS 类似，在服务端有一个计算热度的算法，对所有帖子进行排序，返回到客户端，得到时下流行的帖子列表。然而未名 BBS 缺乏直接的排序功能，比如按照发帖时间，回帖时间，浏览量，回复数等各种属性排序，但我们的匿名论坛将不仅提供按照各种属性排序，也提供按照热度算法排序，让同学们更简洁直观地获取自己想要的热门帖子。

此外，BBS home 页面缺乏对帖子内容的简介，只有一个标题，如图1(b)所示，这让标题党（以标题博人眼球的帖子）产生无谓的点击量。而我们的服务端将保存主帖（帖子的第一层）的部分信息，发送给客户端，直接显示在帖子列表里，让用户提前了解帖子信息，以便决定是否点击。

发帖功能和 BBS 类似，提供发帖的标题和文本框输入，未来我们将支持多种文件格式的发送。

2.3. user.html（图 4）

user 是用户个人主页，目前未名 BBS 的个人主页只能申请个人文集，保存并不是所发的帖子；而树洞仅通过收藏功能，如图1(a)所示，来保存用户感兴趣的帖子。本匿名论坛在用户主页上应该比两大平台更加人性化。

用户可以在 user 页面查看所有自己发送与回复的帖子，通过服务端记录每个回复帖的用户 id 实现。并提供这些帖子的链接。

而且匿名论坛将不会提供其他用户访问该用户主页的机会，通过匹配当前登录用户 id 和访问的用户主页 url 中的 id，如果匹配失败，将不予提供，保护了用户隐私，这也是匿名论坛的初衷。

2.4. thread.html（图 5）

当用户想要浏览某个帖子的内容时，就会点开这个帖子的链接进入该帖子对应的 thread 页面。浏览器向服务端发送请求后，服务端会返回一个记录帖子数据的 json 字符串，这个页面就需要根据这个 json 来构建相应的帖子页面。这个 json 字符串的内容包括返回状态、该帖子的楼层数、以及一个 data 数组表示每一个楼层的内容。楼层内容包括发帖人名字、发帖时间和帖子内容。因为要实现匿名，所以我们不会显示用户真正的用户名，而是会由服务端根据用户的发帖顺序给每个用户分配一个绰号，比如第一个发帖的称为 Alice，第二个称为 Bob，第三个称为 Carol 等。

这个页面还提供了回帖功能，可以针对某个楼层回复，也可以不指定针对哪个楼层回复。回帖需要给服务端发送回复文本、帖子 id 和用户 id 等信息。

3. 服务端设计

服务器的后端实现很灵活，现有的后端服务器部署常用的语言包括 Java, php, python, C# 等。不同的语言对于后端架构的实现有不同的优势，在这里我们选用 python 作为我们的后端部署语言。Python 作为编程语言有其自身的优势，比如灵活、可扩展性强等，另外，python 也有很多 Web 应用框架，使用十分方便，例如 flask，Django 等。我们使用 Django 作为我们的开发框架。

3.1. 服务器端架构

Django 作为一种开源的 Web 应用框架，设计模式遵循 MTV，M 表示模型，T 表示模板，V 表示视图。我们会在后面的具体实现中阐述我们如何利用 Django 的这一设计理念。

3.2. 数据库设计

作为服务器后台，一定会有数据库用来存储用户与信息，我们采用 mac/linux 系统自带的 sqlite3 数据库来管理数据。Sqlite 是一种遵守 ACID 的轻量级的关系型数据库，在 python 的实现中，每一条记录为一个对象，它的属性也对应数据库中的一个属性。Sqlite 提供的接口可以方便地对数据进行增删查改。在 Django 架构中，所有的数据表设计存储在 models.py 文件中，在其他文件中调用。

在设计具体的数据库的数据表时，我们按照论坛的需求分析，设置三种数据格式，分别对应用户 (Person)，帖子 (Post)，楼层 (Floor)。它们的定义及属性如下：

usrid	用户 ID（对已注册用户）
password	用户密码

表 1. 用户 (Person) 数据表定义。

title	标题
content	内容
view	查看人数
re	回复人数
post_time	发帖时间
last_time	最后回复时间
post_id	发帖人 ID
last_id	最后回复人 ID
tid	帖子 ID

表 2. 帖子 (Post) 数据表定义。

tid	帖子 ID
time	时间
text	内容
userid	回帖用户 ID
fid	楼层 ID

表 3. 楼层 (Floor) 数据表定义。

对于表1，需要说明的是在目前的实现中，用户名和密码暂时是明文存储的，并且实现成服务器用 get 方式登陆服务器，这些缺陷会在后续实现中修正。表2 中，每次新发帖，服务器会为这个帖子分配一个新的帖子 ID，便于后面的查找操作。表3 中的 fid 是在每个帖子内部进行独立编号的，不同的帖子都是从 1 开始编号的。

3.3. 与前端交互设计

前端的 JS 代码通过 get 或者 post 来进行服务器端的资源获取，形式一般都是发送 url 作为参数，后端在 url.py 里面进行 url 的解析。我们目前的实现是将前端产生的请求作为参数发送给后端，而不是直接写在 url 中，所以后端处理函数中可以通过 GET 来得到具体的参数。

我们的服务器具体要处理的 url 如图6 所示。可以看到，当前端发起一个 url 的请求，在后端解析后，会找到对应的处理函数，进入到这个函数中。

```
urlpatterns = [
    path('', forum_view.welcome),
    path('send_reply/', forum_view.send_reply),
    path('get_replies/', forum_view.get_replies),
    path('get_user_post/', forum_view.get_user_post),
    path('thread/', forum_view.thread),
    path('user/', forum_view.user),
    path('get_post/', forum_view.get_post),
    path('query_author/<slug:author>', forum_view.query_author),
    path('post/', forum_view.post),
    path('visitor/', forum_view.visitor),
    path('signup/', forum_view.signup),
    path('home/', forum_view.home),
    path('login/', forum_view.index),
    path('admin/', admin.site.urls),
]
```

图 6. url 格式。

另外，服务器端需要访问静态文件，例如 js, css, img 等等，这需要我们在服务器端新建一个 static 文件夹存储资源。此外，只在 url.py 中进行相应的修改，就可以访问静态资源了。我们将在后期的实现中考虑加入这些扩展功能。

3.4. 视图实现

视图，即 Django 框架中的 V，实现在 view.py 中。这部分负责解析完 url 之后进行的处理。从前端发送 url 有两种方式，一种是将参数写在 url 中，通过 url.py 的正则匹配来解析出参数，传给 view.py 的函数。我们这里采用第二种，也就是在函数体中解析参数，例如，我们函数的参数为 request，它可以视作发来的一个 json 数据，通过 request.GET[‘attribute’] 就可以得到名为 attribute 的参数。

我们实现的函数如下：

1. Welcome: 负责登录页面的渲染；
2. Index：负责用户信息验证，服务器收到用户发来的用户名和密码后，到数据库中匹配，并且返回信息表示查询结果，前端收到后进行相应的处理；
3. home：登录成功后，负责渲染 home.html 页面（渲染会在后一节中说明）；
4. get_post：在渲染 home.html 后，前端 js 会发送请求得到第一页的帖子，这个函数负责去数据库中找到 tid 最大的指定数目的帖子，并且装在 json 文件中返回给用户；
5. get_user_post：前端有查看用户自己发帖历史的需求，发来 userid 之后需要服务器到数据库中查询对应用户的帖子，同样要按 fid 排序并且进行分页后传给客户端；
6. thread：负责进行用户点开帖子后帖子页面的渲染；
7. send_reply：在帖子页面，用户可以选择回复，点击发送按钮后会触发这个函数，这个函数创建一个新的 Floor 对象，相当于新盖了一层楼；
8. get_replies：渲染 thread 页面时，前端会发送请求，得到指定页数的回复，与帖子一样，这是由后台去查询并且返回的；
9. user：负责渲染 user.html 页面；
10. signup：负责注册，在 Person 数据表中创建一个新的纪录；
11. vistor：负责游客登陆处理；
12. post：发送一个帖子，在帖子数据表中创建一个新的记录。

3.5. 模板

刚才在视图实现的介绍中，提到了渲染的概念，实际上就是用户请求一个 html 页面，服务器返回对应资源的过程。我们采用 render 函数来进行页面的渲染。前端要给后端发送信息的时候，可以通过 jQuery 的 get 或者 post 功能。相反，服务器要给前端发送消息是用到的就是模板。

Render 函数有一个参数，将 json 格式的数据填充到 html 的模板中。在 html 中可以有类似于编程语言

的语句块，它会自动解析 render 函数的参数，并且渲染出来。所以就可以将类似于 userid 等参数通过 render 函数传到模板，实现模板的渲染。

4. 加密特色

目前，我们的实现中还未加入加密环节，所有信息均以明文形式保存于后端。后期我们考虑在前端实现加密，将信息加密后再连同请求传输给后端，完全隔离后端从而避免从后端获取用户的隐私信息。但是加密完的信息依旧要能够识别出用户，更准确的说是区分每个用户。这里我们先做了调研和框架预设，待聊天系统基本功能完善后加入加密功能。

对于不同的应用环境和保密需求，有不同的加密算法。主流加密算法可以大致分为非对称（密钥）加密算法和对称（密钥）加密算法，以及 MD5 这种严格意义上不算加密算法的散列算法。实际上，MD5 是不可逆的摘要算法，类似于 Hush 函数。

4.1. 加密算法简介

对称加密算法有且仅有一份密钥，所以想要解码加密的信息（密文）则密钥也需要传输到接收信息者处，密钥传输的过程和信息传输的过程没有本质区别，存在被窃听的可能。有时甚至需要级联加密来增强保密性，如图7 所示。典型算法有 DES、3DES 等。

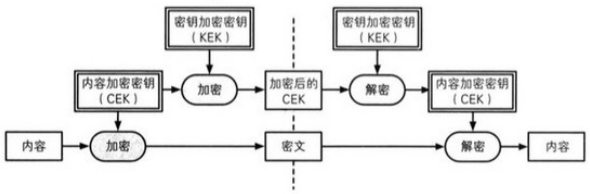

图 7. 级联加密示意图。

非对称加密算法则同时派生一对密钥，即公钥和私钥。工作流程如下：信息接收者将公钥向外公布，信息传递者将信息以公钥加密并发送给私钥持有方（接收者），以公钥加密的信息由且仅能由接收者的私钥解密。这一过程中从始至终私钥都仅由接收方持有，不经传输，安全系数高得多。实际双方通信中应该有两对密钥。典型算法有 RSA、ECC 等。

散列算法不需要密钥，但可能有 salt——系统产生的随机数，与信息 concatenate 后再散列——用以增大建表成本同时减小散列冲突，从而降低窃取可能性。典型的 SHA1 和 MD5 算法，都是将任意大小的数据映射到一个较小、固定长度的唯一值。加密性强的散列一定是不可逆的，这就意味着通过散列结果，无法推出任何部分的原始信息。但可强行枚举，所以安全性是一大缺陷。

4.2. 匿名论坛的加密场景

一般来说，加密算法应用于“双方通信，传输过程加密”的场景，密钥类加密算法的本质就是使传输中的数据或信息呈密文形式，只有获取密钥才可以解析。但在我们的需求有些不同，我们需要保障可能泄露用户身份的信息，如 IP 地址、MAC 地址、Cookie 文件等，在服务器端（维护者角度）不可见。换言之，个人信息只能由用户在本地可见，服务器端均是密文，和银行系统的账号密码管理方式一致。

这样做的好处有两点：第一点是明文数据不储存在服务器数据库，即便服务器信息泄露，如之前提及的 SQL 注入攻击，攻击者也无法获取用户的真实信息；第二点我们的加密逻辑和银行保护用户密码一致，即从技术角度上解决了内部泄露的问题。

4.3. 预设框架

我们初步考虑用多重加密来保障信息安全性，如 SHA512+MD5+RSA，而具体实现时看情况再定。其中使用到的 MD5 等散列算法无密钥，通常情况下，salt 由服务器端生成并会保留在服务器端。但这里，我们考虑将散列过程保留在客户端，包括加密接口调用和随机生成 salt。

对于对称和非对称加密，我们将密钥保留在客户端，只将加密后的信息传递给服务器，服务器再在数据库中比对加密后的数据，如账号密码模式的密码、游客模式的 cookie 等，从而验证及识别用户。其中非对称加密可以将公钥派发给服务器，以支持更多的功能（譬如数字签名）。但是这样也有潜在的威胁性，譬如用户不小心将 cookie 清除了，可能导致密钥缺失，这样一来服务器端就无法获取正确加密后的密码密文，从而验证用户失败。这些缺陷可能需要通过添加“重设密码”的功能来修正（通过验证一些非隐私的问答核实用户身份），毕竟服务器端是不知道真正的密码的。

一种可能的优化方式是，我们可以定期刷新数据

库的密文用户数据。换言之，定期更新密钥、salt，从而更新加密后的密文，类似于现在的动态令牌，唯一的不同只是这些令牌缓存在本机，用户不用知道罢了。但这样操作的实际效果是否会更好，目前尚不得知。如果效果不好或有更好的方式，那么我们会摒弃这种做法。

5. 未来工作

5.1. 文件传输与共享

服务器端维护所有用户上传的文件，并且提供下载功能。我们的匿名论坛将提供多种数据格式的传输，用户可以选择发送文字、语音、图片，甚至视频的数据格式，并可以在线流量这些数据格式的文件，而服务器端如何对这些不同的数据报如何解析，并且呈现在客户端，客户端又将如何渲染这些文件，是需要考虑的问题。

5.2. 异常流量检测与封禁

网络上大量存在恶意攻击的问题，如果某一用户短时间内恶意大量发帖，那么会造成服务器瘫痪。因此，有必要对用户流量进行监控，如果发现某些用户某个时间点的发帖频率超过阈值，那么就通过 IP/MAC 对用户进行封禁。

5.3. 更多用户友好功能

由于帖子主题繁多，将他们分门别类到不同的分区，用户通过选择感兴趣的分区，来高效地浏览与发布帖子。

此外，论坛将增加帖子点赞功能，让用户无需输入文字，即可与帖子互动，表达自己的支持，简洁明了地显示一个帖子的热门程度。