计算机系统安全报告

姓　　名	
班　　级	
实 验 名 称	
开 设 学 期	
开 设 时 间	
报 告 日 期	
评 定 成 绩	

东北大学软件学院

目录

一．报告内容

对学校的单点登录系统评估。

二．简单登录的介绍

1. 一个简单的登录流程

- 用户输入需要访问的URL访问站点，接受用户请求后判断用户是否已经登录，若未登录则跳转到登录页面。
- 用户访问登陆页面，填写相关信息并提交登陆表单。
- Web应用对登录表单进行验证，若验证失败，即用户登陆失败，则返回错误信息给用户。根据登录系统的相关需求来要求用户重新填写信息；若验证成功，则将用户相关的信息（通常为用户id等信息）写入到当前的session中，将session id以cookie的形式发送给用户（同时可以将session中的身份信息以cookie的形式发送给用户，这个是可选的，使用该cookie可以实现自动登录。
- 用户登录后，后续的访问中会将其获得的包含session id的cookie传递给web应用，若web应用能根据session id从session中获取到相应的身份信息（甚至进一步从数据库查找到相关的用户数据），则代表用户成功登录。

2. 退出登录的流程

退出登录主要包括以下两个流程：

- 销毁session信息 （在服务端销毁）
- 销毁客户端的相关cookie（包含session id的cookie）

3. 自动登录的介绍和实现

通常来说，服务器端的session会有一定的过期时间，同样的，客户端中包含session id的cookie也有过期时间。若用户登录后，长时间没有发出访问请求，则等到用户再次访问时，可能服务端的session或客户端的cookie已经失效，导致服务器判断用户为“未登录”，而实际上用户并没有退出登录过。可以通过实现“自动登录”来解决这个问题，即在session失效的时候，可以重新自动登录。而自动登录的基本思路如下：

- 在用户登录后，除了生成session id对应的cookie，还会生成一个包含用户身份信息的cookie（假设这个cookie的key为identity）。identity cookie包含的信息可以有：用户id，用户authKey（这个比较重要，authKey的功能有点类似于password），cookie持续时间等。为了达到自动登录的目的,通常会将该cookie的过期时间设置的特别长（可以是一个星期甚至一个月）。但是某些网站为了极大的保护页面的定时安全性，该cookie的过期时间设置的会比较短，一般来说不超过2小时。例如东北大学的新版教务系统，若长时间停留在该页面，用户进行任何操作都会重置页面到登陆页面，需要用户重新登录才可继续操作使用相关系统的功能。
- 在用户session失效后，用户再次访问web应用，会带上identity cookie（因为该cookie的有效期较长）。web应用首先判断用户为“未登录”状态（因为session失效了）。
- web应用尝试通过identity cookie为用户自动登录。应用从identity cookie中获取“用户id”和“用户authKey”，通过与数据库中的数据进行对比，校验“用

户authKey”的有效性。

- 用户authKey认证有效后，web应用为用户生成新的session，并将新的session id放入cookie发送给用户。（即用户登录成功的标志为获取到包含session id的cookie）

总的来说，在Web应用中实现登录的核心就是cookie和session。由客户端的cookie（其包含session id）可以由session id索引到服务器端的session（其包含用户的身份信息，如user id和user account）。然后再根据session中的信息，可以进一步从数据库中获取用户的详细信息或相关权限。

三．单点登录

1. 单点登录简介

单点登录（Single sign-on，缩写为 SSO），又译为单一签入，一种对于许多相互关连，但是又是各自独立的软件系统，提供访问控制的属性。在拥有这项属性的环境中，当用户在某个系统登录时，就可以获取所有系统的访问权限，不用对每个单一系统都逐一登录。这项功能通常是以轻型目录访问协议（LDAP）来实现，在服务器上会将用户信息存储到LDAP数据库中。相同的，单一注销和单点登录是相对的。单一注销（single sign-off）就是指，只需要单一的注销动作，就可以结束对于多个系统的访问权限。

简单总结来说就是，在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统；用户只需要退出登录一次就可以退出所有的应用系统。

2. 单点登录的优点

- 降低访问第三方网站风险（用户密码不存储或外部管理）,提高了系统安全性:随着我们对应用程序的日益依赖，企业也正面临着“数字化身份管理”的安全挑战。除了身份验证登录之外，“数字化身份管理”还包括分配给某个用户的角色和权限。因此，系统管理员还需要保护这些身份权限免遭盗窃和破坏。但是，一些企业可能没有足够的资源或专业知识来满足快速变化的安全环境所引发的这些需求。和知名的身份提供商(IdP)一起，采用单点登录的方式，可以让企业将这一负担转移到有足够能力应对这些挑战的专家身上。知名提供商包括Okta、Azure AD 和Auth0等。
- 用户不需要在不同的站点使用不同的用户名和密码，减少“密码疲劳”（password fatigue ）：密码输入疲劳是指用户由于必须记住过多的密码才能使用各种应用程序，因此感到困难和疲惫。不同的应用程序具有不同的密码要求，意味着适用于一个系统的密码可能不适用于另一个系统，于是用户会使用相同（并且安全性可能很弱）的基本密码的“变体”来满足这些要求。比如在一个应用程序里的密码是“apple”，然而另一个应用程序却要求密码中必须包含数字，因此用户可能用“apple123”来做密码。然而使用现代密码破解办法能够轻松破解这些“变体”。换句话说，密码“apple123”并不比“apple”更安全。单点登录的集成解决了这种困境。
- 减少在使用不同站点时花费时间来重新输入密码进行身份验证，通过减少登录次数来提高工作效率：对登录的持续需求可能会导致生产力的显著降低。虽然单次登录所需的时间仅需要10-15秒，但考虑到其他因素，例如确定使用

了哪种密码变体、联系IT部门提供密码协助和重置等，都占用一定工作时间。当用户请求密码帮助时，IT部门的时间也会被占用。综上，员工浪费的精力和时间成本就会变得很可观。使用单点登录可以直接解决这些问题，提高工作效率。

- 单点登录为系统管理人员工作提供便利：企业的系统管理团队负责将新应用程序集成到现有系统中，并对新应用程序进行所有用户的账户设置。这对于15到20名用户的小型企业来说很容易，但随着应用程序及员工人数的增长，解决每个新应用程序的登录问题所需的大量时间可能会阻碍系统管理员执行其他工作任务。如果企业需要雇佣更多的管理员来保障业务持续运转，额外的人力也会转化为成本增加。使用单点登录意味着系统管理员只需管理较少数量的用户账户。部署新的应用程序时，可以重复使用现有账户，无需每次为每个用户设置新帐户。此外，如果员工离职，系统管理员能够撤消任何用户对所有应用程序的访问权限，从而限制了窃取机密、恶意攻击的可能性。
- 单点登录简化了用户跟踪审计过程：使用多个登录系统可能很难跟踪用户以进行审计。例如，John Smith可能有一个应用程序的JSmith帐户和另一个应用程序的JohnS帐户。如果有两个同名同姓的John Smiths（这在大型企业中并不少见）为您工作，情况就变得复杂了。利用单点登录，账户管理更加清晰明了，系统管理员可以精准识别每一个登录用户，从而轻松地跨不同应用程序跟踪用户以进行审计和记录。

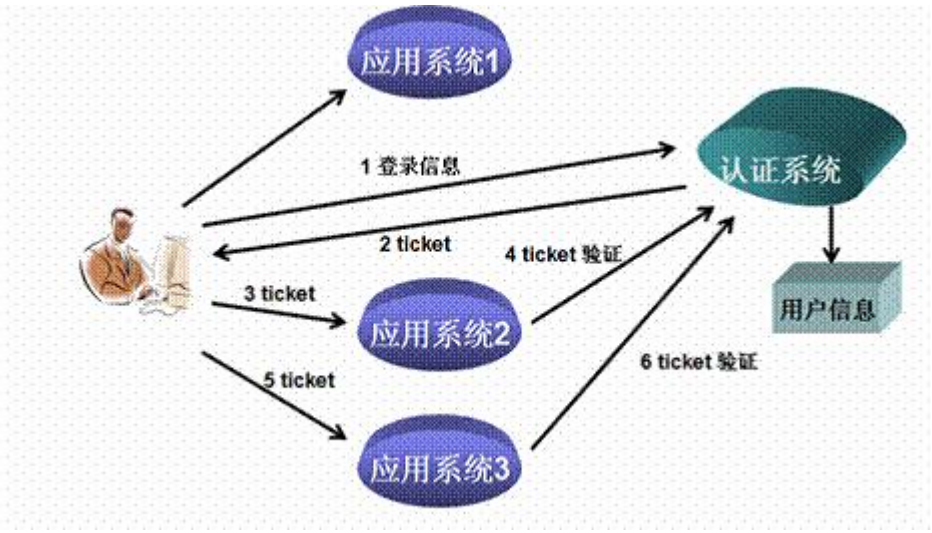
3. 单点登录的缺点

- 在规模较小的系统下，成本较高：SSO的成本可能会在短时间内飙升。对于小型公司，SSO虽然优势巨大，但也可能耗费较高预算。许多SSO厂商按功能单独收费，大多数核心功能都是叠加费用，因此整体费用增长较快。对于CPU而言，在并发量比较高的情况下，如果许多线程反复尝试更新某一个变量，却又一直更新不成功，循环往复，会给CPU带来很大的压力。
- 需要IdP：任何SSO解决方案的主干都是企业的IdP/目录服务。SSO解决方案通常架构于目录之上，企业为了达到理想效果，不得不为不同的解决方案单独付费。当然，目录的解决方案可能与SSO一样成本高昂，包括设置实施的费用以及继续使用的经常性成本。
- 以网络应用为主：IAM领域较为宽泛，覆盖了IT中的大部分职责。使用SSO管理对Web应用的访问只是IAM的冰山一角，IT管理员为了创建完整的IAM解决方案需要将Web应用SSO与大量其他方案结合使用。与此同时，用户仍需访问设备（Mac，Windows，Linux）、服务器基础架构、VPN、WiFi网络、文件服务器、本地应用等，而SSO并不能管理对这些IT资源的访问。
- 需要超强密码：虽然终端用户只需记住SSO这一个密码，但要让SSO发挥最大效用，需要使用高度复杂性和保密性较好的长密码。高强度密码通常有利于整体的身份安全，但也可能被用户忘记或泄露，所以功过相抵。另外，使用单点登录意味着只需要一个主密码即可访问与其相关的所有内容，可能会导致某些重要信息的泄露。这也为整体系统的安全性埋下了隐患。由于SSO与许多关键资源打通，如果SSO提供程序沦为攻击目标，整个用户群都将会遭到影响。另一方面，如果终端用户的SSO门户受到威胁，且未启用MFA，那么用户对应用的访问也将面临风险。在后续为了增强密码和帐户的安全性，可以将多重身份验证(MFA)与单点登录集成，从而提高安全级别，确保黑客

无法轻易破解帐户。

- 不利于系统重构，需要考虑框架设置：在实施单点登录时还应考虑其他因素，由于单点登录仅提供身份验证机制，还要额外注意框架的配置。比如仅仅因为某人可以访问某个系统并不意味着他们应该能够访问该系统上的所有资源。与许多技术创新一样，单点登录的集成不具有排他性。因此单点登录可以实施到一个框架中，该框架还考虑了细粒度的访问控制权限等。SSO与许多IT工具一样，少有“即插即用”的，这意味着IT管理员必须投入时间和精力来集成和定制企业的SSO服务。不仅需要配置应用程序，而且在使用第三方IdP时，SSO集成可能会很复杂，难以实施。 最后，因为涉及到的系统很多，要重构必须要兼容所有的系统，会十分耗时耗力。

4. 单点登录的基本思路



- 当用户第一次访问应用系统1的时候，因为还没有登录，会被引导到认证系统中进行登录（1）
- 根据用户提供的登录信息， 认证系统进行身份校验，如果通过效校验，返回给用户一个认证的凭据ticket（2）
- 用户重新访问应用系统1，此时会带上ticket。应用系统1接受到请求后会把ticket发送到认证系统进行校验，检查ticket的合法性；若ticket合法性验证通过（代表用户已经登录），则用户可以访问应用系统1
- 用户再访问别的应用的时候（3，5）也会将这个ticket 带上，作为自己认证的凭据，应用系统接受到请求之后会把ticket送到认证系统进行效验，检查ticket的合法性（4，6）。如果通过效验，用户就可以在不用再次登录的情况下访问应用系统2和应用系统3了。

ticket是所有系统对用户的统一的认证标志。在具体的实现中，我们可以用cookie来实现ticket的功能。最简单的，一个包含session id的cookie就可以看成是一个ticket。出于安全需要，通常我们会对ticket进行加密。目前来说比较流行的加密方式为RSA和椭圆曲线加密。要实现SSO的功能，让用户只登录一次，就必须让应用系统能够识别已经登录过的用户。应用系统应该能对ticket进行识别和提取，通过与认证系统的通讯，能自动判断当前用户是否登录过，从而完成单点

登录的功能。

总的来说，图片示例中的ticket是整个系统实现登陆的核心。ticket会在用户，应用系统，认证系统的交互中传输，最终达到实现单点登录的目的。Ticket是所有系统对用户的统一的认证标志。

5. 单点登录的分类

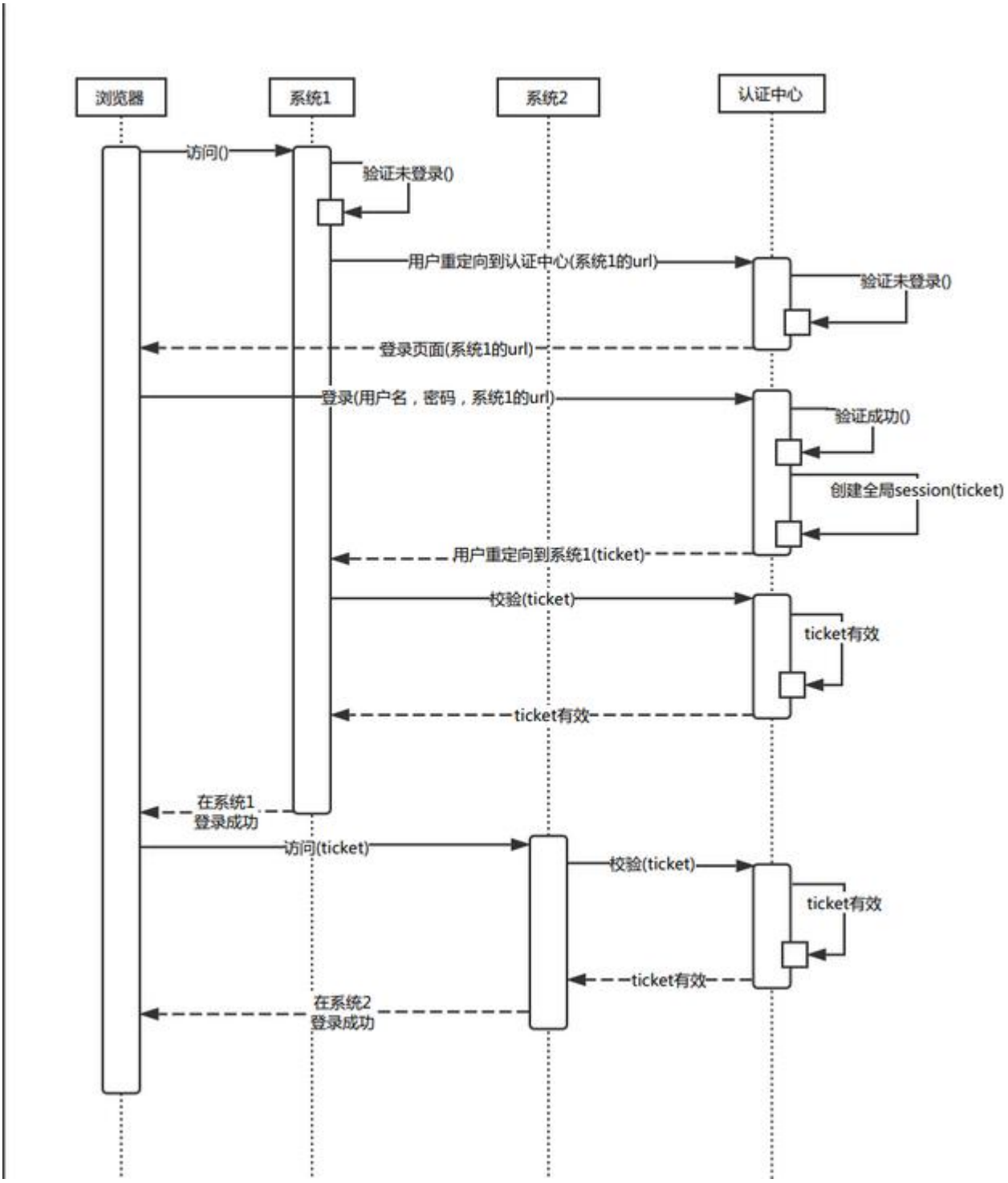
在技术实现上，单点登录可以分为**跨子域单点登录和完全跨域单点登录**。**跨子域单点登录**即在具有相同根域名的站点之间实现单点登录。例如，有以下站点a.example.com, b.example.com, p.example.com(认证中心)，它们都有一个共同的根域名“example.com”。在单点登录写cookie时，把cookie的域设为它们共同的父域（即“example.com”），这样在不同的子域名下都可以使用同一个cookie（即ticket）；与此同时，可以让多个系统共享session信息。**完全跨域单点登录**即具有不同根域名的站点之间实现单点登录。实现思路：每个站点需要有自己的ticket（A-tikcet，B-ticket，P-ticket）；当访问应用系统（A，B）时，若没有相应的ticket，则自动重定向到认证系统（P），在认证系统认证获得P-ticket后重定向到应用系统（A或B），用P-tikcet换取相应的A-ticket或B-ticket；注销的时候要注销所有的ticket(P-ticket,A-ticket,B-ticket)。

我们可以通过简单的进行技术的比较，可以得到在**跨子域单点登录**中实现共享cookie的方式存在若干局限性：

- 应用群域名得统一
- 应用群各系统使用的技术（至少是web服务器）要相同，不然cookie的key值（tomcat为JSESSIONID，php为PHPSESSID）不同，无法维持会话，共享cookie的方式是无法实现跨语言技术平台登录的，比如java、php、.net系统之间第
- cookie本身不安全，需要频繁进行加解密处理。

（1）跨子域单点登录

跨子域单点登录的实现相对比较简单。但是需要满足若干前提：应用群中各个站点的域名需要统一，即具有相同的根域名；应用群使用的技术要相同，这样才能够维持回话（例如tomcat下cookie的key值为JSESSIONID，而PHP技术栈下则为PHPSESSID）。跨子域单点登录可以基于cookie来实现，基本思想和单点登录的思路基本相似，将ticket具体实现为cookie中的session id，并且可以省略应用到认证中心对ticket进行校验的步骤即可。



具体实现的步骤如下：

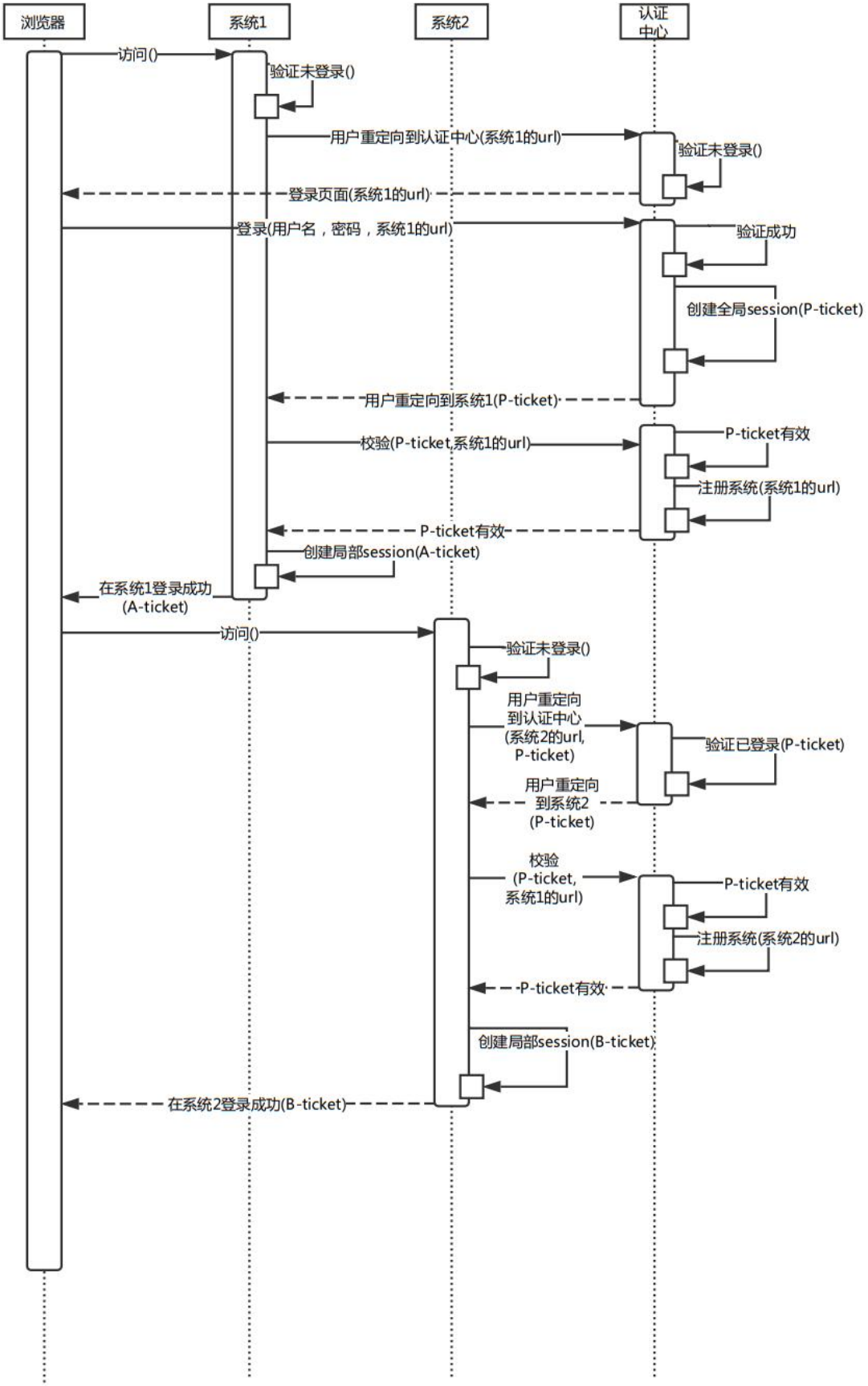
- 当用户访问系统1（域名为a.example.com）时,系统检测到用户的请求中没有ticket（即没有相应的cookie），则判断用户未登录，将用户重定向到认证中心(带上系统1的url，则认证完后认证中心可将用户重定向会系统1)
- 用户提交登录表单到认证中心，验证通过后创建相应的session。将session id作为ticket，以cookie的方式发送给用户，并将用户重定向到系统1。（注意，cookie的域为根域“example.com”）
- 用户重定向到系统1(此时会带上相应的cookie作为ticket)。系统1检测到有

ticket，则向认证中心发起请求，验证ticket的有效性。

- 认证中心验证ticket的有效性（即确实有对应的session），将校验结果返回给系统1。
- 系统1从认证中心得到校验成功的结果后，则可以认为用户“已登录”。
- 用户继续访问系统2(因为系统2的域名为“b.example.com”,其根域名也为“example.com”,故此时会带上相应的cookie作为ticket)。系统2检测到有ticket，则向认证中心发起请求，验证ticket的有效性。
- 认证中心验证ticket的有效性（即确实有对应的session），将校验结果返回给系统2。
- 系统2从认证中心得到校验成功的结果后，则可以认为用户“已登录”。

（2）完全跨域单点登录

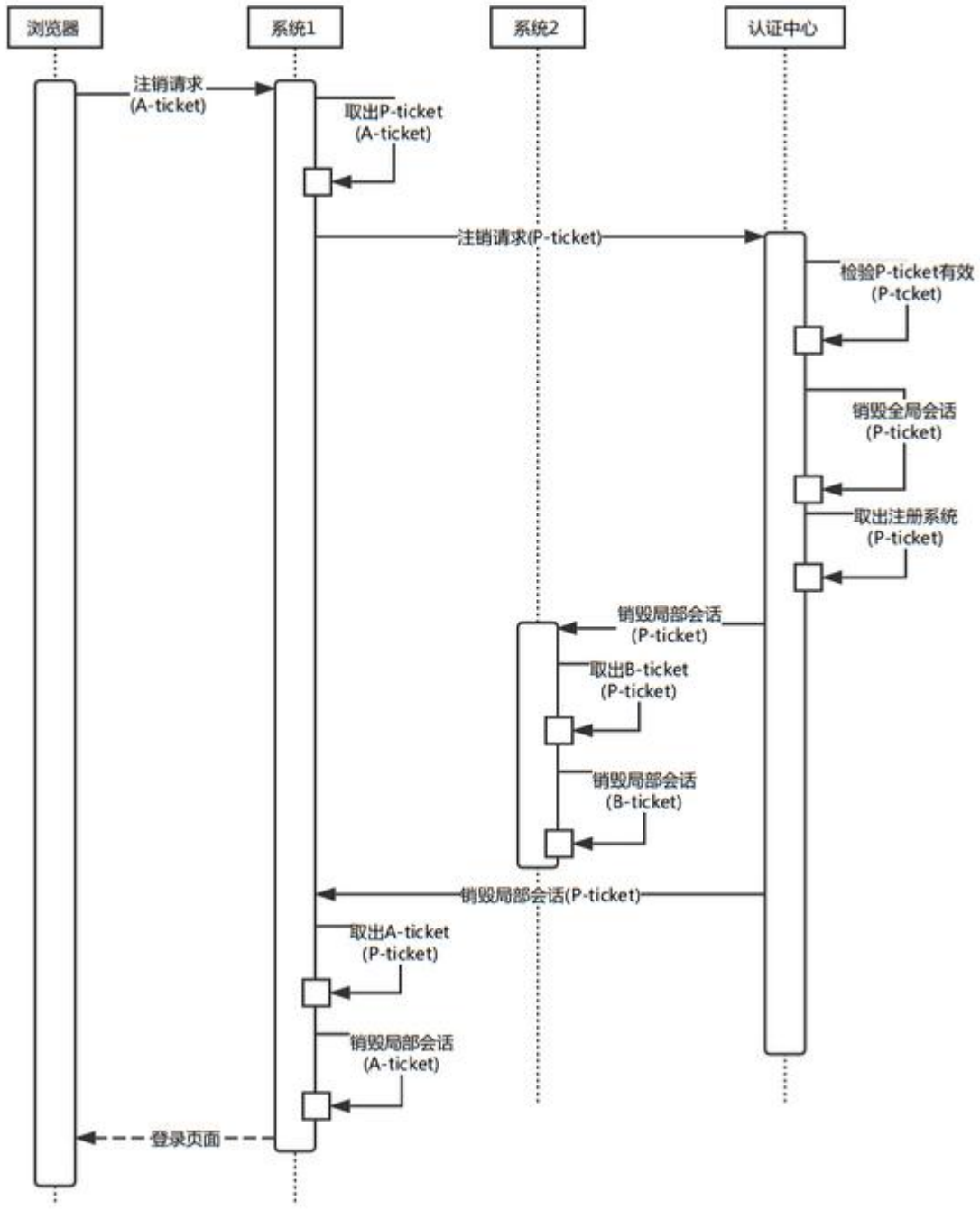
完全跨域单点登录的实现条件及思路如下：首先保证每个站点需要有自己的ticket（假设：系统1为A-tikcet，系统2为B-ticket，认证中心为P-ticket）；当访问应用系统（系统1，系统2）时，若没有相应的ticket，则自动重定向到认证系统（P），在认证系统认证获得P-ticket后重定向到应用系统（系统1或系统2），用P-tikcet换取相应的A-ticket或B-ticket；注销的时候要注销所有的ticket(P-ticket,A-ticket,B-ticket)。


浏览器
系统1
系统2
认证中心
访问()
验证未登录()
用户重定向到认证中心(系统1的url)
验证未登录()
登录页面(系统1的url)
登录(用户名，密码，系统1的url)
验证成功
创建全局session(P-ticket)
用户重定向到系统1(P-ticket)
校验(P-ticket,系统1的url)
P-ticket有效
注册系统(系统1的url)
P-ticket有效
创建局部session(A-ticket)
在系统1登录成功
(A-ticket)
访问()
验证未登录()
用户重定向
到认证中心
(系统2的url,
P-ticket)
验证已登录(P-ticket)
用户重定向
到系统2
(P-ticket)
校验
(P-ticket,
系统1的url)
P-ticket有效
注册系统(系统2的url)
P-ticket有效
创建局部session(B-ticket)
在系统2登录成功(B-ticket)

具体实现的步骤如下：

- 用户访问系统1，系统1发现用户未登录(没有A-ticket)，跳转至认证中心，并带上系统的url作为参数（即回调url，这样认证后就可以跳转回来了）。
- 认证中心发现用户未登录(没有P-ticket)，将用户引导至登录界面。
- 用户提交登录信息到认证中心。
- 认证中心校验用户的登录信息，通过验证后，创建一个全局session，生成一个绑定当前session的P-ticket。然后将P-ticket发送给用户，并将用户重定向到系统1（注意，此时会带上P-ticket）。
- 系统1接收到带有P-ticket的请求后，会向认证中心发出一个请求，验证P-ticket的有效性。
- 认证中心接收到系统1的校验请求，将校验结果返回给系统1。同时若验证通过则会将映射关系(P-ticket,系统1)记录到一个映射表中（称该表为“注册系统表”，即记录那些登录的子系统与P-ticket的对应关系，这样在用户注销的时候就可以向相应的子系统发送请求，销毁相应的局部session）。
- 系统1接受到认证中心的校验结果，若校验通过，则认为用户是“已登录”，系统1在自身系统为用户创建一个session（局部session），并生成一个绑定该session的A-ticket，同时将映射关系（A-ticket，P-ticket）记录到自己的一个映射表中（称该表为“ticket映射表”）。然后系统1将A-ticket发送给用户，在后续用户对系统1的访问中，会带上A-ticket，则可以通过A-ticket判断用户是否已经登录。
- 用户访问系统2，系统2发现用户为登录（没有B-ticket）,跳转到认证中心（此时对认证中心的访问会带上P-ticket），并带上系统2的url作为参数（即回调url，这样认证后就可以跳转回来了）。
- 认证中心发现用户已经登录（因为用户已经持有P-ticket），确认P-ticket的有效性后即可认为用户“已登录”。认证中心将用户重定向到系统2（带上P-ticket）。
- 系统2接收到带有P-ticket的请求后，会向认证中心发出一个请求，验证P-ticket的有效性。
- 认证中心接收到系统2的校验请求，将校验结果返回给系统2。同时若验证通过则会将映射关系(P-ticket,系统2)添加到“注册系统表”。
- 系统2接受到认证中心的校验结果，若校验通过，则认为用户是“已登录”，系统2在自身系统为用户创建一个session（局部session），并生成一个绑定该session的B-ticket，同时将映射关系（A-ticket，P-ticket）记录到自己的一个映射表中（称该表为“ticket映射表”）。然后系统2将B-ticket发送给用户，在后续用户对系统2的访问中，会带上B-ticket，则可以通过B-ticket判断用户是否已经登录。

注销示意图如下：



具体实现的步骤如下：

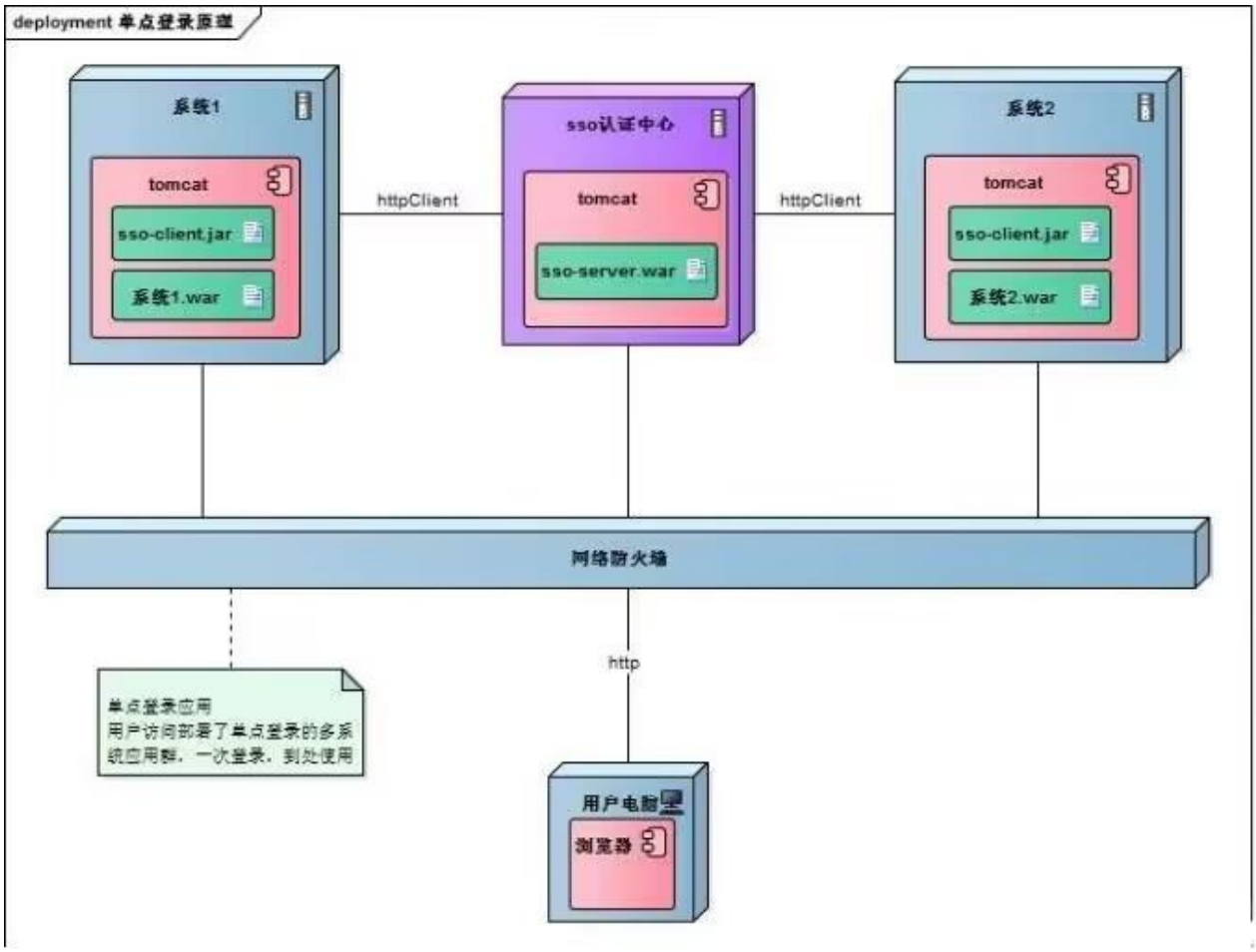
- 用户向系统1发起注销请求（会带上A-ticket）。
- 系统1根据映射表“ticket映射表”，取出相应的P-ticket。系统1向认证中心发起注销请求（带上P-ticket）。
- 认证中心接收到注销请求后，验证P-ticket的有效性。若P-ticket有效，则销毁本地的全局session(根据P-ticket)。认证中心从“注册系统表”中查找出与

P-ticket相关的所有系统，向所有相关的系统发送注销局部会话请求。

- 系统2接收到认证中心的“注销局部会话”请求，根据P-ticket从“ticket映射表”中取出相应的B-ticket，根据B-ticket销毁本地局部会话。
- 系统1接收到认证中心的“注销局部会话”请求，根据P-ticket从“ticket映射表”中取出相应的A-ticket，根据A-ticket销毁本地局部会话。

6. 单点登录的部署图

单点登录涉及sso认证中心与众子系统，子系统与sso认证中心需要通信以交换令牌、校验令牌及发起注销请求，因而子系统必须集成sso的客户端，sso认证中心则是sso服务端，整个单点登录过程实质是sso客户端与服务端通信的过程，用下图描述：



四．多点登录

1. 多点登录简介

多点登录实际上就是同一账号可以在不同终端同时登录，同时收发信息。同时，多点登录可以设置是否存在多点登录限制。例如用户使用应用A登录自己的账号，之后又使用应用B登录同一账号，如果这时应用A的账号经过刷新操作之后账号就被注销登录，那么这个应用对于多点登录就是有限制的。以微信为例：可以PC端、phone端同时登录、同时收发消息。需要注意的是：一个端只能登录

一个实例，例如同一个QQ号，在pc1上登录再到pc2上登录，后者会把前者踢出，pc1会收到通知“你已在别处登录xxoo”。我们的微信账号不可以在同一移动设备同时在线。

既然说到了多点登录，那么就离不开消息漫游这个概念。

Google在2013年的开发者大会（Google I/O）上公布了一款叫“Google 环聊”（网址：http://hangouts.google.com）的移动端IM。其中心思想是：即允许同一账号在多种设备上同时登陆和使用、且消息会以某种逻辑同步到另外的端上（包括自己发出的消息）等特性，使用得移动端IM的消息同步第一次真正的变的友好和易用。

消息漫游的概念：在任何一个终端的任何一个实例登录qq，都能够拉取到所有历史聊天消息，这个就是消息漫游。微信目前只支持“多点登录”同时收发在线消息（如果你同时开过PC端微信和手机端微信，你可以注意到你收到和发出的消息都会在另一个端同时显示出来），没有实现“消息漫游”，潜台词是：登出手机微信，登录PC微信，聊天，再登录手机微信是看不到历史消息的。

2. 多点登录的优缺点

多点登录的优点：

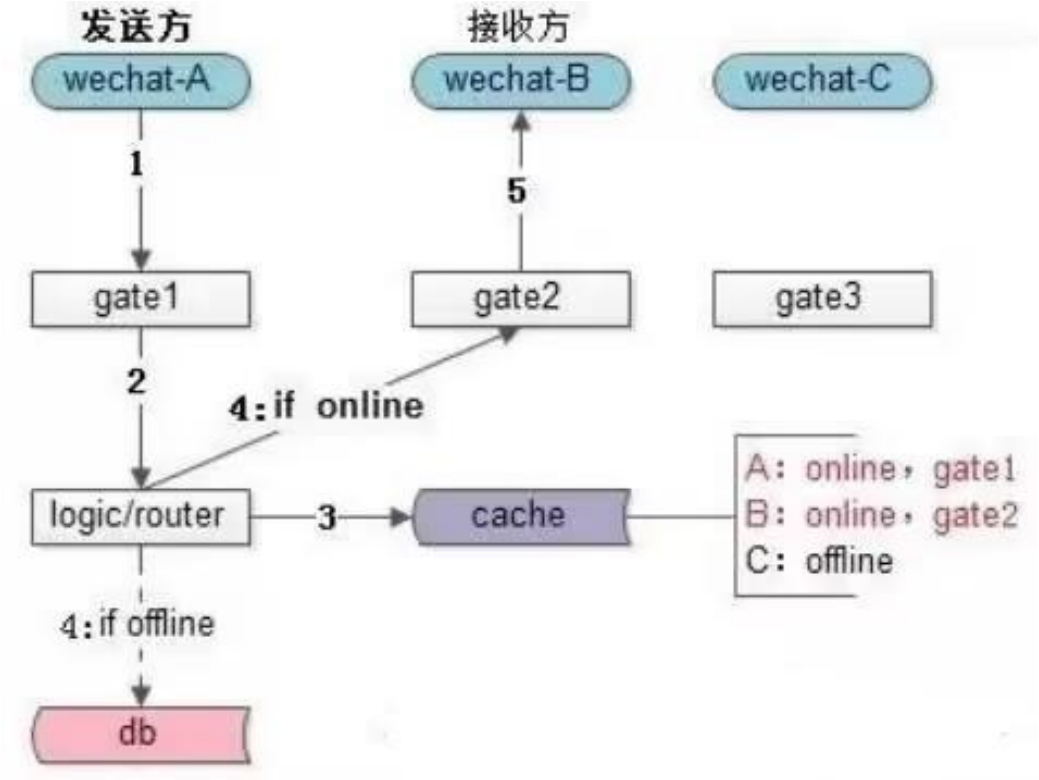
- 利于系统的重构，实现方式较简单：在面对一个比较庞大复杂的系统时，多点登录仅仅需要考虑登录及登陆后的消息通信即可，比较易于实现。并且在系统重构时，也不会太过于复杂。
- 有利于实现多设备的协调工作：多设备可以同步接收和发生相关消息，比较利于协调工作。

多点登录的缺点：

- 消息实现同步会耗费大量资源，可维护成本更高：多点登录为了实现消息漫游需要一个庞大的数据库来存储相关信息，并且还要保证这些数据的合法性和安全性。
- CPU性能要求高：当多个进程在多个设备上进行操作时，会耗费大量的算力资源。
- 对于账号和系统安全性要求更高：系统需要对每次登录进行相关的身份验证和环境识别，过程比较麻烦复杂，并且还无法保证用户相关操作的合法性和安全性。

3. 多点登录实现原理

（1）一个典型的IM消息收发架构



整个IM的架构可以抽象成这么几层：

- 客户端：例如PC微信和手机QQ。
- 服务端：

入口层Gate集群：能够水平扩展，保持与客户端的连接；

逻辑层Logic，路由层Router集群：高可用可扩展，实现业务逻辑，进行消息的路由；

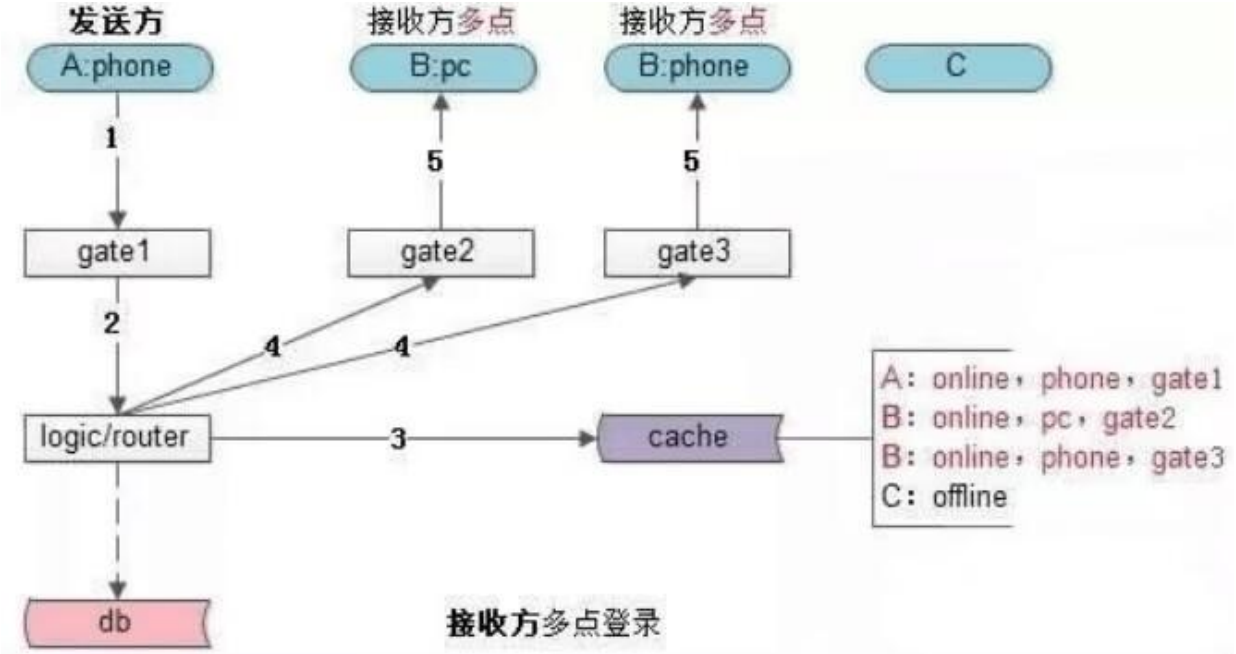
Cache：高可用cache集群，用来存储用户的在线状态，与接入节点（用户具体连接在哪个gate节点）；

DB：固化存储信息，群消息，好友关系链等信息。

一个典型的消息收发流程如上图步骤1-5：

- 用户A登陆在Gate1上，发出消息；
- Gate1将消息给Logic/router。
- Logic/router查询接收方的在线状态（B在线，C不在线）；
- 例如接收方C不在线，存储离线;例如接收方B在线，且登录在gate2上，消息投递给gate2；
- Gate2将消息投递给B。

（2）接收方多点登陆时的消息投递原理



在多点登录中，系统还需要收集用户登陆系统所依托的设备类型：例如Mac，Windows，Android（PC或Mobile）。

接收方多点登录：pc登录、phone也登录，后一端登录不会将前一端踢出，cache中存储状态与登录点时，不再以user_id为key，改为以user_id+终端类型为key即可。

传统的PC端IM时的B客户端信息为：online(状态)，gate2(登录点)。在支持多端登陆的移动端IM时信息应改为：B的pc客户端：online(状态)，gate2(登录点)；B的phone客户端：online(状态)，gate3(登录点)。

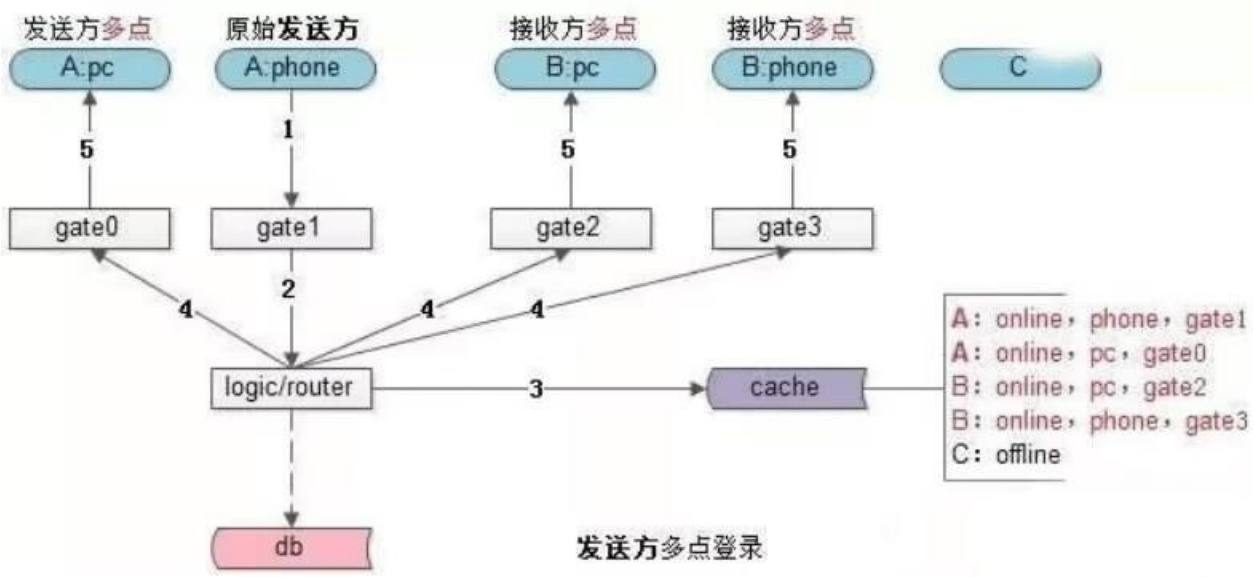
当用户A给用户B发送消息时，取出所有B的登录点，进行消息群发即可（如上图中步骤4与步骤5）。

（3）发送方多点登陆时的消息投递原理

在这里首先为大家解决一个疑问：发送方和多点登录有什么关系？假设用户A登录了两个点：A1和A2；用户B登录了两个点B1和B2。我们将A(A1发出的)发送消息给B(B1和B2)；B(B1发出的)发送消息给A(A1和A2)。这样做有什么问题呢？

A(A1发出的)发送消息给B(B1和B2)，B(B1发出的)发送消息给A(A1和A2)。A2端虽然收到了所有B回复的消息，但消息其实是在A1端发出的，故A2端只知道聊天消息的一半(所有B的回复)，缺失了聊天的上下文(所有A1端的发出)。故：如果发送方也进行了多点登录，发送出去的任何消息，除了要投递给多点登录的接收方，还需要投递给多点登录的发送方。

总结来说就是，自己发出的消息，也要发送到自己登陆的其它客户端上，而不仅仅需要同步其他设备发送给自己的消息。



如上图所示，发送方A和接收方B都进行了多点登陆，服务端cache中存储的信息为：

- A的pc端：online(状态)，gate0(登录点)；
- A的phone端：online(状态)，gate1(登录点)；
- B的pc端：online(状态)，gate2(登录点)；
- B的phone端：online(状态)，gate3(登录点)。

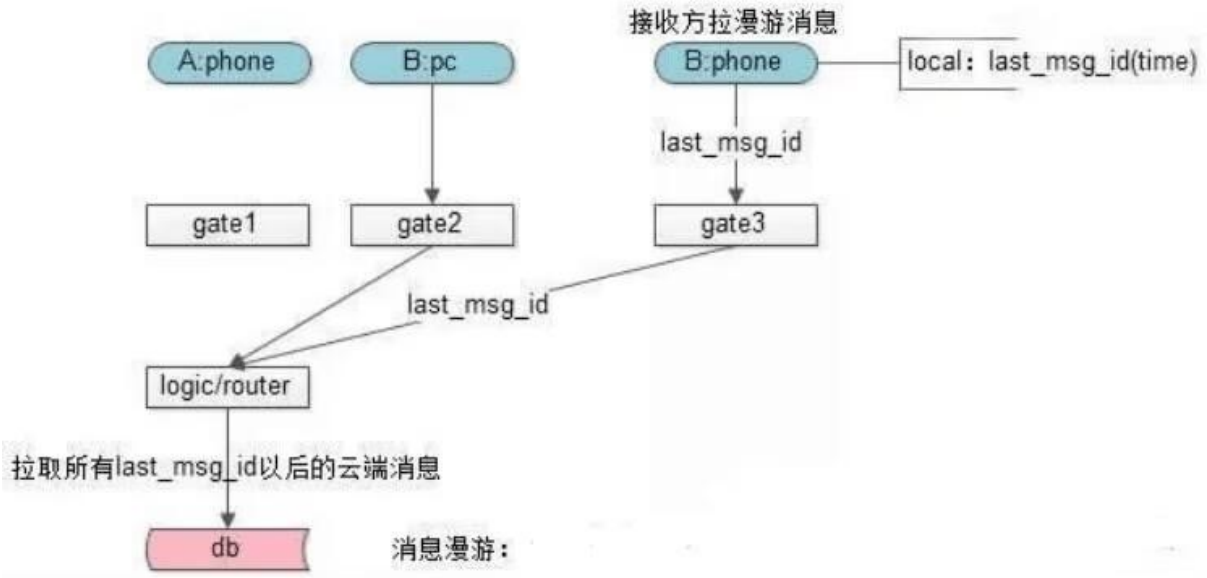
当用户A（phone端）给用户B发送消息时，除了要投递给B的所有多点登录端，还需要投递给A自己多点登陆的其他端（pc端），如上图中步骤4与步骤5。只有这样，才能在所有用户的所有端，恢复与还原双方聊天的上下文。

（4）消息漫游原理

如果业务不需要支持“消息漫游”的功能，对于在线消息，如果用户实时接收到则是不需要存储到数据库的。但如果要支持“换一台机器（指的是用户的客户端）也能看到历史的聊天消息”，就需要对所有消息进行存储。

目前来说，大多数的APP或者Web应用都会提前建立数据库并存储相关的信息，但是具体使用不使用主要还是依托于设置中的具体功能是否开启。

消息投递如上图：用户A发送消息给用户B，虽然B在线，仍然要增加一个步骤2.5，在投递之前进行存储，以备B的其他端登陆时，可以拉取到历史消息。



消息拉取如上图：原本不在线的B(phone端)，又重新登录了，他怎么拉取历史消息？只需要在客户端本地存储一个上一次拉取到的msg_id(time)，到服务端重新拉取即可。这里还有个问题：由于服务端存储所有消息成本是非常高的，所以一般“消息漫游”是有时间（或者消息数）限制，不能拉取所有所有几年前的历史消息，比如只能拉取3个月内的云端消息等。

（5）多点登录的总结

多点登录是指多个端同时登录一个帐号，同时收发消息，其关键点是：

- 需要在服务端存储同一个用户多个端的状态与登陆点；
- 发出消息时，要对发送方的多端与接收端的多端，都进行消息投递。

而“消息漫游”是指一个用户在任何端，都可以拉取到历史消息，关键点是：

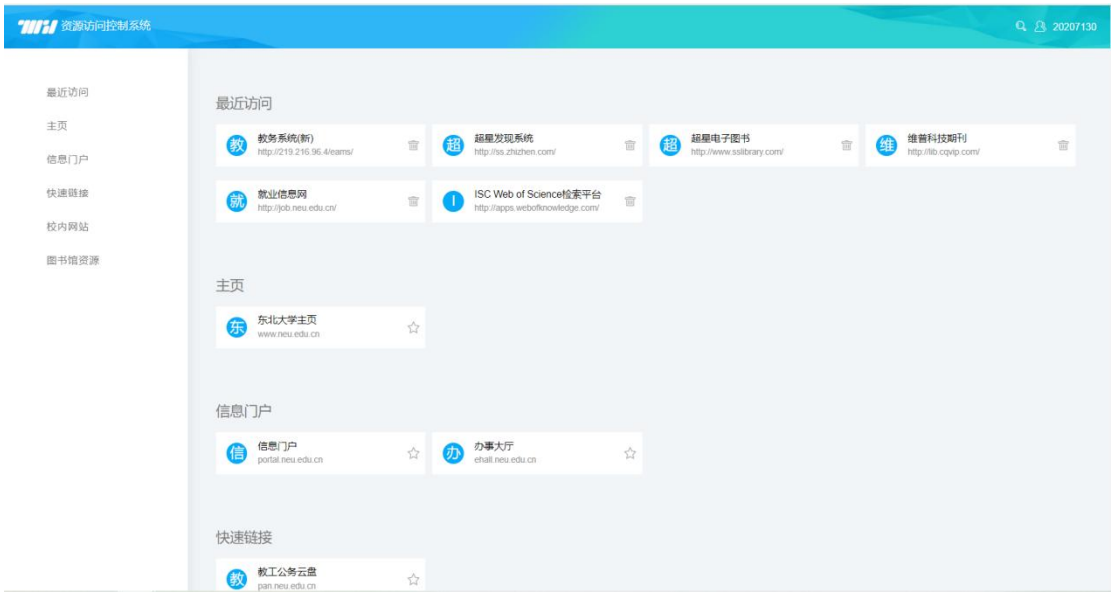
- 所有消息存储在云端；
- 每个端本地存储last_msg_id，在登录时可以到云端同步历史消息；
- 云端存储所有消息成本较高，一般会对历史消息时间(或者条数)进行限制。

五．教务系统登陆的测试

由于暂未返校，因此所有测试均通过VPN实现。

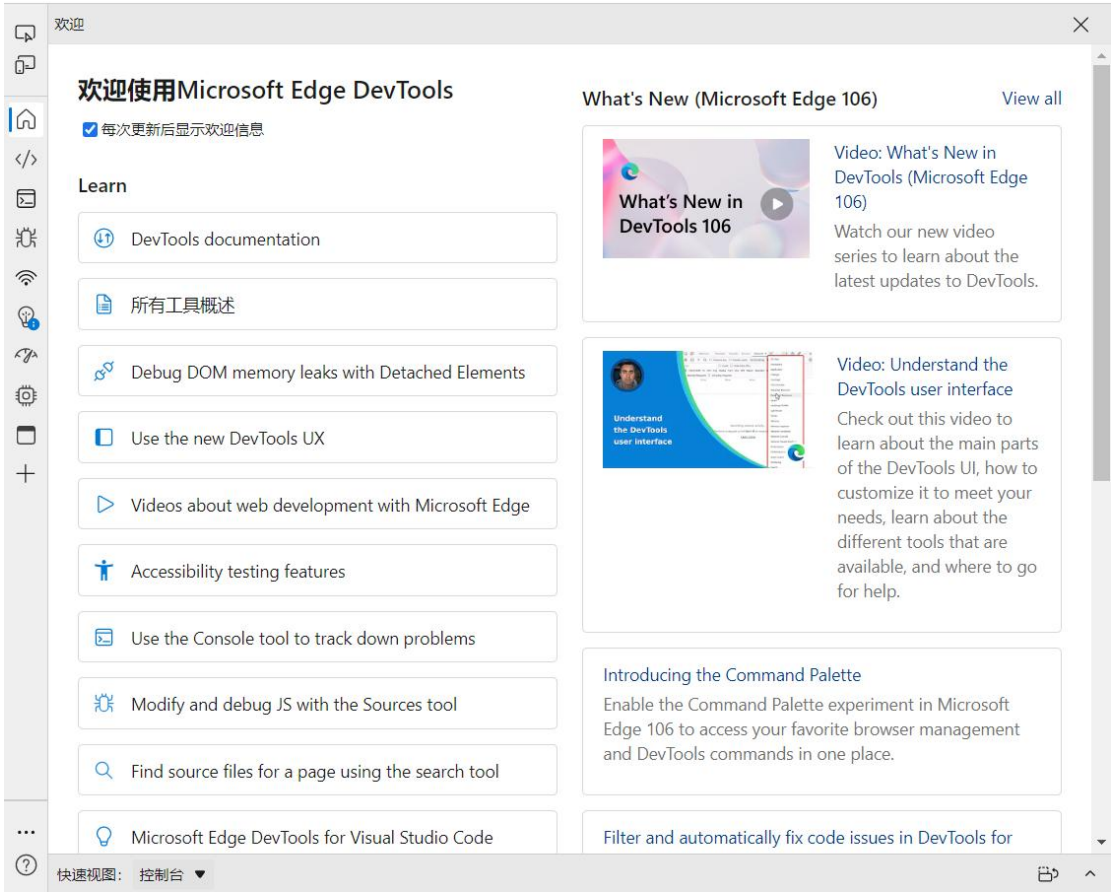
1. 点击校内资源下的VPN服务选项

2. 访问网址https://webvpn.neu.edu.cn/进入到其主页面



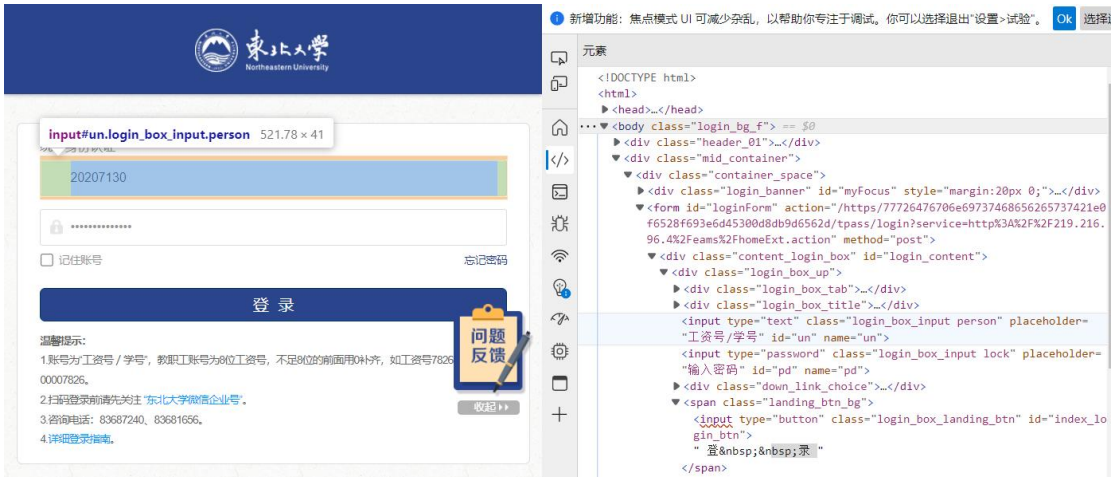
3. 点击教务系统选项，进入登录界面

4. 点击F12选项打开开发者模式



5. 进行相关的分析

首先需要找到登录所对应的表单，通过对整个页面HTML代码结构的分析，找到了登录的表单：



截取对应登陆部分表单：

```
<form id="loginForm" action="/https/77726476706e69737468656265737421e0f6528f693e6d453
00d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action"
method="post"> == $0
```

这里发现学校教务处的登录和传统的页面登陆不太一样：传统的登录会在表单form设置一个监听事件。例如表单提交成功后，设置onsubmit的key选项，对应的value就是触发监听事件所对应的JavaScript函数，例如加密函数处理对应的用户名和密码，或者将其组合在一起加密处理，传递给下一部分网页。

```
<span class="landing_btn_bg">
  <input type="button" class="login_box_landing_btn" id="index_login_btn">
  " 登  录 "
</span>
```

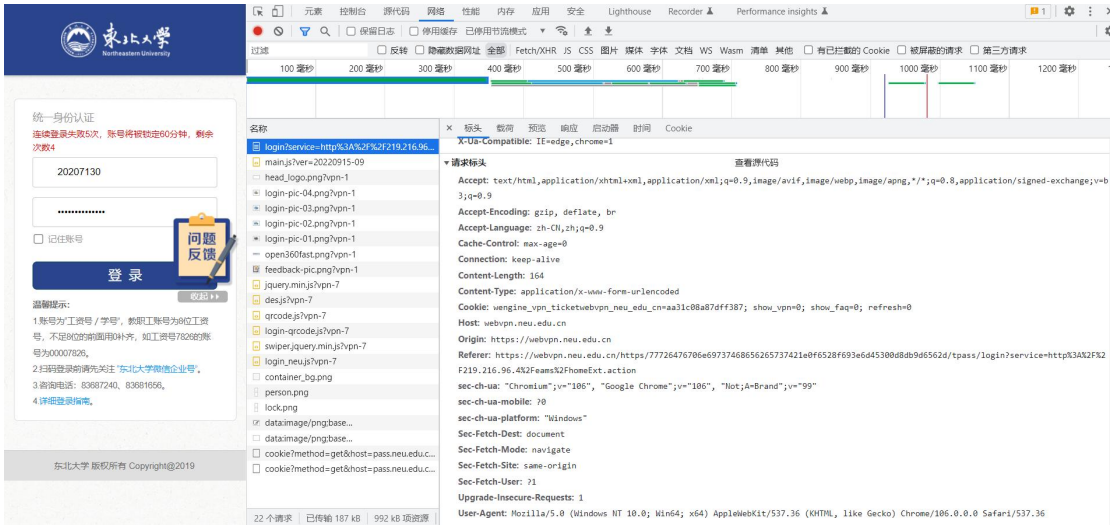
发现对应的登录按钮也没有涉及到相关的数据传递操作。

发现这个表单里面并没有涉及到相关页面逻辑的操作，仅仅存在文件名为https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action的数据包。最终发现在表单处有method=“post”字样，说明我们输入的账号密码应该是被表单加密处理后了的。这也给我们提供了一个思路，就是后续我们可以尝试跟踪分析数据包https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action里面数据的变化即可。

另外补充说明一下，我发现每次初次登录教务系统时，form表单的数据包名字会不一样，名字为：

https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login;jsessionid_tpass=3EoYWwmmK_5V7YlYB4Lbra2r3mJWGL_5MzUK6OQhX4fXP4h1ilLV!433186618?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action%3Bjsessionid%3D4C6F4C40C596000C1D4CBEA5F4B59A27.std10。目前并不知道为什么会有这样的机制。

首先故意登陆失败查看下对应数据的流向。



查看对应数据包的相关信息，发现对应表单Content-Type为application/x-www-form-urlencoded，百度学习了一下。当action为get时候，浏览器用x-www-form-urlencoded的编码方式把form数据转换成一个字串（name1=value1&name2=value2…），然后把这个字串append到url后面，用?分割，加载这个新的url。

输入正确的账号但是输入错误的密码，这里我输入的是1。然后查看数据包https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action的cookie：

```
Cookie: show_vpn=0; show_faq=0; wengine_vpn_ticketwebvpn_neu_edu_cn=bb7fd981e7a78f32; refresh=0
```

紧接着，又分别在数据包：

https://webvpn.neu.edu.cn/wengine-vpn/cookie?method=get&host=pass.neu.edu.cn&scheme=https&path=/tpass/login&vpn_timestamp=1666859453147发现了几乎一样的cookie：

```
Cookie: show_vpn=0; show_faq=0; wengine_vpn_ticketwebvpn_neu_edu_cn=bb7fd981e7a78f32; refresh=1
```

https://webvpn.neu.edu.cn/wengine-vpn/cookie?method=get&host=pass.neu.edu.cn&scheme=https&path=/tpass/login&vpn_timestamp=1666859453217发现了几乎一样的cookie：

```
Cookie: show_vpn=0; show_faq=0; wengine_vpn_ticketwebvpn_neu_edu_cn=bb7fd981e7a78f32; refresh=0
```

发现再提交失败的表单中，会传递请求方法method，请求主机地址host，scheme和时间戳vpn_timestamp所对应的数值：

另外又发现另一个数据包https://webvpn.neu.edu.cn/wengine-vpn/input也存在一个几乎相同的cookie:

Cookie: show_vpn=0; show_faq=0; wengine_vpn_ticketwebvpn_neu_edu_cn=bb7fd981e7a78f32; refresh=0

由于也是POST请求，我们查看input的form data：

发现这里仅仅是登陆表单里面的账号，查看请求的源地址和目的地址：

Host: webvpn.neu.edu.cn

Origin: https://webvpn.neu.edu.cn

Referer: https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action

发现这里出现了另外一个数据包：

https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action。我们猜测这个input数据包提交了对应的账号信息给上述数据包，但是由于我们输入的密码是错误的，因此页面的重定向失败导致这个数据包没有正常的显示出来。

重头戏来了，我们查看数据包：

https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action的源地址和目的地址：

Host: webvpn.neu.edu.cn

Origin: https://webvpn.neu.edu.cn

Referer: https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action

发现这个数据包也是重定向到数据包：

https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action。

接着我们查看数据包：

https://webvpn.neu.edu.cn/https/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action的form data，得到以下的结果：

上面表单的数据，我们发现rsa里面应该就是我们所填写的账号密码之后网页帮我们加密的信息。但是似乎并没有加密的样子呢？？？刚刚登录失败的情况下我们输入的是账号：20207130，密码：1。这里面rsa的前面若干项不就是账号、密码和lt字符串的简单拼接吗？因为现在暂时还不知道rsa里面的字符串后续是怎么处理的，所以这里应该存在着一定的安全隐患。ul应该是username length的缩写，指代的是账号的长度；pl应该是password length的缩写，指代的是密码的长度。Lt,execution和_eventld都暂时不知道什么含义。

到这里我们初步的了解了一些数据包的组成。上面采用登录失败的方法分析了大概的流程，下面就采用登陆成功的方式分析。

成功登陆之后，还是发现数据包：

https://webvpn.neu.edu.cn/http/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action依然为POST请求。查看下请求的源地址和目的地址：

```
Host: webvpn.neu.edu.cn
Origin: https://webvpn.neu.edu.cn
Referer: https://webvpn.neu.edu.cn/http/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action
```

紧接着查看对应的cookie：

```
Cookie: wengine_vpn_ticketwebvpn_neu_edu_cn=aa31c08a87dff387; show_vpn=0; show_faq=0; refresh=1
```

接着又发现接着出现了3个数据包（第一个数据包就是前面我们刚刚分析的）：

名称	状态	类型	启动器	大小	时间	瀑布
login?service=http%3A%2F%2F219.216.96.4%2Feams...	302	document / 重...	其他	502 B	327 毫秒	
homeExt.action?ticket=ST-1099047-JyK5SJC3TXSLNAa...	302	document / 重...	login?service=http%3A%2F...	303 B	92 毫秒	
homeExt.action	200	document	homeExt.action?ticket=ST-...	6.2 kB	132 毫秒	

将剩下的数据包：

https://webvpn.neu.edu.cn/http/77726476706e69737468656265737421a2a618d275613e1e275ec7f8/eams/homeExt.action?ticket=ST-1099047-JyK5SJC3TXSLNAa6AJsS-tpass记作数据包A。

https://webvpn.neu.edu.cn/http/77726476706e69737468656265737421a2a618d275613e1e275ec7f8/eams/homeExt.action记作数据包B。

通过上图我们发现，通过状态302和200可以知道数据包和页面发生了重定向，紧接着前面的数据包又作为后面数据包的启动器。并且重定向是成功的。

分别查看数据包A和B的cookie：

```
Cookie: wengine_vpn_ticketwebvpn_neu_edu_cn=aa31c08a87dff387; show_vpn=0; show_faq=0; refresh=1
Cookie: wengine_vpn_ticketwebvpn_neu_edu_cn=aa31c08a87dff387; show_vpn=0; show_faq=0; refresh=1
```

发现三个数据包的cookie是完全相同的。

接着我们分析数据包：

https://webvpn.neu.edu.cn/http/77726476706e69737468656265737421e0f6528f693e6d45300d8db9d6562d/tpass/login?service=http%3A%2F%2F219.216.96.4%2Feams%2FhomeExt.action的form data：

非常令人气愤的是在rsa选项里面，自己输入的账号和密码居然一个不少的出现在了表单rsa里面。（这也是为什么我打了马赛克的原因）。所以这里推测学校在这里的表单提交里面并没有使用RSA算法进行加密。然后网络上搜索到了一篇文章，里面这样写道：

传输数据一般用到的方式是post和get，先来说一下他们的特点：

get：传输的数据量小，传输的速度相对较快，但是传输的数据会显示在地址栏中，能够看到，所以不安全；

post：传输的数据量比get大，所以相应的速度会较慢，但是地址栏不会显示传输的数据，相对安全；

根据他们的特点我们会发现：

如果我们传输的数据不需要加密，地址栏显示数据也不影响，推荐用get，一般查询即从服务端上获取数据都用的get；

但是我们传输的数据需要加密，则推荐用post，比如你账号的登录、注册都用post；

但是post加密只是相对安全，因为js都是明文的，解密难度并不高，所以一般比较重要的数据传输会采用在浏览器安装安全控件进行加密，比如你在浏览器上登录银行账号（ps：这个方式安全系数高，但是成本比较高），还有ssl方式加密，不过这些一般就是在后端加密了。

由于这里提交表单肯定是采用POST方法，POST方法是需要对数据加密的，因此这里又猜测学校教务系统没有在前端对这些数据进行加密，而是在后端采用RSA算法进行加密的。

由于数据包A是GET请求，因此我们可以查询对应GET请求的字符串：

这也验证了我们在前文所分析的单点登录的原理。这里我们可以复习一下，在技术实现上，单点登录可以分为跨子域单点登录和完全跨域单点登录。跨子域单点登录即在具有相同根域名的站点之间实现单点登录。而这里的ticket应该就是实现单点登录所对应的令牌。

在后续的分析中，还发现一个比较有趣的东西。双击教务系统里面的数据包：https://webvpn.neu.edu.cn/http/77726476706e69737468656265737421a2a618d27561

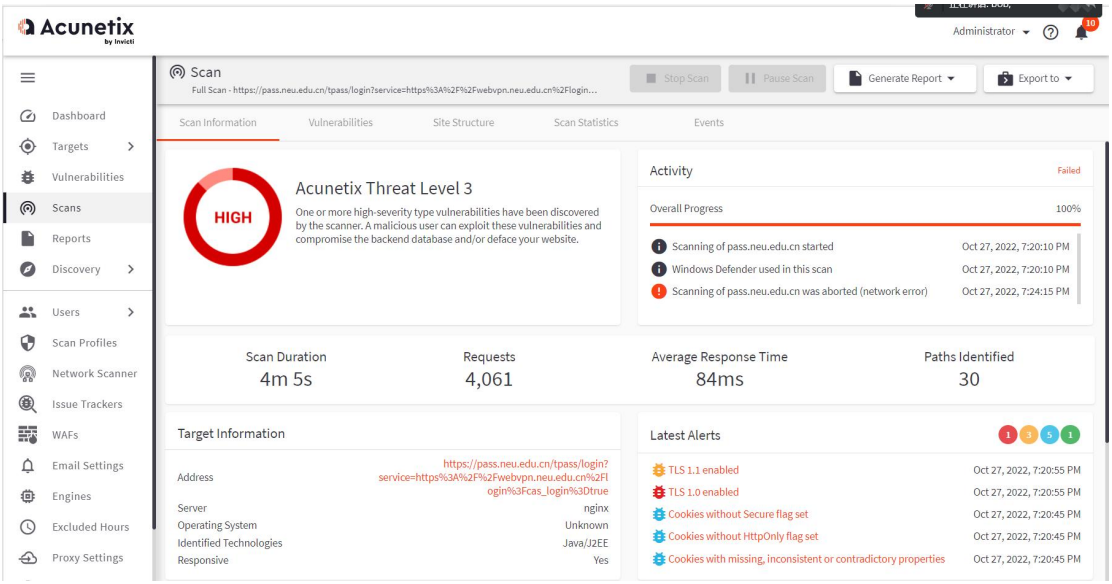
3e1e275ec7f8/eams/static/scripts/my97/My97DatePicker.htm，居然会发现跳转到了这样的页面：

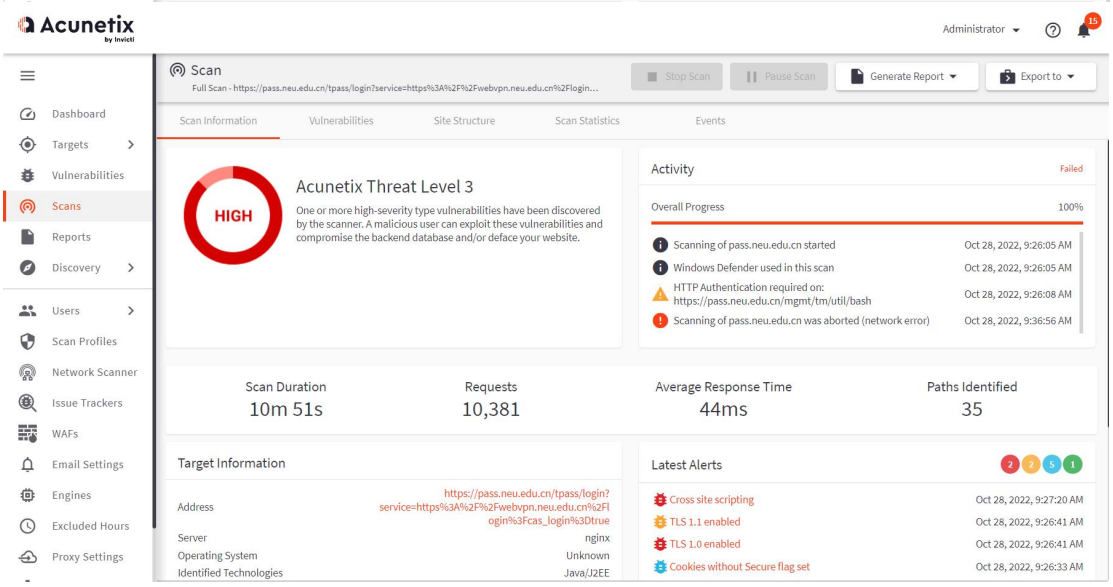
这里推测应该是教务系统自用的日历API.....

考虑到可能校园网下的登录系统和VPN下的登录系统可能存在着差异，因此在这里请求了在校同学的协助下，也测试分析了校园网下的相关登录流程，发现和VPN是几乎完全一样的。账号密码在提交表单过程中依然采用明文传输，这个是十分危险的！！！查阅资料得知表单如果数据没有加密可以采用HTTPS传输协议安全传输数据，但是也没发现学校教务系统使用的是HTTPS协议呢。不知道学校当时设计这个教务系统在登陆表单的数据传输时，是如何设计加密传输的。

六．采用AWVS分析登录网站安全性

1. 分析概要





点击Targets，选择Add targets创建新目标，输入目标的网址（这里输入东北大学教务系统的登录网站：

https://pass.neu.edu.cn/tpass/login?service=https%3A%2F%2Fwebvpn.neu.edu.cn%2Flogin%3Fcas_login%3Dtrue），点击扫描，扫描选项选择最简单的基本配置即可。

AWVS显示，东北大学教务系统安全等级处于Acunetix Threat Level 3,处于高威胁High。尽管后续由于网络问题失去对网站：

https://pass.neu.edu.cn/tpass/login?service=https%3A%2F%2Fwebvpn.neu.edu.cn%2Flogin%3Fcas_login%3Dtrue的连接，但是仍然发现若干问题，安全性很低。

整个扫描持续4分05秒（10分51秒）左右，共计向目标网站发送4061（10381）个请求，平均回应时间为84ms（44ms），已识别的路由有30个(35个)。

学校采用的服务器为NGINX。Nginx (engine x) 是一个高性能的HTTP和反向

代理web服务器，Nginx是一款轻量级的Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，在BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好。NGINX具有：高并发、高性能、可扩展性好、高可靠性、热部署和BSD许可证的特点。

2. NGINX

这里从网上截取了关于NGINX服务器性能的对比表格。

表 1-2　Nginx 与 Apache、Lighttpd 的综合对比

Web 服务器	Nginx	Apache	Lighttpd
反向代理	非常好	好	一般
Rewrite 规则	非常好	好	一般
FastCGI	好	差	非常好
热部署	支持	不支持	不支持
系统压力比较	很小	小	很大
稳定性	非常好	好	一般
安全性	一般	好	一般
技术资料	很少	非常多	一般
静态文件处理	非常好	一般	好
虚拟主机	支持	支持	支持
内存消耗	非常小	很大	非常小

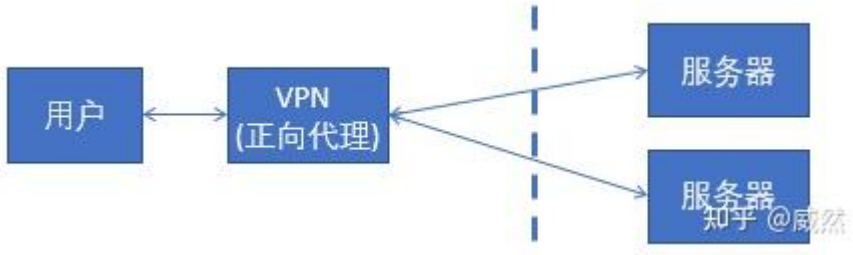
Nginx (engine x) 是一款轻量级的Web 服务器 、反向代理服务器及电子邮件（IMAP/POP3）代理服务器。

(1) 反向代理

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

(2) 正向代理

是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端才能使用正向代理。



反向代理是作用在服务器端的，是一个虚拟ip(VIP)。对于用户的一个请求，会转发到多个后端处理器中的一台来处理该具体请求。

(3) 总结

东北大学教务系统采用NGINX服务器，也是考虑到了其高并发、高性能、可扩展性好、高可靠性和热部署的相关特点。这也为登录功能的高效率和高速度提供了技术支持。

3. 对登陆系统安全漏洞的详细分析

（1）安全漏洞概要

Severity	Vulnerability	URL	Parameter	Status	Confidence %
High	TLS 1.0 enabled	https://pass.neu.edu.cn/		Open	100
Medium	Same site scripting	https://pass.neu.edu.cn/		Open	95
Medium	TLS 1.1 enabled	https://pass.neu.edu.cn/		Open	100
Medium	Vulnerable JavaScript libraries	https://pass.neu.edu.cn/		Open	100
Low	Clickjacking: X-Frame-Options header	https://pass.neu.edu.cn/		Open	95
Low	Cookies with missing, inconsistent or contradictory properties	https://pass.neu.edu.cn/		Open	100
Low	Cookies without HttpOnly flag set	https://pass.neu.edu.cn/		Open	100
Low	Cookies without Secure flag set	https://pass.neu.edu.cn/		Open	100
Low	HTTP Strict Transport Security (HSTS) not implemented	https://pass.neu.edu.cn/		Open	95
Informational	Content Security Policy (CSP) not implemented	https://pass.neu.edu.cn/		Open	95

Severity	Vulnerability	URL	Parameter	Status	Confidence %
High	Cross site scripting	https://pass.neu.edu.cn/tpass/logout	service	Open	100
High	TLS 1.0 enabled	https://pass.neu.edu.cn/		Open	100
Medium	TLS 1.1 enabled	https://pass.neu.edu.cn/		Open	100
Medium	Vulnerable JavaScript libraries	https://pass.neu.edu.cn/		Open	100
Low	Clickjacking: X-Frame-Options header	https://pass.neu.edu.cn/		Open	95
Low	Cookies with missing, inconsistent or contradictory properties	https://pass.neu.edu.cn/		Open	100
Low	Cookies without HttpOnly flag set	https://pass.neu.edu.cn/		Open	100
Low	Cookies without Secure flag set	https://pass.neu.edu.cn/		Open	100
Low	HTTP Strict Transport Security (HSTS) not implemented	https://pass.neu.edu.cn/		Open	95
Informational	Content Security Policy (CSP) not implemented	https://pass.neu.edu.cn/		Open	95

对学校的登录网站进行了多次扫描，上图仅仅展示了其中两次比较有代表性的扫描漏洞结果。大约存在着1-2个高危漏洞（High）,2-3个中等危险漏洞（medium），5个低危漏洞（low）和一个信息漏洞（informational）。

高危漏洞（High）：Cross site scripting，TLS 1.0 enabled。

中等危险漏洞（medium）：TLS 1.1 enabled，Vulnerable JavaScript libraries，Same site scripting。

低危漏洞（low）：Clickjacking: X-Frame-Options header，Cookies with missing、inconsistent or contradictory properties，Cookies without HttpOnly flag set，Cookies without Secure flag set，HTTP Strict Transport Security (HSTS) not implemented。

信息漏洞（informational）：Content Security Policy (CSP) not implemented。

（2）High：TLS 1.1 enabled

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

SSL 服务器(端口号:443)使用 TLSv1.0 对流量进行加密。

③脆弱性描述（Vulnerability Description）

web 服务器支持通过 TLS 1.0 加密，但由于固有的安全问题，该协议已于 2021 年 3 月正式弃用。此外，TLS 1.0 在用于保护传输到或来自网站的敏感信息时，不被认为是 PCI 数据安全标准 3.2(1)所定义和要求的“强密码学”。根据 PCI，“2018 年 6 月 30 日是禁用 SSL/早期 TLS 和实施更安全的加密协议 TLS 1.1 或更高(强烈鼓励 TLS v1.2)的最后期限，以满足保护支付数据的 PCI 数据安全标准(PCI DSS)。”

④漏洞影响（The impact of this vulnerability）

攻击者可能会利用这个问题进行中间人攻击，并解密受影响的服务和客户端之间的通信。

⑤如何修复漏洞（How to fix this vulnerability）

建议禁用 TLS 1.0，使用 TLS 1.2 及以上版本。

⑥漏洞分类（Classification）

CWE	CWE-326
CVSS	Base Score: **5.4** - CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:N Attack Vector: Network Attack Complexity: High Privileges Required: None User Interaction: None Scope: Changed Confidentiality: Low Integrity: Low Availability: None

（3）High:Cross site scripting

①URL 来源

https://pass.neu.edu.cn/tpass/logout

②攻击细节（Attack Detail）

URL 编码 GET 输入服务设置为:
https://portal.neu.edu.cn/tp_up'"()&%<acx></acx>

③脆弱性描述（Vulnerability Description）

跨站脚本(XSS)是指客户端代码注入攻击，攻击者可以在合法网站或web应用程序中执行恶意脚本。当web应用程序在其生成的输出中使用未经验证或未编码的用户输入时，就会发生XSS。

④漏洞影响（The impact of this vulnerability）

恶意 JavaScript 可以访问与 web 页面其他部分相同的所有对象，包括访问 cookie 和本地存储(通常用于存储会话令牌)。如果攻击者能够获得用户的会话 cookie，他们就可以模拟该用户。

此外，JavaScript 可以读取并对显示给用户的页面内容进行任意修改。因此，XSS 结合一些巧妙的社会工程为攻击者提供了许多可能性。

⑤如何修复漏洞（How to fix this vulnerability）

对页面上呈现的用户输入应用依赖于上下文的编码和/或验证.

⑥漏洞分类（Classification）

CWE	CWE-79
CVSS	Base Score: **5.3** - CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: None Scope: Unchanged Confidentiality: None Integrity: Low Availability: None

（4）Medium:TLS 1.1 enabled

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

SSL 服务器(端口号:443)使用 TLSv1.1 对流量进行加密。

③脆弱性描述（Vulnerability Description）

web 服务器支持通过 TLS 1.0 加密，但由于固有的安全问题，该协议已于 2021 年 3 月正式弃用。此外，TLS 1.0 在用于保护传输到或来自网站的敏感信息时，不被认为是 PCI 数据安全标准 3.2(1)所定义和要求的“强密码学”。根据 PCI，“2018 年 6 月 30 日是禁用 SSL/早期 TLS 和实施更安全的加密协议 TLS 1.1 或更高(强烈鼓励 TLS v1.2)的最后期限，以满足保护支付数据的 PCI 数据安全标准(PCI DSS)。”

④漏洞影响（The impact of this vulnerability）

攻击者可能会利用这个问题进行中间人攻击，并解密受影响的服务和客户端之间的通信。

⑤如何修复漏洞（How to fix this vulnerability）

建议禁用 TLS 1.0，使用 TLS 1.2 及以上版本。

⑥漏洞分类（Classification）

CWE	CWE-326
CVSS	Base Score: **5.4** - CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:L/A:N Attack Vector: Network Attack Complexity: High Privileges Required: None User Interaction: None Scope: Changed Confidentiality: Low Integrity: Low Availability: None

（5）Medium:Vulnerable JavaScript libraries

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

URL: https://pass.neu.edu.cn/tpass/comm/js/jquery.min.js

检测方法:根据文件的语法指纹和内容确定库的名称和版本。Acunetix使用该文件唯一的语法指纹验证了库版本和相关漏洞，该文件的语法指纹与Acunetix所期望的语法指纹匹配。

③脆弱性描述（Vulnerability Description）

您正在使用一个或多个易受攻击的 JavaScript 库。该库的这个版本报告了一个或多个漏洞。有关受影响的库和报告的漏洞的更多信息，请参阅攻击详细信息和 Web 参考。

④漏洞影响（The impact of this vulnerability）

暂不明确。

⑤如何修复漏洞（How to fix this vulnerability）

建议更新到最新版本。

⑥漏洞分类（Classification）

CWE	CWE-937
CVSS	Base Score: **6.5** - CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: None Scope: Unchanged Confidentiality: Low Integrity: Low Availability: None

（6）Medium:Same site scripting

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

Host: localhost.pass.neu.edu.cn解析为127.0.0.2。

③脆弱性描述（Vulnerability Description）

Tavis Ormandy报告了一个常见的DNS错误配置，它会导致web应用程序的一个小安全问题。

安装形式为“localhost”的记录是一种常见而明智的做法。然而，奇怪的是，在命名服务器配置的A 127.0.0.1中，管理员经常错误地去掉后面的点，引入了一种有趣的跨站点脚本(XSS)变体，我称之为同站点脚本。缺少的圆点表示记录不是完全限定的，因此将解析形式为“localhost.example.com”的查询。虽然表面上这似乎是无害的，但实际上它确实允许攻击者欺骗RFC2109 (HTTP状态管理机制)同源限制，从而劫持状态管理数据。”

④漏洞影响（The impact of this vulnerability）

攻击者可以欺骗 RFC2109 (HTTP 状态管理机制)的同源限制，从而劫持状态管理数据。

⑤如何修复漏洞（How to fix this vulnerability）

建议将非 FQ 的 localhost 条目从托管依赖 HTTP 状态管理的网站的域名服务器配置中删除。

⑥漏洞分类（Classification）

CWE	CWE-16
CVSS	Base Score: **3.7** - CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N Attack Vector: Network Attack Complexity: High Privileges Required: None User Interaction: None Scope: Unchanged Confidentiality: None Integrity: Low Availability: None

（7）Low：Clickjacking: X-Frame-Options header

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

没有安全XFO头的路径:https://pass.neu.edu.cn/tpass/login

③脆弱性描述（Vulnerability Description）

点击劫持(用户界面纠正攻击，UI纠正攻击，UI纠正)是一种恶意的技术，它欺骗Web用户点击一些与用户所感知的不同的内容，从而有可能泄露机密信息或在点击看似无害的网页时控制用户的计算机。

服务器没有返回值为DENY或SAMEORIGIN的X-Frame-Options报头，这意味着该网站可能存在点击劫持攻击的风险。X-Frame-Options HTTP响应报头可用于指示是否允许浏览器在框架或iframe中呈现页面。网站可以利用这一点来避免点击劫持攻击，确保他们的内容没有嵌入到不受信任的网站。

④漏洞影响（The impact of this vulnerability）

影响取决于受影响的 web 应用程序。

⑤如何修复漏洞（How to fix this vulnerability）

配置你的web服务器，包括一个X-Frame-Options报头和一个带有frame-祖宗指令的CSP报头。

⑥漏洞分类（Classification）

CWE	CWE-1021
CVSS	Base Score: **5.8** - CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:L/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: None Scope: Changed Confidentiality: None Integrity: Low Availability: None

（8）Low：Cookies with missing, inconsistent or contradictory properties

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

缺少、不一致或矛盾属性的cookie列表:

- https://pass.neu.edu.cn/tpass/login

Cookie被设置为：

Set-Cookie: Language=zh_CN; expires=Thu, 03-Nov-2022 11:20:12 GMT; path=/

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

- https://pass.neu.edu.cn/tpass/login

Cookie 被设置为：

Set-Cookie:

jsessionid_tpass=BYUZK896fcayib4ky_HJQ4vkV7bsohHZ1NLam7J3BzwHB_dAaSQY!-1734464462; path=/; HttpOnly

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

- https://pass.neu.edu.cn/tpass/login

Cookie 被设置为：

Set-Cookie:

jsessionid_tpass=yHIZK8-JGQ9mDP29XYLFeKTHnEIvbqfCBo-CqFEMmb3yxaf_H_uF!-1734464462; path=/; HttpOnly

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

- https://pass.neu.edu.cn/

Cookie 被设置为：

Set-Cookie:

JSESSIONID=szMZLC0n6ZpDRHqj0zMDCy7QP-5Ok3_Gdr7BuRak-3avPzJbi4mC!-498418953; path=/; HttpOnly

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

- https://pass.neu.edu.cn/tpass/login

Cookie 被设置为：

Set-Cookie: Language=zh_CN; expires=Thu, 03-Nov-2022 11:20:36 GMT; path=/

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

- https://pass.neu.edu.cn/tpass/login

Cookie 被设置为：

Set-Cookie: jsessionid_tpass=FQwZLC3IwAgOFtho9m69vaJTnj16qAFDThnN68HoAG4W_O0Kuz8-!-498418953; path=/; HttpOnly

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

- https://pass.neu.edu.cn/tpass/login

Cookie 被设置为：

Set-Cookie: jsessionid_tpass=I3UZLC3OkiLdc2B6lXcTMweYbGm3WE_luWWLsK_PEwjgE2dV4BWg!-498418953; path=/; HttpOnly

Cookie 有以下的信息：

- Cookie without SameSite attribute.

When cookies lack the SameSite attribute, Web browsers may apply different and sometimes unexpected defaults. It is therefore recommended to add a SameSite attribute with an appropriate value of either "Strict", "Lax", or "None".

③脆弱性描述（Vulnerability Description）

下列cookie属性中的至少一个会导致cookie无效或与同一cookie的不同属性或与使用该cookie的环境不兼容。尽管这本身不是一个漏洞，但它可能会导致应用程序出现意料之外的行为，进而可能导致次要安全问题。

④漏洞影响（The impact of this vulnerability）

网络浏览器不会存储或提交cookie。

⑤如何修复漏洞（How to fix this vulnerability）

请确保cookie配置符合相关标准。

⑥漏洞分类（Classification）

CWE	CWE-284
CVSS	Base Score: **0** - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: Required Scope: Unchanged Confidentiality: None Integrity: None Availability: None

（9）Low：Cookies without HttpOnly flag set

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

没有设置HttpOnly标志的cookie:

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: Language=zh_CN; expires=Thu, 03-Nov-2022 11:20:12 GMT; path=/

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: Language=zh_CN; expires=Thu, 03-Nov-2022 11:20:36 GMT; path=/

③脆弱性描述（Vulnerability Description）

一个或多个cookie没有设置HttpOnly标志。当cookie被设置为HttpOnly标志时，它指示浏览器该cookie只能由服务器访问，而不能由客户端脚本访问。这是对会话cookie的重要安全保护。

④漏洞影响（The impact of this vulnerability）

可以通过客户端脚本访问 cookie。

⑤如何修复漏洞（How to fix this vulnerability）

如果可能的话，您应该为这些cookie设置HttpOnly标志。

⑥漏洞分类（Classification）

CWE	CWE-1004
CVSS	Base Score: **0** - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: Required Scope: Unchanged Confidentiality: None Integrity: None Availability: None

（10）Low：Cookies without Secure flag set

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

没有设置安全标志的cookie:

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: Language=zh_CN; expires=Thu, 03-Nov-2022 11:20:12 GMT; path=/

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: jsessionid_tpass=BYUZK896fcayib4ky_HJQ4vkV7bsohHZ1NLam7J3BzwHB_dAaSQY!-1734464462; path=/; HttpOnly

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: jsessionid_tpass=yHIZK8-JGQ9mDP29XYLFeKTHnEIvbqfCBo-CqFEMmb3yxaf_H_uF!-1734464462; path=/; HttpOnly

- https://pass.neu.edu.cn/

Set-Cookie: JSESSIONID=szMZLC0n6ZpDRHqj0zMDCy7QP-5Ok3_Gdr7BuRak-3avPzJbi4mC!-498418953; path=/; HttpOnly

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: Language=zh_CN; expires=Thu, 03-Nov-2022 11:20:36 GMT; path=/

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: jsessionid_tpass=FQwZLC3IwAgOFtho9m69vaJTnj16qAFDThnN68HoAG4W_O0Kuz8-!-498418953; path=/; HttpOnly

- https://pass.neu.edu.cn/tpass/login

Set-Cookie: jsessionid_tpass=I3UZLC3OkiLdc2B6lXcTMweYbGm3WE_luWWLsK_PEwjgE2dV4BWg!-498418953; path=/; HttpOnly

③脆弱性描述（Vulnerability Description）

一个或多个cookie没有设置安全标志。当cookie被设置为Secure标志时，它指示浏览器只能通过安全的SSL/TLS通道访问该cookie。这是对会话cookie的重要安全保护。

④漏洞影响（The impact of this vulnerability）

cookie可以通过未加密的通道发送。

⑤如何修复漏洞（How to fix this vulnerability）

如果可能的话，应该为这些cookie设置Secure标志。

⑥漏洞分类（Classification）

CWE	CWE-614
CVSS	Base Score: **0** - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: Required Scope: Unchanged Confidentiality: None Integrity: None Availability: None

（11）Low：HTTP Strict Transport Security (HSTS) not implemented

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

未启用HSTS的url:https://pass.neu.edu.cn/tpass/login

③脆弱性描述（Vulnerability Description）

HTTP严格传输安全(HSTS)告诉浏览器一个网站只能通过HTTPS访问。检测到您的web应用程序没有实现HTTP严格传输安全(HSTS)，因为响应中缺少严格传输安全报头。

④漏洞影响（The impact of this vulnerability）

HSTS可用于防止和/或减轻某些类型的中间人(MitM)攻击。

⑤如何修复漏洞（How to fix this vulnerability）

建议在您的web应用程序中实现HTTP严格传输安全(HSTS)。

⑥漏洞分类（Classification）

CWE	CWE-16
CVSS	Base Score: **0** - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:N/I:N/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: Required Scope: Changed Confidentiality: None Integrity: None Availability: None

（12）Informational：Content Security Policy (CSP) not implemented

①URL 来源

https://pass.neu.edu.cn/

②攻击细节（Attack Detail）

没有CSP头的路径:https://pass.neu.edu.cn/tpass/login

③脆弱性描述（Vulnerability Description）

内容安全策略(CSP)是一个额外的安全层，它有助于检测和减轻某些类型的攻击，包括跨站脚本(XSS)和数据注入攻击。

内容安全策略(CSP)可以通过添加内容安全策略标头来实现。此头的值是一个字符串，包含描述内容安全策略的策略指示。要实现CSP，您应该为站点使用的所有资源类型定义允许的源列表。例如，如果你有一个简单的网站，需要加载脚本，样式表和本地托管的图像，以及从jQuery库从他们的CDN, CSP头文件可能看起来像这样:

```
Content-Security-Policy:
    default-src 'self';
    script-src 'self' https://code.jquery.com;
```

检测到您的web应用程序没有实现内容安全策略(CSP)，因为响应中缺少CSP头。建议在web应用程序中实现内容安全策略(CSP)。

④漏洞影响（The impact of this vulnerability）

CSP可用于防止和/或减轻涉及内容/代码注入的攻击，如跨站点脚本/XSS攻击、需要嵌入恶意资源的攻击、涉及恶意使用iframes的攻击，如点击劫持攻击等。

⑤如何修复漏洞（How to fix this vulnerability）

建议在web应用程序中实现内容安全策略(CSP)。配置内容安全策略包括向网页添加内容安全策略HTTP标头，并给它赋值以控制允许用户代理为该页面加载的资源。

⑥漏洞分类（Classification）

CWE	CWE-1021
CVSS	Base Score: **0** - CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:N/I:N/A:N Attack Vector: Network Attack Complexity: Low Privileges Required: None User Interaction: Required Scope: Changed Confidentiality: None Integrity: None Availability: None

4. 对东北大学教务系统安全性分析的总结

通过AWVS软件分析出来的结果也验证了我在这个部分前面一个部分里面的部分猜想。下面做一些总结：

- TLS 传输协议版本老旧。攻击者可能会利用这个问题进行中间人攻击，并解密受影响的服务和客户端之间的通信。建议禁用 TLS 1.0，使用 TLS 1.2 及以上版本。
- 网站容易受到 XSS 攻击。登录网站恶意 JavaScript 可以访问与 web 页面其他部分相同的所有对象，包括访问 cookie 和本地存储(通常用于存储会话令牌)。如果攻击者能够获得用户的会话 cookie，他们就可以模拟该用户。此外，JavaScript 可以读取并对显示给用户的页面内容进行任意修改。攻击者容易直接进入系统窃取关键信息。建议对表单数据提前进行验证处理。
- JavaScript 库容易受到攻击和篡改。建议使用最新版本的 JS 库。
- DNS 存在一个错误配置。Host: localhost.pass.neu.edu.cn 解析为 127.0.0.2。攻击者可以欺骗 RFC2109 (HTTP 状态管理机制)的同源限制，从而劫持状态管理数据。建议将非 FQ 的 localhost 条目从托管依赖 HTTP 状态管理的网站的域名服务器配置中删除。
- Cookie 不符合要求规范。会导致 cookie 无效或与同一 cookie 的不同属性或与使用该 cookie 的环境不兼容。尽管这本身不是一个漏洞，但它可能会导致应用程序出现意料之外的行为，进而可能导致次要安全问题。建议配置符合相关标准的 cookie。
- 一个或多个 cookie 没有设置 HttpOnly 标志。当 cookie 被设置为 HttpOnly 标志时，它指示浏览器该 cookie 只能由服务器访问，而不能由客户端脚本访问。这是对会话 cookie 的重要安全保护。攻击者可以通过客户端脚本访问 cookie。建议为这些 cookie 设置 HttpOnly 标志。
- 一个或多个 cookie 没有设置安全标志。当 cookie 被设置为 Secure 标志时，它指示浏览器只能通过安全的 SSL/TLS 通道访问该 cookie。这是对会话 cookie 的重要安全保护。可能会导致 cookie 可以通过未加密的通道发送，进而被窃取。建议为这些 cookie 设置 Secure 标志。
- 没有实现 HTTP 严格传输安全(HSTS)。这也验证了我的猜想。HSTS 可用于防止和/或减轻某些类型的中间人(MitM)攻击。建议在您的 web 应用程序中实现 HTTP 严格传输安全(HSTS)。
- web 应用程序没有实现内容安全策略(CSP)。CSP 可用于防止和/或减轻涉及内容/代码注入的攻击，如跨站点脚本/XSS 攻击、需要嵌入恶意资源的攻击、

涉及恶意使用 iframes 的攻击，如点击劫持攻击等。建议在 web 应用程序中实现内容安全策略(CSP)。配置内容安全策略包括向网页添加内容安全策略 HTTP 标头，并给它赋值以控制允许用户代理为该页面加载的资源。

附录：

《计算机系统安全报告》成绩评定表

评价表格

考核标准	得分
（1）正确理解和掌握实验所涉及的概念和原理（20%）；	
（2）按实验要求合理设计数据结构和程序结构，运行结果正确（40%）；	
（3）实验过程中，具有严谨的学习态度和认真、踏实、一丝不苟的科学作风，实验报告规范；认真记录实验数据，原理及实验结果分析准确（40%）；	
合计	