第三次作业

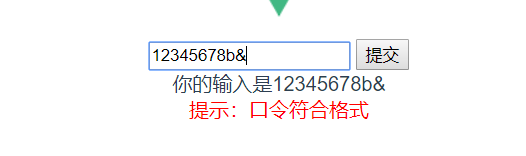
（1）测试点3-1

口令检测程序，要求：必须包含大写字母，小写字母，数字，特殊字符中的三种，要求长度8到30位。

语言：js，思路：使用正则表达式验证的方法，

核心代码

|  |
| --- |
| watch: {  *// eslint-disable-next-line vue/no-dupe-keys* password: *function* () {  *// eslint-disable-next-line max-len  if* (/^(?![a-zA-Z]+$)(?![A-Z0-9]+$)(?![A-Z\W\_!@#$%^&\*`~()-+=]+$)(?![a-z0-9]+$)(?![a-z\W\_!@#$%^&\*`~()-+=]+$)(?![0-9\W\_!@#$%^&\*`~()-+=]+$)[a-zA-Z0-9\W\_!@#$%^&\*`~()-+=]{8,30}$/.test(*this*.password)) {  *this*.tip = '口令符合格式'  } *else* {  *this*.tip = '口令不符合格式，数字大小写字母特殊字符四者之三，8到30位'  }  } } |

结果：

（2）测试点3-2

1. needhamschroeder协议缺陷和改进

中间人攻击：中间人在第三步A->B发送消息时，截获消息，并伪装B向A发送一个格式相同的随机数，A使用Kab解密，等同于加密，然后发送给B，中间者随即也能截获。

重放攻击：攻击者可以直接跳到第三步发送一个旧的Kab给B，从而不需要认证就能和B进行通信。

改进方案：造成缺陷的原因是不确定消息是否新鲜，以及B无法认证A,所以需要加入时间戳以及公钥证书。

2.公钥密码的Needham-Schroeder的分析。

风险：攻击者可以两次运行NS公钥协议进行攻击。

攻击者第1次NS公钥协议运行

消息1.1：A→Z(A)：{Na，A}Kz

此时，入侵者Z开始第2次NS协议运行：

消息2.1：Z(A)→B：{Na，A}Kb

消息2.2：B→Z(A)：{Na，Nb}Ka

消息1.2：Z(A) →A：{Na，Nb}ka

消息1.3：A→Z(A)：{Nb}Kz

消息2.3：Z(A)→B：{Nb}Kb

入侵者Z通过解密消息1.1和消息1.3获取发送消息2.1和消息2.3所需的Na和Nb，消息1.2则是消息2.2的重放上述协议运行完，主体B认为他与A共享秘密Nb，实际上他与共享Nb，Z假冒A成功，攻击有效.对于网络系统中任何一个合法用户，只要接收到发给自己的NS公钥协议消息1，就可以发起上述攻击，以欺骗另外一个用户，故NS公钥协议是不安全的。

改进方案：

A→B：{Na，A}Kb

B→A：{Na，Nb，B}Ka

A→B：{Nb}Kb