Web安全技术

韩伟力

内容安排

- 安全技术简介
- 数据源层的安全
- 应用服务层的安全
- HTTP安全
- 展示层安全
  - XSS及其防御
  - CSRF及其防御
  - 钓鱼及其攻击
  - 口令管理
  - CAPTCHA

内容安排

1. 安全技术简介
2. 数据源层的安全
3. 应用服务层的安全
4. HTTP安全
5. 展示层安全

什么是安全

- 信息安全是一个十分广泛而又复杂的话题
- 美国国家电信和信息安全委员会 (NSTISSC)
  - 信息安全是对信息、系统以及使用、存储和传输信息的硬件的保护。但是要保护信息及其相关系统，诸如策略、认识、培训和教育以及技术手段都是必要的。
- 微软公司 M. Howard, D. LeBlance
  - 一个安全的产品应该是指在信息所有者或者系统管理员控制下能够保护客户数据的机密性、完整性和可用性，能够保护资源处理过程的完整性和有效性。
- 机密性 (Confidentiality)、完整性 (Integrity) 和可用性 (Availability) 的组合

机密性

- **机密性**是指保证计算机相关的有价值财产（信息）只能被授权过的用户所访问。
- 机密性的保护
  - 认证和访问控制
  - 加密

完整性

- **完整性**是指这些计算机相关的有价值财产（信息）只能被授权过的用户所修改，或者通过授权过的过程所修改。
- 完整性的保护
  - 认证和访问控制
  - 加密

可用性

- **可用性**是指这些有价值的财产（信息）在需要的时候必须能够被授权的用户访问或者修改。
- 可用性的破坏
  - DOS: Denial Of Service

可用性破坏案例

- SYN Flood的基本原理
  - SYN Flood是当前最流行的DoS（拒绝服务攻击）与DDoS（分布式拒绝服务攻击）的方式之一，这是一种利用TCP协议缺陷，发送大量伪造的TCP连接请求，从而使得被攻击方资源耗尽（CPU满负荷或内存不足）的攻击方式。
  - 要明白这种攻击的基本原理，还是要从TCP连接建立的过程开始说起：TCP与UDP不同，它是基于连接的，也就是说：为了在服务端和客户端之间传送TCP数据，必须先建立一个虚拟电路，也就是TCP连接，建立TCP连接的标准过程是这样的：
    - 首先，请求端（客户端）发送一个包含SYN标志的TCP报文，SYN即同步（Synchronize），同步报文会指明客户端使用的端口以及TCP连接的初始序号；
    - 第二步，服务器在收到客户端的SYN报文后，将返回一个SYN+ACK的报文，表示客户端的请求被接受，同时TCP序号被加一，ACK即确认（Acknowledgement）。
    - 第三步，客户端也返回一个确认报文ACK给服务器端，同样TCP序列号被加一，到此一个TCP连接完成。
    - 以上的连接过程在TCP协议中被称为三次握手（Three-way Handshake）。

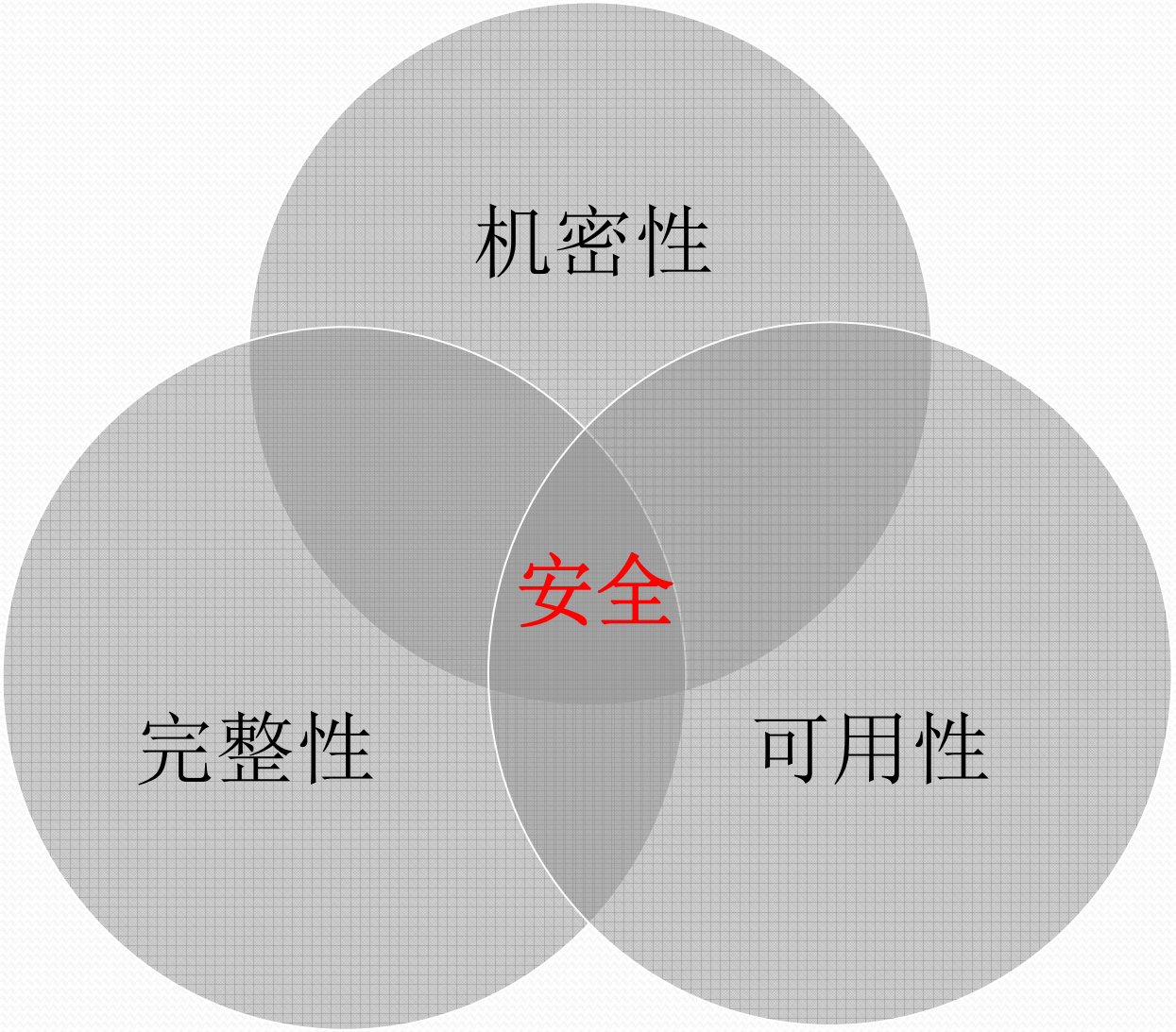
可用性破坏案例(续)

- 问题就出在TCP连接的三次握手中，假设一个用户向服务器发送了SYN报文后突然死机或掉线，那么服务器在发出SYN+ACK应答报文后是无法收到客户端的ACK报文的（第三次握手无法完成），这种情况下服务器端一般会重试（再次发送SYN+ACK给客户端）并等待一段时间后丢弃这个未完成的连接，这段时间的长度我们称为SYN Timeout，一般来说这个时间是分钟的数量级（大约为30秒-2分钟）；一个用户出现异常导致服务器的一个线程等待1分钟并不是什么很大的问题，但如果有一个恶意的攻击者大量模拟这种情况，服务器端将为了维护一个非常大的半连接列表而消耗非常多的资源--数以万计的半连接，即使是简单的保存并遍历也会消耗非常多的CPU时间和内存，何况还要不断对这个列表中的IP进行SYN+ACK的重试。实际上如果服务器的TCP/IP栈不够强大，最后的结果往往是堆栈溢出崩溃--即使服务器端的系统足够强大，服务器端也将忙于处理攻击者伪造的TCP连接请求而无暇理睬客户的正常请求（毕竟客户端的正常请求比率非常之小），此时从正常客户的角度看来，服务器失去响应，这种情况我们称作：服务器端受到了SYN Flood攻击（SYN洪水攻击）。

可用性破坏案例(续)

- 从防御角度来说，有几种简单的解决方法：
  - 第一种是缩短SYN Timeout时间，由于SYN Flood攻击的效果取决于服务器上保持的SYN半连接数，这个值=SYN攻击的频度 x SYN Timeout，所以通过缩短从接收到SYN报文到确定这个报文无效并丢弃改连接的时间，例如设置为20秒以下（过低的SYN Timeout设置可能会影响客户的正常访问），可以成倍的降低服务器的负荷。
  - 第二种方法是设置SYN Cookie，就是给每一个请求连接的IP地址分配一个Cookie，如果短时间内连续受到某个IP的重复SYN报文，就认定是受到了攻击，以后从这个IP地址来的包会被丢弃。
- 可是上述的两种方法只能对付比较原始的SYN Flood攻击，缩短SYN Timeout时间仅在对方攻击频度不高的情况下生效，SYN Cookie更依赖于对方使用真实的IP地址，如果攻击者以数万/秒的速度发送SYN报文，同时利用SOCK_RAW随机改写IP报文中的源地址，以上的方法将毫无用武之地。

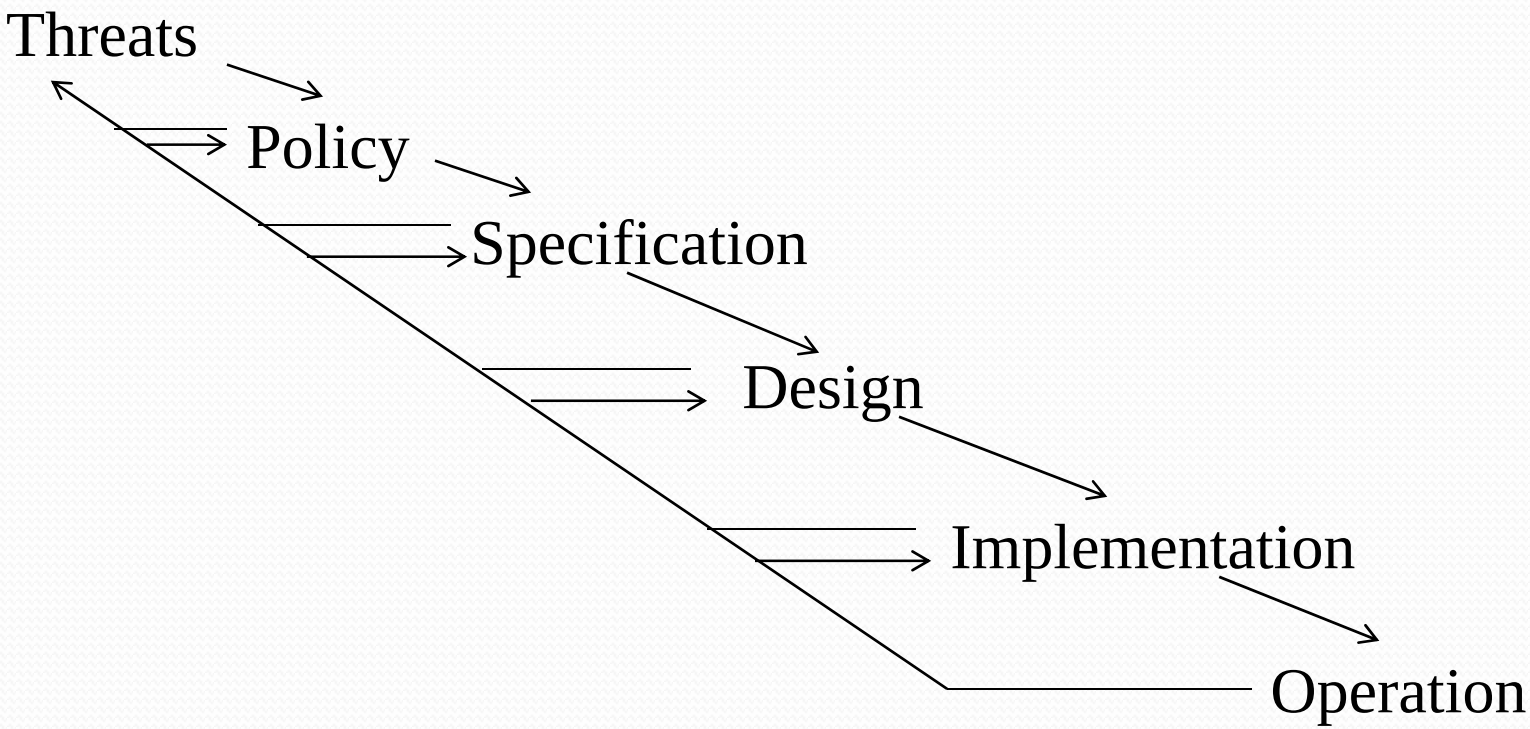
机密性，完整性和可用性及其之间的关系



C, I, A分类

- 小李拷贝了小王的作业
- 小李让小王的计算机崩溃了
- 小李将小王的支票从100元修改到1000元
- 小李冒用死去的老张的签名
- 小李注册了一个域名www.fudan.org，并拒绝复旦大学购买并且使用这个域名
- 小李得到小王的信用卡卡号并让信用卡公司删除这个卡，然后重新办理了新的卡，并使用原有卡的信用
- 小李哄骗小王计算机的IP检测，得到了访问小王计算机的访问许可

研究安全的方法学


Threats
Policy
Specification
Design
Implementation
Operation

保护信息系统安全的技术

- 认证
- 授权与访问控制
- 审计
- 加密和解密

身份认证

- 客户端需要向一个或者多个服务器或者服务证实自己的身份
- 三种基本方式

  ① 用户所知道的某个秘密信息，如自己的口令；

  ②用户持有的某个秘密信息，如智能卡中存储的用户个人化参数；

  ③用户所具有的某些生物学特征，如指纹、声音、DNA图案等。
- 你知道什么，你拥有什么，你是什么。
- 其中第一种最为常用

授权与访问控制

- 访问控制的三个主要研究对象：

  Subject，Object，Policy

- 访问控制的三个主要功能：

  Authorization, Revocation, Evaluation

- 访问控制定义：

是解决谁（主体）对某个特定对象（客体）具有何种权限的一项系统安全技术。它包括两方面的关键技术，一方面是安全策略的制定技术，即如何表达主体对客体有何种权限，另一方面是安全策略的实现技术，即如何将制定的策略在系统中有效地执行。

审计

- 审计：一个过程－该过程中，为了汇报可以计量的信息同已建立的标准之间的关系，由一个独立的个体积累和评估事实
- 检查可接受标准一致性的过程称为审计
- 负责独立检查计算机安全的部门称为审计部门
- 审计促进了实际环境同已建立的策略和指导原则的一致性，从而提高了企业计算安全的整体级别
- 审计部门必须同他所要服务的计算群体保持一个良好的关系

加密和解密

- 对称加密
  - DES，3DES，AES
- 非对称加密
  - RSA，ECC
- 单向加密
  - MD5，SHA-1，SHA-256

内容安排

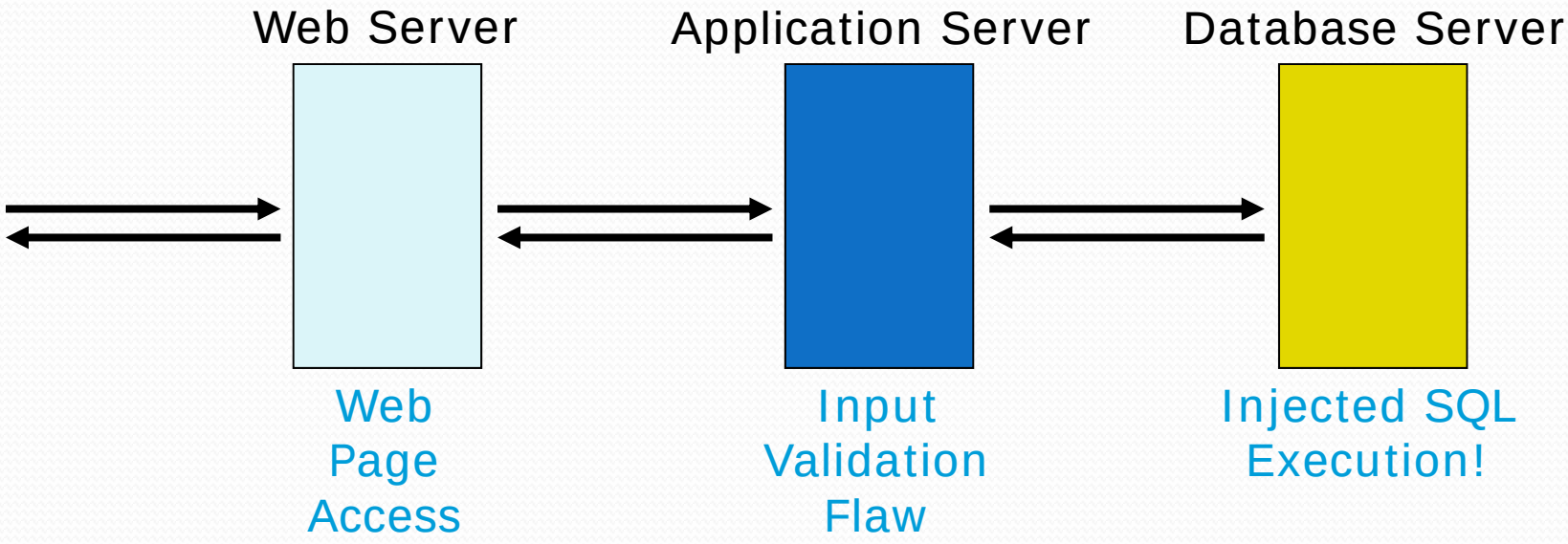
1. 安全技术简介
2. 数据源层的安全
3. 应用服务层的安全
4. HTTP安全
5. 展示层安全

数据源层的安全问题

- 本身的安全机制
  - 认证
  - 访问控制
  - 审计
- SQL注入
- 加密存储

SQL注入攻击

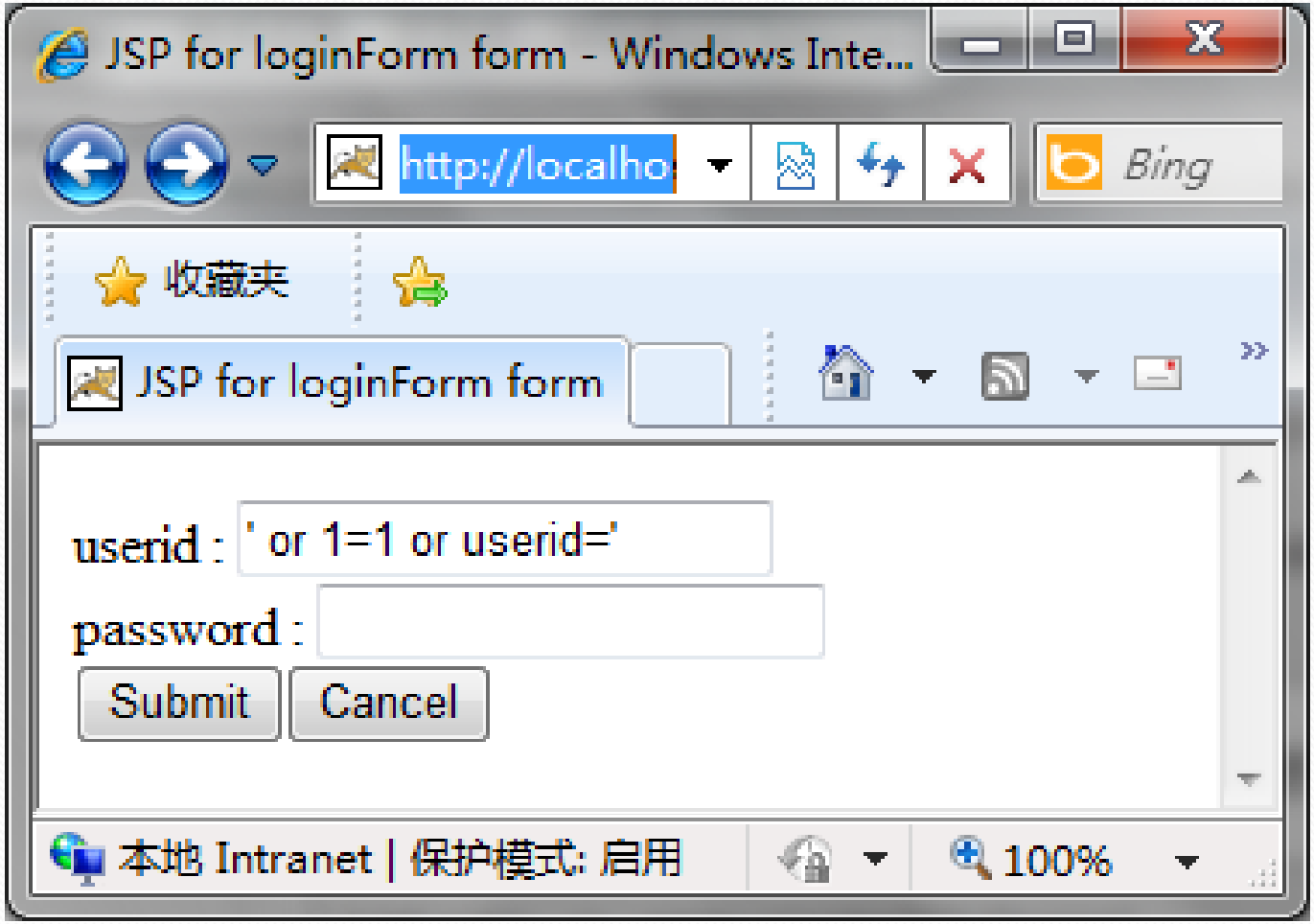
- 应用层攻击
  - 很难防御的攻击行为
- 在应用操作界面直接操作数据库，以获取对敏感数据的非法访问许可



SQL注入的简单例子 – 数据库表格

```
create table Users(
    inner_id        int         not null
            unique identity(1, 1),
    userid                      char (64)
            primary key,
    passwd                      varchar (64),
    username        varchar (128),
    gender                      varchar (1)
            not null,
    email                       varchar (64)
            not null,
    status int      not null
            default 0
)
```

SQL注入的简单例子－page



SQL注入的简单例子－code

```
public boolean validateLoginSQLInjection2(String userid, String password,
    String authcode){
        List<User> userlist = userDAO.getHibernateTemplate().find("from User where userid = '" + userid +"' and
            password='"+password+"'" );
            try{
                if (userlist.size()>0)
                    return true;
            }
            catch(Exception e){
                    return false;
            }
            return false;
}
```

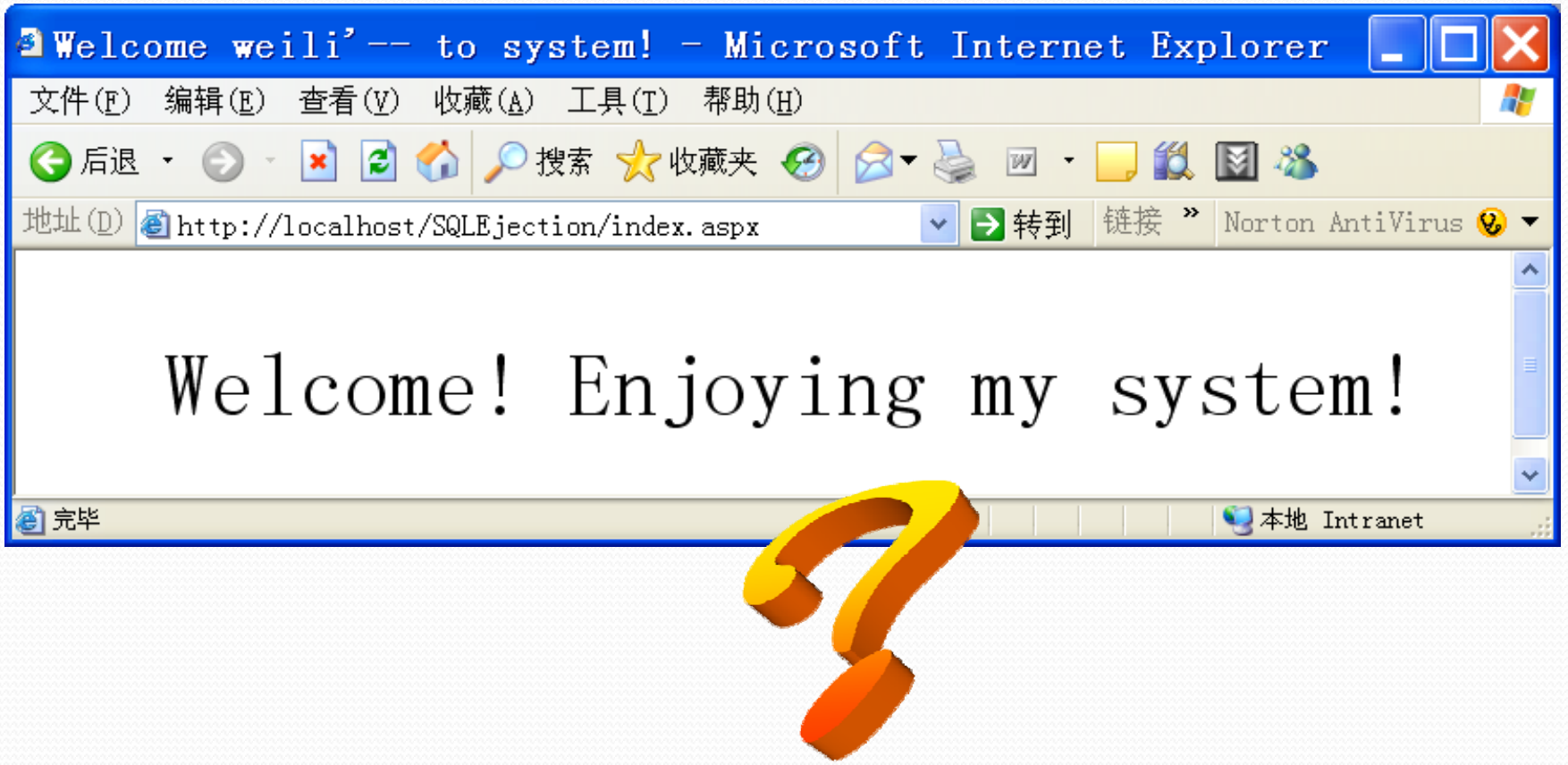
SQL注入的简单例子 – initial data

insert into Users (userid, passwd, username, gender, email, status)

values

('weili', 'weili', 'han weili', 'M', 'wlhan@fudan.edu.cn', 0)

跳过检查


Welcome weili' -- to system! - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/SQLEjection/index.aspx
转到 链接 Norton AntiVirus
Welcome! Enjoying my system!
完毕
本地 Intranet

通过SQL注入实现DoS

- weili' shutdown --

修改URL

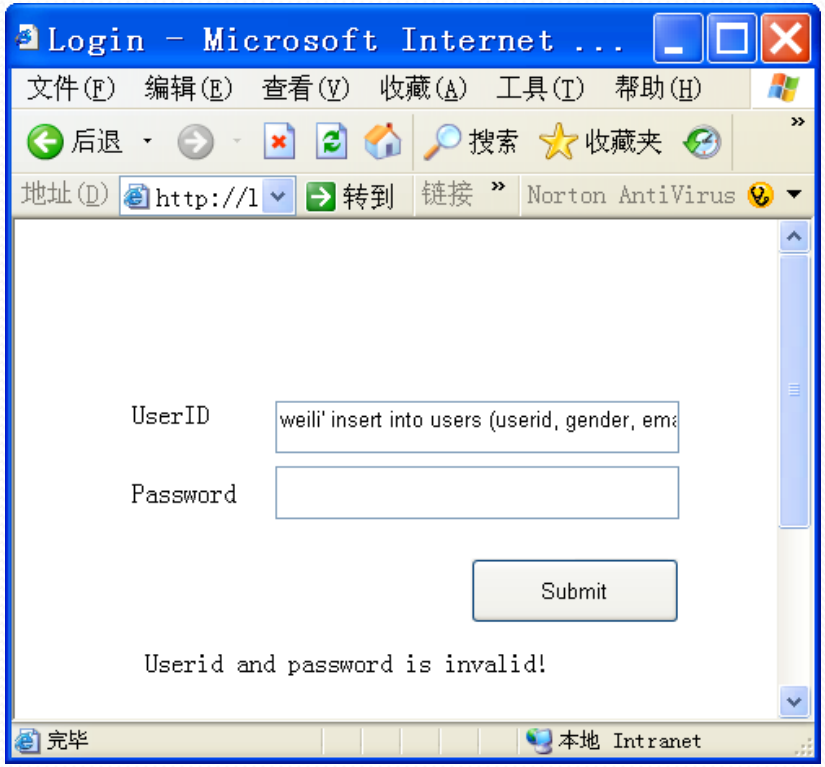
- http://***.***.***/showarticle?id=12
- http://***.***.***/showarticle?id=12 or 1=1
- http://www.fudan.edu.cn/fudannews/news_content.php?channel=1&id=7923

存储过程和SQL注入攻击

- 存储过程可以有效减少SQL注入的发生
- 但是不能通过存储过程彻底消灭SQL注入的发生

```
create Procedure proc_SQLEject
    @Password char(64),
    @UserID     char(64)
as
    SELECT count(*) FROM users WHERE
            UserID = @UserID AND Passwd = @Password;
    if @@RowCount = 0
    begin
            return 0;
    end
    else
    begin
            return 1;
    end
```

- weili' insert into users (userid, gender, email, status) values ('weili2', 'm', 'sdd', 1) --



最具有破坏力的存储过程

```
create procedure sp_MySProc @input varchar(128)
as
   exec(@input)
   go
```

防御SQL注入的有效方案－程序员的视角

- 不得使用sa来连接数据。而是使用个替代的最小特权用户进行连接；
- 不能在程序中直接生成SQL语句。而是使用参数进行创建；
- 永远不允许使用空口令来连接数据库；
- 代码严格地限制合法的输入形式，比如用户ID只能为字符串和数字的混合；
- 代码尽量使用存储过程来实现对数据库的操作，目的是在如果代码被破坏，也可以尽量隐藏应用程序逻辑；
- 发生错误以后，攻击者除了知道“发生错误”以外，得不到任何信息；
- 不管代码是否失败，总是关闭对数据库的连接。

内容安排

1. 安全技术简介
2. 数据源层的安全
3. 应用服务层的安全
4. HTTP安全
5. 展示层安全

应用服务层的安全

- 本身的安全机制
  - 认证
  - 访问控制
  - 审计

内容安排

1. 安全技术简介
2. 数据源层的安全
3. 应用服务层的安全
4. **HTTP安全**
5. 展示层安全

HTTP安全 - HTTPS

- HTTP+SSL/TLS = HTTPS
- 用于敏感信息的传输
- 实现端到端的数据机密性和完整性
- 与IPSec的关系

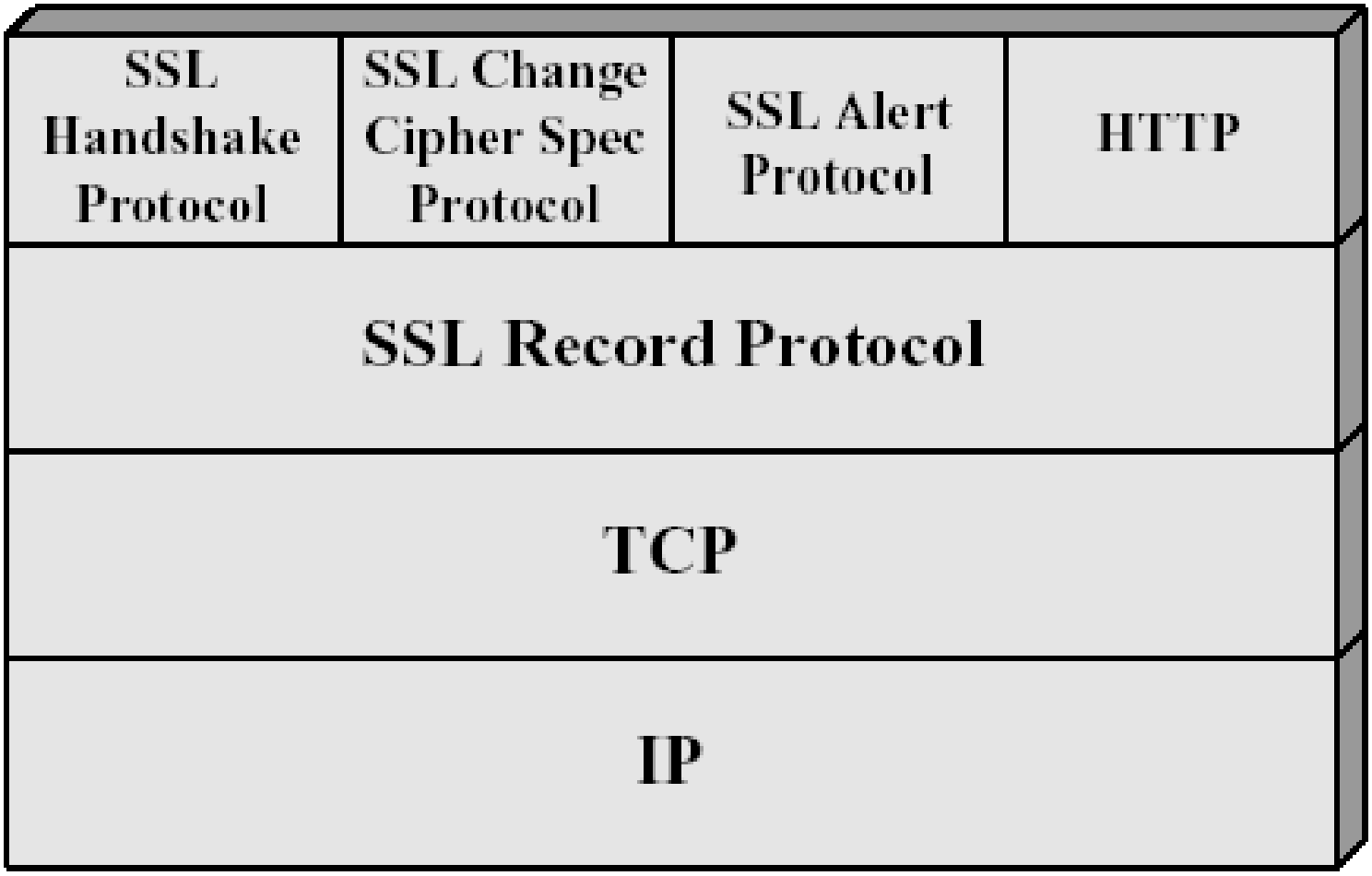
SSL的加密的安全性

- SSL用公共密钥加密，来在客户与服务器之间交换一个进程密钥，这个密钥用来加密HTTP传输过程(包括请求和响应)。每次传输采用不同的密钥，因而即使某些人能够破译某次传输，并不意味着他们发现了服务器的密码，如果他们想要破译下一次，他们就必须付出像第一次那样的时间和努力。

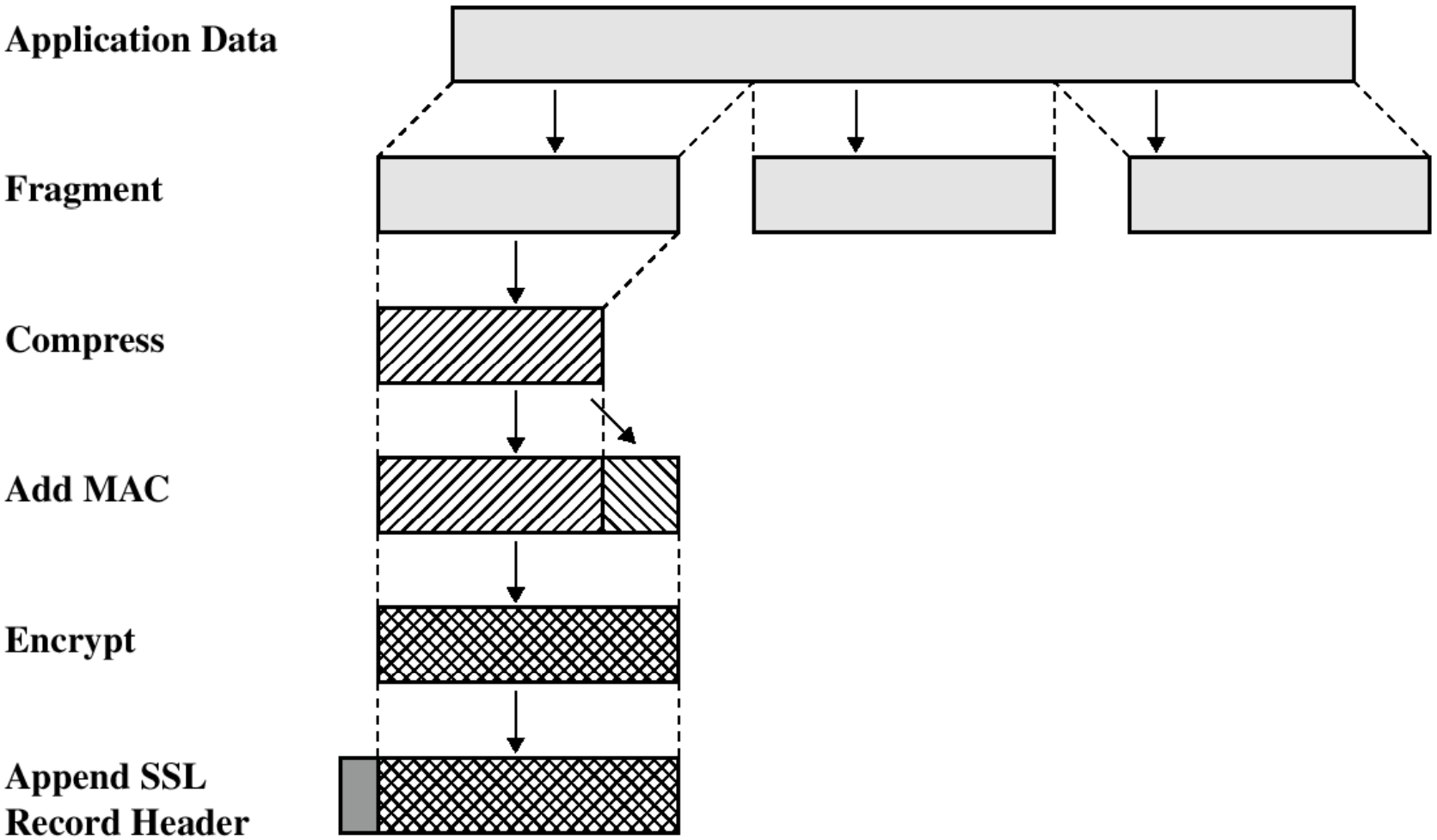
SSL和TLS

- 安全套结层（SSL）由Netscape公司提出
- SSLv3作为Internet的草案文档
- 专门成立了TLS（传输层安全）工作组来开发公共标准
- TLSv1接近于SSLv3.1

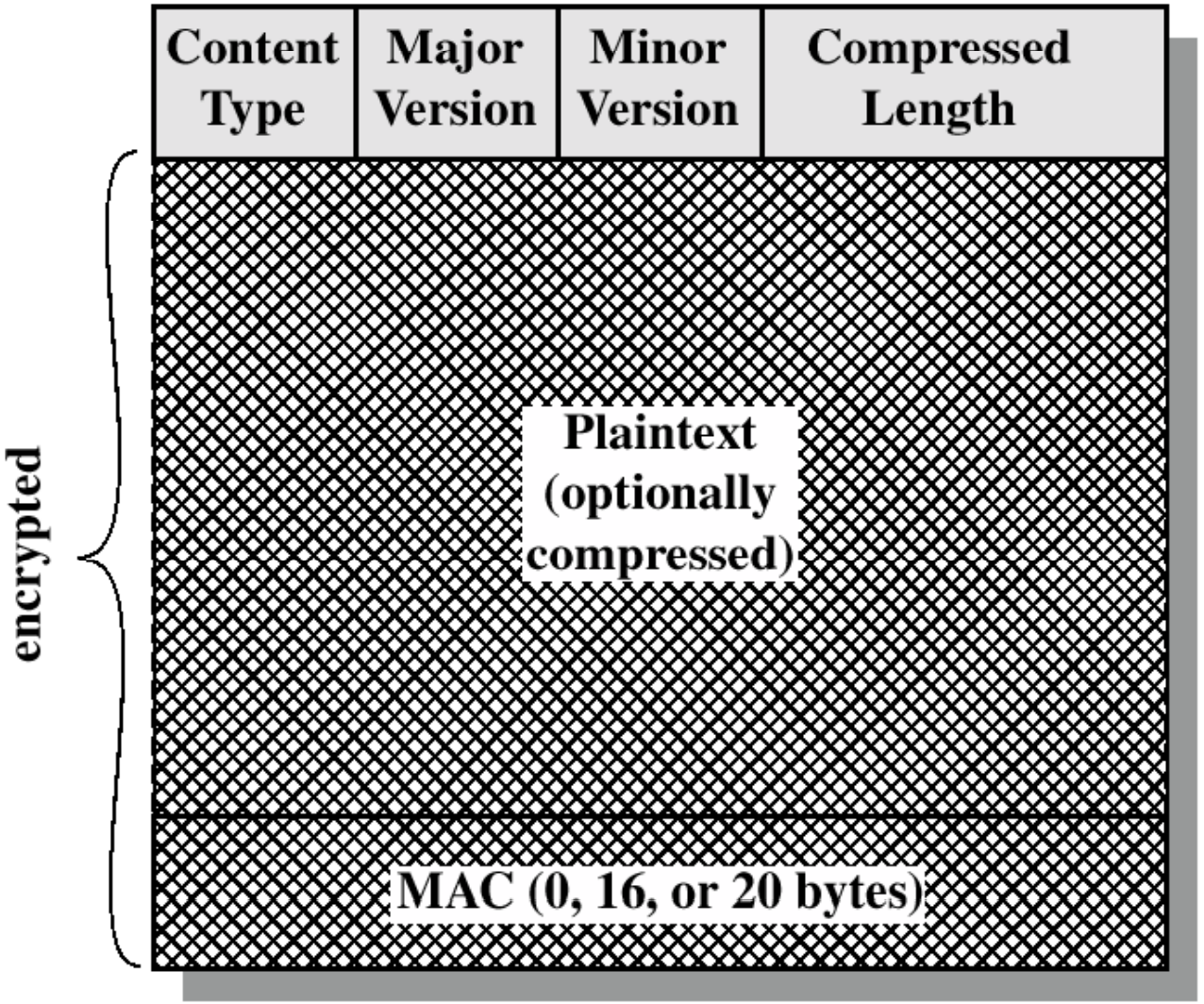
SSL协议栈


SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP
SSL Record Protocol
TCP
IP

SSL记录协议的操作

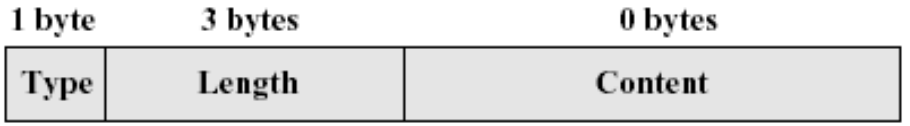

Application Data
Fragment
Compress
Add MAC
Encrypt
Append SSL
Record Header

SSL记录的格式

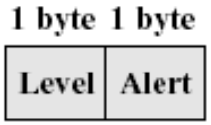

Content Type
Major Version
Minor Version
Compressed Length
Plaintext (optionally compressed)
MAC (0, 16, or 20 bytes)
encrypted

SSL记录协议的有效数据

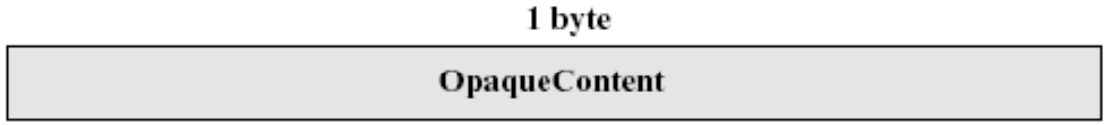
(a) Change Cipher Spec Protocol



(c) Handshake Protocol



(b) Alert Protocol

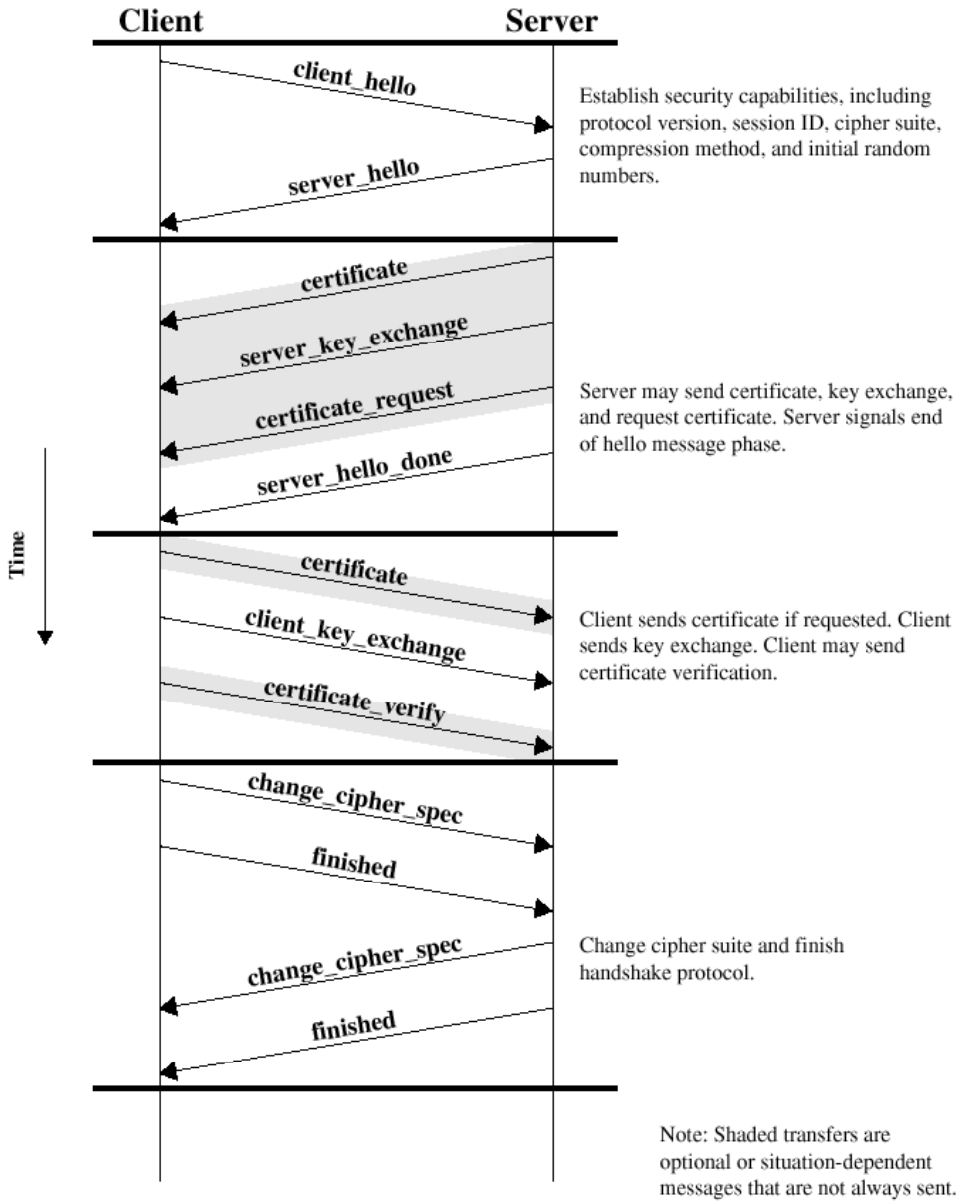


(d) Other Upper-Layer Protocol (e.g., HTTP)

握手协议

- **SSL中最复杂的部分**
- 使服务器和客户能够相互鉴别
- 协商加密和**MAC**算法以及加密密钥
- 在传输应用数据前，先执行握手协议

握手协议动作序列


Client
Server
client_hello
server_hello
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

阶段1：建立安全能力

- 客户端发起交换，与服务端交换了一个client_hello和server_hello报文，具有相同的参数：
  - 版本：客户理解的最高SSL版本
  - 随机数：客户端生成的随机数，防止重放攻击
  - 安全ID：会话标识符
  - 密文族：客户支持的加密算法的组合列表
  - 压缩方法：客户支持的压缩方法的列表

阶段2：服务器鉴别和密钥交换

- 如果服务器需要被鉴别，首先发送自己的CA证书certificate
- 发送服务器密码交换报文server_key_exchange
- 服务器向客户请求证书certificate_request
- 发送hello结束报文server_hello_done

阶段3：客户鉴别和密钥交换

- 发送自己的CA证书certificate，如果需要
- 发送客户密码交换报文client_key_exchange
- 发送证书验证报文certificate_verify

阶段4：结束

- 客户发送change_cipher_spec报文
- 发送在新算法、新密钥下的finished报文，验证密码交换和鉴别过程是成功的
- 服务器发送change_cipher_spec报文
- 服务器发送finished报文

传输层安全TLS

- 记录格式与SSL记录格式相同
- 在RFC2246中定义
- 非常类似于SSLv3

HTTP安全考虑

- HTTP协议允许远程用户对远程服务器请求通信及远程执行命令，这会危及Web服务器和客户
- HTTP的服务器日志。Web服务器会记录下各种用户的大量请求及数据信息。HTTP可能对这些信息的检索不加任何限制，从而泄露用户信息。

安全超文本传输协议1

- 安全超文本传输协议S-HTTP是保护Internet上所传输的敏感信息的安全协议。
- 随着Intemet和Web对身份验证的需求日益增长，用户在彼此收发加密文件之前需要身份验证。
- S-HTTP的目标是保证蓬勃发展的商业交易的传输安全。S-HTTP使Web客户和服务器均处于安全保护之下，其信息交换也是安全的。
- 利用S-HTTP，安全服务器以加密和签名信息回答请求，同样，安全客户验证签名并验证身份。验证是通过服务器的私钥实现的，该私钥用来产生服务器的数字签名，当信息发送给客户时，服务器将其公钥证书和签名信息一起发往客户，客户便可以验证发送者身份。服务器也可以用同样的过程来验证发自客户的数字签名。

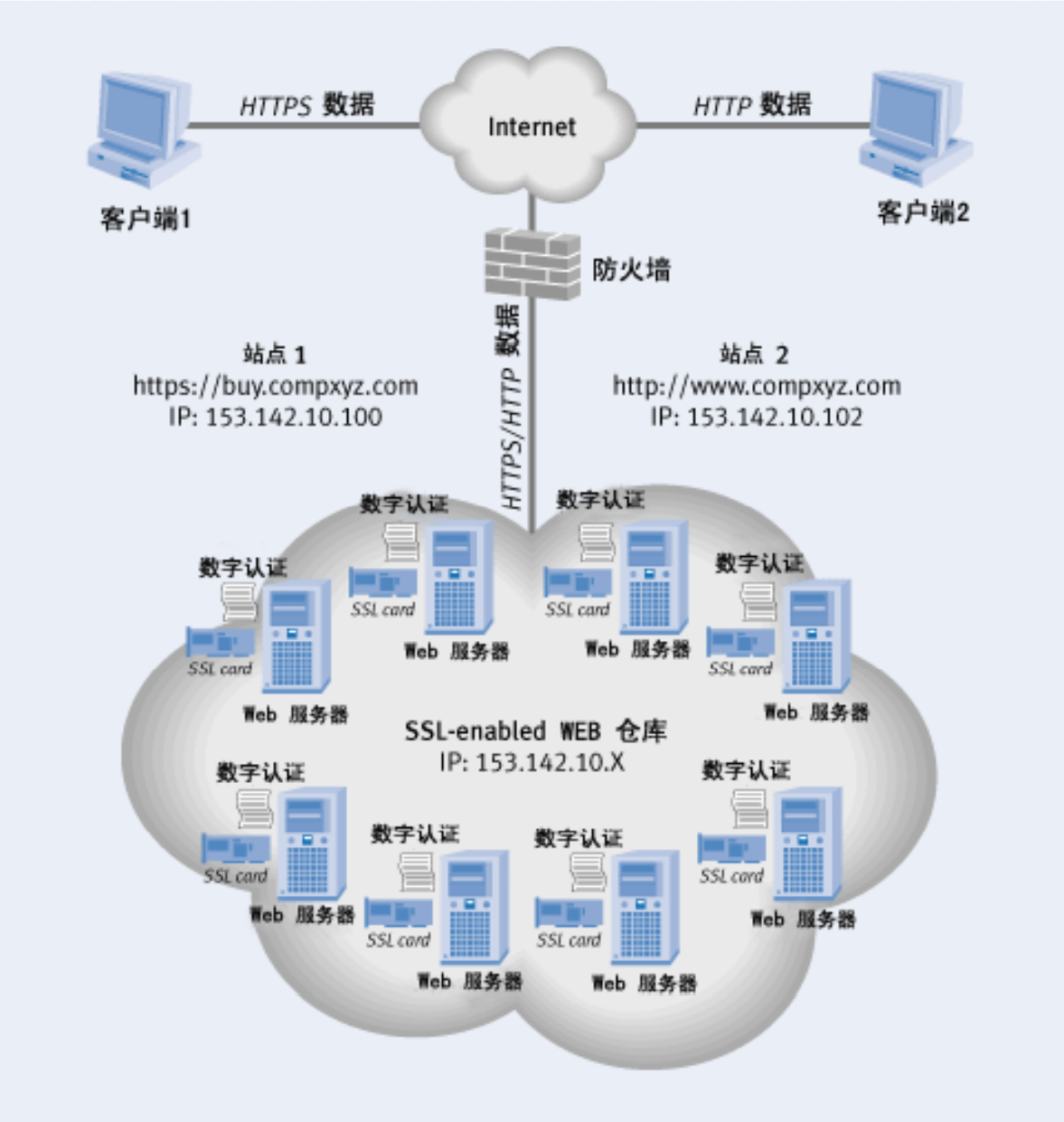
安全超文本传输协议2

- 数据加密可以用私钥、公钥或对称密钥加密。如果数据是用公钥加密，则无需客户公钥，信息即可被交换及解密。因为服务器将其私钥保存在密钥数据库文件中，且由Webmaster的口令加密。
- 加密和签名通过CGI脚本程序控制，这是本地安全配置文件，CGI脚本程序对S-HTTP信息头进行处理，以决定服务器是否签名、加密。

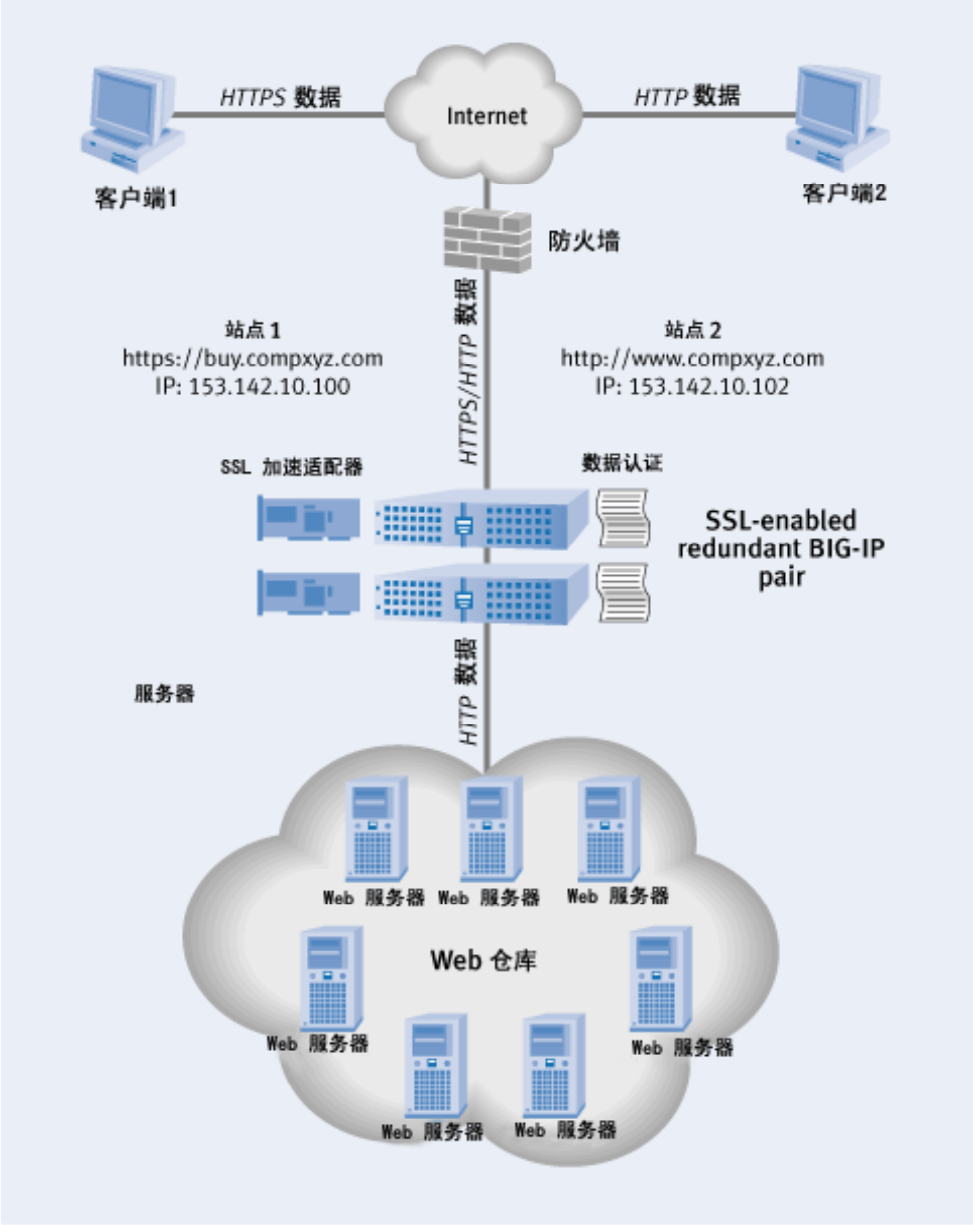
S-HTTP与SSL

- SSL不同于S-HTTP之处在于后者是HTTP的超集，只限于Web的使用；而前者则是通过Socket发送信息，SSL是一个通过Socket层对客户和服务器之间的事务处理进行安全处理的协议，适用于所有TCP／IP应用。

一个HTTPS应用实例



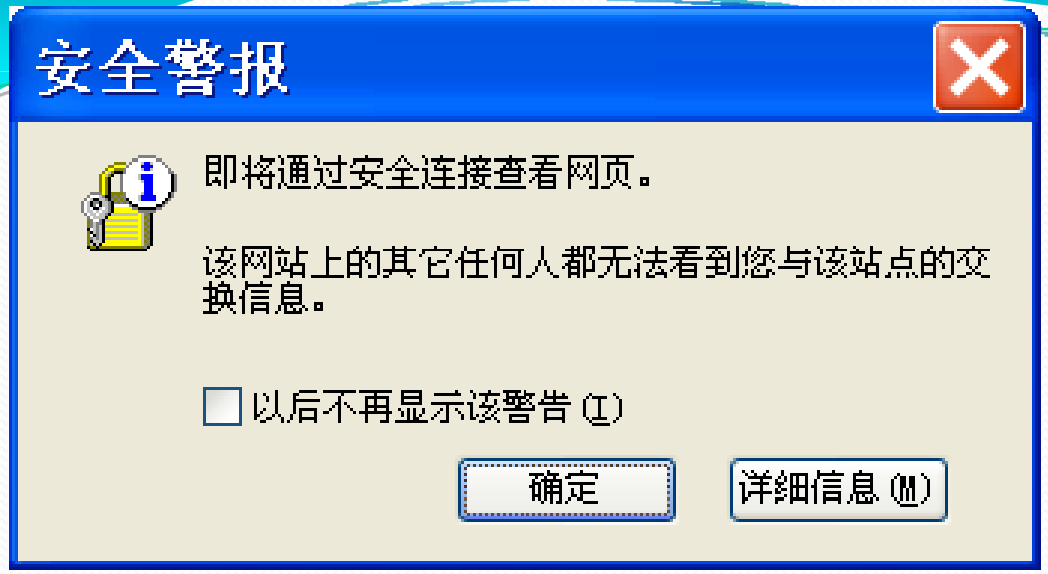
一个HTTPS应用实例

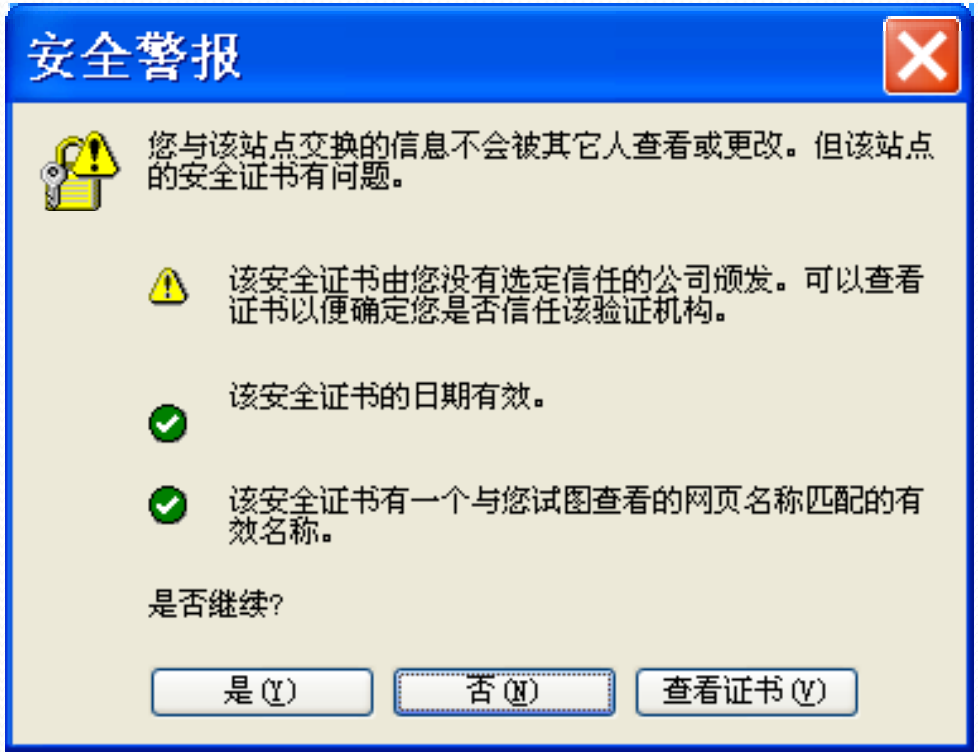


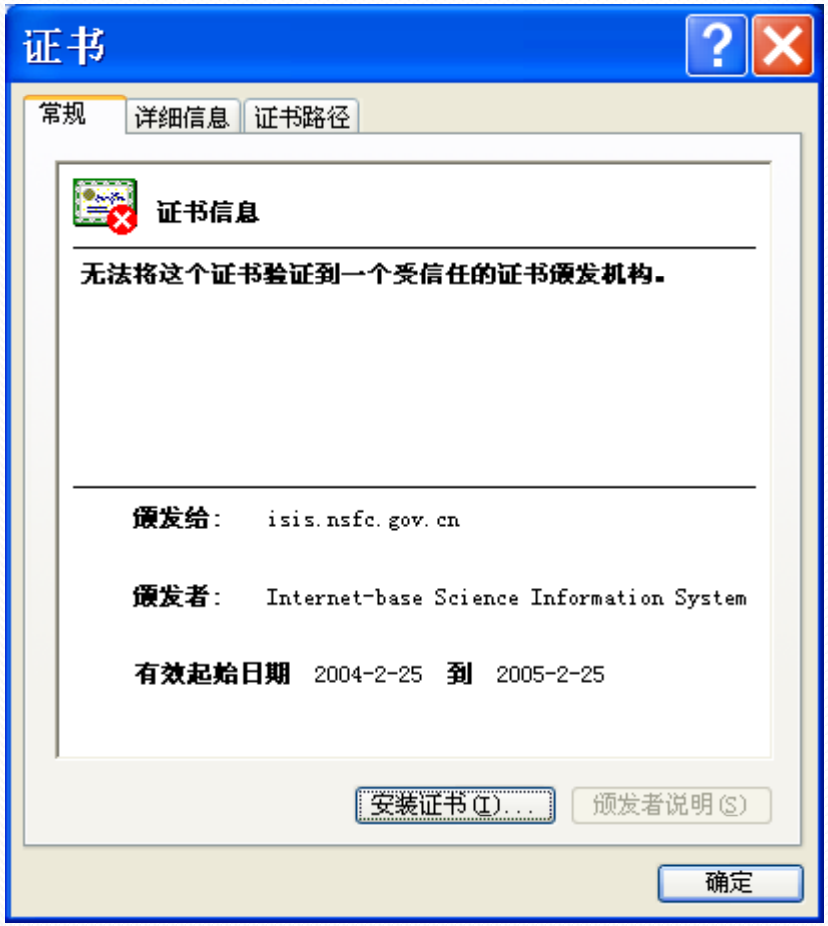
SSL/HTTPS的安全性

- 无法防止中间人攻击
- 受到美国密码算法出口法限制，Web浏览器的加密算法强度并不是特别高
  - 1999年，白宫批准出口密钥长度超过128位的加密软件（PGP）
  - 大部分只能用56位密钥长度的DES和512位密钥长度的RSA算法

HTTPS建立过程


安全警报
即将通过安全连接查看网页。
该网站上的其它任何人都无法看到您与该站点的交换信息。
以后不再显示该警告(I)
确定
详细信息(M)


安全警报
您与该站点交换的信息不会被其它人查看或更改。但该站点的安全证书有问题。
该安全证书由您没有选定信任的公司颁发。可以查看证书以便确定您是否信任该验证机构。
该安全证书的日期有效。
该安全证书有一个与您试图查看的网页名称匹配的有效名称。
是否继续?
是(Y)
否(N)
查看证书(V)


证书
常规
详细信息
证书路径
证书信息
无法将这个证书验证到一个受信任的证书颁发机构。
颁发给: isis.nsfc.gov.cn
颁发者: Internet-base Science Information System
有效起始日期 2004-2-25 到 2005-2-25
安装证书(I)...
颁发者说明(S)
确定

内容安排

1. 安全技术简介
2. 数据源层的安全
3. 应用服务层的安全
4. HTTP安全
5. 展示层安全

展示层安全

- XSS及其防御
- CSRF及其防御
- 钓鱼攻击及其防御
- 口令管理
- CAPTCHA

XSS及其防御

- 大多数现有的浏览器都能解释和执行来自 Web 服务器的嵌入到 Web 页面下载部分的脚本（用 JavaScript、JScript、VBScript 等脚本语言创建）。当攻击者向用户提交的动态表单引入恶意脚本时，就会产生跨站点脚本 (XSS) 攻击。
- XSS 攻击会导致不良后果。例如攻击者将能够获取会话信息，窃取 ID、密码、信用卡信息、家庭住址和电话、社会保险/税 ID 等用户的隐私资料。如果目标 Web 站点不检查这类恶意代码，就可能造成恶意使用用户信息。

XSS及其防御（续）

- 为了让脚本不被识别为恶意脚本，攻击者会用不同的编码方法对其进行编码，比如使用 HEX。通过这种方法，Web 站点就会像显示站点中的有效内容一样在页面上显示恶意内容。如果 Web 应用程序没有验证输入，攻击者只需诱导用户选择恶意链接，然后 Web 应用程序就会从用户那里收集机密信息。这就让攻击者就能够获取用户会话信息并窃取用户凭证，重定向到另一个 Web 站点上的网页，然后插入恶意代码来破坏 cookie、公开 SSL 连接、访问受限或私人站点，甚至触发一系列这样的攻击。

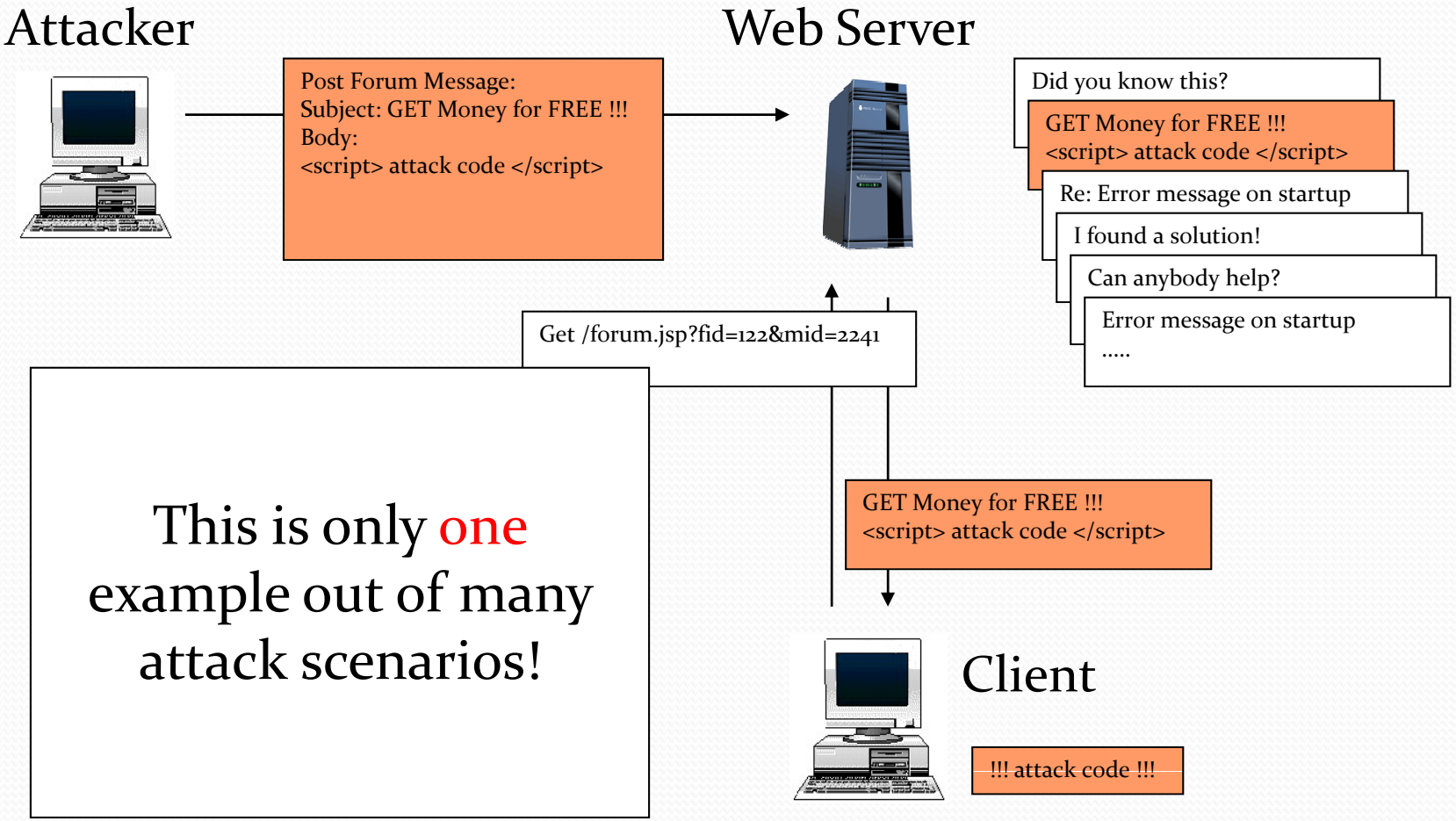
发现易受 XSS 攻击的漏洞

- 假设一个攻击者试图窃取虚拟的网银站点（www.simplebank.com）上的一个用户信息。要想登录这个站点，攻击者首先要输入一个测试用户 ID ("test") 和密码 ("test")。当错误信息页面出现，告知用户 ID 和密码组合 错误时，这个攻击者就找到了一个向 Web 页面中插入恶意代码的理想时机。
  - 首先将下面的信息输入 ID 文本框：<script>alert('Test')</script>。
  - 然后提交表单并看到这个 JavaScript 警告信息："TO BE DONE."，现在他知道这个站点容易受 XSS 类型的攻击。
  - 接下来攻击者会将类似于 清单 1 的脚本引入到 URL 中，这个 URL 将把所提交的用户信息重定向到 malicioussite.com。
  - 该代码通常会将任何登录到此 Web 站点的用户的 ID 及密码信息传递到攻击者的 Web 站点。
  - 窃取用户 ID 和密码的脚本完成后，攻击者就通过这个链接向网银站点的用户发送具有诱人优惠信息的电子邮件和帖子。
  - 受这些诱人的优惠信息的鼓动，用户有可能会单击链接并登录到网银站点。于是浏览器会执行攻击者引入的恶意代码，数据将被传递到黑客的网站。这样，黑客就轻而易举地利用了受害者的凭证登录到网银站点。

XSS攻击发生的根源

- Web 服务器没有采取充分的步骤，确保生成适当编码的页面。
- 允许用户输入并显示用户的输入，但没有正确验证输入。

XSS的简要过程


Attacker
Web Server
Post Forum Message:
Subject: GET Money for FREE !!!
Body:
<script> attack code </script>
Did you know this?
GET Money for FREE !!!
<script> attack code </script>
Re: Error message on startup
I found a solution!
Can anybody help?
Error message on startup
.....
Get /forum.jsp?fid=122&mid=2241
GET Money for FREE !!!
<script> attack code </script>
Client
!!! attack code !!!
This is only one example out of many attack scenarios!

谁将被XSS影响？

- XSS攻击的首要目标是用户
  - 用户信任公司
  - 但浏览器运行了恶意程序

- XSS攻击的第二个目标就是公司
  - 损失公共形象
  - 损失客户信任
  - 损失金钱

XSS攻击的影响

- 首要的影响是攻击了窃取了用户的Web认证凭据
  - Cookies, Username and Password
    - XSS不是一个无害的攻击!
  - 普通用户
    - 访问个人数据 (Credit card, Bank Account)
    - 访问业务数据 (Bid details, construction details)
    - 误用账号 (订购一些昂贵的物品)
  - 特权用户
    - 控制整个Web应用
    - 控制或者访问Web应用所在的主机
    - 控制或者访问后端系统或者数据库系统

XSS攻击的影响（续）

- 拒绝服务攻击
  - 令用户浏览器崩溃
  - 无限弹出新窗口
  - 重新定向
- 访问用户计算机
  - 使用ActiveX控制用户计算机
  - 将本地数据上传到攻击者计算机上
- 毁坏公司公共形象

XSS防御措施 by用户

- 下列措施可以减少XSS攻击的可能性
  - 不必要时禁用脚本功能。
  - 不要轻信电子邮件或消息栏内到其他站点的链接。它们可能包含具有破坏性的恶意代码。
  - 除非特别信任，否则不要单击站点上那些到涉及个人和商业信息的安全敏感页面的链接。
  - 通过地址直接访问涉及敏感信息的站点，而不是通过第三方站点。
  - 获得各种攻击及其发生的站点和公告栏的列表，访问时要格外小心。

XSS防御措施 by 开发者

- 保证 Web 站点的页面只有在验证了用户输入没有恶意代码之后才返回用户输入。
- 在验证的同时过滤输入中的 Meta 字符。（这是消除 XSS 攻击的一个重要检查点。虽然它不能消除全部的 XSS 问题，但它能警告 Web 维护人员在站点安全性方面存在的不足）。
- 不要完全相信使用了 HTTPS（Secure Sockets Layer）的 Web 站点就不会受到 XSS 攻击；HTTPS 只能确保安全连接，处理用户输入的数据是应用程序内部的事情。如果应用程序有 XSS 漏洞，攻击者就可能会发送能被应用程序执行的恶意脚本，导致 XSS 侵入。
- 在搜索引擎和论坛内显示用户输入之前，将所有非字母和数字字符转变成 HTML 字符。
- 在应用程序正式付诸使用之前，在设计阶段大量使用测试工具，消除这类 XSS 漏洞。
- 开发一些具有私有和公共密钥的标准或标志脚本，这些脚本要确实进行检查，以保证所引入的脚本真正经过了身份验证。（要大规模地实现，Internet 规则就必须进行标准化，以得出一套由 W3C 等主要参与者参与的公认方法）。

以下不是XSS的防御方法

- TLS/SSL
  - 攻击并不是通过通信中存在的漏洞发起
  - 攻击发起的原因在于应用层的漏洞
- 客户端的输入验证
  - 非常容易转换
  - 可以直接通过URL进行访问

```
<form method="GET" action="/file.jsp">
<input type="text" name="fname" maxlength="10">
```

GET /file.jsp?fname=1234567890**12345**

展示层安全

- XSS及其防御
- **CSRF及其防御**
- 钓鱼攻击及其防御
- 口令管理
- CAPTCHA

CSRF及其防御

- CSRF（Cross Site Request Forgery, 跨站域请求伪造），也称为XSRF，是一种网络的攻击方式，它在 2007 年曾被列为互联网 20 大安全隐患之一。
- CSRF 攻击可以在受害者毫不知情的情况下以受害者名义伪造请求发送给受攻击站点，从而在并未授权的情况下执行在权限保护之下的操作。
- CSRF发音*sea-surf*【*WIKI Pedia*】
- 对于大多数人来说，CSRF 却依然是一个陌生的概念。即便是大名鼎鼎的 Gmail, 在 2007 年底也存在着 CSRF 漏洞，从而被黑客攻击而使 Gmail 的用户造成巨大的损失。
  - http://directwebremoting.org/blog/joe/2007/01/01/csrf_attacks_or_how_to_avoid_exposing_your_gmail_contacts.html

CSRF 攻击的前提

- 前提
  - 用户在登录合法网站的情况下，不退出，但通过某种方式进入攻击网站
  - 攻击网站诱骗用户点击伪造的攻击链接
  - 合法网站不检查HTTP header中的referer内容或者检查的方法存在问题（后者极少发生）
  - 用户点击完后，在未知情的情况下，合法网站上的合法信息被恶意操作。
- 关键要点：Session信息在恶意网站被利用。

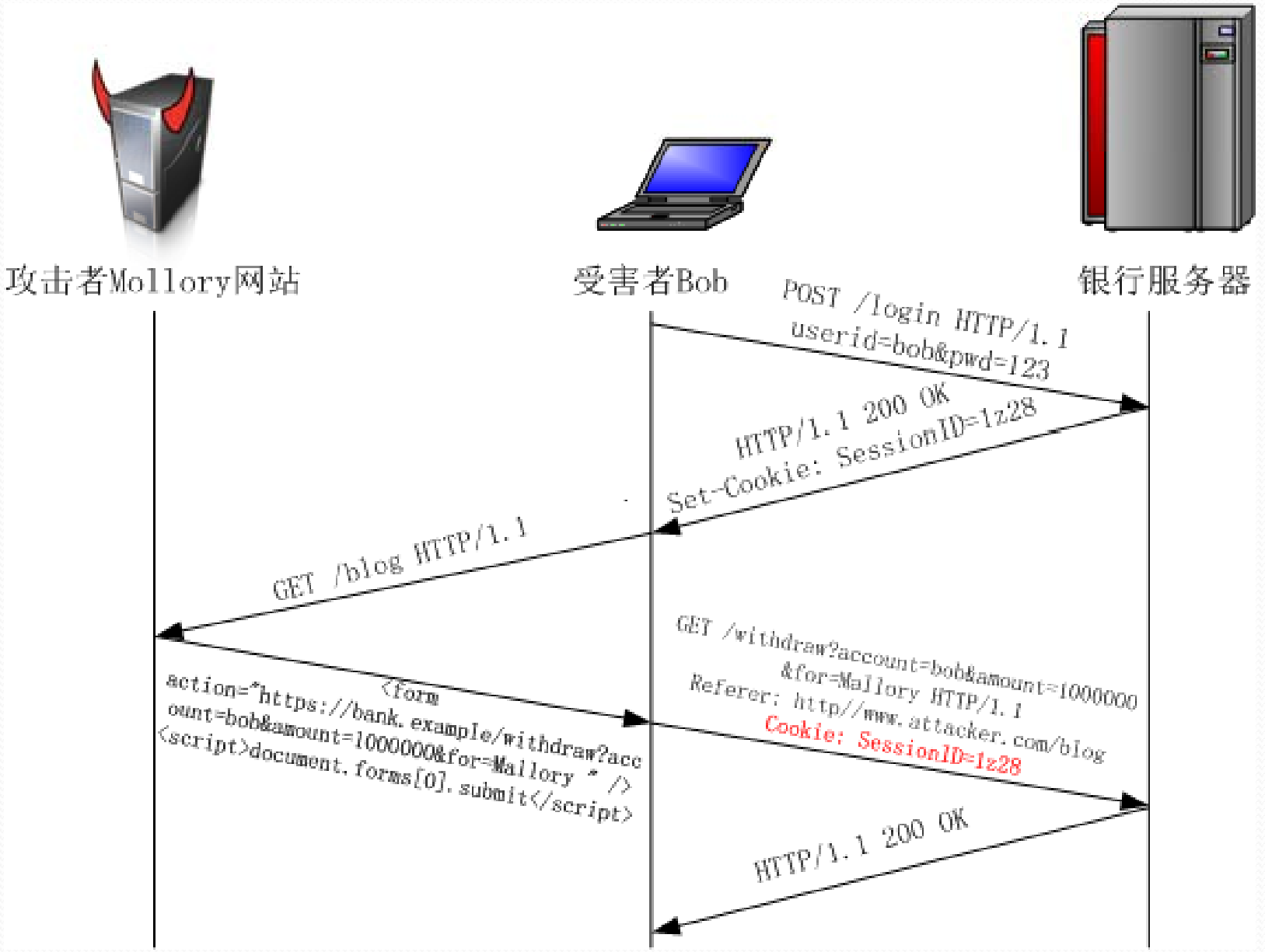
CSRF的攻击案例

- 受害者 Bob 在银行有一笔存款，通过对银行的网站发送请求 https://bank.example/withdraw?account=bob&amount=1000000&for=bob2 可以使 Bob 把 1000000 的存款转到 bob2 的账号下。通常情况下，该请求发送到网站后，服务器会先验证该请求是否来自一个合法的 session，并且该 session 的用户 Bob 已经成功登陆。
- 黑客 Mallory 自己在该银行也有账户，他知道上文中的 URL 可以把钱进行转帐操作。Mallory 可以自己发送一个请求给银行：https://bank.example/withdraw?account=bob&amount=1000000&for=Mallory。但是这个请求来自 Mallory 而非 Bob，他不能通过安全认证，因此该请求不会起作用。
- 这时，Mallory 想到使用 CSRF 的攻击方式，他先自己做一个网站，在网站中放入如下代码：src=”https://bank.example/withdraw?account=bob&amount=1000000&for=Mallory”，并且通过广告等诱使 Bob 来访问他的网站。当 Bob 访问该网站时，上述 url 就会从 Bob 的浏览器发向银行，而这个请求会附带 Bob 浏览器中的 cookie 一起发向银行服务器。
- 大多数情况下，该请求会失败，因为他要求 Bob 的认证信息。但是，如果 Bob 当时恰巧刚访问他的银行后不久，他的浏览器与银行网站之间的 session 尚未过期，浏览器的 cookie 之中含有 Bob 的认证信息。这时，悲剧发生了，这个 url 请求就会得到响应，钱将从 Bob 的账号转移到 Mallory 的账号，而 Bob 当时毫不知情。
- 等以后 Bob 发现账户钱少了，即使他去银行查询日志，他也只能发现确实有一个来自于他本人的合法请求转移了资金，没有任何被攻击的痕迹。而 Mallory 则可以拿到钱后逍遥法外。

CSRF的攻击对象

- 在讨论如何抵御 CSRF 之前，先要明确 CSRF 攻击的对象，也就是要保护的对象。
  - 从以上的例子可知，CSRF 攻击是黑客借助受害者的 cookie 骗取服务器的信任，但是黑客并不能拿到 cookie，也看不到 cookie 的内容。另外，对于服务器返回的结果，由于浏览器同源策略的限制，黑客也无法进行解析。因此，黑客无法从返回的结果中得到任何东西，他所能做的就是给服务器发送请求，以执行请求中所描述的命令，在服务器端直接改变数据的值，而非窃取服务器中的数据。
  - 我们要保护的对象是那些可以直接产生数据改变的服务，而对于读取数据的服务，则不需要进行 CSRF 的保护。比如银行系统中转账的请求会直接改变账户的金额，会遭到 CSRF 攻击，需要保护。而查询余额是对金额的读取操作，不会改变数据，CSRF 攻击无法解析服务器返回的结果，无需保护。

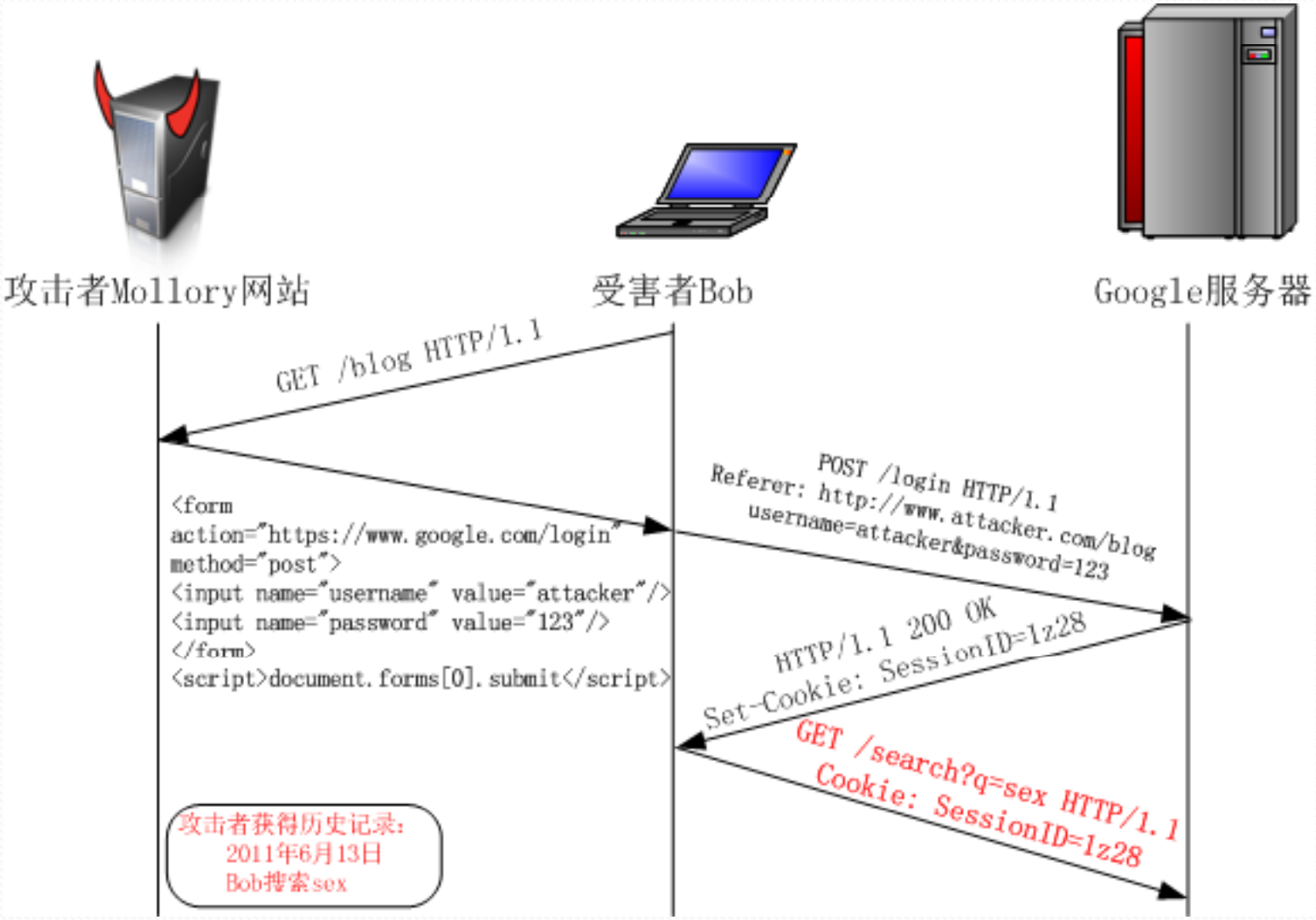
CSRF图例


攻击者Mollory网站
受害者Bob
银行服务器
POST /login HTTP/1.1
userid=bob&pwd=123
HTTP/1.1 200 OK
Set-Cookie: SessionID=1z28
GET /blog HTTP/1.1
<form
action="https://bank.example/withdraw?acc
ount=bob&amount=1000000&for=Mallory " />
<script>document.forms[0].submit</script>
GET /withdraw?account=bob&amount=1000000
&for=Mallory HTTP/1.1
Referer: http//www.attacker.com/blog
Cookie: SessionID=1z28
HTTP/1.1 200 OK

更复杂的CSRF： login CSRF

- 攻击者将自己的账号贡献给受害者
- 受害者*被登录*后，会在session中留下行为记录
- 攻击者登到自己的账号中，获取受害者的行为记录

Login CSRF图例



CSRF攻击的防御

- 验证 HTTP Referer 字段
- 在请求地址中添加 token 并验证
- 在 HTTP 头中自定义属性并验证

验证 HTTP Referer 字段

- 根据 HTTP 协议，在 HTTP 头中有一个字段叫 Referer，它记录了该 HTTP 请求的来源地址。在通常情况下，访问一个安全受限页面的请求来自于同一个网站，比如需要访问 http://bank.example/withdraw?account=bob&amount=1000000&for=Mallory，用户必须先登陆 bank.example，然后通过点击页面上的按钮来触发转账事件。这时，该转帐请求的 Referer 值就会是转账按钮所在的页面的 URL，通常是以 bank.example 域名开头的地址。而如果黑客要对银行网站实施 CSRF 攻击，他只能在他自己的网站构造请求，当用户通过黑客的网站发送请求到银行时，该请求的 Referer 是指向黑客自己的网站。因此，要防御 CSRF 攻击，银行网站只需要对于每一个转账请求验证其 Referer 值，如果是以 bank.example 开头的域名，则说明该请求是来自银行网站自己的请求，是合法的。如果 Referer 是其他网站的话，则有可能是黑客的 CSRF 攻击，拒绝该请求。

验证 HTTP Referer 字段方法的问题

- Referer 的值是由浏览器提供的，虽然 HTTP 协议上有明确的要求，但是每个浏览器对于 Referer 的具体实现可能有差别，并不能保证浏览器自身没有安全漏洞。使用验证 Referer 值的方法，就是把安全性都依赖于第三方（即浏览器）来保障，从理论上来讲，这样并不安全。事实上，对于某些浏览器，比如 IE6 或 FF2，目前已经有一些方法可以篡改 Referer 值。
- 即便是使用最新的浏览器，黑客无法篡改 Referer 值，这种方法仍然有问题。因为 Referer 值会记录下用户的访问来源，有些用户认为这样会侵犯到他们自己的隐私权，特别是有些组织担心 Referer 值会把组织内网中的某些信息泄露到外网中。因此，用户自己可以设置浏览器使其在发送请求时不再提供 Referer。当他们正常访问银行网站时，网站会因为请求没有 Referer 值而认为是 CSRF 攻击，拒绝合法用户的访问。

在请求地址中添加 token 并验证

- CSRF 攻击之所以能够成功，是因为黑客可以完全伪造用户的请求，该请求中所有的用户验证信息都是存在于 cookie 中，因此黑客可以在不知道这些验证信息的情况下直接利用用户自己的 cookie 来通过安全验证。要抵御 CSRF，关键在于在请求中放入黑客所不能伪造的信息，并且该信息不存在于 cookie 之中。可以在 HTTP 请求中以参数的形式加入一个随机产生的 token，并在服务器端建立一个拦截器来验证这个 token，如果请求中没有 token 或者 token 内容不正确，则认为可能是 CSRF 攻击而拒绝该请求。

在请求地址中添加 token 并验证方法的问题

- 需要给每个请求添加token，这很繁琐
- 该方法还有一个缺点是难以保证 token 本身的安全。特别是在一些论坛之类支持用户自己发表内容的网站，黑客可以在上面发布自己个人网站的地址。由于系统也会在这个地址后面加上 token，黑客可以在自己的网站上得到这个 token，并马上就可以发动 CSRF 攻击。

在 HTTP 头中自定义属性并验证

- 这种方法也是使用 token 并进行验证，和上一种方法不同的是，这里并不是把 token 以参数的形式置于 HTTP 请求之中，而是把它放到 HTTP 头中自定义的属性里。通过 XMLHttpRequest 这个类，可以一次性给所有该类请求加上 csrftoken 这个 HTTP 头属性，并把 token 值放入其中。这样解决了上种方法在请求中加入 token 的不便，同时，通过 XMLHttpRequest 请求的地址不会被记录到浏览器的地址栏，也不用担心 token 会透过 Referer 泄露到其他网站中去。

在 HTTP 头中自定义属性并验证的问题

- XMLHttpRequest 请求通常用于 Ajax 方法中对于页面局部的异步刷新，并非所有的请求都适合用这个类来发起，而且通过该类请求得到的页面不能被浏览器所记录下，从而进行前进，后退，刷新，收藏等操作，给用户带来不便。
- 对于没有进行 CSRF 防护的遗留系统来说，要采用这种方法来进行防护，要把所有请求都改为 XMLHttpRequest 请求，这样几乎是要重写整个网站，这代价无疑是不能接受的。

展示层安全

- XSS及其防御
- CSRF及其防御
- 钓鱼攻击及其防御
- 口令管理
- CAPTCHA

钓鱼攻击及其防御

- 在信息安全领域里，钓鱼式攻击（Phishing，与钓鱼的英语fishing发音一样，又名“网钓法”或“网络网钓”，以下简称网钓）是一种企图从电子通信中，通过伪装成信誉卓著的法人媒体以获得如用户名、密码和信用卡明细等个人敏感信息的犯罪诈骗过程。这些通信都声称（自己）来自于风行的社交网站（YouTube、Facebook、MySpace）、拍卖网站（eBay）、网络银行、电子支付网站（PayPal）、或网络管理者（雅虎、互联网服务供应商、公司机关），以此来诱骗受害人的轻信。
- 网钓通常是通过e-mail或者实时通信进行。它常常导引用户到URL与接口外观与真正网站几无二致的假冒网站输入个人数据。就算使用强式加密的SSL服务器认证，要侦测网站是否仿冒实际上仍很困难。
- 网钓是一种利用社会工程技术来愚弄用户的实例。它凭恃的是现行网络安全技术的低亲和度。种种对抗日渐增多网钓案例的尝试涵盖立法层面、用户培训层面、宣传层面、与技术保全措施层面。
- 网钓技术最早于1987年问世，而首度使用“网钓”这个术语是在1996年。该辞是英文单词钓鱼（fishing）的变种之一，意味着放线钓鱼以“钓”取受害人财务数据和密码。

网络钓鱼的历史

- **Phreaking** + **Fishing** = **Phishing**
  - **Phreaking** = 1970年代免费打电话的骗子
  - **Fishing** = 使用诱饵诱惑受害者

- 1995年的钓鱼网络钓鱼攻击

  目标: AOL 用户

  目的：得到合法用户的口令以便免费上网

  威胁级别：低

  技术：使用类似名字(www.aol.com 替代www.aol.com), 社会工程

- 2001年的网络钓鱼攻击

  目标：电子购买者和主要银行

  目的：得到信用卡号和账号

  威胁级别：中

  技术：与1995年类似, keylogger

- 2007年的网络钓鱼攻击

  目标：Paypal, banks, ebay用户

  目的：银行账户

  威胁级别：高

  技术：浏览器漏洞，链接模糊

Paypal的网络钓鱼攻击

- 在PayPal网钓示例里，电子邮件里的拼写错误以及非PayPal网域链接的存在（显示在状态栏红色框里）都是线索，指出这是一个网钓的企图。另一种网钓法是无个人问候的赠品，尽管显示的个人数据并不保证其正当性。一个合法的PayPal通信总是以用户的真实姓名问候，而非一个普通的问候如：“敬启者”、“亲爱的用户”（Dear Accountholder）。其他的信息欺诈的迹象像是简单不过的字拼写错误、文法拙劣、以及威胁收信人若不遵照信息指示办理的话会遭帐号停用的处分。

- 请注意，许多网络网钓电子邮件会如同来自PayPal的一封真正的电子邮件般包括一则永不将您的密码泄漏以防网钓攻击的重大警告。这些警告用户网钓攻击的可能性，并提供链接到帮助如何避免或识别此种攻击的网站的种种，是使得该网钓电子邮件如此虚伪以便欺骗。在这个例子里，网钓电子邮件警告用户，PayPal绝对不会要求您提供敏感信息。该信件言而有信不问您敏感信息，反而邀请用户点击一个链接，以“确认”其帐户;这一步将导引这些受害人进一步造访网钓网站，其设计看起来与PayPal网站很像。而在那里会问这些受害人的个人机密信息。

Dear PayPal valued member

PayPal

Dear valued PayPal® member

It has come to our attention that your **PayPal®** account information needs to be updated as part of our continuing commitment to protect your account and to reduce the instance of fraud on our website. If you could please take 5-10 minutes out of your online experience and update your personal records you will not run into any future problems with the online service.

However, failure to update your records will result in account suspension.
Please update your records on or before **June 5, 2006**.

Once you have updated your account records, your **PayPal®** session will not be interrupted and will continue as normal.

To update your **PayPal®** records click on the following link:
https://www.paypal.com/cgi-bin/webscr?cmd=_login-run

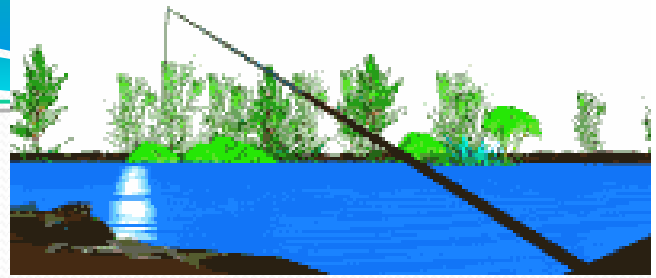
Thank You.
PayPal® UPDATE TEAM

Accounts Management As outlined in our User Agreement, **PayPal®** will periodically send you information about site changes and enhancements.

Visit our Privacy Policy and User Agreement if you have any questions.
http://www.paypal.com/cgi-bin/webscr?cmd=p/gen/ua/policy_privacy-outside

网络钓鱼造成的损失

- 网钓所造成的损害范围从拒绝访问电子邮件到巨大财务损失都有。这种形式的身份盗窃正在普及，因为给信任的人方便往往泄露个人信息给网钓者，这些信息包括信用卡号码、社会安全号码（美国）、身分证号码（台湾）、和母亲婚前姓名。也有人担心身份窃贼仅仅通过访问公开纪录就可以添加此类信息到它们取得的知识库中。一旦这信息被取得了，网钓者可能会利用个人数据明细以受害者姓名创造假帐号。然后他们可以毁掉受害者的信用，或者甚至让受害人无法访问自己的帐户。
- 据估计，从2004年5月和2005年5月，大约120万电脑用户在美国遭受网钓所造成的损失，总计约92900万美元。随着为美国企业的客户成为受害者，该国企业估计每年损失20亿美元。2007年网钓攻击升级。截至2007年8月前在美国360万成年人于12个月内失去32亿美元。在英国，网络银行诈骗的损失—大多来自网钓—几乎增加了一倍从2004年1220万英镑到2005年2320英镑，而在2005年，每20个电脑用户中就有一个声称因网钓造成的财务损失。
- 英国银行机构APACS采取的立场是：“客户还必须采取合理的预防措施...，如此对罪犯而言他们才不会好欺负。”同样，2006年9月当第一次大量的网钓攻击登陆爱尔兰银行业界时，爱尔兰银行起初拒绝补偿客户所遭受的损失（而且它仍然坚持认为，它的政策不应如此），虽然会补偿损失的金额上限定调为1万1300英镑还算不错。



网络钓鱼的经济账

- 发送2,000,000个假冒电子邮件，诱惑用户点击
- 5%的邮件绕过了过滤软件到达终端用户– 100,000 (APWG)
- 5%的用户点击了钓鱼链接– 5,000 (APWG)
- 2% 的用户在钓鱼页面输入了账户信息–100 (Gartner)
- $1,200 的钱从这些账户中转出 (FTC)
- 潜在的回报: **$120,000**

•所需要的付出：一封精心准备的钓鱼邮件+2,000,000个电子邮件地址+一个粗劣的钓鱼网站+等待时间

在2005 David Levi 通过钓鱼攻击从160个ebay用户中获取了$360,000。

鱼叉网钓 (spear phishing)

- 社会感知攻击
  - ✓ 通过公开数据获取社会关系
  - ✓ 钓鱼邮件将送给一些已知的用户
  - ✓ 使用一些被信赖的组织来获取用户的信任t
  - ✓ 诱骗用户更新他们的账户
  - ✓ 对那些不予以回音的用户威胁中止账户
  - ✓ 所用礼物或者奖金作为诱饵
  - ✓ 基于安全保证。

- 上下文相关的攻击
  - ✓ “你拍卖到了淘宝宝贝!”
  - ✓ “你要买的书现在正在打折!”

反网络钓鱼

- 用户意识：打击网钓的策略之一，是试着培养人们识别网钓，并教导怎样处理这些问题。
- 技术手段：反网钓措施已经实现将其功能内嵌于浏览器，作为浏览器的扩展或工具栏，以及网站的登录程序的一部分。
- 法律对策：
  - 在2004年1月26日，美国联邦贸易委员会提交了涉嫌网钓者的第一次起诉。被告是个美国加州少年，据说他设计建造了一个网页看起来像美国在线网站，并用它来窃取信用卡数据。
  - 在美国，参议员派崔克·莱希（Patrick Leahy）在2005年3月1日向美国国会提审2005反网钓法案。这项法案，如果它已成为法律，将向创建虚假网站、发送虚假电子邮件以诈欺消费者的罪犯求处罚款高达25万美元并且可监禁长达5年。英国在2006年以《2006年欺诈罪法令》（Fraud Act 2006）强化了其打击仿冒欺诈的法律武器，该法令采用一般欺诈罪，可求刑监禁多达10年，并禁止开发或意图欺诈下拥有网钓软件包。

网址嫁接 (Pharming) 和网络钓鱼 (Phishing)

- 类似网络钓鱼。但是网址嫁接不会直接骗取个人或企业资讯，而是挟持合法的网站，如“www.mybank.com”），然后透过 DNS 将网站重新导向到看似与原始网站无异的错误 IP 位址。这些假冒的网站会透过图形使用者界面来收集受到保护的信息，使用者根本不会发现有异样。因为要进行网址嫁接必须具备非常高超的技术，再加上 DNS 非常难操控，所以网址嫁接远比网络钓鱼少见。不过，网址嫁接在不久的将来还是有可能成为不小的威胁。

DNS的安全问题

- 2010年1月12日上午7时左右，百度出现长时间无法访问故障。此次事件的核心是由于百度所使用的域名注册商程序被入侵，导致baidu.com域名的DNS服务器信息被恶意修改，然后对百度的域名进行了劫持，在国内外造成巨大影响。
- DNSSEC – DNS security extension
  - DNSSEC被设计用来保护Internet免于某些攻击，如DNS缓冲中毒。它是DNS的一组扩展协议。

展示层安全

- XSS及其防御
- CSRF及其防御
- 钓鱼攻击及其防御
- 口令管理
- CAPTCHA

口令（password）

- 原来的意义是口头暗号的意思，换英文Password，通常被叫成密码，也叫“通行字”，是一个用于身份验证的保密的字符串，它被用来保护不想被别人看到的隐私以及防止未经授权的操作。

- 用户名（帐号）和口令经常被用来登录受到保护的操作系统、手机、自动取款机等。通常，计算机用户需要口令来登录系统、收发电子邮件、以及控制程序、数据库、网络和网站，甚至在线浏览新闻等。

口令管理存在的问题

- 记忆问题
  - 口令太长不容易记忆
  - 口令太复杂不容易记忆
  - 口令太多记不过来
  - 不知道哪个口令与哪个账号对应
- 输入问题
  - 口令容易输错
  - 口令在错误的界面输入
- 存储问题
  - 明文存储口令
    - 最近的Sony爆出的安全问题

如何生成一个合格的口令

- 选择自己熟悉的短语
  - 比如：朗文当代英语字典
- 抽取中间的拉丁字母
  - 比如：lwddyyzd
- 添加数字和特殊字符
  - 比如：1wd#y#zd

- 千万不要前往所谓专业网站测试自己的密码强度！
  - 比如：http://www.microsoft.com/china/athome/security/privacy/password_checker.mspx
  - Absolutely NOT！
  - 原因：字典攻击

如何为输入设计良好的口令

- 储值卡密码的设计
  - **** **** ****
  - **** **** ****
- 银行卡号的设计
  - **** **** **** ****

- 合理的口令可以减少输错的可能，这可以降低系统的负载

如何存储口令

- 口令+用户名+网站个性串 通过Hash算法生成密码串存储于硬盘中

猜口令的方法

- 社会工程
- 字典攻击
- 启发式攻击
- 暴力破解

展示层安全

- XSS及其防御
- CSRF及其防御
- 钓鱼攻击及其防御
- 口令管理
- **CAPTCHA**

CAPTCHA (验证码)

- 全自动区分计算机和人类的测试（英语：Completely Automated Public Turing Test to tell Computers and Humans Apart，可简称为CAPTCHA）是一种区分用户是计算机和人的公共全自动程序。在CAPTCHA测试中，作为服务器的计算机会自动生成一个问题由用户来解答。这个问题可以由计算机生成并评判，但是必须只有人类才能解答。由于计算机无法解答CAPTCHA的问题，所以回答出问题的用户就可以被认为是人类。

CAPTCHA的历史

- CAPTCHA 这个词最早是在2002年由卡内基梅隆大学（Carnegie Mellon University）的Luis von Ahn、Manuel Blum、Nicholas J. Hopper以及IBM的John Langford所提出。卡内基梅隆大学曾试图申请此词使其成为注册商标，但该申请于2008年4月21日被拒绝。一种常用的CAPTCHA测试是让用户输入一个扭曲变形的图片上所显示的文字或数字，扭曲变形是为了避免被光学字符识别（OCR, Optical Character Recognition）之类的电脑程式自动辨识出图片上的文数字而失去效果。由于这个测试是由计算机来考人类，而不是标准图灵测试中那样由人类来考计算机，人们有时称CAPTCHA是一种反向图灵测试。
- 为了无法看到图像的身心障碍者，替代的方法是改用语音读出文数字。

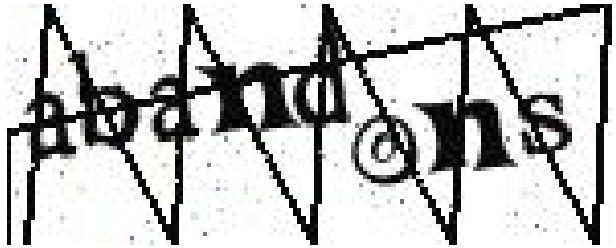
CAPTCHA的例子

- Google

- Baidu

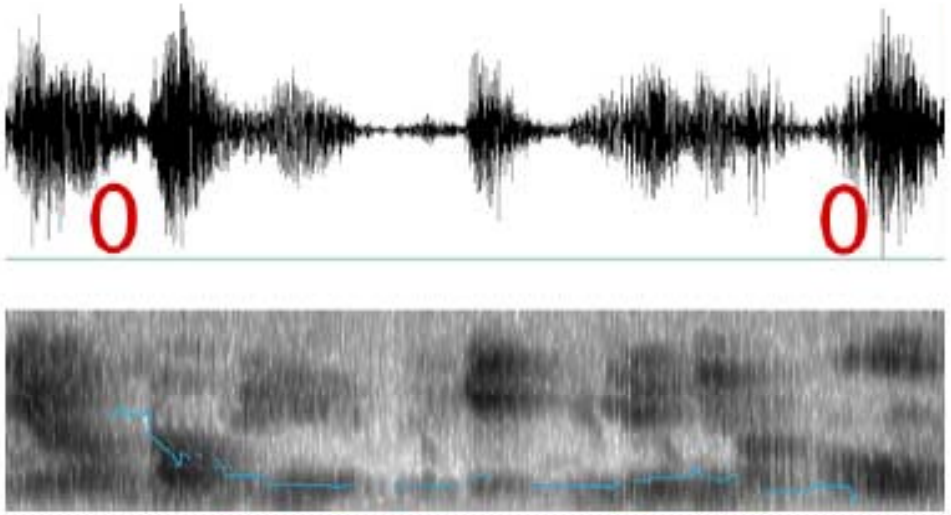
- Captchas.net

- Slashdot

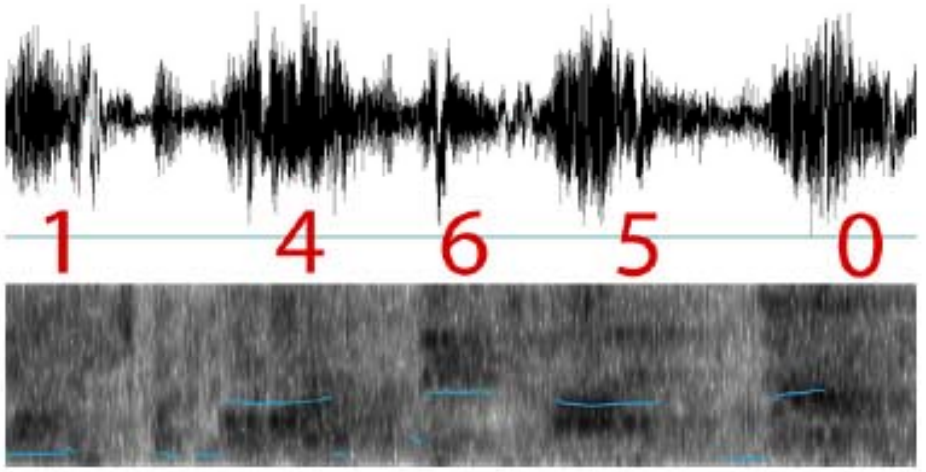


音频Captchas (Cont.)

- Google



- Microsoft



CAPTCHA的破解

- CAPTCHA也有一些方法可以破解。
  - 俄罗斯的一个黑客组织使用一个自动识别软件在2006年破解了Yahoo的CAPTCHA。准确率大概是15%，但是攻击者可以每天尝试10万次，相对来说成本很低。
  - 而在2008年，Google的CAPTCHA也被俄罗斯黑客所破解。攻击者使用两台不同的电脑来调整破解进程，可能是用第二台电脑学习第一台对CAPTCHA的破解，或者是对成效进行监视。
  - 甚至有人工破解验证码的例子，Gmail邮箱注册验证系统的破解可能即是经由此方法。reCAPTCHA（en:Human-based computation）的一种，由CAPTCHA的发明者之一Luis von Ahn所发起。）亦是人工破解验证码的应用。
  - *captcha-bypass.com*

重要的网站

- OWASP (http://www.owasp.org)
- http://www.sqlsecurity.com
- http://www.securityfocus.com/infocus/1768