

软件安全与漏洞分析

2.4 SQL注入和数组越界访问漏洞

Previously in Software Security

- 整数计算的原理
- 整数溢出漏洞及其潜在危害
- 格式化字符串漏洞的成因及利用

SQL注入和数组越界访问漏洞

- 本节主题 -- 1. SQL注入漏洞
 - 实现原理及后果
 - 现有的相关应对措施
- 本节主题 -- 2. 数组越界访问漏洞
 - 数组越界的产生
 - 数组越界防范的困难性

数据库与SQL简介

- 数据库：大型软件系统的关键基础设施
 - 长期储存、有组织的、可共享的数据集合
 - 具有尽可能小的冗余度、较高的数据独立性和易扩展性
 - 应用场景 --- **Web**后台数据、企业级联机事务处理



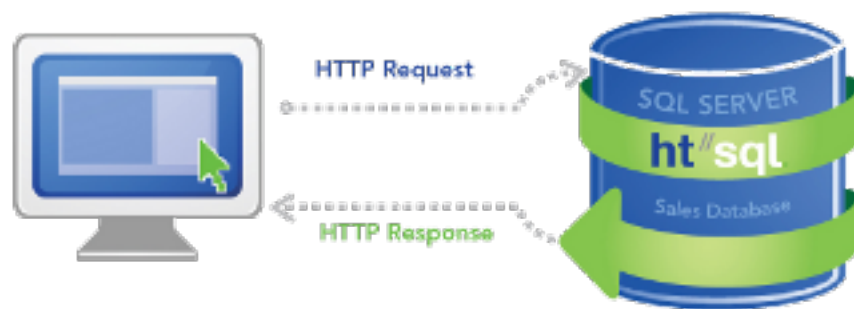
SQL、Mysql等

数据库与SQL简介

□ 结构化查询语言(Structured Query Language, SQL)

- 特殊目的编程语言
- 数据库查询 + 程序设计语言
- 屏蔽底层细节的高级数据库操作语言

用户可以远程发起SQL操作



数据库与SQL简介

□ SQL语言的核心结构

- **Sth. to do**

SELECT *

UPDATE

- **@ target**

FROM table

table

- **Sth. to do (optional)**

SET col = new

- **under condition (optional)**

WHERE con = true

WHERE col = old

对SQL语言的解析，与格式化字符串的解析非常相似

SQL注入的原理

□ SQL语言中的一些特殊符号

- 分号，意味着指令结束，有可能开始下一个指令
- 单引号，用于字符串常量，系统在输入命令中顺序匹配“
- #或--，意味着注释，出现在其后的内容会被系统忽略掉

上述符号的存在，使得经过设计的SQL语句可以篡改设计者事先定义的查询语义

SQL注入的原理

User ID:

Password:

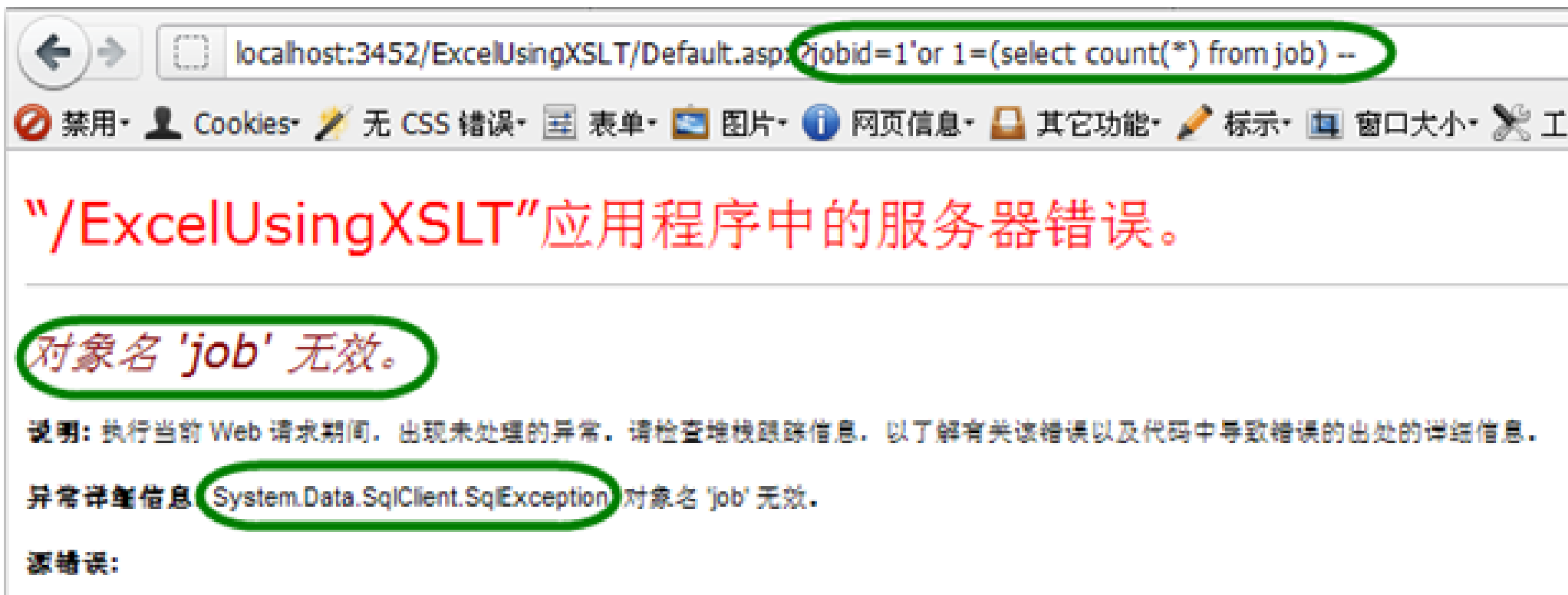
Corporate:

SELECT * From SomeTable WHERE Name='XX' and Password='YY' and Corp='ZZ'

SELECT * From SomeTable WHERE Name=' ' and Password='' and Corp=''

结果：预设SQL操作命令中的条件判断部分被无效化，**权限控制被绕过**

SQL注入的原理



结果：由于服务器端输出了过多的错误信息给客户端，攻击者可以借此探查远端数据库结构

SQL注入的原理

□ 其他更为高级的SQL注入攻击

◦ 联合查询攻击

SELECT accounts FROM users WHERE login="" **UNION**

SELECT cardNo FROM CreditCards WHERE acctNo=10032 -- AND pass="" AND pin=

◦ 背负查询攻击

SELECT accounts FROM users WHERE login='doe' AND pass='';

DROP table users -- ' AND pin=123

- 更多SQL注入攻击类型参考文献：Halfond W G, Viegas J, Orso A. A classification of SQL-injection attacks and countermeasures[C]//Proceedings of the IEEE International Symposium on Secure Software Engineering. IEEE, 2006, 1: 13-15.

SQL注入的原理

□ SQL注入攻击的常见途径

- 通过用户输入注入
- 通过cookies注入
- 通过服务器参数注入
- 二阶注入攻击

例：在某网站注册一个ID “admin'--” ， 然后修改密码

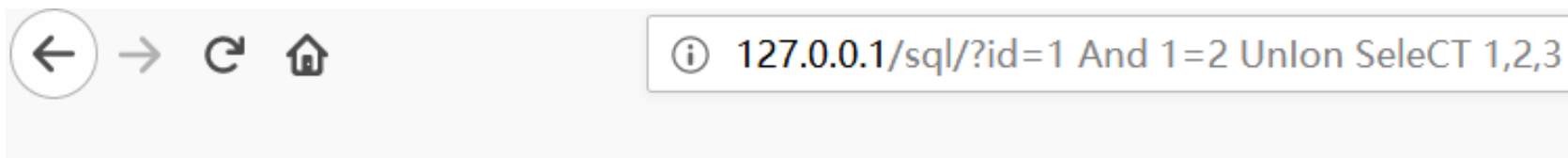
```
queryString={
  UPDATE users
  SET password='newPassword' + ""
  WHERE user Name='admin'--' AND password='oldpwd'
  AND password='' + oldPassword + ""
}
```

SQL注入 vs. 关键词过滤

□ 一个DEMO

```
function filter($str){  
    $str = preg_replace('/and|or|select|union/', '', $str);  
    return $str;  
}
```

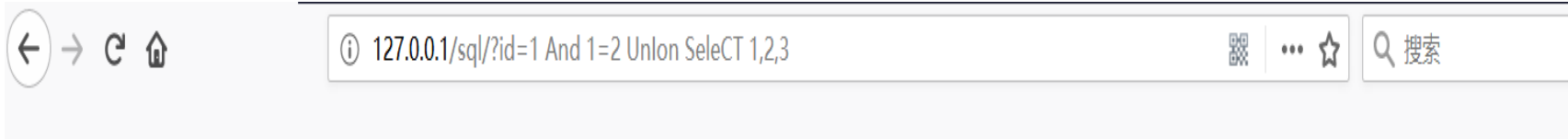
□ BUT, 这个过滤啊, 乃衣服! 可以大小写绕过



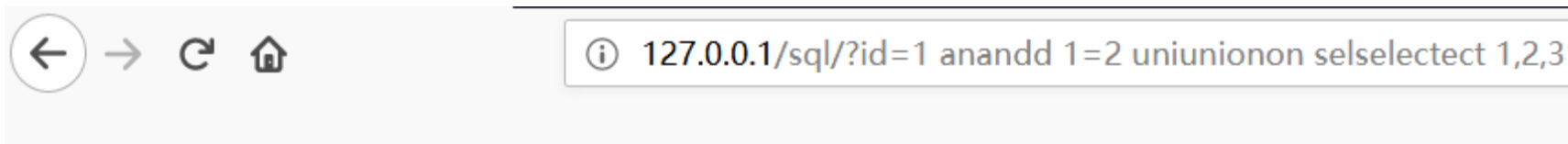
SQL注入 vs. 关键词过滤

- 修改filter以后，无法大小写绕过

```
function filter($str){  
    $str = preg_replace('/and|or|select|union/i', '', $str);  
    return $str;  
}
```



- BUT，只过滤了一次，于是可以双写绕过



SQL注入 vs. 关键词过滤

□ 循环过滤，防双写绕过

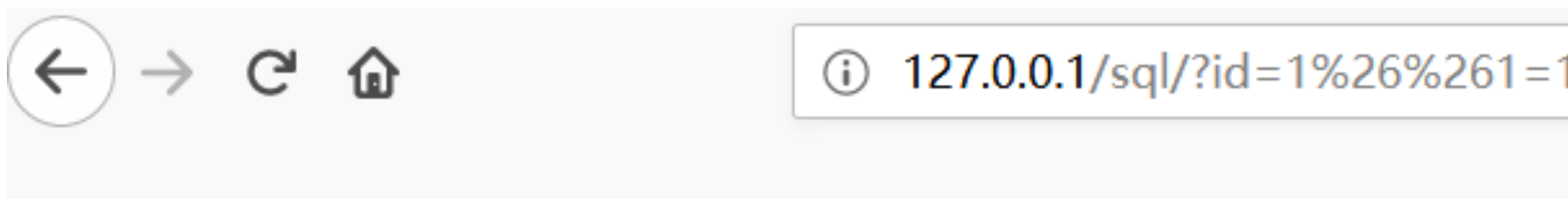
```
function filter($str){  
    while(preg_match('/and|or|select|union/i', $str)){  
        $str = preg_replace('/and|or|select|union/i', '', $str);  
    }  
    return $str;  
}
```

□ HOWEVER

- SQL十分“强大”，甚至可以说接近自然语言
- 相应的，SQL存在大量的等效情形

SQL注入 vs. 关键词过滤

- ❑ 屏蔽掉关键词 “and” / “or” ， 又如何？
 - “and” 可以用 “&&” 代替(%26为URL编码后的 “&”)
 - “or” 可以用 “||” 代替

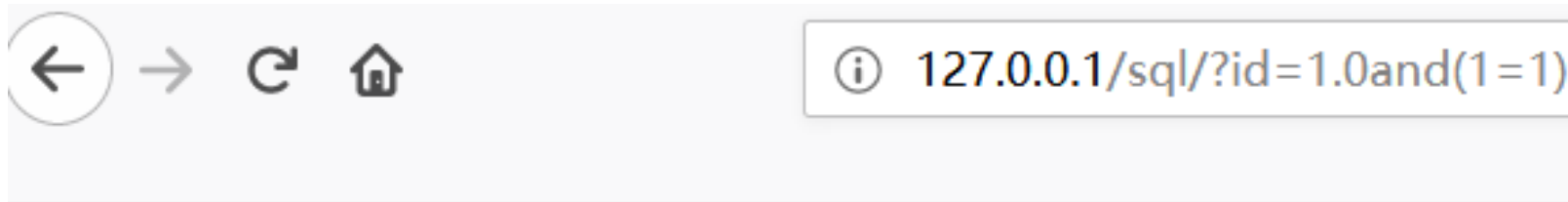
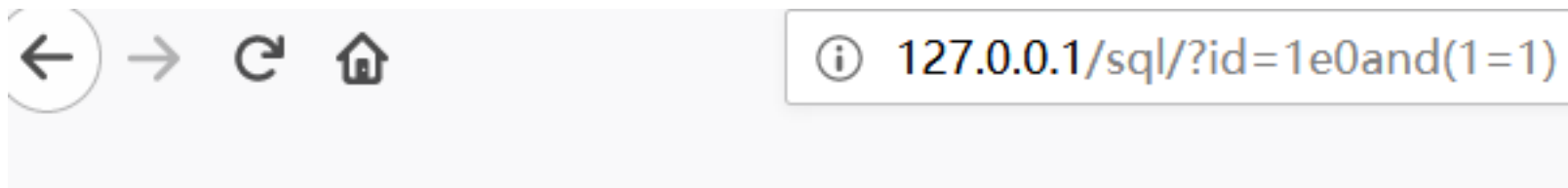


BbB 15

SQL注入 vs. 关键词过滤

□ 不允许空格，又怎样？

- 科学计数法+括号绕过
- 浮点数+括号绕过

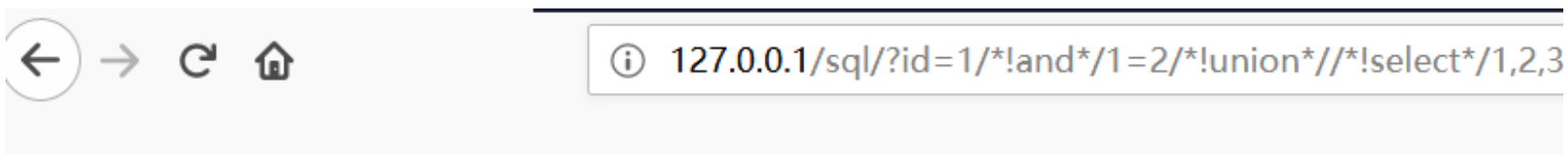


BbB 15

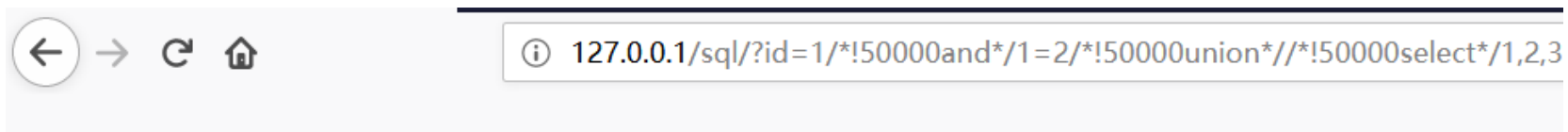
SQL注入 vs. 关键词过滤

□ 不允许空格，又怎样？

- 内联注释(一些WAF不过滤注释里的内容，因此还可用于绕过一些WAF的关键字过滤)



2 3



2 3

SQL注入 vs. 关键词过滤

❑ 禁止逗号？都行，可以，没关系

◦ 子查询+join联合绕过

```
mysql> select 1,2,3;
+----+----+----+
| 1 | 2 | 3 |
+----+----+----+
| 1 | 2 | 3 |
+----+----+----+
1 row in set (0.00 sec)

mysql> select * from (select 1)a join (select 2)b join (select 3)c;
+----+----+----+
| 1 | 2 | 3 |
+----+----+----+
| 1 | 2 | 3 |
+----+----+----+
1 row in set (0.00 sec)
```

现有针对SQL注入的防御措施

对SQL请求的静态+动态解析

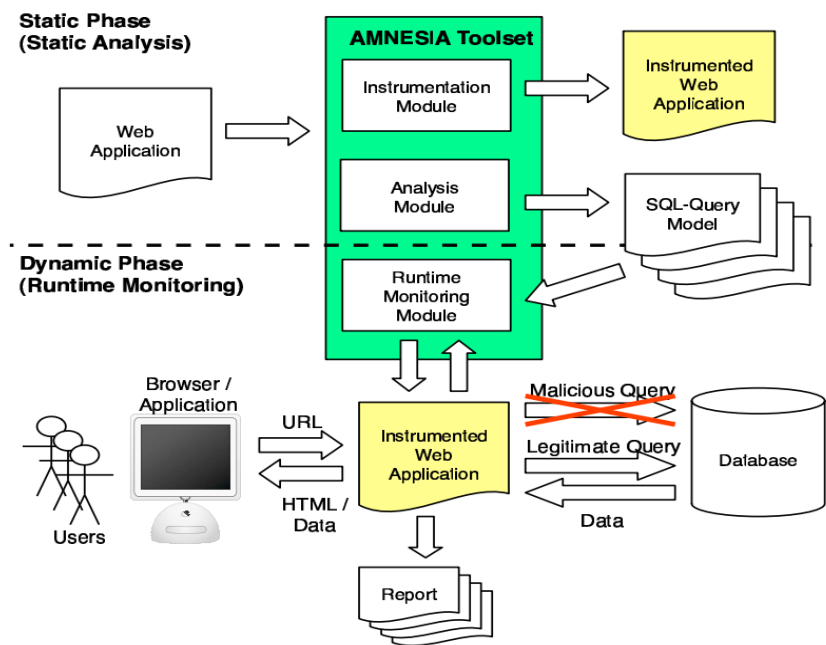
login [doe]

(a) SELECT info FROM users WHERE login='doe' AND

SELECT → info → FROM → users → WHERE → login → = → ' → doe → [

(b) SELECT info FROM users WHERE login='' OR 1=1

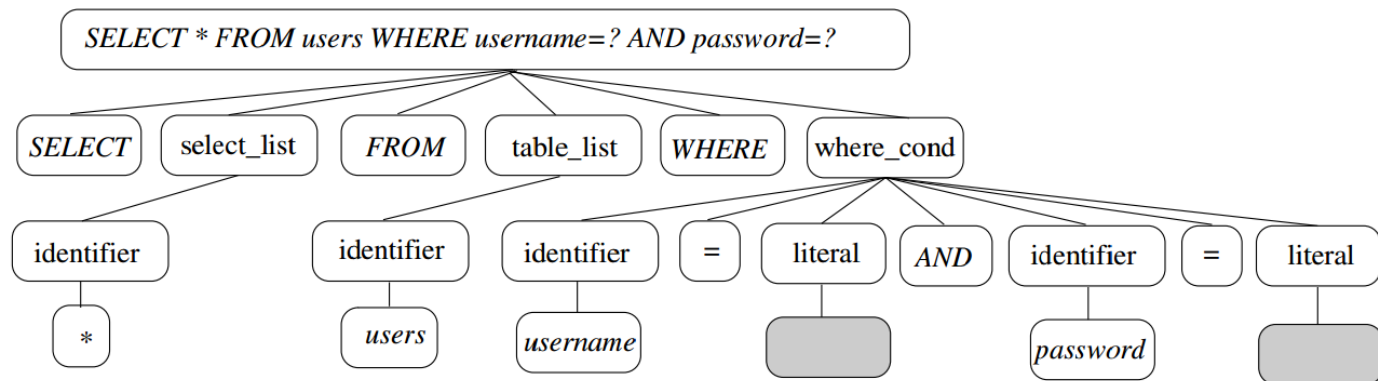
SELECT → info → FROM → users → WHERE → login → = → ' → ε → [



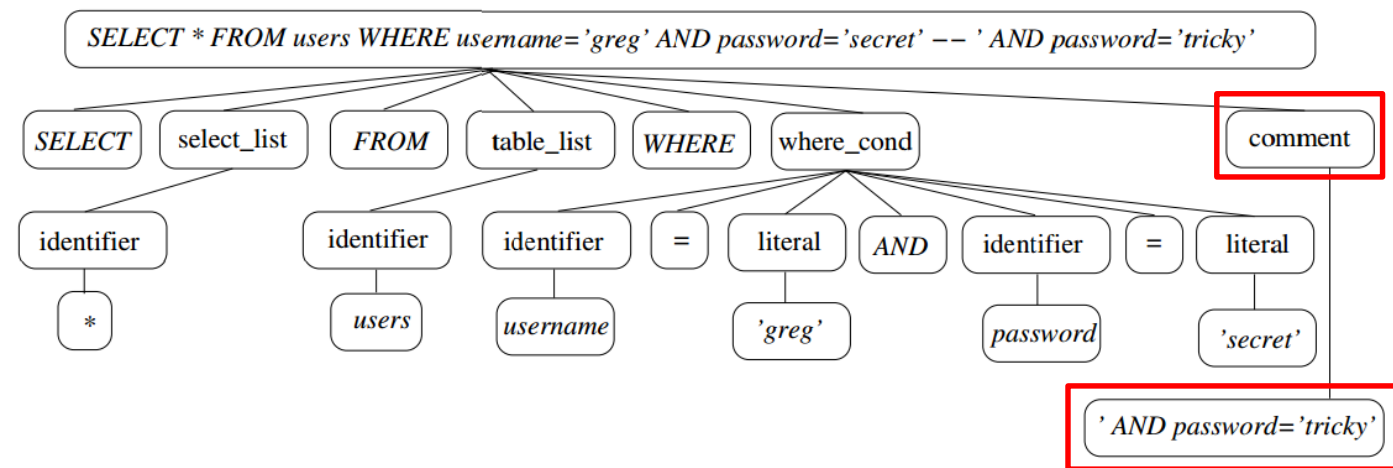
- 参考文献: Halfond W G J, Orso A. AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks[C]//Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, 2005: 174-183.

现有针对SQL注入的防御措施

对SQL请求的动态解析树



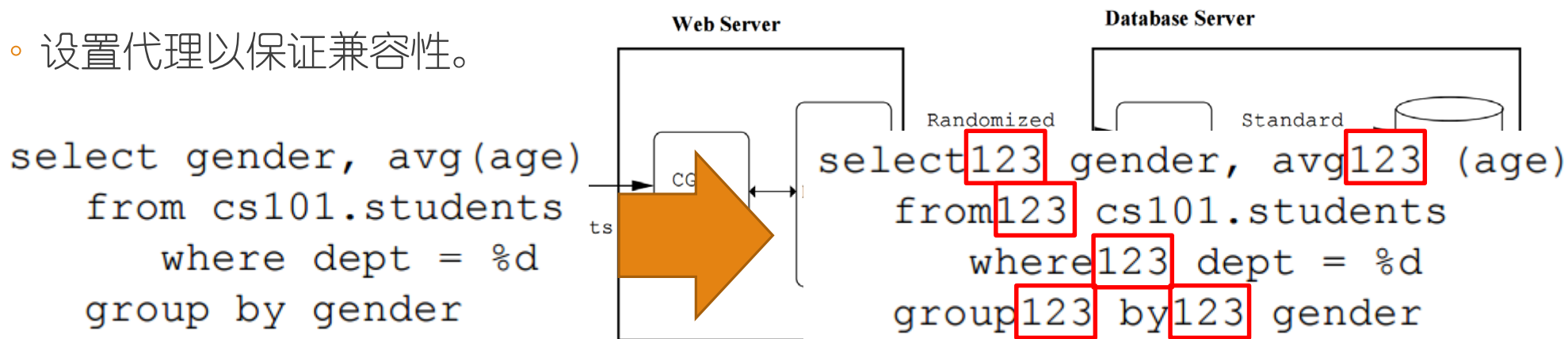
- 参考文献: Buehrer G, We prevent SQL injection attack Software engineering and n



现有针对SQL注入的防御措施

□ 对SQL请求的随机化

- 为SQL查询语句创建随机化的实例（目标：CGI脚本和数据库解析器中的查询模板）
- 设置代理以保证兼容性。



- 参考文献：Boyd S W, Keromytis A D. SQLrand: Preventing SQL injection attacks[C]//International Conference on Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2004: 292-302.

数组越界访问漏洞

□ 溢出与越界不完全相等

- 数组越界分为读/写两种情况，而溢出漏洞属于越界写入
- 一些溢出漏洞的本质是数组越界

□ 数组越界是怎样产生的？

```
{  
    int x[10];  
    cout << x[10];  
}
```

直观的数组越界可以在编译时
较容易地检查出来，但是.....

```
{  
    int i, x[10];  
    ...  
    对i进行算术运算  
    ...  
    cout << x[i];  
}
```

用法是允许的，存在
越界的可能，但编译器
无法指出

数组越界访问漏洞

□ 例1： CVE-2014-0160 “OpenSSL数组越界访问漏洞” (Heartbleed心脏滴血)

- 成因 -- 处理heartbeat response时没有检查包长度的合法性，直接分配相应大小的内存

心跳包字段	长度	说明
ContentType	1byte	心跳包类型，IANA组织把type编号定义为24 (0x18)
ProtocolVersion	2bytes	TLS的版本号，目前主要包括含有心跳扩展的TLS版本：TLSv1.0, TLSv1.1, TLSv1.2
length	2bytes	HeartbeatMessage的长度
HeartbeatMessageType	1byte	Heartbeat类型 01表示heartbeat_request 02表示heartbeat_response
payload_length	2bytes	payload长度
payload	payload_length个bytes	payload的具体内容
padding	>=16bytes	padding填充，最少为16个字节

数组越界访问漏洞

□ 例1： CVE-2014-0160 “OpenSSL数组越界访问漏洞” (Heartbleed心脏滴血)

```
unsigned char *p = &s->s3->rrec.data[0];
```

```
/* Read type and payload length first */  
hbtype = *p++;  
n2s(p, payload);  
p1 = p;
```

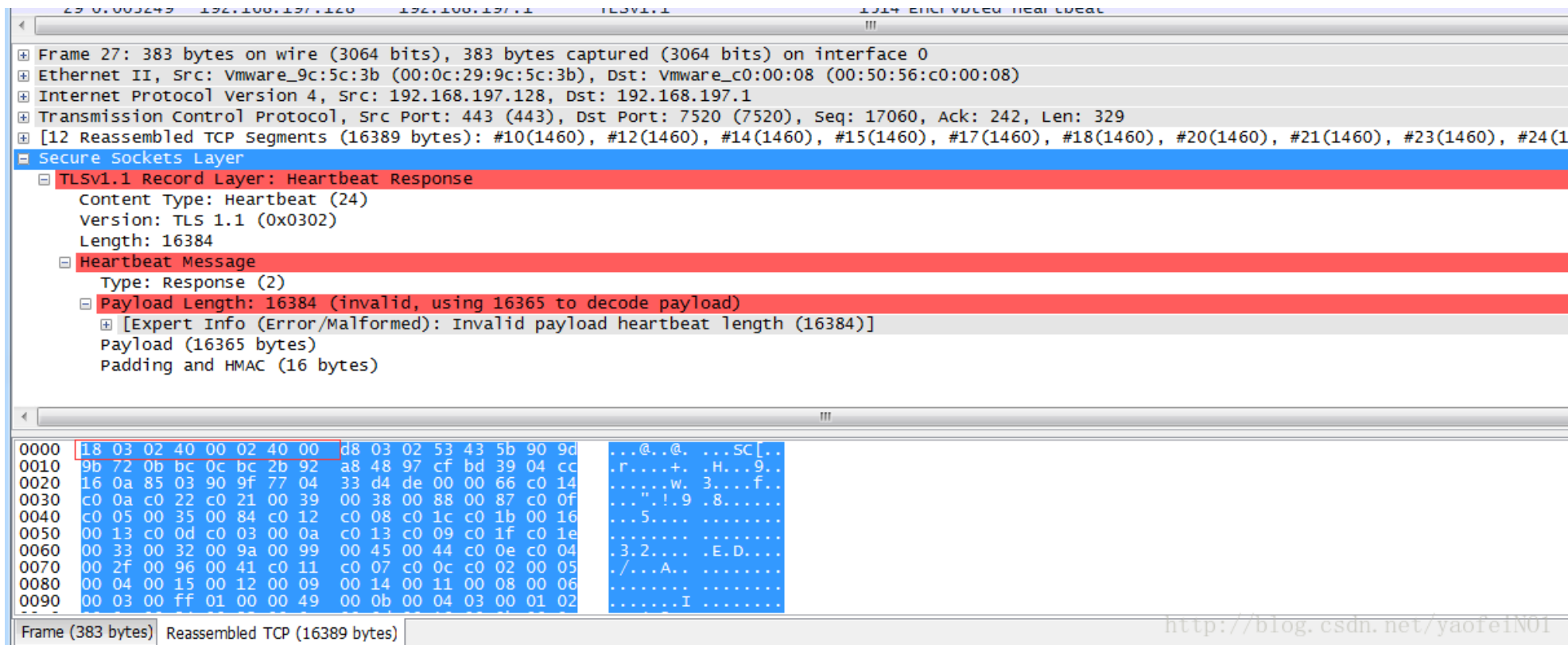

数组越界访问漏洞

□ 例1： CVE-2014-0160 “OpenSSL数组越界访问漏洞” (Heartbleed心脏滴血)

```
/* Allocate memory for the response, size is 1 byte  
 * message type, plus 2 bytes payload length, plus  
 * payload, plus padding  
 */  
buffer = OPENSSL_malloc(1 + 2 + payload + padding);  
bp = buffer;
```

数组越界访问漏洞

例1： CVE-2014-0160 “OpenSSL数组越界访问漏洞” (Heartbleed心脏滴血)



The image shows a Wireshark packet capture of a TLSv1.1 Heartbeat Response. The packet details pane shows the following structure:

- Secure Sockets Layer
 - TLSv1.1 Record Layer: Heartbeat Response
 - Content Type: Heartbeat (24)
 - Version: TLS 1.1 (0x0302)
 - Length: 16384
 - Heartbeat Message
 - Type: Response (2)
 - Payload Length: 16384 (invalid, using 16365 to decode payload)
 - [Expert Info (Error/Malformed): Invalid payload heartbeat length (16384)]
 - Payload (16365 bytes)
 - Padding and HMAC (16 bytes)

The packet bytes pane shows the raw data of the heartbeat message, starting with the type 02 and the payload length 16384 (0x000010000).

Offset	Hex	ASCII	
0000	18 03 02 40 00 02 40 00	d8 03 02 53 43 5b 90 9d	...@..@..SC[.
0010	9b 72 0b bc 0c bc 2b 92	a8 48 97 cf bd 39 04 cc	.r....+.H...9.
0020	16 0a 85 03 90 9f 77 04	33 d4 de 00 00 66 c0 14w.3...f..
0030	c0 0a c0 22 c0 21 00 39	00 38 00 88 00 87 c0 0f	...".!9.8.....
0040	c0 05 00 35 00 84 c0 12	c0 08 c0 1c c0 1b 00 16	...5....
0050	00 13 c0 0d c0 03 00 0a	c0 13 c0 09 c0 1f c0 1e	..3.2....E.D...
0060	00 33 00 32 00 9a 00 99	00 45 00 44 c0 0e c0 04	./...A..
0070	00 2f 00 96 00 41 c0 11	c0 07 c0 0c c0 02 00 05I.....
0080	00 04 00 15 00 12 00 09	00 14 00 11 00 08 00 06	
0090	00 03 00 ff 01 00 00 49	00 0b 00 04 03 00 01 02	

Frame (383 bytes) Reassembled TCP (16389 bytes)

数组越界访问漏洞

□ C语言的数组越界为何难以检查？

- 性能 \leftrightarrow 安全

```
for (size_t i = 0; i < n; ++i)
    c[i] = a[i] + b[i];
```

```
Pointer ap = &a[0], bp = &b[0], cp = &c[0], aend = ap + n;
while (ap < aend) {
    *cp++ = *ap++ + *bp++;
}
```

- 指向数组元素的指针和数组本身是完全独立的
- 指针运算的可行性
- 用指针表示范围，不仅需要指针本身存在且可用，还需要指针指向的内存可用

What's next?

- 较为高级的内存访问漏洞
 - 双重释放漏洞
 - 释放重引用漏洞
- 其他