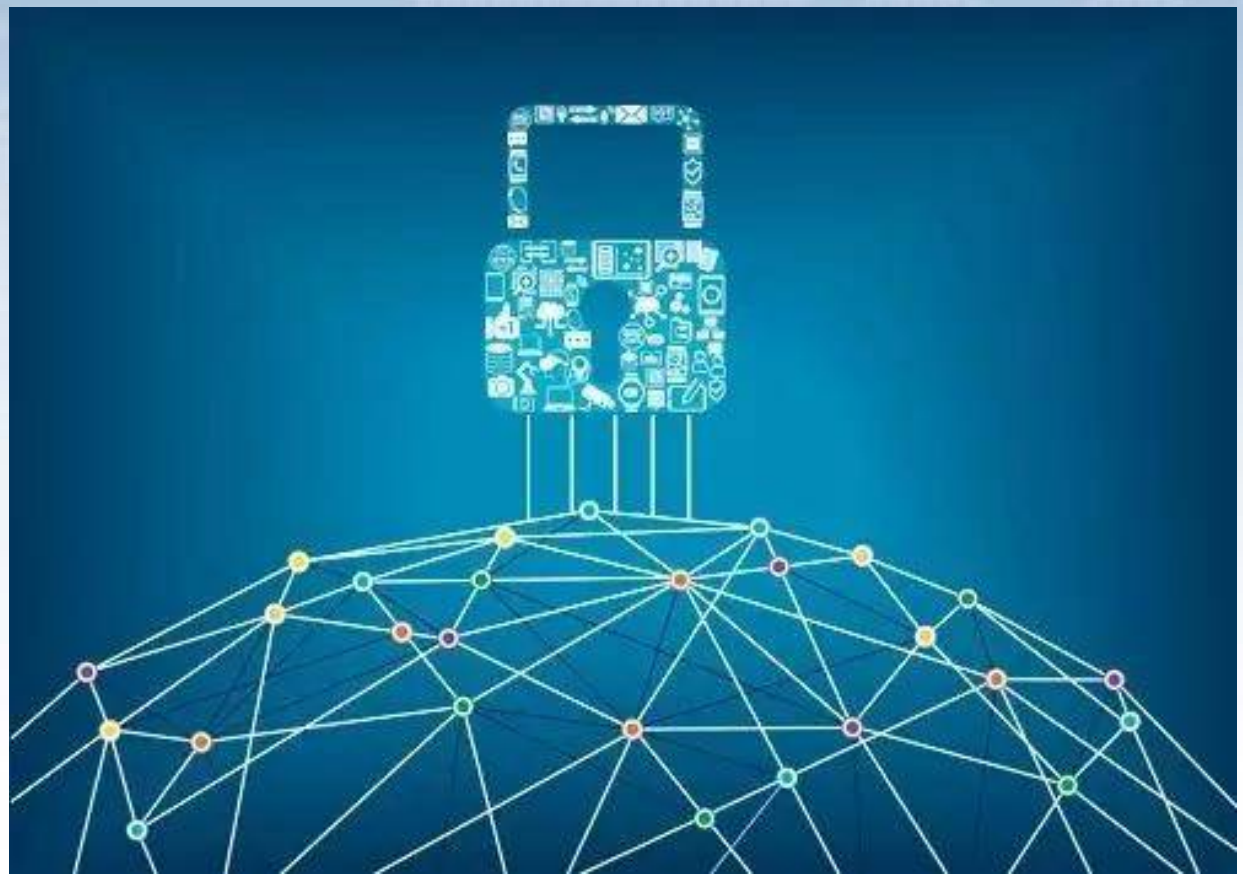
CTF 中的密码学

- 东北大学 NEX

前言

密码学在现代特别指对信息以及其传输的数学性研究，常被认为是数学和计算机科学的分支，和信息论也密切相关。

前言



OWASP
Open Web Application Security Project

前言

CTF中的密码学

- 学习难度 > 得分收入，学习门槛较高--需要有较强的数学功底
- 趋势：密码类型题目趋于论文化 ,和 Reverse、PWN、WEB等其他类型题目结合。

前言

- 古典密码学 & 现代密码学
- 古典密码学
  - 移位密码
  - 替代密码
- 现代密码学
  - 对称加密
  - 非对称加密
  - 杂凑函数(哈希)
  - 签名

- 唯密文攻击:攻击者只拥有密 文;
- 已知明文攻击:攻击者拥有一 些密文对应的明文;
- 选择明文攻击:攻击者可以进 行加密,能够获取指定明文加 密后的密文;
- 选择密文攻击:攻击者可以进 行解密,能够获取指定密文解 密后的明文。

前言

- 古典密码学（单表替代、多表替代、其他）
  - 通常可以拿到的：
    - 加密算法
    - 密文
  - 决定了大部分的古典密码学题目是唯密文攻

解题思路

- 爆破（空间小） • 词频统计（英语语言特性）

前言

- 分组密码
  - 通常可以拿到的：
    - 加密算法
    - 密文（唯密文攻击）/// 密文+若干对已知明密文（已知明文攻击）/// 加 密或者解密的oracle（选择明文攻击、选择密文攻击）
- 解题思路
  - 考察模式的问题，和加密算法本身无关：分析我们可以拿到的信息是否 能够满足几种攻击模式（CBC等模式的问题）
  - 密码学分析：差分、积分、代数等攻击方式

前言

- 序列密码
  - 通常可以拿到的：
    - 加密算法
    - 明密文（直接异或出乱数，或直接给乱数）
  - 解题思路
    - 一般为逆推寄存器初态

前言

- 公钥密码
  - 通常可以拿到的
    - 加密算法
    - 密文+公钥 / oracle
  - 解题思路
    - 不会考察算力，通常会考察公钥密码体制使用不当而产生的一些问题
    - 分析手中条件，符合哪种攻击模式

前言

- 杂凑函数
  - 通常可以拿到的
    - 哈希算法
    - 哈希值
  - 解题思路：
    - MD结构的一些问题
    - 碰撞和爆破

公钥密码

- RSA
  - Alice：随机选择两个不同大质数 p 和 q，计算 m=p×q
  - Alice：根据欧拉函数，可以求出
  - Alice：选择r的一对互逆的整数e和d
  - Alice：公开n和e->做为公钥
  - Bob：使用公钥对明文m进行加密，并发送c给alice
    - c=m^e mod n # c=pow(m,e,n)
  - Alice：使用私钥对c进行解密
    - m=c^d mod n
    - e*d=1 mod f(n)
    - modinv(e,(p-1)*(q-1))
    - m=pow(c,d,n)
  - 攻击者：c,e,n

公钥密码

- 直接模数分解
- 公约数模数分解
- 共模攻击
- 小指数明文爆破
- RSA-选择密文攻击
- LLL-attack
- Wiener Attack & Boneh_durfee Attack
- 广播攻击
- 相关消息攻击

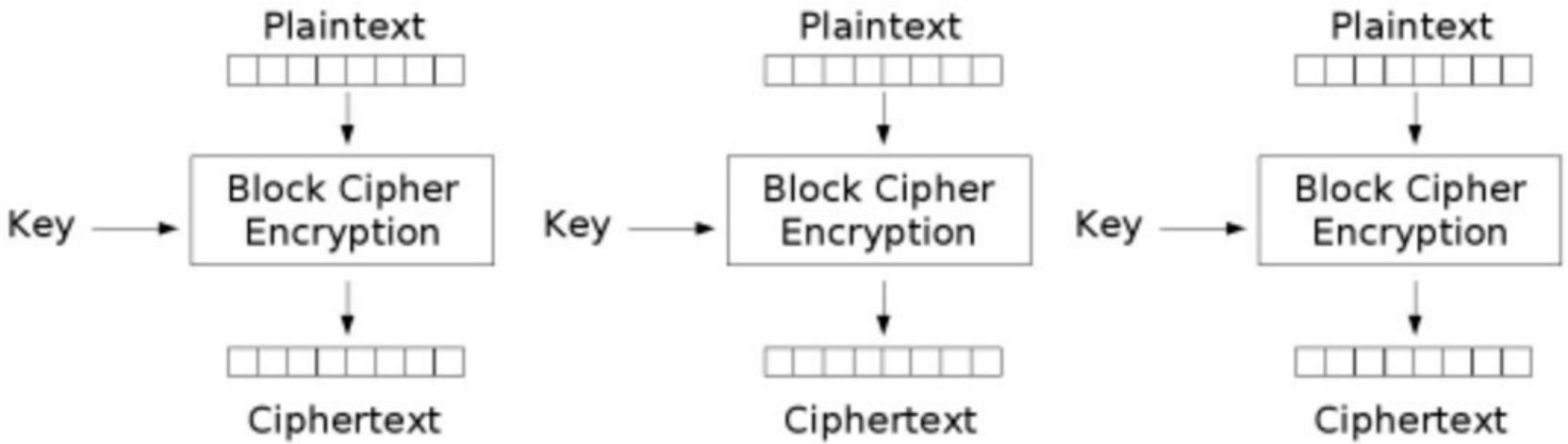
CTF WEB中的密码学

- 加密模式

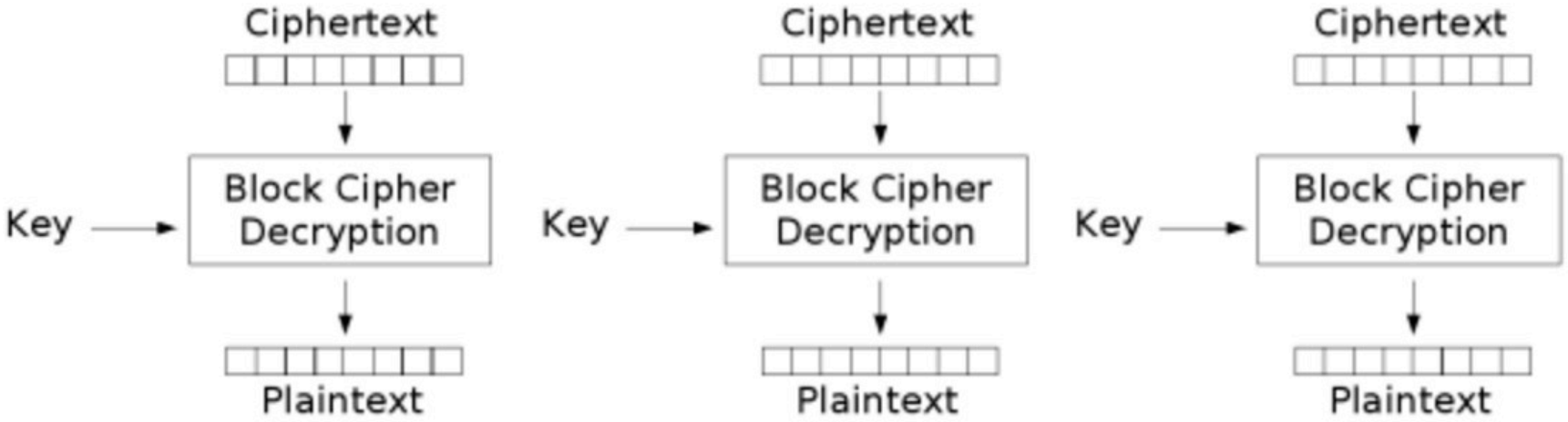
密码学中，区块（Block）密码的工作模式有六种（ECB、CBC、PCBC、CFB、OFB、CTR）。

首先讨论最简单的ECB模式：

ECB模式。需要加密的消息按照块密码的块大小被分为数个块，并对每个块进行独立加密。



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

CTF WEB中的密码学

- 这种模式最大的问题就是它是使用对所有的密文使用同一个密钥进行加密，所以同样的明文一定会加密成同样的密文。所以，如果存在多组明文进行加密，那么我们只需要观察一组明文-密文对就能得到所有明文的现象。

例子

p1.php

Users > DELL > Desktop > p1.php > ...

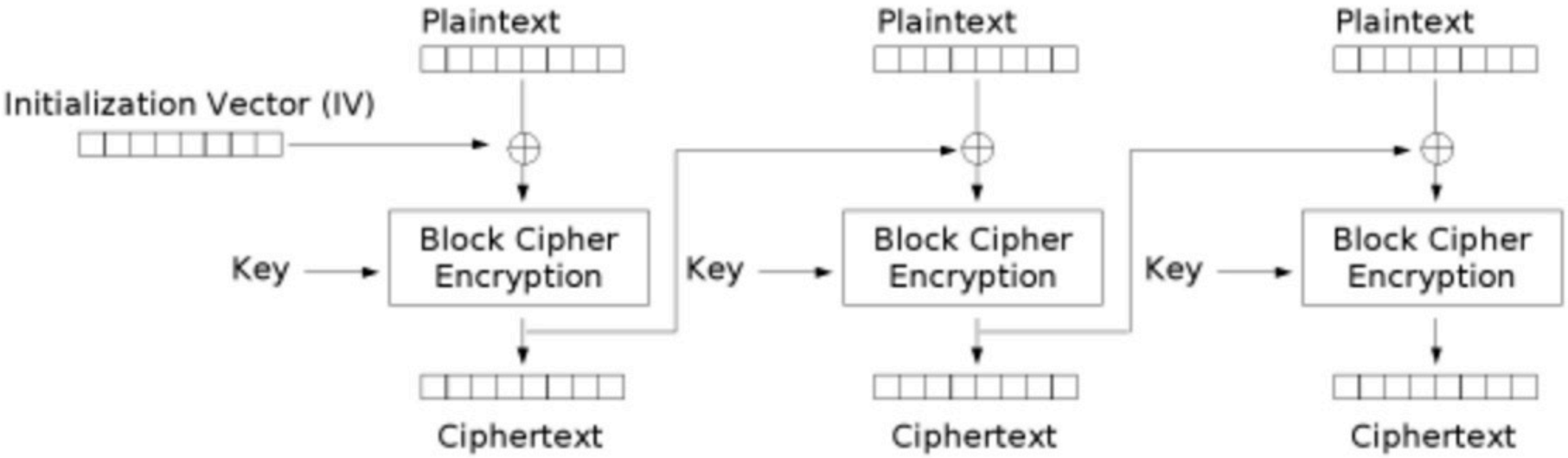
```php
<?php
function AES($data){
    $privateKey = "12345678123456781234567812345678";
    $encrypted = mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $privateKey, $data, MCRYPT_MODE_ECB);
    $encryptedData = (base64_encode($encrypted));
    return $encryptedData;
}
function DE__AES($data){
    $privateKey = "12345678123456781234567812345678";
    $encryptedData = base64_decode($data);
    $decrypted = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $privateKey, $encryptedData, MCRYPT_MODE_ECB);
    $decrypted = rtrim($decrypted, "\0") ;
    return $decrypted;
}
if (@$_GET['a']=='reg'){
    setcookie('uid', AES('9'));
    setcookie('username', AES($_POST['username']));
    header("Location: http://127.0.0.1/ecb.php");
    exit();
}
if (@!isset($_COOKIE['uid'])||@!isset($_COOKIE['username'])){
    echo '<form method="post" action="ecb.php?a=reg"> Username:<br> <input type="text" name="username"> <br> Password:<br> <input type="text" name="password" > <br><br> <input type="submit" value="注册"> </form> ';
}
else{
    $uid = DE__AES($_COOKIE['uid']);
    if ( $uid != '4'){
        echo 'uid:' .$uid .'<br/>';
        echo 'Hi ' . DE__AES($_COOKIE['username']) .'<br/>';
        echo 'You are not administrotor!!';
    }
    else {
        echo "Hi you are administrotor!!" .'<br/>';
        echo 'Flag is 360 become better';
    }
}
?>
```

例子

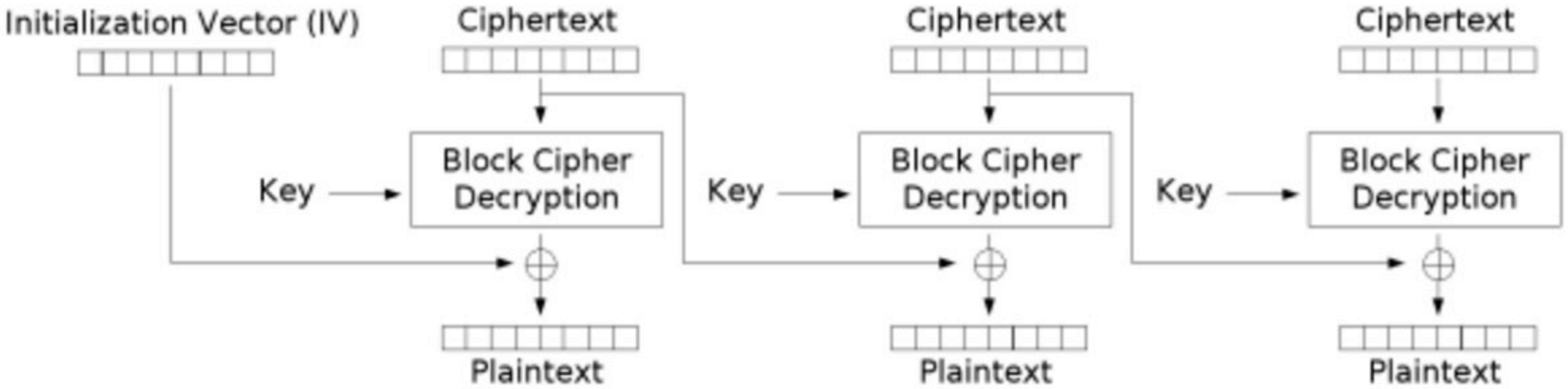
在注册的时候，我们可以控制我们的用户名，但是我们的UID被默认设置为9。而后面，我们需要修改uid到4来提升我们的权限到administrotor。因为这里采用的是ECB模式，所以我们可以依据username的明文操纵生成我们想要的uid密文。这里AES采用了128位的加密，即16个字节。所以，我们可以注册17个字节，多出的那一个字节就可以是我们希望的UID的值，而此时我们查看username的密文增加部分就是UID的密文，即可伪造UID。（因为第十七个字节单独为一组，前面十六个字节为一组）

CTF WEB中的密码学

- CBC模式字节翻转攻击
- 这里，会有两个攻击点：
- 更改iv向量，影响第一个明文分组
- 如果我们改变Ciphertext N-1的字节，其会影响到Ciphertext N块的解密过程。 这个就是CBC 比特翻转攻击的原理。



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Padding Oracle 攻击

- 这是CBC模式下存在的攻击，与具体的加密算法无关（当然，必须是分组加密）。不过，实际上Padding Oracle不能算CBC模式的问题，它的根源在于应用程序对异常的处理反馈到了用户界面（即服务器对解密后数据Padding规则的校验。若不符合Padding规则，则返回500.其它，返回200），是算法在生产环境中使用不当造成的问题。

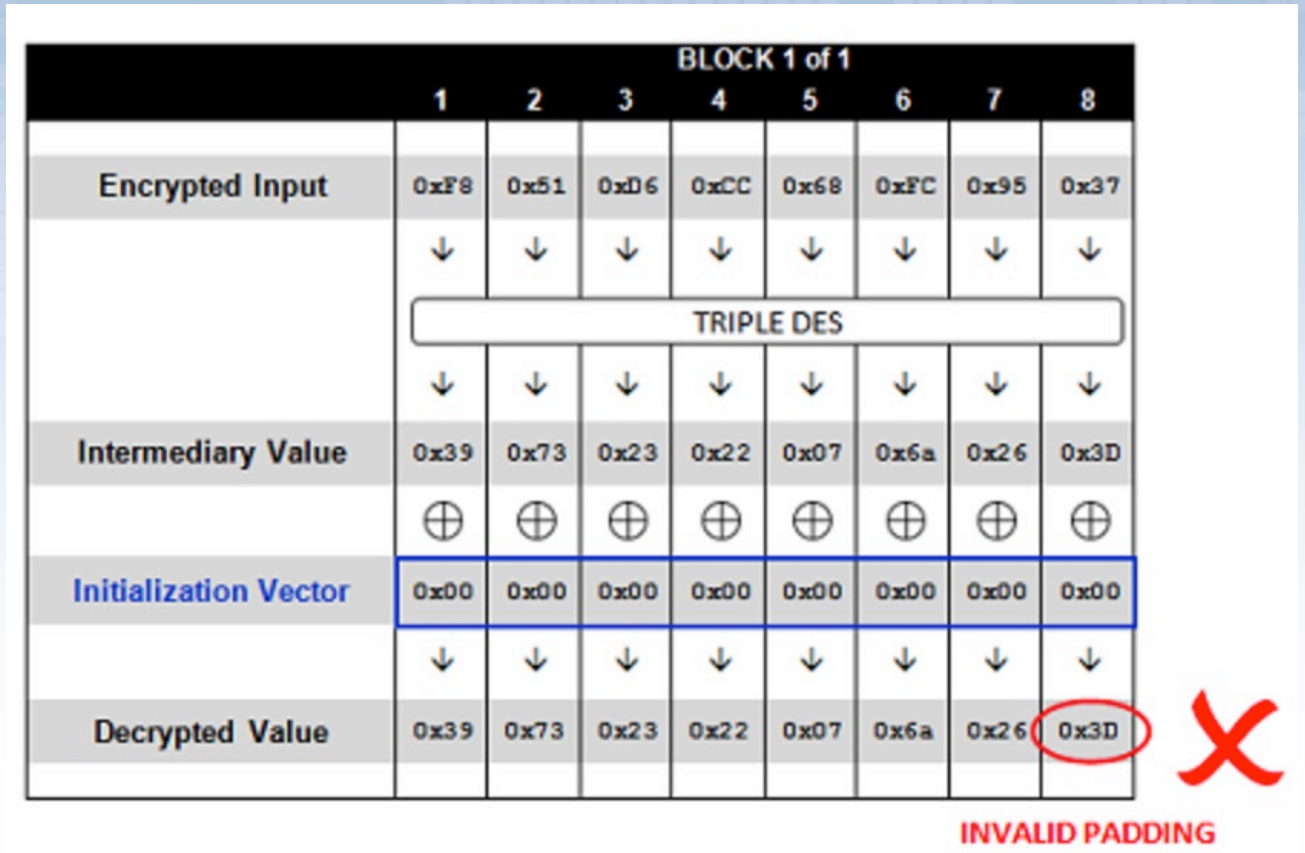
Padding Oracle 攻击

- PKCS#5是一种Padding规则。简单来说，它在数据填充中，使用缺失的位数长度来统一填充。缺5位就用5个0x05填充，缺2位就用2个0x02填充；如果正好为8位，就需要扩展8个0x08填充。

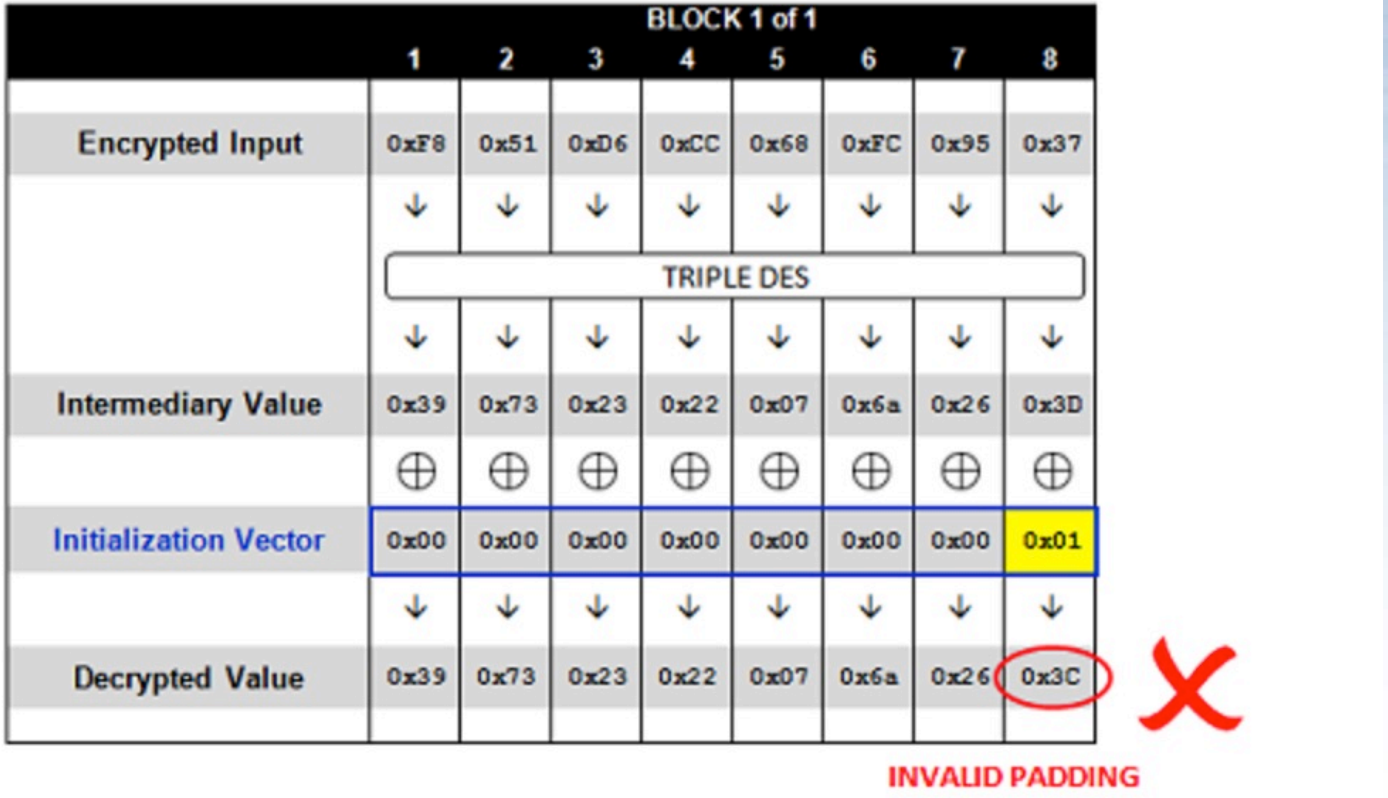
Padding Oracle 攻击

- 我们可以构想如下一个可以被利用的漏洞服务器，来解释其特点：对于请求，会有如下反馈：
- 如果解密过程没有问题，明文验证（如用户名密码验证）也通过，则会返回正常 HTTP 200；
- 如果解密过程没有问题，但是明文验证出错（如用户名密码验证），则还是会返回 HTTP 200，只是内容上是提示用户用户名密码错误；
- 如果解密过程出问题，比如Padding规则核对不上，则会爆出 HTTP 500错误。
- 这就是先前说的，服务器对解密过程的异常反馈到了用户界面，这种反馈会导致整体的危险性。

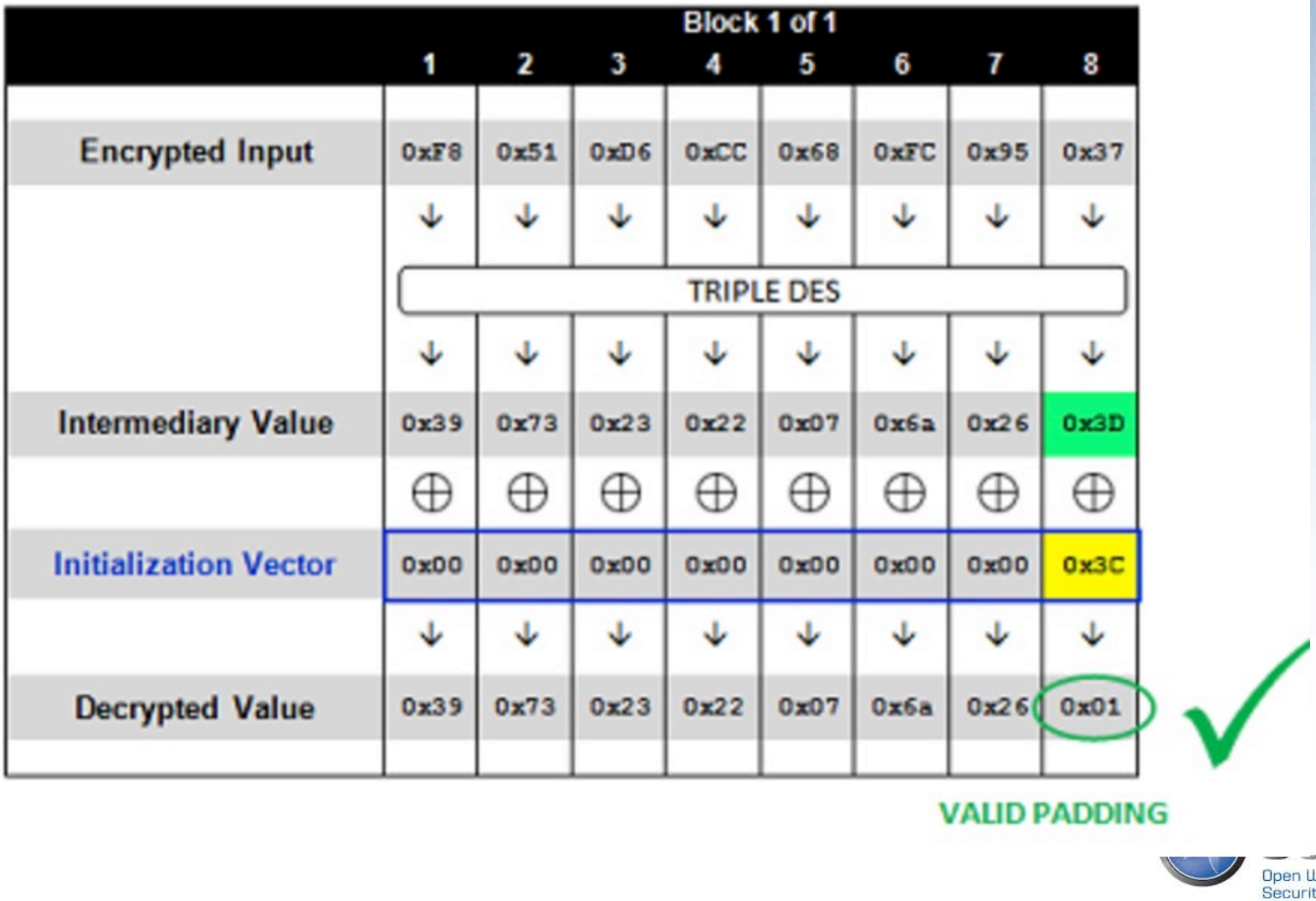
Padding Oracle 攻击



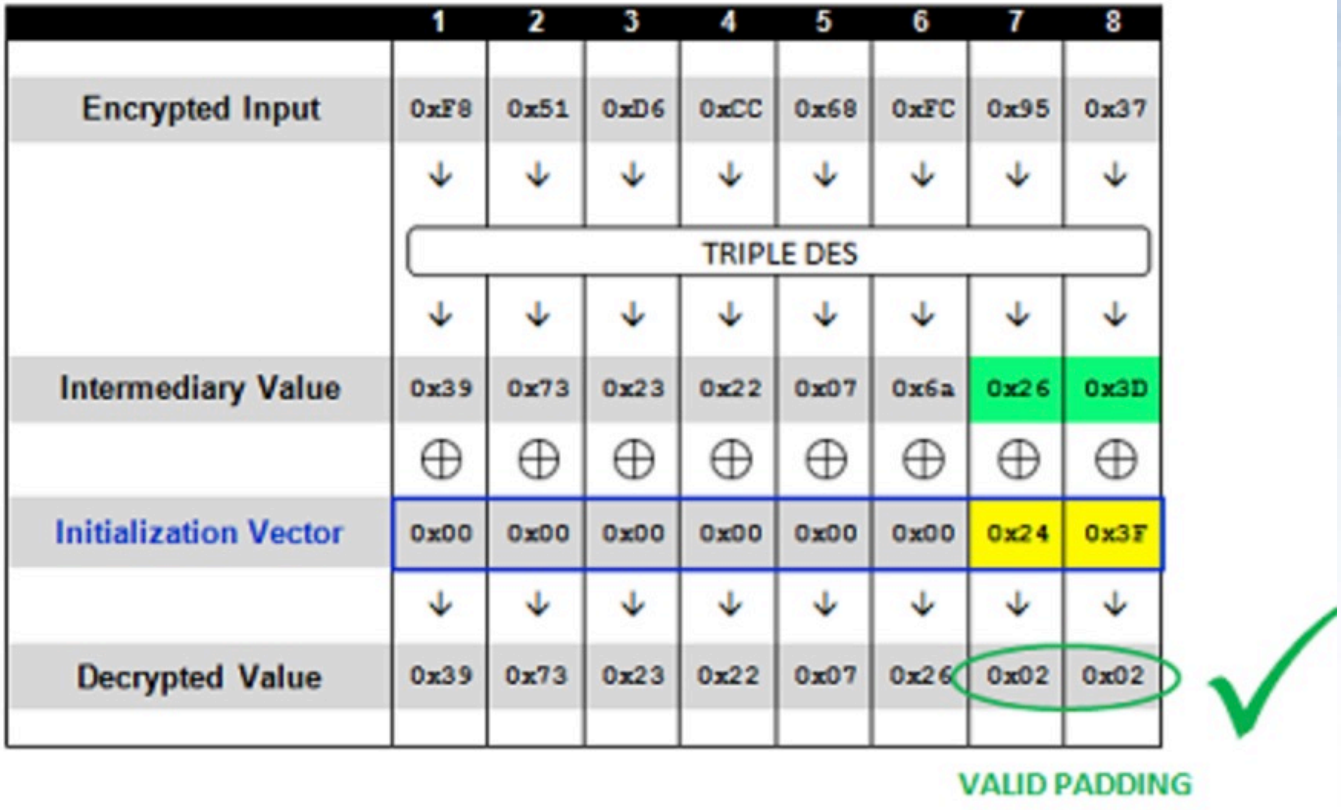
Padding Oracle 攻击

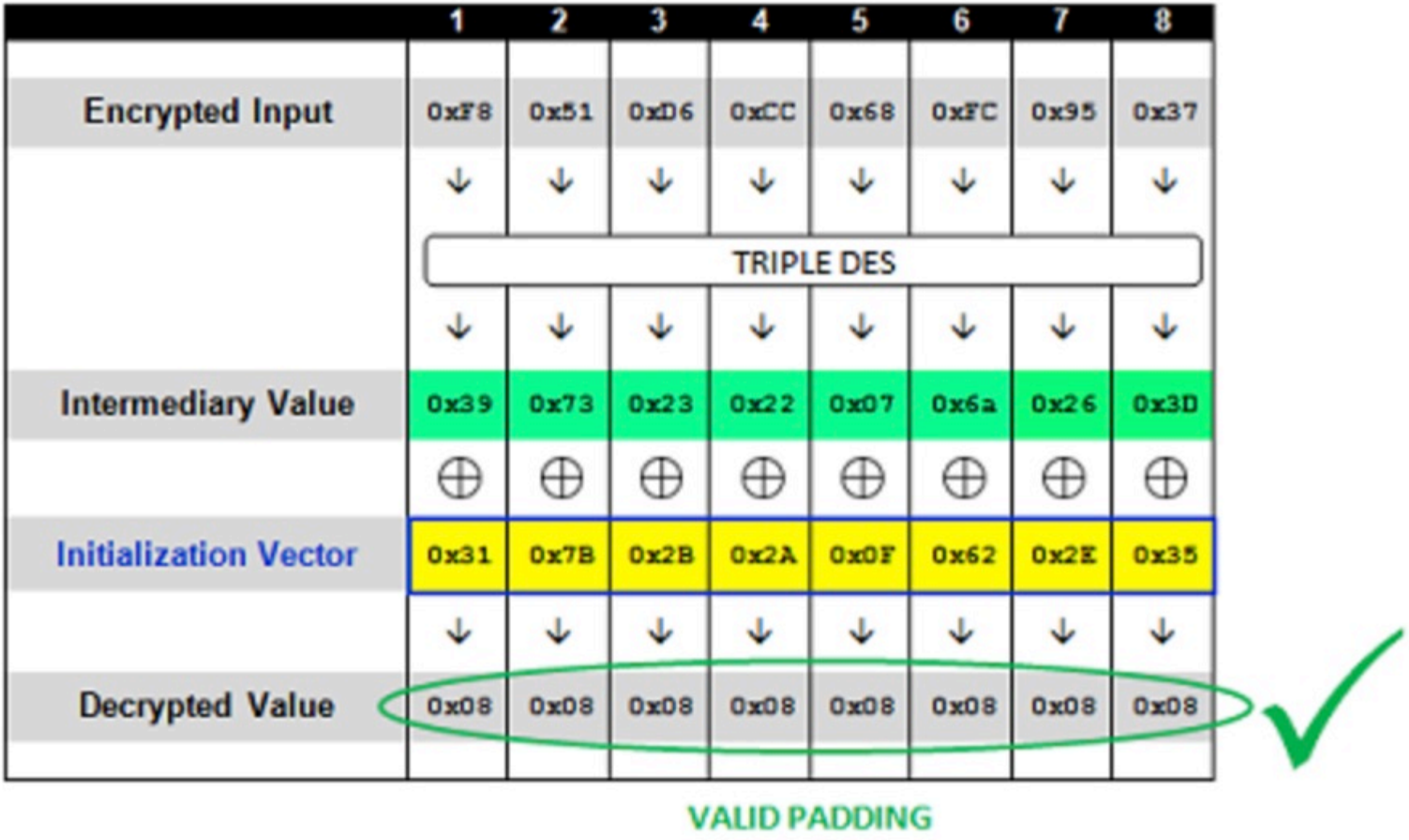


Padding Oracle 攻击



Padding Oracle 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

Padding Oracle 攻击

现在，我们得到中第一个块的中间值和已知的初始化向量，我们就可以知道第一个块的明文了。依次类推，我们就可以得到后面块的中间值，再得到明文。

哈希扩展攻击

- 这里简单说一下MD-5，其算法流程大体如下：
  - 将消息内容按64字节分组
  - 最后一组的长度模512（64字节）小于448（56字节）的使用空字节（0x00）填充，空字节开始处使用0x80标识，若大于等于448（56字节）则填充本组至64字节后，再向下填充一个分组至56字节，结尾8字节用于填写整个消息的长度。
- 每个分组进行64轮数学计算，上一组的计算结果作为下一组计算的初始输入，最开始的输入为IV。

哈希扩展攻击

所以，当知道MD5(secret)时，我们可以在不知道secret的情况下，可以轻易地推出MD5(secret||padding||m')

哈希扩展攻击

- 修复

可以将secret放在末尾，如 MD5(m+secret)。这样，如果希望推导出MD5(m+secret||padding||m')，结果由于自动附加secret在末尾的关系，会变成MD5(m||padding||m'||secret)，从而导致Length Extension run不起来。


OWASP
Open Web Application
Security Project