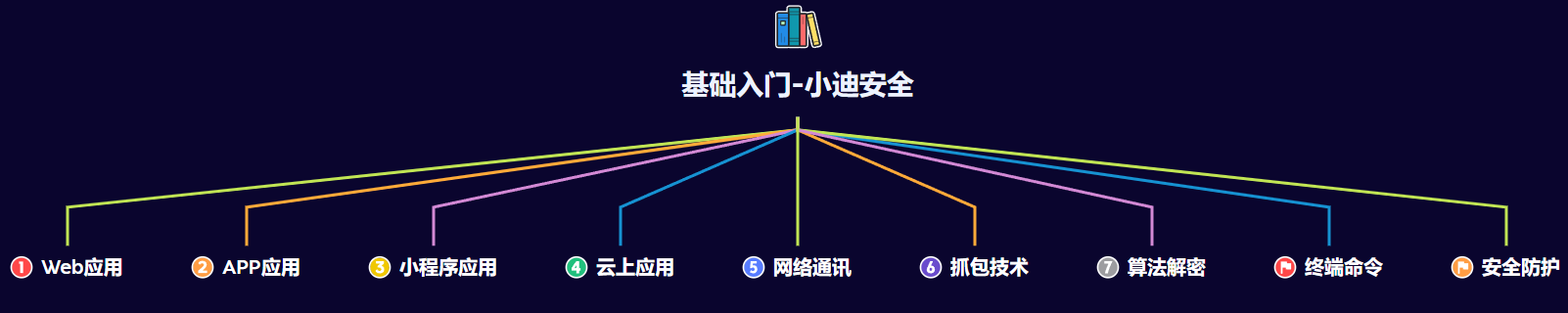
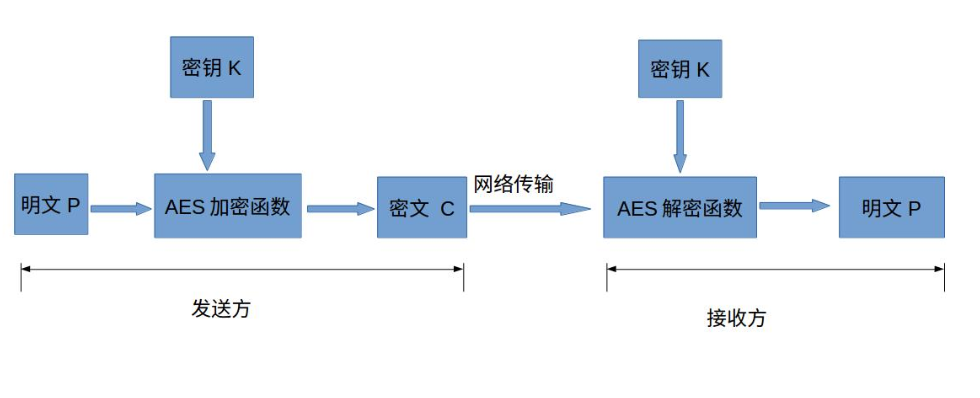
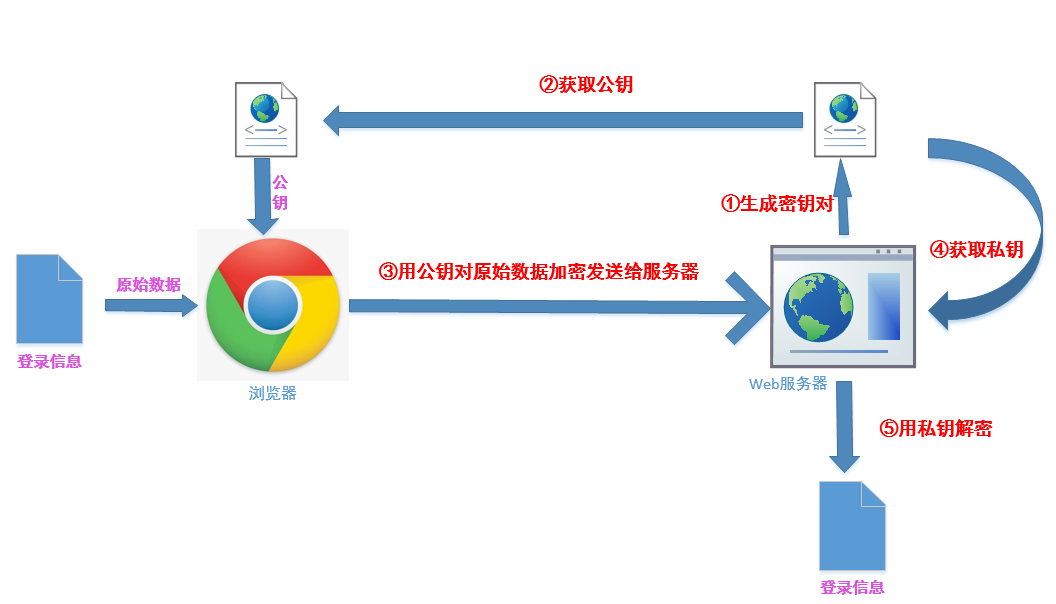
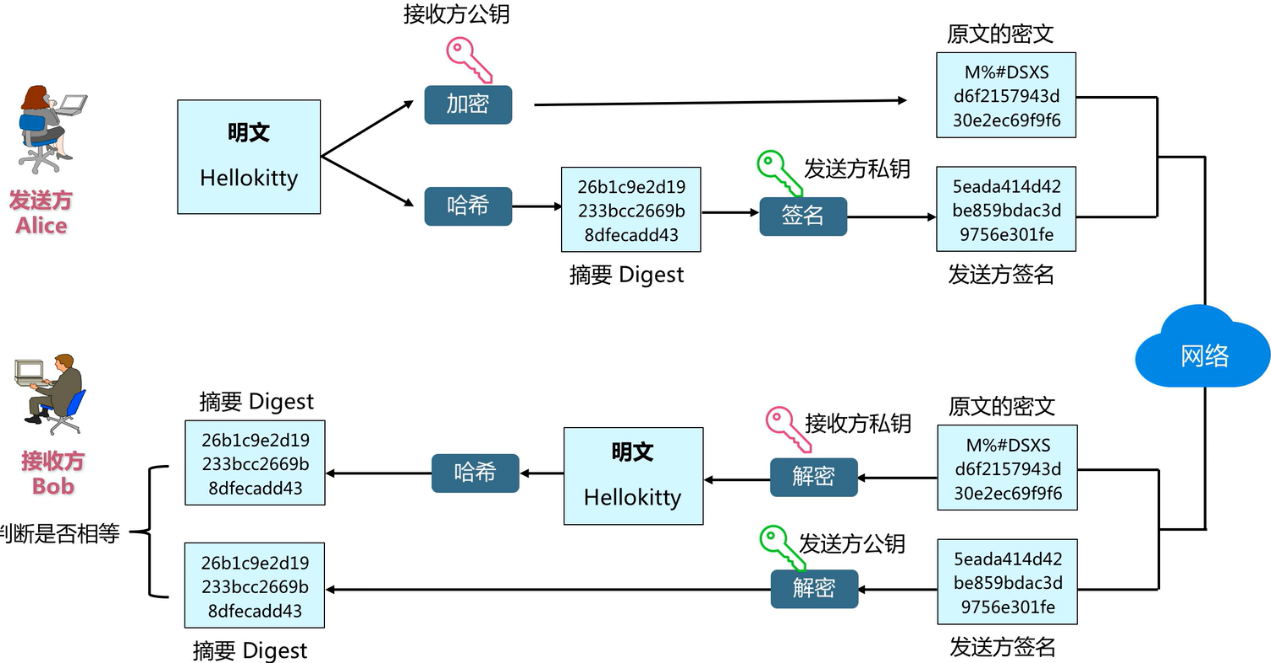
基础入门-算法解密&散列对称非对称&字典碰撞&前后端逆向&MD5&AES&DES&RSA



#知识点：  
1、基础入门-Web应用-域名上的技术要点  
2、基础入门-Web应用-源码上的技术要点  
3、基础入门-Web应用-数据上的技术要点  
4、基础入门-Web应用-解析上的技术要点  
-----------------------------------------------------  
1、基础入门-Web应用-搭建架构上的技术要点  
2、基础入门-Web应用-源码类别上的技术要点  
-----------------------------------------------------  
1、基础入门-Web应用-防护产品-WAF保护  
2、基础入门-Web应用-加速服务-CDN节点  
3、基础入门-Web应用-文件托管-OSS存储  
4、基础入门-Web应用-通讯服务-反向代理  
5、基础入门-Web应用-运维安全-负载均衡  
-----------------------------------------------------  
1、基础入门-Web应用-蜜罐系统  
2、基础入门-Web应用-堡垒机运维  
3、基础入门-Web应用-内外API接口  
4、基础入门-Web应用-第三方拓展架构  
-----------------------------------------------------  
1、基础入门-APP应用-开发架构安全问题  
2、基础入门-小程序应用-开发架构安全问题  
-----------------------------------------------------  
1、抓包技术-Web应用-http/s-Burp&Yakit  
2、抓包技术-APP应用-http/s-Burp&Yakit  
3、抓包技术-PC端应用-http/s-Burp&Yakit  
4、抓包技术-WX小程序-http/s-Burp&Yakit  
5、抓包技术-软件联动-http/s-Proxifier  
6、抓包技术-通用方案-http/s-ReqableApi  
7、抓包技术-其他工具-http/s-Fiddler&Charles  
-----------------------------------------------------  
1、抓包技术-HTTP/S双层代理-扶墙环境  
2、抓包技术-HTTP/S上游下游-项目联动  
3、抓包技术-全局协议-WireShark&科来  
-----------------------------------------------------  
1、数据不回显原因和解决-带外延迟反弹写文件  
2、数据不出网原因和解决-出入站策略正反向连接  
-----------------------------------------------------  
1、反弹Shell-项目&命令&语言等  
2、系统渗透命令-网络&文件&操作等  
-----------------------------------------------------  
1、传输格式&数据-类型&编码&算法  
2、密码存储&混淆-不可逆&非对称性  
-----------------------------------------------------  
0、算法类型-单向散列&对称性&非对称性  
1、算法识别加解密-MD5&AES&DES&RSA  
2、解密条件寻找-逻辑特征&源码中&JS分析  
  
#章节点：(待补充)  
Web架构，App架构，小程序架构，前后端分离，容器技术，云产品服务，  
数据加解密算法，数据包抓取，数据包解析，正反向网络通讯，内外网通讯，  
防火墙出入站，Windows&Linux渗透命令，WAF产品，负载均衡，加壳保护等  
  
#具体点：(待补充)  
架构：WEB,APP,小程序,前后端,容器化等  
服务：OSS存储,CDN加速,云数据库,负载均衡等  
网络：不回显,反向代理,防火墙出入站,内外网,正反向连接等  
算法：MD5,Base64,AES,DES,Salt,自定义,代码加密算法等  
命令：Windows&Linux,文件下载,网络查看,反弹权限,用户等  
防护：WAF防护,蜜罐系统,CDN加速,权限设置,加壳加密反调试等







演示案例：

* 算法识别加解密-MD5&AES&DES&RSA
* 解密条件-有源码找逻辑&无源码JS逆向

#传输格式  
JSON XML WebSockets HTML 二进制 自定义  
WebSockets：聊天交互较常见（豆包等应用）  
https://zhuanlan.zhihu.com/p/712032652  
https://cloud.tencent.com/developer/article/1917215  
文件上传：  
https://www.cnblogs.com/wanglei1900/p/17177303.html  
影响：安全后渗透测试必须要统一格式发送  
  
#传输数据  
例：  
MD5  
Base64  
自定义算法  
影响：安全后渗透测试必须要统一加密发送  
  
#密码存储  
例：  
-ZBlog&Dz  
-Win&Linux  
-MSSQL&MYSQL  
影响：安全后渗透测试必须要做到算法解密  
  
#代码混淆  
例：  
-PHP&JS混淆加密  
-DLL&JAR代码保护  
影响：代码审计，逆向破解  
  
由上述内容发现还需那些内容学习：  
1、加密算法的识别与解密  
2、自定义算法的识别解密

安全测试中：  
密文-有源码直接看源码分析算法（后端必须要有源码才能彻底知道）  
密文-没有源码1、猜识别 2、看前端JS（加密逻辑是不是在前端）  
#算法加密-概念&分类&类型  
1. 单向散列加密 -MD5   
单向散列加密算法的优点有(以MD5为例)：  
方便存储，损耗低：加密/加密对于性能的损耗微乎其微。  
单向散列加密的缺点就是存在暴力破解的可能性，最好通过加盐值的方式提高安全性，此外可能存在散列冲突。我们都知道MD5加密也是可以破解的。  
常见的单向散列加密算法有：  
MD5 SHA MAC CRC  
\*解密条件：密文即可，采用碰撞解密，几率看明文复杂程度  
  
2. 对称加密 -AES  
对称加密优点是算法公开、计算量小、加密速度快、加密效率高。  
缺点是发送方和接收方必须商定好密钥，然后使双方都能保存好密钥，密钥管理成为双方的负担。  
常见的对称加密算法有：  
DES AES RC4  
\*解密条件：密文及密钥偏移量，采用逆向算法解密，条件成立即可解密成功  
  
3. 非对称加密 -RSA  
非对称加密的优点是与对称加密相比，安全性更好，加解密需要不同的密钥，公钥和私钥都可进行相互的加解密。  
缺点是加密和解密花费时间长、速度慢，只适合对少量数据进行加密。  
常见的非对称加密算法：  
RSA RSA2 PKCS  
\*解密条件：密文和公钥或私钥，采用逆向算法解密，条件成立即可解密成功  
  
#加密解密-识别特征&解密条件  
MD5密文特点：  
1、由数字“0-9”和字母“a-f”所组成的字符串  
2、固定的位数 16 和 32位  
解密需求：密文即可，但复杂明文可能解不出  
  
BASE64编码特点：  
0、大小写区分，通过数字和字母的组合  
1、一般情况下密文尾部都会有两个等号，明文很少的时候则没有  
2、明文越长密文越长，一般不会出现"/""+"在密文中  
  
AES、DES密文特点：  
同BASE64基本类似，但一般会出现"/"和"+"在密文中  
解密需求：密文，模式，加密Key，偏移量，条件满足才可解出  
  
RSA密文特点：  
特征同AES,DES相似，但是长度较长  
解密需求：密文，公钥或私钥即可解出  
  
其他密文特点见：  
1.30余种加密编码类型的密文特征分析（建议收藏）  
https://mp.weixin.qq.com/s?\_\_biz=MzAwNDcxMjI2MA==&mid=2247484455&idx=1&sn=e1b4324ddcf7d6123be30d9a5613e17b&chksm=9b26f60cac517f1a920cf3b73b3212a645aeef78882c47957b9f3c2135cb7ce051c73fe77bb2&mpshare=1&scene=23&srcid=1111auAYWmr1N0NAs9Wp2hGz&sharer\_sharetime=1605145141579&sharer\_shareid=5051b3eddbbe2cb698aedf9452370026#rd  
  
2.CTF中常见密码题解密网站总结（建议收藏）  
https://blog.csdn.net/qq\_41638851/article/details/100526839  
  
3.CTF密码学常见加密解密总结（建议收藏）  
https://blog.csdn.net/qq\_40837276/article/details/83080460  
  
1、密码存储（后端处理）  
X3.2-md5&salt  
DZ对应代码段-/uc\_server/model/user.php  
 function add\_user() {  
 $password = md5(md5($password).$salt);  
 }  
   
<?PHP  
$h = 'd7192407bb4bfc83d28f374b6812fbcd';  
$hash=md5(md5('123456').'3946d5');  
if($h==$hash){  
 echo 'ok';  
}else{  
 echo 'no';  
}  
?>  
  
  
<?php  
function reverse\_md5\_hash\_with\_salt($final\_hash, $salt, $dictionary\_file) {  
 // 打开字典文件  
 $file = fopen($dictionary\_file, "r");  
 if (!$file) {  
 die("无法打开字典文件！");  
 }  
  
 // 遍历字典文件中的每一行（即每一个密码）  
 while (($password = fgets($file)) !== false) {  
 // 去除密码两侧的空白字符（包括换行符）  
 $password = trim($password);  
  
 // 第一次 MD5 加密密码  
 $md5\_hash = md5($password);  
  
 // 拼接 md5($password) 和 salt  
 $salted\_hash = $md5\_hash . $salt;  
  
 // 第二次 MD5 加密  
 $final\_attempt = md5($salted\_hash);  
  
 // 检查是否匹配  
 if ($final\_attempt === $final\_hash) {  
 fclose($file);  
 return $password; // 找到匹配的密码  
 }  
 }  
  
 // 如果没有找到匹配的密码  
 fclose($file);  
 return null;  
}  
  
// 给定的目标加密值和盐值  
$final\_hash = "125648dbf16531ab7e6b2f8ec8003ea7"; // 目标 MD5 值  
$salt = "3e3790"; // 盐值  
  
// 字典文件路径  
$dictionary\_file = "weaksauce.txt"; // 字典文件路径  
  
// 调用函数进行密码破解  
$password = reverse\_md5\_hash\_with\_salt($final\_hash, $salt, $dictionary\_file);  
  
// 输出结果  
if ($password) {  
 echo "破解成功，密码是: $password";  
} else {  
 echo "密码破解失败";  
}  
?>  
  
  
  
X3.5-hash  
DZ对应代码段-/uc\_server/model/user.php  
 function add\_user() {  
 $salt = '';  
 $password = $this->generate\_password($password);  
 }  
  
 function generate\_password($password) {  
 $algo = $this->get\_passwordalgo();  
 $options = $this->get\_passwordoptions();  
 $hash = password\_hash($password, $algo, $options);  
 }  
  
<?PHP  
$hash = '$2y$10$KA.7VYVheqod8F3X65tWjO3ZXfozNA2fC4oIZoDSu/TbfgKmiw7xO';  
if (password\_verify('123456', $hash)) {  
 echo 'ok';  
} else {  
 echo 'error';  
}  
?>  
  
  
<?PHP  
$hash = '$2y$10$PDACNCRyZzcsknF8zvL4yu7YHIPQTN8F635PxQeXSB8QxxDZSXrd.';  
  
  
$dictionary\_file = "weaksauce.txt";  
$file = fopen($dictionary\_file, "r");  
 if (!$file) {  
 die("无法打开字典文件！");  
 }  
  
 // 遍历字典文件中的每一行（即每一个密码）  
 while (($password = fgets($file)) !== false) {  
 // 去除密码两侧的空白字符（包括换行符）  
 $password = trim($password);  
 echo $password."<br>";  
   
 if (password\_verify($password, $hash)) {  
 echo 'ok';  
 } else {  
 echo 'error';  
 }   
 }  
?>  
  
2、数据通讯  
-博客登录-zblog（前端处理）  
<script src="script/md5.js" type="text/javascript"></script>  
$("#btnPost").click(function(){  
 var strPassWord=$("#edtPassWord").val();  
 $("form").attr("action","cmd.php?act=verify");  
 $("#password").val(MD5(strPassWord));  
console.log(MD5("123456"));  
  
-博客登录-混合加密（前端处理）  
<script src="/Scripts/Vip/Login.js?v=20241202154949"></script>  
function Login() {  
 logindata.UserName = encodeURI(encrypt.encrypt(numMobile));  
 logindata.Mobile = encodeURI(encrypt.encrypt(numMobile));;  
 logindata.Password = encodeURI(encrypt.encrypt(numPassword));  
}  
var encrypt = new JSEncrypt();  
encodeURI(encrypt.encrypt('13554365566'));  
  
3、密文  
明确以下三种加密算法加解密条件  
对称AES/DES加解密，非对称RSA加解密  
解密：http://tool.chacuo.net/cryptdes  
  
应用场景：  
1、发送数据的时候自动将数据加密发送（自需加密即可）  
安全测试思路：我们需要将我们的Payload也要加密发送过去，这样才符合正常的业务逻辑，所以我们就只需要调用应用的JS加密逻辑进行提交发送测试即可！  
2、比如要得到数据的明文（必须要拿到解密算法）  
由于各种算法的解密条件不一，密钥，偏移量，私钥等不一定能拿到。  
  
对于后续安全测试影响较大的：  
1、目标：API接口 前后端分离应用居多  
2、漏洞发现：未授权 各类漏洞测试 爆破等  
3、其他

涉及资源：[资源下载地址](https://docs.qq.com/doc/DQ3Z6RkNpaUtMcEFr)