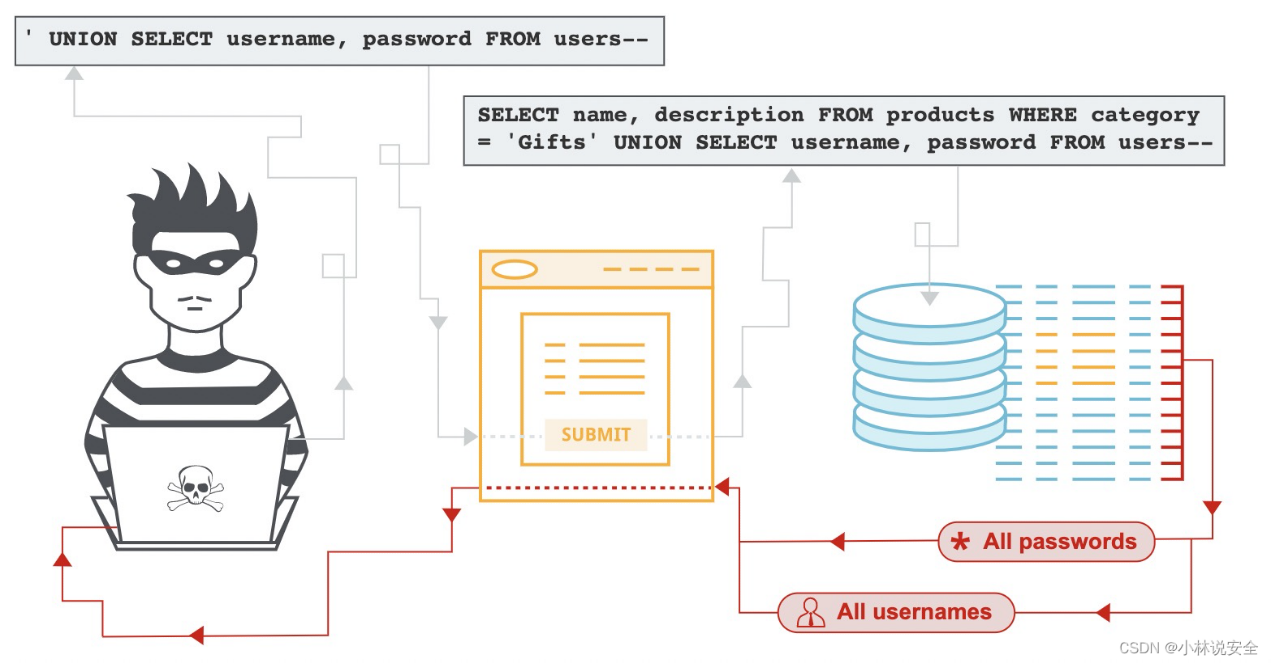
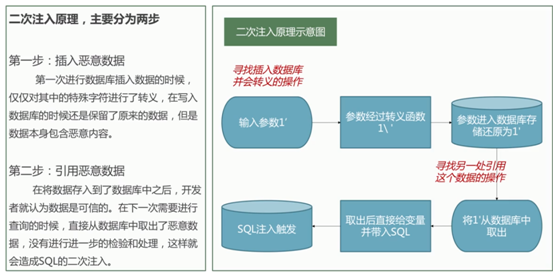
Web攻防-SQL注入&二次攻击&堆叠执行&SQLMAP&Tamper编写&指纹修改&分析调试

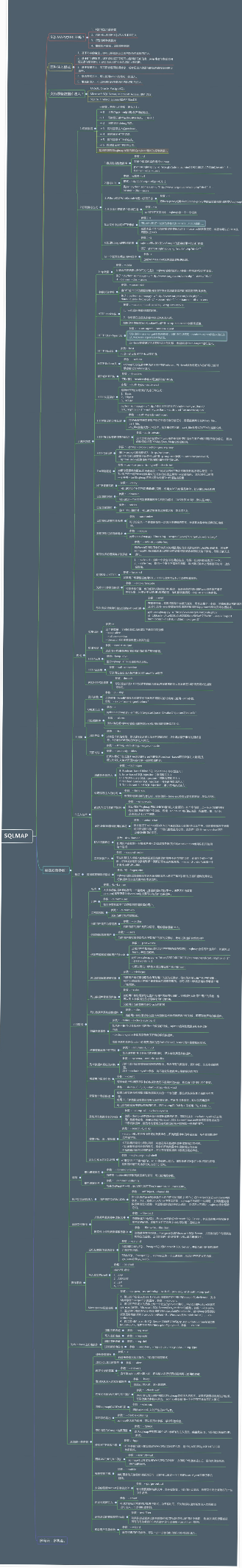


#知识点：  
1、Web攻防-SQL注入-产生原理&应用因素  
2、Web攻防-SQL注入-各类数据库类型利用  
----------------------------------------------  
1、Web攻防-SQL注入-参数类型\*参数格式  
2、Web攻防-SQL注入-XML&JSON&BASE64等  
3、Web攻防-SQL注入-数字字符搜索等符号绕过  
----------------------------------------------  
1、Web攻防-SQL注入-操作方法&增删改查  
2、Web攻防-SQL注入-HTTP头&UA&Cookie  
3、Web攻防-SQL注入-HTTP头&XFF&Referer  
----------------------------------------------  
1、Web攻防-SQL注入-操作方法&增删改查  
2、Web攻防-SQL注入-布尔&延时&报错&盲注  
----------------------------------------------  
1、Web攻防-SQL注入-高权限用户差异  
2、Web攻防-SQL注入-跨库&文件读写带外  
----------------------------------------------  
1、Web攻防-SQL注入-堆叠&二次注入  
2、Web攻防-SQL注入-SQLMAP进阶使用  
   
#章节点：(待补充)  
OWTOP10，SQL注入，文件安全（上传,下载,读取,删除,包含等），目录遍历，  
CSRF，SSRF，CRLF，SSTI注入，XML&XXE安全，RCE执行（代码或命令等）  
反序列化，业务逻辑（验证码，接口枚举，支付购买，机制验证，越权，并发等）  
未授权访问，失效访问控制，弱口令安全，第三方组件安全，CORS，JSONP等









演示案例：

* WEB攻防-SQL注入-数据库类型利用
* WEB攻防-SQL注入-参数类型&符号干扰
* WEB攻防-SQL注入-参数格式&参数编码
* WEB攻防-SQL注入-增删改查&无回显报错
* WEB攻防-SQL注入-HTTP头&UA&Cookie
* WEB攻防-SQL注入-HTTP头&XFF&Referer
* WEB攻防-SQL注入-布尔&延时&报错&盲注
* WEB攻防-SQL注入-高权限&跨库&文件读写
* WEB攻防-SQL注入-高权限&文件读写&带外
* WEB攻防-SQL注入-堆叠注入&二次注入
* WEB攻防-SQL注入-SQLMAP&Tamper使用

#数据库知识：  
1、数据库名，表名，列名，数据  
2、自带数据库，数据库用户及权限  
3、数据库敏感函数，默认端口及应用  
4、数据库查询方法（增加删除修改更新）  
   
#SQL注入产生原理：  
代码中执行的SQL语句存在可控变量导致  
   
#影响SQL注入的主要因素：  
1、数据库类型（权限操作）  
2、数据操作方法（增删改查）  
3、参数数据类型（符号干扰）  
4、参数数据格式（加密编码等）  
5、提交数据方式（数据包部分）  
6、有无数据处理（无回显逻辑等）  
   
#常见SQL注入的利用过程：  
1、判断数据库类型  
2、判断参数类型及格式  
3、判断数据格式及提交  
4、判断数据回显及防护  
5、获取数据库名，表名，列名  
5、获取对应数据及尝试其他利用  
   
#黑盒/白盒如何发现SQL注入  
1、盲对所有参数进行测试  
2、整合功能点脑补进行测试  
白盒参考后期代码审计课程  
   
利用过程：  
获取数据库名->表名->列名->数据（一般是关键数据，如管理员）  
   
案例说明：  
在应用中，数据库用户不同，可操作的数据库和文件读写权限不一，所有在注入过程中可以有更多的利用思路，如直接写入后门，获取数据库下洽谈网站的数据等。  
   
#堆叠注入  
堆叠注入触发的条件很苛刻，因为堆叠注入原理就是通过结束符同时执行多条sql语句，例如php中的mysqli\_multi\_query函数。与之相对应的mysqli\_query()只能执行一条SQL，所以要想目标存在堆叠注入,在目标主机存在类似于mysqli\_multi\_query()这样的函数,根据数据库类型决定是否支持多条语句执行.  
1、目标存在sql注入漏洞  
2、目标未对";"号进行过滤  
3、目标中间层查询数据库信息时可同时执行多条sql语句  
支持堆叠数据库：MYSQL MSSQL Postgresql等  
   
#二次注入  
原理：注册（写入payload）-登录后修改-修改时以当前用户名为条件触发  
黑盒思路：分析功能有添加后对数据操作的地方（功能点）  
白盒思路：insert后进入select或update的功能的代码块  
注入条件：插入时有转义函数或配置，后续有利用插入的数据  
   
#SqlMap使用  
#参考：<https://www.cnblogs.com/bmjoker/p/9326258.html>  
   
#数据猜解-库表列数据&字典  
测试：常规数据获取  
--current-db  
--tables -D ""  
--columns -T "" -D ""  
--dump -C "" -C "" -T ""  
   
#权限操作-文件&命令&交互式  
测试：高权限操作  
引出权限：  
--is-dba --privileges  
引出文件：  
--file-read --file-write --file-dest   
引出命令：  
--os-cmd= --os-shell --sql-shell  
   
#提交方法-POST&HEAD&JSON  
测试：Post Cookie Json等  
--data ""  
--cookie ""  
-r 1.txt  
   
#绕过模块-Tamper脚本-使用&开发  
测试：base64+json注入&再加有过滤的注入  
--tamper=base64encode.py  
--tamper=test.py  
from lib.core.convert import encodeBase64  
from lib.core.enums import PRIORITY  
   
\_\_priority\_\_ = PRIORITY.LOW  
   
def dependencies():  
 pass  
   
def tamper(payload, \*\*kwargs):  
 if payload:   
 payload = payload.replace('SELECT','sElEct')  
 payload = payload.replace('select','sElEct')  
 payload = payload.replace('OR','Or')  
 payload = payload.replace('or','Or')  
 payload = payload.replace('AND','And')  
 payload = payload.replace('and','And')  
 payload = payload.replace('XOR','xOr')  
 payload = payload.replace('xor','xOr')  
 payload = payload.replace('SLEEP','SleeP')  
 payload = payload.replace('sleep','SleeP')  
 payload = payload.replace('ELT','Elt')  
 return encodeBase64(payload, binary=False) if payload else payload  
   
   
#分析拓展-代理&调试&指纹&风险&等级  
1、后期分析调试：  
-v=(0-6) #详细的等级(0-6)  
--proxy "http://xx:xx" #代理注入  
   
2、打乱默认指纹：  
绕过流量设备识别sqlmap  
--user-agent "" #自定义user-agent  
--random-agent #随机user-agent  
--time-sec=(2,5) #延迟响应，默认为5  
   
3、使用更多的测试：测试Header注入  
--level=(1-5) #要执行的测试水平等级，默认为1   
--risk=(0-3) #测试执行的风险等级，默认为1

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