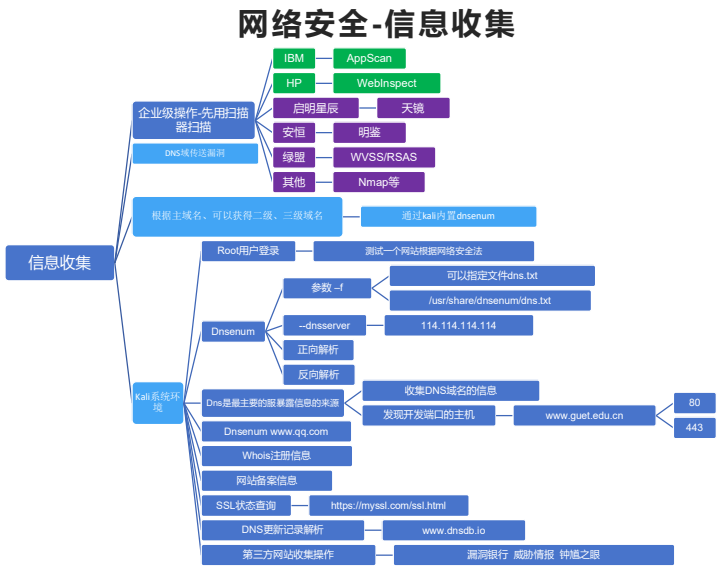


NMAP脚本-脚本目录

脚本目录cd /usr/share/nmap/scripts

暴力破解方式nmap --script=brute 10.60.69.228

NMAP脚本
https://nmap.org/nsedoc/

- auth: 负责处理鉴权证书（绕开鉴权）的脚本
- broadcast:在局域网内探查更多服务开启状况
- brute: 提供暴力破解方式，针对常见的应用如http/snmp等
- default: 使用-sC或-A选项扫描时候默认的脚本，提供基本脚本扫描能力
- discovery: 对网络进行更多的信息，如SMB枚举、SNMP查询等
- dos: 用于进行拒绝服务攻击
- exploit: 利用已知的漏洞入侵系统
- external: 利用第三方的数据库或资源，例如进行whois解析
- fuzzer: 模糊测试的脚本，发送异常的包到目标机，探测出潜在漏洞 intrusive: 入侵性的脚本，此类脚本可能引发对方的IDS/IPS的记录或屏蔽
- malware: 探测目标机是否感染了病毒、开启了后门等信息
- safe: 此类与intrusive相反，属于安全性脚本
- version: 负责增强服务与版本扫描（Version Detection）功能的脚本
- vuln: 负责检查目标机是否有常见的漏洞（Vulnerability），如是否有MS08_06

nmap脚本存放位置：/usr/share/nmap/scripts

脚本简介

1.nmap --script=auth [ip]

通权☐ 扫描：使用 --script=auth 可以对目标主机或目标主机所在的网段应用身份认证。

2.nmap --script=brute [ip]

暴力破解攻击： nmap☐ 具有暴力破解的功能，可对数据库，smb,snmp等进行简单密码的暴力猜解。

3.nmap --script=vuln [ip]

扫描常见的漏洞：nmap具备漏洞扫描的功能，可以检查目标主机或网段是否存在常见的漏洞。

4.nmap --script=realvnc-auth-bypass [ip]

应用服务扫描：nmap具备很多常见应用服务的扫描脚本，例如VNC服务，mysql服务，telnet服务，rsync服务等，此处以vnc服务为例。

5.nmap --script=broadcast [ip]

广播探测：探测同域网内更多服务开启的情况。

6.nmap --script=external [域名]

用于第三方服务，例如whois解析。

whois解析：利用第三方的数据库或资源查询目标地址的信息

7.nmap --script=default

默认，执行脚本（-sC）。

8.map --script=discovery

与主机和服务发现相关的脚本。

9.map --script=dos

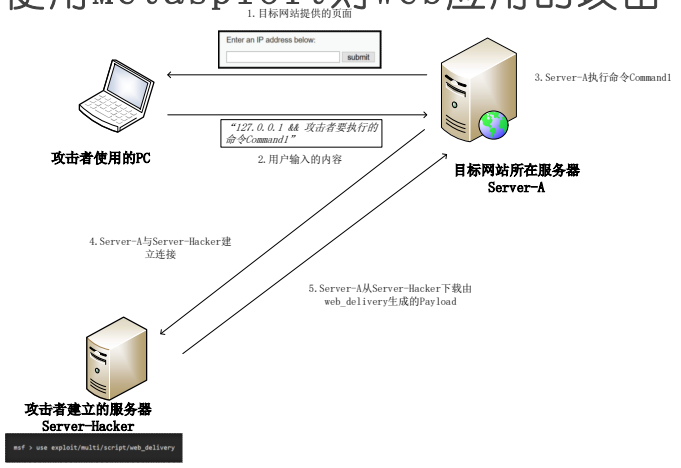
与拒绝服务攻击有关的脚本。

10.map --script=exploit

利用安全漏洞。

11.map --script=fuzzer

7.6使用Metasploit对Web应用的攻击



漏洞扫描-缓冲区溢出简介

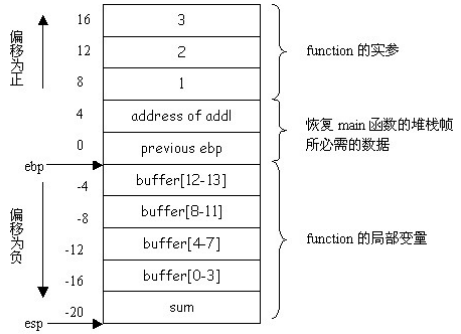
- 缓冲区溢出-Buffer Overflow--缓冲区就是应用程序用来保存用户输入数据、程序临时数据的内存空间-数组 。
- 如果用户输入的数据长度超出了程序为其分配的内存空间，这些数据就会覆盖程序为其它数据分配的内存空间，形成所谓的缓冲区溢出。
- 通过修改某些内存区域，把一段恶意代码存储到一个buffer中,并且使这个buffer被溢出，以便当前进程被非法利用(执行这段恶意的代码)
- 如果覆盖缓冲区的是一段精心设计的机器指令序列，可能通过溢出，改变返回地址，将其指向自己的指令序列，从而改变该程序的正常流程。
- 溢出漏洞发掘起来需要较高的技巧和知识背景，一旦编写出溢出代码，则用起来非常简单。

漏洞扫描-缓冲区溢出简介

函数调用发生时，新的堆栈帧被压入堆栈；当函数返回时，相应的堆栈帧从堆栈中弹出。

```
int function(int a, int b, int c)
{
char buffer[14];
int sum;
sum = a + b + c;
return sum;
}

void main()
{
int i;
i = function(1,2,3);
}
```




```
#include <stdio.h>
#include <string.h>
void foo(const char* input)
{
  char buf[10];
  printf("my stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");//打印堆栈的值
  strcpy(buf,input);
  printf("%s\n",buf);//接受输入并拷贝到缓冲区
  printf("now my stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
}
void bar()
{
    printf("Augh!I've been hacked\n");
}
 int main(int argc,char*argv[])
{
        char str[50];
        scanf("%s",str);
        printf("address of foo=%p\n",foo);   //打印函数的地址
       printf("address of bar=%p\n",bar);
       foo(str);
       return 0;
}
```