主机扫描那点事儿

Aug 7, 2017

主机扫描那点事儿

0x00 资产扫描、汇总、实时监控

0X01 解决方案

0X02 Nmap简介、目录结构、扫描流程、nse Engine

0X03 一个简单的扩展打开世界

0X04 扩展结构设计

0X05 More ideas

0X06 共同探讨

0x00 资产扫描、汇总、实时监控

资产扫描能够有利于企业内部查看终端、监控终端、对终端进行安全加固。周期性的扫描能有效快速修补漏洞、降低办公网络风险。

如何进行汇总、实时监控？

在我们要进行汇总的时候，有如下几个可以考虑的方案。

- PDF
- Excel
- Text
- Database / SQL

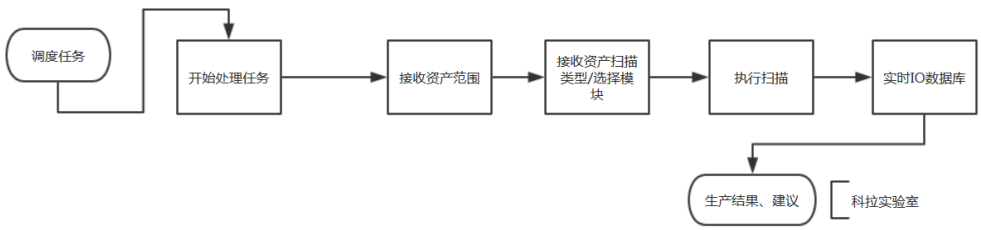
PDF && Excel && Text 都不适合实时View

Database / SQL 有利于生成数据汇总、图表，并且可移植性很高。

Database -> Web -> Excel/PDF .... 可行性都变得高了起来

实时监控采用任务调度，数据库采用IO效率高的NO SQL产品，详细信息采用普通的数据库：MySQL、SQL Server、Oracel...

0X01 解决方案



Nmap简介、目录结构、扫描流程、Nse Engine

0X02 简介

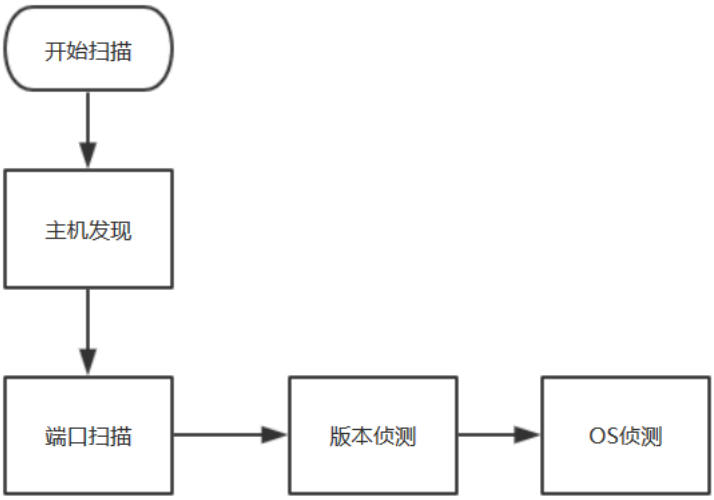
Nmap (“Network Mapper(网络映射器)”) 是一款开放源代码的 网络探测和安全审核的工具。它的设计目标是快速地扫描大型网络，当然用它扫描单个 主机也没有问题。Nmap以新颖的方式使用原始IP报文来发现网络上有哪些主机，那些 主机提供什么服务(应用程序名和版本)，那些服务运行在什么操作系统(包括版本信息)， 它们使用什么类型的报文过滤器/防火墙，以及一堆其它功能。虽然Nmap通常用于安全审核，许多系统管理员和网络管理员也用它来做一些日常的工作，比如查看整个网络的信息， 管理服务升级计划，以及监视主机和服务的运行。

目录结构

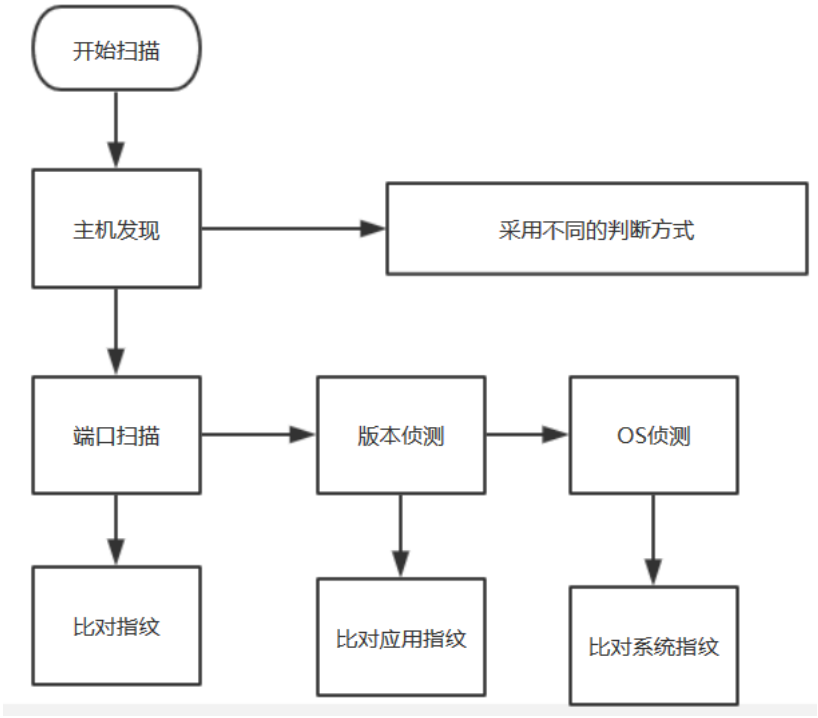
```
root@debian:/usr/share/nmap# tree -L 1
.
├── nmap.dtd
├── nmap-mac-prefixes
├── nmap-os-db
├── nmap-payloads
├── nmap-protocols
├── nmap-rpc
├── nmap-service-probes
├── nmap-services
├── nmap.xsl
├── nselib
├── nse_main.lua
└── scripts

2 directories, 10 files
root@debian:/usr/share/nmap#
```

文件名称	文件说明
nmap.dtd	Nmap输出的XML格式内部变量的定义
nmap-mac-prefixes	是Nmap针对不同终端的MAC地址所收集的指纹（常用于内网扫描）
nmap-os-db	Nmap针对不同终端的操作系统返回的数据包特征所收集的指纹
nmap-payloads	是Nmap在扫描时将payload向扫描目标发送的数据
nmap-protocols	Nmap 用来存储目标端口对应服务描述的db文件
nmap-rpc	Nmap在扫描的时候调用RPC进行服务发现的db文件
nmap-service-probes	Nmap针对响应数据包内容进行正则匹配从而判断服务的db文件
nmap-services	Nmap存储一个TCP/UDP服务的db文件
nmap-xsl	Nmap导出xml文件的模板
nselib	Nmap的脚本引擎扩展库
nse_main.lua	在调用任何Nmap脚本都会提前自动调用的预处理Lua脚本
Scripts	Nmap的脚本扩展

扫描流程 V1



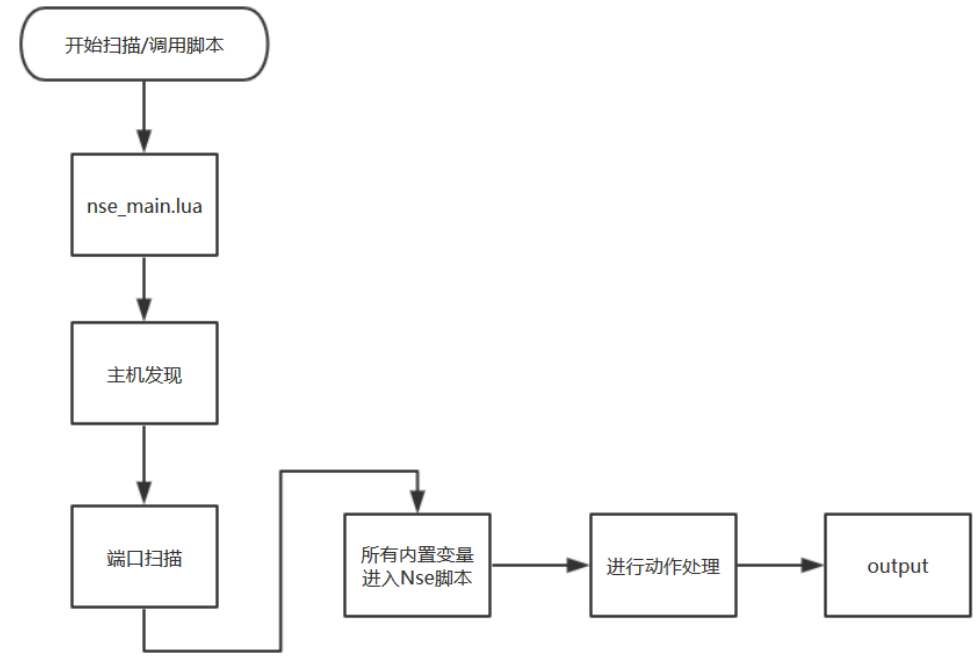
扫描流程 V2



Nse Engine（Nmap 脚本引擎）

Nmap Nse 脚本引擎用于针对发现的OS、主机、端口进行不同的操作，例如：Fuzz测试、漏洞发现、漏洞利用等。这对Nmap又增添了一大亮点，所以说Nmap不只是一个扫描工具，在黑客的手中，更是一款爱不释手的渗透工具。

Nse Engine的执行流程



0X03 一个简单的扩展打开世界

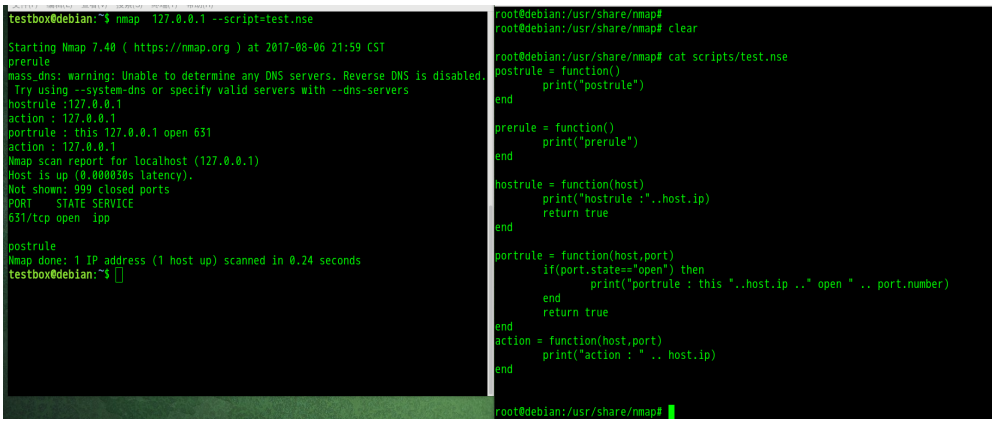
https://nmap.org/book/nse-api.html

Nmap扩展主要由以下几个变量构成。编码方式：变量绑定函数

变量名称	函数执行顺序
prerule()	最先执行
hostrule(host)	第二步执行
portrule(host,port)	第二步执行
postrule()	最后一步执行

顺序为： `Prerule -> Hostrule or Portrule -> Action -> Postrule`

当 `Hostrule` 或者 `Portrule` 的绑定函数返回 `true` 的时候，都会执行一次 `Action` 的绑定函数。



```
testbox@debian:~$ nmap  127.0.0.1 --script=test.nse

Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-06 21:59 CST
prerule
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
hostrule :127.0.0.1
action : 127.0.0.1
portrule : this 127.0.0.1 open 631
action : 127.0.0.1
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000030s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
631/tcp open  ipp

postrule
Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
testbox@debian:~$
```

```
root@debian:/usr/share/nmap#
root@debian:/usr/share/nmap# clear

root@debian:/usr/share/nmap# cat scripts/test.nse
postrule = function()
        print("postrule")
end

prerule = function()
        print("prerule")
end

hostrule = function(host)
        print("hostrule :"..host.ip)
        return true
end

portrule = function(host,port)
        if(port.state=="open") then
                print("portrule : this "..host.ip .." open " .. port.number)
        end
        return true
end
action = function(host,port)
        print("action : " .. host.ip)
end

root@debian:/usr/share/nmap#
```

入库

获取参数->连接数据库 -> 采用不同条件判断期望获取到的值->执行SQL

获取参数：stdnse lib [stdnse.get_script_args()]

连接数据库：mysql lib

获取socket : nmap socket lib

socket = nmap.new_socket() – 获得一个新的socket套接字

socket:set_timeout(1000) – 设置超时时间

socket:connect(host_ip,port_number) – 连接目标

socket:close() – 关闭套接字连接

我自己根据API封装了一个执行MySQL Query的扩展：

```
local mysql = require "mysql"
local nmap = require "nmap"
local stdnse = require "stdnse"

description =[[
This is a Lua script that performs the MySQL statement.
]]
author = "China CoraLab payloads"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"external"}

socket = nmap.new_socket()
socket:set_timeout(1000)
try = nmap.new_try(function() socket:close() end)
optionsConfig = {
    username = "root",
    password = "root",
    database = "test",
    sqlQuery = "SELECT MD5('admin'),",
    host     = "127.0.0.1",
    port     =  3306
}


optionsConfig.username = stdnse.get_script_args(SCRIPT_NAME .. ".username") or optionsConfi
optionsConfig.password = stdnse.get_script_args(SCRIPT_NAME .. ".password") or optionsConfi
optionsConfig.database = stdnse.get_script_args(SCRIPT_NAME .. ".database") or optionsConfi
optionsConfig.host = stdnse.get_script_args(SCRIPT_NAME .. ".host") or optionsConfig.host
optionsConfig.port = stdnse.get_script_args(SCRIPT_NAME .. ".port") or optionsConfig.port
optionsConfig.sqlQuery = stdnse.get_script_args(SCRIPT_NAME .. ".sql") or optionsConfig.sql

function mysqlLogin(socket, username, password)
    local status, response = mysql.receiveGreeting( socket )
    if ( not(status) ) then
        return response
    end
    return mysql.loginRequest( socket, { authversion = "post41", charset = response.charset
end
portrule = function (host,port)
    return true
end
action = function(host,port)
    if(socket:connect(optionsConfig.host,optionsConfig.port))
    then
        local status,response = mysqlLogin(socket,optionsConfig.username,optionsConfig.pass
        if(status)
        then
            local status, rs = mysql.sqlQuery( socket, optionsConfig.sqlQuery )
            socket:close()
            local result = mysql.formatResultset(rs, { noheaders = true })
            stdnse.debug(string.format("[*]Query is %s | Result is  %s",optionsConfig.sqlQu
            return stdnse.format_output(true,result)
        end
    else
        print("Connect to mysql Failed !!!")
        return nil
    end
end
```

`stdnse.get_script_args` 是用于获取参数的函数，对应Nmap的–script-args参数。

遇到的问题

在Nse扩展库中，无法与外部地址(除去扫描范围以外的目标)进行Socket连接。

因为大部分函数都接收绑定函数中的Host与Port参数，他们是一个table数据类型。

于是我翻阅API扩展，发现了一个方法可以直接获取一个IP+端口的套接字：

```
socket = nmap.new_socket()
socket:set_timeout(1000)
socket:connect(HOST_ADDRESS,PORT_NUMBER)
```

如上方法我们可以直接传递IP地址与端口号就可以获得套接字了。

0X04 扩展结构设计

nse_main.lua 是声明全局变量、装载script db的预处理脚本，我们可以在其中将连接数据库的行为载入，然后用portrule函数进行数据库的读写。

> nmap –script=scan_save_database 127.0.0.1

执行流程：

- nse_main.lua 装载
- 寻找 scan_save_database 脚本
- 主机发现
- 端口扫描
- 版本侦测
- 系统指纹侦测
- ....
- 执行prerule函数
- 执行hostrule函数（如果返回true，执行action）
- 执行portrule函数（如果返回true，执行action）
- 执行postrule函数

主要是在扩展脚本中的portrule or hostrule 过滤我们的想要的数据，最后再action中对数据库进行读写。

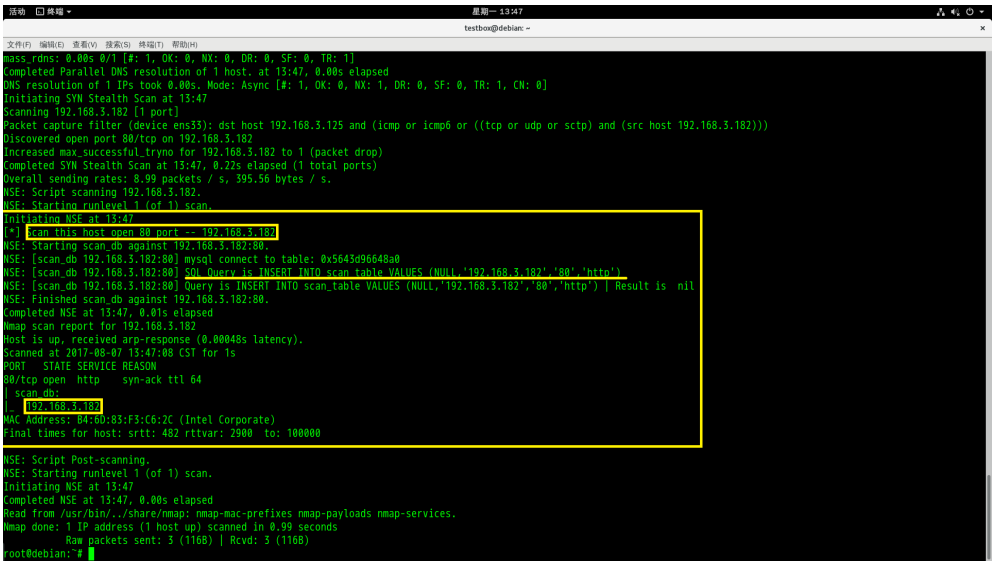
下面演示一下获取开启80端口的所有主机：

```
local mysql = require "mysql"
local nmap = require "nmap"
local stdnse = require "stdnse"
local shortport = require "shortport"
description =[[
This is a Lua script that performs the MySQL statement.
]]
author = "China CoraLab payloads"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"external"}

socket = nmap.new_socket()
socket:set_timeout(1000)
try = nmap.new_try(function() socket:close() end)
optionsConfig = {
    username = "root",
    password = "root",
    database = "test",
    sqlQuery = "INSERT INTO scan_table VALUES (NULL,'%s','%s','%s')",
    host     = "127.0.0.1",
    port     =  3306
}


optionsConfig.username = stdnse.get_script_args(SCRIPT_NAME .. ".username") or optionsConfi
optionsConfig.password = stdnse.get_script_args(SCRIPT_NAME .. ".password") or optionsConfi
optionsConfig.database = stdnse.get_script_args(SCRIPT_NAME .. ".database") or optionsConfi
optionsConfig.host = stdnse.get_script_args(SCRIPT_NAME .. ".host") or optionsConfig.host
```

```
    optionsConfig.port = stdnse.get_script_args(SCRIPT_NAME .. ".port") or optionsConfig.port
    optionsConfig.sqlQuery = stdnse.get_script_args(SCRIPT_NAME .. ".sql") or optionsConfig.sql

    function mysqlLogin(socket, username, password)
        local status, response = mysql.receiveGreeting( socket )
        if ( not(status) ) then
            return response
        end
        return mysql.loginRequest( socket, { authversion = "post41", charset = response.charset
    end
    portrule=function(host,port)
            if(port.number==80) then
                    print("[*] Scan this host open 80 port -- " .. host.ip)
                    return true
            end
    end
    --
    -- portrule = shortport.portnumber({80,443,8080},"tcp",{"open","open|filtered"})
    -- portrule = shortport.port_or_service({80,443,8080},"http","tcp",{"open","open|filtered"}
    -- portrule = service("http","tcp",{"open"})
    --
    action=function(host,port)
            -- load mysql lib
            -- get scan info
            optionsConfig.sqlQuery = string.format("INSERT INTO scan_table VALUES (NULL,'%s','%
            if(socket:connect(optionsConfig.host,optionsConfig.port))
        then
            local status,response = mysqlLogin(socket,optionsConfig.username,optionsConfig.pass
            if(status)
            then
                local status, rs = mysql.sqlQuery( socket, optionsConfig.sqlQuery )
                socket:close()
                local result = mysql.formatResultset(rs, { noheaders = true })
                stdnse.debug(string.format("Query is %s | Result is  %s",optionsConfig.sqlQuery
                return stdnse.format_output(true,host.ip)
            end
        else
            print("Connect to mysql Failed !!!")
            return nil
        end
    end
```

执行扫描： `nmap --script=scan_db --script-args username=root 192.168.3.183 -d -p 80`

执行过程：



不开启debug...

```
root@debian:~# nmap --script=scan_db --script-args username=test,password=123456,host=192.168.3.182 192.168.3.182  -p 80

Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-07 13:51 CST
[*] Scan this host open 80 port -- 192.168.3.182
Nmap scan report for 192.168.3.182
Host is up (0.00042s latency).
PORT   STATE SERVICE
80/tcp open  http
| scan_db:
|_  192.168.3.182
MAC Address: B4:6D:83:F3:C6:2C (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 0.80 seconds
root@debian:~#
```

结果：

```
Packet capture filter (device ens33): dst host 10.10.10.125 and (icmp or icmp6 or ((tcp or udp or sctp) and (src host 10.
Discovered open port 80/tcp on 10.10.10.1
NSE: MySQL [(none)]> use test;
NSE: Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
NSE: Database changed
MySQL [test]> select * from scan_table;
+----+------------+------+---------+
| id | ip         | port | service |
+----+------------+------+---------+
|  1 | 10.10.10.1 | 80   | http    |
|  2 | 10.10.10.1 | 80   | http    |
|  3 | 10.10.10.1 | 80   | http    |
|  4 | 10.10.10.1 | 80   | http    |
|  5 | 10.10.10.1 | 80   | http    |
|  6 | 10.10.10.1 | 80   | http    |
|  7 | 10.10.10.1 | 80   | http    |
|  8 | 10.10.10.1 | 80   | http    |
|  9 | 10.10.10.1 | 80   | http    |
| 10 | 10.10.10.1 | 80   | http    |
| 11 | 10.10.10.1 | 80   | http    |
+----+------------+------+---------+
11 rows in set (0.01 sec)
MySQL [test]>

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 09:39
Completed NSE at 09:39, 0.00s elapsed
Read from /usr/bin/../share/nmap: nmap-mac-prefixes nmap-payloads nmap-services.
Nmap done: 1 IP address (1 host up) scanned in 14.80 seconds
           Raw packets sent: 3 (116B) | Rcvd: 3 (116B)
root@debian:~#
```

testbox@debian: ~

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

0X05 More ideas

更多插件...

实用的函数手册...

代码规范

共同维护

有效的任务调度

0x06 共同探讨

倾旋 Email:payloads@aliyun.com

倾旋

安全从业者，ShadowTeam 成员，从事安全产品开发，渗透测试，Web漏洞研究。

Email

Newest Posts

- 主机扫描那点事儿
- 端口转发工具小结
- docker install openvas
- How to write sqlmap tamper script?
- 光阴是酒，醉了来人

TOC