Testes de Penetração: Explorador de Portas

Segurança da Informação

Charles Tim Batista Garrocho

Instituto Federal de São Paulo – IFSP
Campus Campos do Jordão

garrocho.ifspcjo.edu.br/SEGA6

charles.garrocho@ifsp.edu.br

Curso Superior de TADS

O Worm Morris

6000 estações de trabalho foram **infectadas** (10% de todos os computadores).

Entre 10 e 100 milhões de dólares para **erradicar** os danos.

Então, como isso funcionou?

Aproveitou de diversas **vulnerabilidades** de programas e serviços padrões que eram executados nas estações de trabalho daquela época.

Utilizou de listas de nomes e usuários **comuns**.

Atualmente, é possível fazer o que Morris fez?

Explorador de Portas – Contextualização

Reconhecimento serve como o primeiro passo em qualquer bom ataque cibernético.

Um invasor deve descobrir onde as **vulnerabilidades** estão antes de selecionar e escolher explorações para um alvo.

A seguir, vamos construir um pequeno script de reconhecimento que verifique um host alvo para **portas TCP abertas**. No entanto, para interagir com as portas TCP, precisamos primeiro construir sockets TCP.

Python, fornece acesso à interface de soquete. Através de uma série de funções de API de soquete, podemos **criar, ligar, ouvir, conectar ou enviar** tráfego em soquetes TCP / IP.

Explorador de Portas – Contextualização

A maioria dos aplicativos acessíveis à Internet **residem no TCP**. Por exemplo, em uma organização de destino, o servidor da Web pode residir na porta TCP 80, no servidor de e-mail na porta TCP 25 e no servidor de transferência de arquivos na porta TCP 21.

Para se conectar a qualquer um desses serviços, um invasor deve conhecer tanto o endereço do protocolo Internet quanto a porta TCP **associada ao serviço**. Embora alguém familiarizado com a organização tenha acesso a essa informação, um invasor não pode.

Um tipo de varredura de porta inclui o envio de um pacote TCP SYN para uma série de portas comuns e aguardando uma resposta TCP ACK que resultará na sinalização de uma porta aberta. Em contraste, um TCP Connect Scan usa o handshake de três vias completo para determinar a **disponibilidade** do serviço ou da porta.

Explorador de Portas – Implementação

O script aceitará o nome do host e a(s) porta(s) alvo. Para isso, será utilizado a biblioteca **optparse** para analisar as opções de entrada.

```
37 def inicio():
38     analisador = optparse.OptionParser('use explorador_portas '+\
39       '-H <host alvo> -p <porta(s) alvo>')
40     analisador.add_option('-H', dest='host', type='string',\
41       help='especifique o host alvo')
42     analisador.add_option('-p', dest='porta', type='string',\
43       help='especifique a porta[s] alvo separadas por virgula')
44
45     (opcoes, args) = analisador.parse_args()
46
47     host = opcoes.host
48     porta = str(opcoes.porta).split(',')
49
50     if (host == None) | (porta[0] == None):
51         print analisador.usage
52         exit(0)
53
54     portaScan(host, porta)
```

Explorador de Portas – Implementação

A função **portaScan** irá resolver o endereço IP do host passado como argumento, e iniciar threads para explorar as portas com **conexaoScan**.

```
19 def portaScan(host, portas):
20     try:
21         IPSite = gethostbyname(host)
22     except:
23         print "[-] Nao conseguiu resolver o host '%s'" % host
24         return
25
26     try:
27         nomeSite = gethostbyaddr(IPSite)
28         print '\n[+] Resultados para: ' + nomeSite[0]
29     except:
30         print '\n[+] Resultados para: ' + IPSite
31
32     setdefaulttimeout(1)
33     for porta in portas:
34         t = Thread(target=conexaoScan,args=(host,int(porta)))
35         t.start()
```

Explorador de Portas – Implementação

A função **conexaoScan** irá tentar estabelecer uma conexão com o host e porta definidos. Se bem sucessido, irá imprimir uma mensagem de porta aberta, caso contrário, de porta fechada.

```
 7 def conexaoScan(host, porta):
 8     try:
 9         soquete = socket(AF_INET, SOCK_STREAM)
10         soquete.connect((host, porta))
11         soquete.send('Segurança da Informação\r\n')
12         resultados = soquete.recv(100)
13         print '[+] %d/tcp aberta' % porta
14     except:
15         print '[-] %d/tcp fechada' % porta
16     finally:
17         soquete.close()
```

Explorador de Portas – Executando

O script abaixo explora as portas 22, 25, e 80 do github.com. É analisado se as portas de serviços de SSH, FTP e HTTP estão abertas.

Lista de Portas TCP e UDP: `https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`

```
tim@charles:~$ python explorador_portas.py -H github.com -p 22,25,80

[+] Resultados para: lb-192-30-253-112-iad.github.com
[+] 22/tcp aberta
[+] 80/tcp aberta
[-] 25/tcp fechada
tim@charles:~$
```

Atividades

Faça uma busca na Internet sobre os protocolos TCP e UDP, os serviços que executam nestes protocolos e suas respectivas portas.

Faça uma busca na Internet e encontre versões de serviços que tem vulnerabilidades.

Utilize o script e encontre portas abertas de algum alvo na Internet.

Modifique o script de forma que ele analise um intervalo de portas determinado pelo usuário. Por exemplo:
python explorador_portas.py -H github.com -p 1,1024