****

**Visão**

Com a crescente demanda sobre Tecnologias, percebemos que muitas pessoas apesar de buscarem informações, não possuem fontes que queiram realmente passar o conhecimento da maneira como ela deve ser, livre e com embasamento técnico que permita ser aplicado e utilizado quando necessário, além de serem testados em sua criação, tornando esta informação útil e confiável.

**Missão**

O Laboratório foi criado com a intenção de buscar e disseminar o conhecimento de uma maneira clara e objetiva, de forma gratuita, auxiliando na evolução dos membros e da sociedade na qual estas informações são compartilhadas, buscando o crescimento de todos os envolvidos nesta criação de valores.

**Licença**



Figura 01 – Licença Criative Commons – by-nc-as

Esta licença permite que outros remixem, adapte, e criem obras derivadas sobre a obra original, desde que com fins não comerciais e contanto que atribuam crédito ao autor e licenciem as novas criações sob os mesmos parâmetros. Outros podem fazer download ou redistribuir a obra da mesma forma que na licença anterior, mas eles também podem traduzir, fazer remixes e elaborar novas histórias com base na obra original. Toda nova obra feita a partir desta deverá ser licenciada com a mesma licença, de modo que qualquer obra derivada, por natureza, não poderá ser usada para fins comerciais.

This license lets other remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms.

Para maiores informações sobre o método de licenciamento acesse os seguintes sites:

Brasil:

<http://creativecommons.org.br/as-licencas/>

<http://creativecommons.org/licenses/by-nc-sa/3.0/br/>

Internacional:

<http://creativecommons.org/licenses/by-nc-sa/3.0/>

<http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>

**1 – Proteção de Acesso Remoto**

Assim como nós protegemos o acesso a console também devemos proteger os acessos remotos, até com mais cautela que o acesso a console, até porque, o acesso físico muitas vezes tem outros mecanismos de proteção, como salas fechadas e com restrições, rack com chave, etc (isso não é praxe e acredite em muitos ambientes os equipamentos são expostos =), já no acesso remoto, o atacante pode estar em outro local da empresa, ou ainda está fora da empresa e mesmo assim com poderes de acesso aos serviços expostos.

**2 – TCP Wrapper**

Iremos ver como utilizar o **TCP Wrapper**, permitindo ou bloqueando o acesso aos **daemons** que são iniciados pelo **inetd/xinetd** ou que foram compilados com a opção de aceitar **TCP Wrapper**.

Para verificarmos se um daemon possui suporte ao **TCP Wrapper**, basta analisarmos se o mesmo utilizar a biblioteca compartilhada **libwrap.so**, como no exemplo abaixo:

root@fusion:~# ldd /usr/sbin/sshd

linux-vdso.so.1 => (0x00007fffe30f5000)

libwrap.so.0 => /lib/libwrap.so.0 (0x00007fd0bb73c000)

libpam.so.0 => /lib/libpam.so.0 (0x00007fd0bb530000)

libselinux.so.1 => /lib/libselinux.so.1 (0x00007fd0bb311000)

libcrypto.so.0.9.8 => /usr/lib/libcrypto.so.0.9.8 (0x00007fd0baf70000)

libutil.so.1 => /lib/libutil.so.1 (0x00007fd0bad6d000)

libz.so.1 => /usr/lib/libz.so.1 (0x00007fd0bab55000)

libcrypt.so.1 => /lib/libcrypt.so.1 (0x00007fd0ba91e000)

libgssapi\_krb5.so.2 => /usr/lib/libgssapi\_krb5.so.2 (0x00007fd0ba6e9000)

libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0x00007fd0ba420000)

libcom\_err.so.2 => /lib/libcom\_err.so.2 (0x00007fd0ba21d000)

libc.so.6 => /lib/libc.so.6 (0x00007fd0b9ebb000)

libnsl.so.1 => /lib/libnsl.so.1 (0x00007fd0b9ca2000)

libdl.so.2 => /lib/libdl.so.2 (0x00007fd0b9a9e000)

/lib64/ld-linux-x86-64.so.2 (0x00007fd0bbbc9000)

libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0x00007fd0b9878000)

libkrb5support.so.0 => /usr/lib/libkrb5support.so.0 (0x00007fd0b966f000)

libkeyutils.so.1 => /lib/libkeyutils.so.1 (0x00007fd0b946d000)

libresolv.so.2 => /lib/libresolv.so.2 (0x00007fd0b9257000)

libpthread.so.0 => /lib/libpthread.so.0 (0x00007fd0b903a000)

Não é recomendado o uso do **TCP Wrapper** por termos tecnologias muito melhores e com maior nível de granularidade de acesso como o **Netfilter** (que você poderá ver mais detalhes no Artigo **Firewall Linux com Netfilter**), porém no mundo onde nem tudo é perfeito, temos diversos sistemas embarcados e customizados que não tem suporte ao **Netfilter (iptables/ipchains)**. Para este tipo de situação ainda podemos contar com o **TCP Wrapper** para criamos regras de acesso.

**2.1 – Instalando o XINETD**

Caso o seu sistema não tenha instalado o **inetd (Internet Daemon)** ou o **xinetd (Extended Internet Daemon)** basta instalar o pacote **openbsd-inetd** ou **xinetd**.

No nosso caso iremos instalar o **xinetd**, o qual possui melhorias com relação ao **inetd**.

root@fusion:~# aptitude install xinetd

O daemon **xinetd** é um **super-server daemon** de serviços do **TCP Wrapper** que controla o acesso a um subconjunto de serviços de rede populares, incluindo o **FTP**, o **IMAP** e o **Telnet**. O **xinetd** também oferece opções de configuração especificas de serviços para controle de acesso, registro aprimorado, vinculação, redirecionamento e controle de utilização de recursos.

Quando um cliente tenta conectar a um serviço de rede controlado pelo **xinetd**, o **super-server** recebe o pedido e verifica a existência de quaisquer regras de controle de acesso do **TCP Wrapper**.

Se o acesso for permitido, o **xinetd** verifica se a conexão é permitida sob as suas próprias regras de acesso para o serviço em questão. O **xinetd** também verifica se o serviço pode ter mais recursos alocados e certifica-se de que o serviço não está violando nenhuma regra definida.

Se todas as condições foram satisfeitas, o **xinetd** então inicia uma instância do serviço e passa o controle da conexão para o serviço. Após a conexão ter sido estabelecida, o **xinetd** não participa mais da comunicação entre o cliente e o servidor.

Os arquivos do **xinetd** são:

* ***/etc/xinetd.conf*** 🡪 Arquivo de configuração global do **xinetd**.
* ***/etc/xinetd.d/*** 🡪 Diretório contendo todos os arquivos de serviços específicos.

**2.2.1 – Arquivo /etc/xinetd.conf**

Este é o arquivo de configuração global, o que for atribuído neste arquivo irá refletir em todos os serviços que usa o **xinetd**.

Vamos configurar nosso arquivo ***/etc/xinetd.conf***.

root@fusion:~# vi /etc/xinetd.conf

**# Simple configuration file for xinetd**

**#**

**# Some defaults, and include /etc/xinetd.d/**

**defaults**

**{**

**# Please note that you need a log\_type line to be able to use log\_on\_success**

**# and log\_on\_failure. The default is the following :**

**# log\_type = SYSLOG daemon info**

**instances = 60**

**log\_type = SYSLOG authpriv**

**log\_on\_sucess = HOST PID**

**log\_on\_failure = HOST**

**cps = 25 30**

**}**

**includedir /etc/xinetd.d**

"/etc/xinetd.conf" 20L, 398C written

root@fusion:~#

As opções usadas no arquivo ***/etc/xinetd.conf*** são:

* **Instances <N>** 🡪 A falta de limite de instâncias para um determinado serviço pode deixar o sistema vulnerável a ataques de negação de serviço e indisponibilidade de informações.
* **log\_type** 🡪 Configura o **xinetd** para usar o **log authpriv**, o qual escreve entradas de registro no arquivo ***/var/log/authpriv***. Uma diretiva como **FILE /var/log/xinetd.log** criaria um arquivo de registro personalizado chamado **xinetd.log** no diretório ***/var/log***.
* **log\_on\_sucess** 🡪 Configura o **xinetd** para registrar as conexões bem sucedidas.
* **log\_on\_failure** 🡪 Configura o **xinetd** para registrar as tentativas de conexão mal sucedidas ou recusadas.
* **cps = <N> <N>** 🡪 Permitir um número grande de novas conexões por segundo a um determinado serviço, pode expor o sistema a ataques de negação de serviço ou consumir os recursos do sistema de forma anormal. Convém que seja configurado quando possível um limite para o número máximo de novas conexões por segundo, de forma que o sistema desabilite o serviço por um período de tempo caso o limite seja excedido. No nosso caso, o **xinetd** para permitir até 25 conexões por segundo para qualquer serviço. Caso esse número for excedido ele aguardara 30 segundo para permitir uma nova requisição.
* **includedir** 🡪 Inclui os arquivos com as configurações dos serviços que será gerenciado pelo **xinetd**.

Também devemos habilitar o log do sistema, a fim de facilitar os processos de auditoria em caso de ataques ou falhas. Convém que cada estação que conecte no sistema seja devidamente identificada, bem como suas alterações no mesmo.

Para isso, devemos adicionar as linhas abaixo na seção **defaults** do arquivo ***/etc/xinetd.conf***:

root@fusion:~# vi /etc/xinetd.conf

**# Simple configuration file for xinetd**

**#**

**# Some defaults, and include /etc/xinetd.d/**

**defaults**

**{**

**# Please note that you need a log\_type line to be able to use log\_on\_success**

**# and log\_on\_failure. The default is the following :**

**# log\_type = SYSLOG daemon info**

**instances = 60**

**log\_type = SYSLOG authpriv**

**log\_on\_sucess = HOST PID USERID DURATION EXIT**

**log\_on\_failure = HOST USERID RECORD**

**cps = 25 30**

**per\_source = 20**

**max\_load = 50**

**access\_times = 08:00-18:00**

**}**

**includedir /etc/xinetd.d**

"/etc/xinetd.conf" 20L, 398C written

root@fusion:~#

Outras opções que devemos nos atentar são:

* **per\_source = <N>** 🡪 Convém estabelecer um limite para o número de conexões a partir de uma mesma origem para os serviços gerenciados pelo **xinetd**. Permitir muitas conexões de uma mesma origem a um determinado serviço pode expor o sistema a ataques de negação de serviço, consumindo os recursos do sistema de forma anormal.
* **max\_load = <N>** 🡪 Aqui configuramos a porcentagem do processador que pode ser utilizada. A falta de controle da carga que um serviço pode receber expõe o sistema à indisponibilidade de serviços e informações. Por este motivo, convém que seja feito quando possível um controle da carga que cada serviço pode utilizar.
* **access\_times = <HH1>:<MM1>-<HH2>:<MM2>** 🡪 Deixar serviços expostos a atacantes durante mais tempo que o necessário aumenta o risco de ocorrer ataques. Geralmente, os ataques ocorrem em horários onde não há utilização e onde não há monitoramento humano. Vale lembrar que, <HH1>:<MM1> é o início do horário em horas e minutos e <HH2>:<MM2> é o término do mesmo.

Para nosso laboratório, vamos levantar o serviço **ssh** pelo **xinetd**. Para isso devemos criar o arquivo com as informações referente ao serviço em ***/etc/xinetd.d***, ficando assim:

root@fusion:~# netstat -patun | grep 22

tcp 0 **0 0.0.0.0:22** 0.0.0.0:\* LISTEN **1099/sshd** **🡨 O serviço esta ouvindo através do daemon sshd**

tcp 0 48 192.168.5.138:22 192.168.5.1:58711 ESTABLISHED 1141/0

tcp6 0 0 :::22 :::\* LISTEN 1099/sshd

root@fusion:~# > /etc/xinetd.d/ssh

root@fusion:~# vi /etc/xinetd.d/ssh

**service ssh**

**{**

**socket\_type = stream**

**protocol = tcp**

**wait = no**

**port = 22**

**user = root**

**server = /usr/sbin/sshd**

**server\_args = -i**

**disable = no**

**}**

"/etc/xinetd.d/ssh" 11L, 151C written

root@fusion:~# /etc/init.d/ssh stop ; /etc/init.d/xinetd restart

Stopping OpenBSD Secure Shell server: sshd.

Stopping internet superserver: xinetd.

Starting internet superserver: xinetd.

root@fusion:~# netstat -patun | grep 22

tcp 0 **0 0.0.0.0:22** 0.0.0.0:\* LISTEN **1231/xinetd** **🡨 O serviço esta ouvindo através do super-daemon xinetd**

tcp 0 48 192.168.5.138:22 192.168.5.1:58711 ESTABLISHED 1141/0

root@fusion:~#

**Observação:** Lembre-se que todas as vezes que alteramos as configurações do **xinetd** devemos reiniciar os serviços e o **xinetd**.

Pronto, nosso serviço de acesso remoto agora passará pelos filtros do **TCP Wrapper**. Vamos agora entender um pouco sobre os arquivos de configuração dos filtros.

**2.2.2 – Arquivo /etc/hosts.deny**

O arquivo ***/etc/hosts.deny*** é um arquivo de configuração com regras que descrevem quais computadores não possuem permissão de acesso a serviços no servidor.

Um modelo simples deste arquivo se parece com o seguinte:

root@fusion:~# cat /etc/hosts.deny

# /etc/hosts.deny: list of hosts that are \_not\_ allowed to access the system.

# See the manual pages hosts\_access(5) and hosts\_options(5).

#

# Example: ALL: some.host.name, .some.domain

# ALL EXCEPT in.fingerd: other.host.name, .other.domain

#

# If you're going to protect the portmapper use the name "portmap" for the

# daemon name. Remember that you can only use the keyword "ALL" and IP

# addresses (NOT host or domain names) for the portmapper, as well as for

# rpc.mountd (the NFS mount daemon). See portmap(8) and rpc.mountd(8)

# for further information.

#

# The PARANOID wildcard matches any host whose name does not match its

# address.

#

# You may wish to enable this to ensure any programs that don't

# validate looked up hostnames still leave understandable logs. In past

# versions of Debian this has been the default.

# ALL: PARANOID

root@fusion:~#

Vamos acrescentar a entrada **ALL:ALL** no arquivo ***/etc/hosts.deny*** para bloquear qualquer conexão que não tenha uma entrada de permissão no arquivo ***/etc/hosts.allow***. Esta é a configuração mais segura.

root@fusion:~# echo "ALL: ALL" >> /etc/hosts.deny

root@fusion:~# cat /etc/hosts.deny

--==[ Resumido ]==--

# You may wish to enable this to ensure any programs that don't

# validate looked up hostnames still leave understandable logs. In past

# versions of Debian this has been the default.

# ALL: PARANOID

ALL: ALL

root@fusion:~#

Qualquer modificação no arquivo ***/etc/hosts.deny*** ou em ***/etc/hosts.allow*** só entrará em vigor após reinicialização do daemon **xinetd**.

Isso pode ser feito da seguinte forma:

root@fusion:~# ps aux | grep xinetd

**root 1231 0.0 0.3 2400 872 ? Ss 03:20 0:00 /usr/sbin/xinetd -pidfile /var/run/xinetd.pid -stayalive -inetd\_compat -inetd\_ipv6**

root 1420 0.0 0.2 3304 756 pts/0 S+ 04:12 0:00 grep xinetd

root@fusion:~# kill -HUP 1231

root@fusion:~#

Com o comando **kill -HUP [pid do xinetd]**, solicitamos aos **xinetd** reler seus arquivos de configuração. Para descobrirmos o **PID (Process IDentification)** do **xinetd** podemos usar o comando **ps aux|grep xinetd**.

**2.2.3 – Arquivo /etc/hosts.allow**

O arquivo ***/etc/hosts.allow*** contém as regras de permissão de acesso aos serviços. Seu formato é muito simples e de fácil configuração.

root@fusion:~# cat /etc/hosts.allow

# /etc/hosts.allow: list of hosts that are allowed to access the system.

# See the manual pages hosts\_access(5) and hosts\_options(5).

#

# Example: ALL: LOCAL @some\_netgroup

# ALL: .foobar.edu EXCEPT terminalserver.foobar.edu

#

# If you're going to protect the portmapper use the name "portmap" for the

# daemon name. Remember that you can only use the keyword "ALL" and IP

# addresses (NOT host or domain names) for the portmapper, as well as for

# rpc.mountd (the NFS mount daemon). See portmap(8) and rpc.mountd(8)

# for further information.

#

root@fusion:~#

O arquivo tem a seguinte sintaxe:

Lista\_de\_Serviço: Lista\_de\_Hosts : Comando

* **Lista de Serviços** 🡪 É uma lista de nomes de serviços separados por vírgula que esta regra se aplica. Exemplos de nomes de serviços são: **sshd, ftpd, telnetd** e **fingerd**.
* **Lista de Hosts** 🡪 É uma lista de nomes de hosts separada por vírgula. Podemos também usar endereços IP aqui. Adicionalmente, podemos especificar nomes de computadores ou endereços IP usando caracteres coringas para atingirmos grupos de hosts. Exemplos incluem: **fox.how2security.com.br,** para conferir com um endereço de computador específico, **.how2security.com.br** para atingir qualquer endereço de computador finalizando com aquela **string**. Use **192.168.252.** para conferir com qualquer endereço IP iniciado com estes três octetos. Existem alguns parâmetros especiais para simplificar a configuração, alguns destes são:
  + **ALL** 🡪 Atinge todos os endereços;
  + **LOCAL** 🡪 Atinge qualquer computador que não contém um **”.”** (ie. está no mesmo domínio de sua máquina);
  + **PARANOID** 🡪 Atinge qualquer computador que o nome não confere com seu endereço (falsificação de nome);
  + **EXCEPT 🡪** Um último parâmetro que também é útil permitindo fazermos uma lista de exceções. Isto será coberto em um exemplo adiante.
* **Comando 🡪** É um parâmetro opcional. Este parâmetro é o caminho completo de um comando que deverá ser executado toda vez que esta regra conferir. Ele pode executar um comando para tentar identificar quem está conectado pelo host remoto, ou gerar uma mensagem via e-mail ou algum outro alerta para um administrador de rede, que alguém está tentando se conectar. Existe um número de expansões que podem ser incluídos. Alguns exemplos comuns são: %h expande o nome do computador que está conectado ou endereço se ele não possuir um nome, %d o nome do daemon sendo chamado.

**Nota:** A sintaxe mostrada acima se aplica tanto ao ***hosts.allow*** como ao ***hosts.deny***.

Se o computador tiver permissão de acessar um serviço através do ***/etc/hosts.allow***, então o ***/etc/hosts.deny*** não será consultado e o acesso será permitido, ou seja, o primeiro arquivo consultado é o ***hosts.allow***, caso não exista uma regra válida para determinada conexão, o arquivo ***hosts.deny*** é consultado. Se também não houver nenhuma regra que rejeite o acesso, o acesso será concedido

Como exemplo, vamos liberar o acesso **ssh** para toda a rede local:

root@fusion:~# vi /etc/hosts.allow

--==[ Resumido]==--

# If you're going to protect the portmapper use the name "portmap" for the

# daemon name. Remember that you can only use the keyword "ALL" and IP

# addresses (NOT host or domain names) for the portmapper, as well as for

# rpc.mountd (the NFS mount daemon). See portmap(8) and rpc.mountd(8)

# for further information.

#

sshd: 192.168.5.

"/etc/hosts.allow" 14L, 597C written

root@fusion:~#

Com isso, liberamos o acesso **ssh** toda a rede **192.168.5.0/24**.

**Observação:** Como vimos acima, o **SSH** possui suporte nativo ao **TCP Wrapper**, desta forma não precisaríamos adicioná-lo ao **xinetd**.

Sempre que adicionarmos ou editarmos arquivos no diretório /***etc/xinet.d/*** ou o arquivo principal do **xinetd**, precisaremos reiniciar o daemon para que as alterações entrem em vigor.

Isso pode ser feito da seguinte forma:

root@fusion:~# ps aux | grep xinetd

**root 1231 0.0 0.3 2400 872 ? Ss 03:20 0:00 /usr/sbin/xinetd -pidfile /var/run/xinetd.pid -stayalive -inetd\_compat -inetd\_ipv6**

root 1420 0.0 0.2 3304 756 pts/0 S+ 04:12 0:00 grep xinetd

# kill -HUP 1231

root@fusion:~#

Com o comando **kill -HUP [pid do xinetd]**, solicitamos aos **xinetd** reler seus arquivos de configuração, para descobrir o **PID** do **xinetd** você pode usar o comando **ps aux|grep xinetd**.

Vamos acessar o **ssh** da máquina **Corola** e validar se o acesso está funcionando para toda a rede.

root@corola:~# ssh W3ll@192.168.5.138

W3ll@192.168.5.138's password:

W3ll@fusion:~$

Agora vamos colocar como exceção o IP de outra máquina na rede (um Windows com o IP **192.168.5.1**) e refazer os testes.

Para isso:

root@fusion:~# vi /etc/hosts.allow

--==[ Resumido]==--

sshd: 192.168.5. EXCEPT 192.168.5.1

"/etc/hosts.allow" 14L, 616C written

root@fusion:~# ps aux | grep xinetd

root 1231 0.0 0.3 2400 896 ? Ss 03:20 0:00 /usr/sbin/xinetd -pidfile /var/run/xinetd.pid -stayalive -inetd\_compat -inetd\_ipv6

root@fusion:~# kill -HUP 1231

root@fusion:~#

Agora vamos testar da máquina Windows:

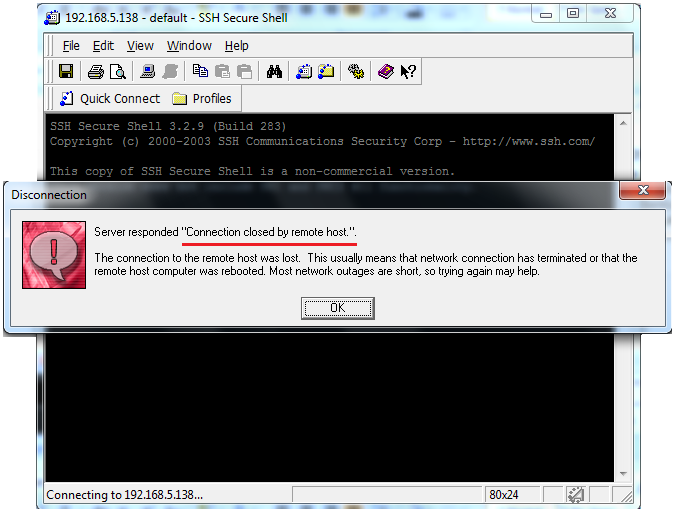


Figura 02 – Acesso negado pelo TCP Wrapper

No **Corola**:

root@corola:~# ssh W3ll@192.168.5.138

W3ll@192.168.5.138's password:

W3ll@fusion:~$

Como podemos ver, o acesso do servidor **Corola** continua funcionando, porém, o host **192.168.5.1** não conseguiu o acesso. Vamos olhar no log o registro gerado pelo **TCP Wrappers**:

root@fusion:~# tail -n 5 /var/log/auth.log

**Aug 7 04:47:45 fusion xinetd[1556]: FAIL: ssh libwrap from=192.168.5.1**

Aug 7 04:55:50 fusion sshd[1569]: Accepted password for W3ll from 192.168.5.131 port 47368 ssh2

Aug 7 04:55:50 fusion sshd[1569]: pam\_unix(sshd:session): session opened for user W3ll by (uid=0)

Aug 7 05:00:50 fusion sshd[1571]: Received disconnect from 192.168.5.131: 11: disconnected by user

Aug 7 05:00:50 fusion sshd[1569]: pam\_unix(sshd:session): session closed for user W3ll

root@fusion:~#

**2.2.4 – Validando as Configurações com o tcpdchk e tcpdmatch**

O utilitário **tcpdchk** é útil para verificar problemas nos arquivos ***/etc/hosts.allow*** e ***/etc/hosts.deny***. Ele é usado para verificar se as sintaxes dos arquivos estão corretas.

root@fusion:~# tcpdchk -v -i /etc/xinetd.conf

Using network configuration file: /etc/xinetd.conf

>>> Rule /etc/hosts.allow line 13:

daemons: sshd

clients: 192.168.5. EXCEPT 192.168.5.1

warning: /etc/hosts.allow, line 13: host address 192.168.5.1->name lookup failed

access: granted

>>> Rule /etc/hosts.deny line 21:

daemons: ALL

clients: ALL

access: denied

root@fusion:~#

Como podemos ver ele nos fornece um relatório sobre nossas configurações. Utilizamos a opção **–v** para mostrar a lista de todas as regras e **–i** para apontar o arquivo que queremos fazer a validação. No relatório vimos que o único problema que ele achou foi referente à resolução de nome.

Outro utilitário é o **tcpdmatch**, com o qual podemos simular o acesso dos endereços IP para validar as regras de acesso.

É importante mostrar na prática como o **tcpdmatch** funciona para simularmos os acessos, garantindo assim que as regras estão corretas, ajudando a mapear possíveis problemas de acesso.

Relembrando nossas regras estão assim:

root@fusion:~# tail -n 2 /etc/hosts.deny

ALL: ALL

root@fusion:~# tail -n 2 /etc/hosts.allow

sshd: 192.168.5. EXCEPT 192.168.5.1

root@fusion:~#

Então como política no arquivo ***/etc/hosts.deny*** bloqueamos tudo que não tenha uma permissão no arquivo de ***/etc/hosts.allow***. E no arquivo ***/etc/hosts.allow*** permitimos o serviço **sshd** para toda a rede local com exceção do host **192.168.5.1**.

Então vamos validar isso:

root@fusion:~# tcpdmatch -i /etc/xinetd.conf sshd 127.0.0.1

warning: sshd: no such process name in /etc/xinetd.conf

client: address 127.0.0.1

server: process sshd

matched: /etc/hosts.deny line 21

access: denied

root@fusion:~# tcpdmatch -i /etc/xinetd.conf sshd 192.168.5.10

warning: sshd: no such process name in /etc/xinetd.conf

client: address 192.168.5.10

server: process sshd

matched: /etc/hosts.allow line 13

access: granted

root@fusion:~# tcpdmatch -i /etc/xinetd.conf sshd 192.168.5.1

warning: sshd: no such process name in /etc/xinetd.conf

client: address 192.168.5.1

server: process sshd

matched: /etc/hosts.deny line 21

access: denied

root@fusion:~#

Como podemos ver, o endereço de **loopback** não foi permitido e a regra que está bloqueando está na **linha 21** do arquivo ***/etc/hosts.deny,*** assim como o endereço IP que colocamos como uma exceção também está bloqueada nesta linha. Enquanto outro endereço da rede local tem acesso pela regra que está na **linha 13** do arquivo ***/etc/hosts.allow***.

**Observação:** Só utilize o **TCP Wrappers** se você não tiver outra opção. Use firewall onde você possa granular mais as regras, como por exemplo o **Netfilter**, **PF**, etc.

**3 – Proteção de Acesso Remoto Via SSH**

Agora vamos usar as melhores práticas para deixarmos o serviço de **SSH (Secure Shell)** mais seguro, o qual é utilizado para administração remota.

**3.1 – Configuração do Arquivo /etc/ssh/sshd\_config**

Vamos editar o arquivo de configuração do **ssh** em ***/etc/ssh/sshd\_config***. Podemos deixá-lo mais seguro fazendo algumas modificações básicas.

root@fusion:~# cp /etc/ssh/sshd\_config /root/auditoria/sshd\_config.bkp

root@fusion:~# cat /root/auditoria/sshd\_config.bkp | grep -v ^# > /etc/ssh/sshd\_config

root@fusion:~# vi /etc/ssh/sshd\_config

**Port 43322**

**Protocol 2**

**ListenAddress 192.168.5.138**

**LoginGraceTime 60**

**PermitRootLogin no**

**PermitEmptyPasswords no**

**PermitUserEnvironment no**

**MaxAuthTries 4**

**IgnoreRhosts yes**

**ClientAliveInterval 300**

**ClientAliveCountMax 0**

**HostbasedAuthentication no**

**UsePAM yes**

**AllowUsers W3ll@192.168.5.131**

**AllowGroups supgrp**

**Banner /etc/issue.net**

"/etc/ssh/sshd\_config" 50L, 704C written

root@fusion:~#

Vamos explicar o que é cada uma dessas variáveis:

* **Port 🡪** Aqui definimos a porta que o **ssh** irá ouvir. Uma boa prática é que a porta, que por padrão é a **22,** seja alterada para uma porta bem alta, acima de **40000** dificultando a vida dos bisbilhoteiros que usam ferramentas automatizada em busca das portas padrões (**PortScan**).
* **ListenAddress 🡪** Aqui é o endereço IP que o serviço de **ssh** irá usar para acesso. Uma boa prática é fixarmos um endereço IP quando a máquina tiver mais que um endereço IP.
* **Protocol 🡪** Aqui configuramos a versão que será usada do protocolo **ssh.** Por questão de segurança, use a versão dois (**v2**), pois ela é mais segura que sua antecessora.
* **LoginGraceTime 🡪** Aqui configuramos o tempo máximo de inatividade durante o login, antes que um usuário complete o logon com sucesso. **0** desabilita a limitação, aguardando indeterminadamente pelo usuário digitar o **username** ou **password**.
* **PermitRootLogin 🡪** A opção **PermitRootlogin** especifica se o usuário root poderá fazer login usando SSH. Proibir o login do root via SSH faz com que os administradores se autentiquem usando sua própria conta individual para, em seguida, usar o comando **sudo** ou **su** para realizar tarefas como root. Ao desabilitar esse opção, a ocorrência de não-repúdio será reduzida e uma trilha de auditoria clara, no caso de um incidente de segurança, será fornecida. Além disso, esta conta é padrão nos sistemas **UNIX-Like** e o usuário predileto dos crackers para fazer **Brute-Force**.
* **PermitEmptyPasswords 🡪** A opção **PermitEmptyPasswords** especifica se o servidor permitirá o login de contas com senhas em branco. Proibir o acesso via shell remoto para contas que não possuem senhas reduz a probabilidade de acesso não autorizado ao sistema.
* **PermitUserEnvironment 🡪** A opção **PermitUserEnvironment** permite aos usuários configurar variáveis de ambiente para o daemon SSH, possibilitando que os usuários burlem os controles de segurança (por exemplo, definindo um caminho de execução que leve a um trojan).
* **MaxAuthTries 🡪** A opção **MaxAuthTries** especifica o número máximo de tentativas de autenticação permitidos por conexão. Quando a contagem de logins mal sucedidos atinge metade desse limite, as mensagens de erro são gravadas no arquivo **syslog** detalhando a falha do login. Definir essa opção para um número pequeno irá minimizar o risco de ataques de **Brute-Force** bem sucedidos no servidor SSH.
* **HostbasedAuthentication no 🡪** A opção **HostbasedAuthentication** especifica se a autenticação, através de hosts confiáveis, será permitida através do uso do ***.rhosts*** ou ***/etc/hosts.equiv*** juntamente com a autenticação de chave pública do host cliente. Esta opção só se aplica ao protocolo SSH versão 2. Desabilitar o uso do ***.rhosts*** no servidor SSH fornece uma camada adicional de proteção.
* **IgnoreRhosts 🡪** A opção **IgnoreRhosts** especifica que os arquivo ***.rhosts*** e ***.shosts*** não serão utilizados para autenticação. A definição desta opção força os usuários a digitar uma senha durante a autenticação do servidor SSH.
* **ClientAliveInterval e ClientAliveCountMax 🡪** Podem haver situações em que um usuário venha a esquecer de finalizar a sessão SSH, deixando a mesma autenticada com suas credenciais. Recomenda-se definir um período de timeout de forma que passado um determinado tempo o sistema finalize a sessão automaticamente. O objetivo é reduzir os riscos de acesso indevido ao sistema, evitando que atacantes possam enviar e-mails, acessar arquivos e executar programas no contexto do usuário que está conectado.
* **AllowUsers 🡪** Aqui podemos colocar os usuário que terão acesso ao serviço (O ideal é utilizar o **PAM**).
* **AllowGroups 🡪** Aqui colocamos o grupo que tem permissão para utilizar o serviço. (O ideal é utilizar o **PAM**).
* **Banner 🡪** Aqui apontamos um arquivo de banner que será apresentado para o usuário toda vez que ele fizer o acesso ao sistema. Aqui devemos informa ao usuário que, o uso do sistema por pessoas autorizadas será auditado, assim como, o acesso não autorizado poderá acarretar em punições (ou por políticas da empresa ou por questões legais).

Um exemplo de banner:

root@fusion:~# cat /etc/issue.net

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WELCOME! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WARNING NOTICE: This is a private system for use by **MyCompany** the unautorized

access or attempt, use or modification of this system is strictly prohibited.

Individuals undertaking such unauthorized access, use or modification are

subject to company disciplinary procedings and/or criminal and civil penalties under applicable domestic and foreign laws. The use of this system may be monitored and recorded for administrative and security reasons in accordance with local law.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BEM VINDO! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AVISO: Este e um sistema privado para uso exclusivo da **MyCompany**. Acesso nao

autorizados ou tentativas, uso ou modificacao deste sistema sao estritamente

restritos. Individuos que executam tal acesso, usam ou modificam sem autorização estao sujeitos a procedimentos disciplinares e penalidades civis com base em leis domesticas e estrangeiras aplicaveis. O uso deste sistema pode ser monitorado e armazenado para uso posterior conforme a lei local.

root@fusion:~#

Após as alterações no arquivo ***/etc/ssh/sshd\_config***, devemos reiniciar o serviço para que as novas configurações entrem em vigor.

root@fusion:~# /etc/init.d/ssh restart

Restarting OpenBSD Secure Shell server: sshd.

root@fusion:~#

Agora vamos validar as configurações, usando para isso o servidor **Corola** (**192.168.5.131**).

root@corola:~# ssh w3ll@192.168.5.138 -p 43322

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WELCOME! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WARNING NOTICE: This is a private system for use by MyCompany the unautorized

access or attempt, use or modification of this system is strictly prohibited.

Individuals undertaking such unauthorized access, use or modification are

subject to company disciplinary procedings and/or criminal and civil penalties under applicable domestic and foreign laws. The use of this system may be monitored and recorded for administrative and security reasons in accordance with local law.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BEM VINDO! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AVISO: Este e um sistema privado para uso exclusivo da MyCompany. Acesso nao

autorizados ou tentativas, uso ou modificacao deste sistema sao estritamente

restritos. Individuos que executam tal acesso, usam ou modificam sem autorização estao sujeitos a procedimentos disciplinares e penalidades civis com base em leis domesticas e estrangeiras aplicaveis. O uso deste sistema pode ser monitorado e armazenado para uso posterior conforme a lei local.

w3ll@192.168.5.138's password:

**Permission denied**, please try again. **🡨 Acesso negado para o usuário w3ll**

root@corola:~# ssh W3ll@192.168.5.138 -p 43322

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WELCOME! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WARNING NOTICE: This is a private system for use by MyCompany the unautorized

--==[ Resumido ]==--

W3ll@192.168.5.138's password:

W3ll@fusion:~$

**Observação:** Vamos tirar a limitação dos usuários que podem utilizar o serviço de **ssh** do arquivo ***/etc/ssh/sshd\_config***, pois utilizaremos o **PAM** para fazer esse bloqueio no próximo tópico.

**3.2 – Limitando o Acesso por Tempo**

Podemos utilizar o **PAM** para limitar o horário de acesso ao serviço de **ssh**. Para isso, vamos editar o arquivo ***/etc/pam.d/sshd*** e adicionar o seguinte módulo:

root@fusion:~# vi /etc/pam.d/sshd

--==[ Resumido ]==--

# Limitando o acesso por tempo

account required pam\_time.so

--==[ Resumido ]==--

"/etc/pam.d/sshd" 42L, 1334C written

root@fusion:~#

Após isso, vamos limitar o usuário **W3ll** editando o arquivo ***/etc/security/time.conf*** para limitar o horário.

root@fusion:~# vi /etc/security/time.conf

--==[ Resumo ]==--

sshd;\*;W3ll;Al0800-1800

sshd;\*;W3ll;!SaSu0000-2400

#

"/etc/security/time.conf" 69L, 2258C written

root@fusion:~#

Neste exemplo, limitamos o acesso do usuário **W3ll** para logar via **ssh** todos os dias da semana das **08:00** as **18:00** horas com exceção de sábado e domingo.

Vamos validar usando o servidor Corola.

root@corola:~# date

Tue Aug 7 16:35:29 BRT 2012

root@corola:~# ssh W3ll@192.168.5.138

W3ll@192.168.5.138's password:

W3ll@fusion:~$ exit

logout

Connection to 192.168.5.138 closed.

root@corola:~#

Para agilizar o teste, iremos no servidor **Fusion** e mudaremos a hora para:

root@fusion:~# date 080718012012

Tue Aug 7 18:01:00 BRT 2012

root@fusion:~#

E refazermos o teste.

root@corola:~# ssh W3ll@192.168.5.138

W3ll@192.168.5.138's password:

**Connection closed by 192.168.5.138**

root@corola:~#

**3.3 – Autenticação por Chaves**

Vamos criar um conjunto de chaves no cliente para acessar os servidores remotos via **ssh** sem que seja necessário digitar uma senha, em vez disso devemos digitar uma frase utilizando a chave, ou seja, estabelecer uma relação de confiança entre cliente e servidor via chave privada e pública.

Em nosso cenário a máquina **Corola** é a cliente e o servidor o **Fusion**. Desta forma, vamos logar no **Corola** (**Cliente**) com o usuário **W3ll** e criar as chaves de autenticação **RSA** com o comando ***ssh-keygen*** como mostrado a seguir:

W3ll@corola:~$ id

uid=1001(W3ll) gid=1001(W3ll) groups=1001(W3ll)

W3ll@corola:~$ ssh-keygen -t rsa -b 2048

Generating public/private rsa key pair.

Enter file in which to save the key (/home/W3ll/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/W3ll/.ssh/id\_rsa.

Your public key has been saved in /home/W3ll/.ssh/id\_rsa.pub.

The key fingerprint is:

45:3b:f7:a2:48:e0:e3:93:08:d4:53:d3:57:33:14:66 W3ll@corola

The key's randomart image is:

+--[ RSA 2048]----+

| o. .oE. |

| . . ....+ o |

| . o . .+ . |

| . o . . o . |

| . o S . . |

| . o + . . . |

| . + . . |

| . |

| |

+-----------------+

W3ll@corola:~$

Será indicado um local para a criação das chaves, deixe o padrão. Digite uma frase que deverá ser passada no momento do acesso.

**Observação:** Não é obrigatório atribuir uma frase, podemos usar essa facilidade na transferência de dados entre servidores sem a necessidade de autenticar com uma senha, apenas utilizando os certificados.

Serão gerados dois arquivos no home do usuário, no diretório ***/home/W3ll/.ssh***, chamados ***id\_rsa*** e ***id\_rsa.pub***. O primeiro é sua chave privada que fica no cliente e o segundo arquivo é a chave pública que deve ser enviada via canal seguro para o servidor de acesso remoto.

Vamos transferir a chave pública para o servidor remoto:

W3ll@corola:~$ scp -p 22 /home/W3ll/.ssh/id\_rsa.pub W3ll@192.168.5.138:/home/W3ll

W3ll@192.168.5.138's password:

22: No such file or directory

id\_rsa.pub 100% 393 0.4KB/s 00:00

W3ll@corola:~$

Agora devemos colocar o certificado, transferido para o servidor, no banco de dados de certificados aceitos da seguinte forma (no servidor **Fusion**).

W3ll@fusion:~$ mv id\_rsa.pub .ssh/authorized\_keys

W3ll@fusion:~$

**Observação:** O diretório ***/home/<User>/.ssh*** só existirá se você tiver usado o **OpenSSH** pelo menos uma vez. Caso não tenha, basta cria-lo com o **mkdir**.

Existe outra forma de transferir sua chave pública para os servidores com o utilitário **ssh-copy-id**, porém não temos a opção de trocar a porta de acesso, tendo que usar a padrão que é a porta **22/TCP**. Esse utilitário cria o diretório .**ssh** no ***home*** do usuário, não precisando criá-lo.

W3ll@corola:~$ ssh-copy-id -i .ssh/id\_rsa.pub W3ll@192.168.5.138

W3ll@192.168.5.138's password:

Now try logging into the machine, with "ssh 'W3ll@192.168.5.138'", and check in:

.ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting.

W3ll@corola:~$

Agora vamos testar:

W3ll@corola:~$ ssh W3ll@192.168.5.138

**Enter passphrase for key '/home/W3ll/.ssh/id\_rsa':**

Linux fusion 2.6.32-5-686 #1 SMP Sun May 6 04:01:19 UTC 2012 i686

The programs included with the Debian GNU/Linux system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent

permitted by applicable law.

Last login: Tue Jul 31 08:58:55 2012 from 192.168.5.128

W3ll@fusion:~$

Com isso inibimos os ataques de **Brute-Force**, já que não somos dependentes apenas de usuário e senha, mas também de um certificado.

No Windows podemos usar o **PuTTY** (**PuTTY**, **PuTTYGen**, **pageant**, **pscp.exe**).

**Nota:** Caso não possuamos os utilitários acima, podemos baixá-los no seguinte link: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Vamos gerar um certificado para o usuário root como exemplo.

Clique em **PuTTYGen.exe**, selecione **SSH-2 RSA** em **Type of key to generate** (parte inferior da janela) e **2048** em **Number of bits in a generated key** como parâmetro para a criação, e clique no botão **Generate**. Posicione o mouse abaixo da barra de progresso e comece a movimenta-lo para a criação de dados aleatórios.

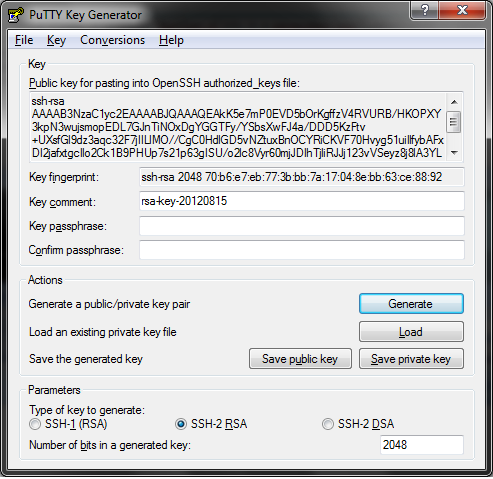


Figura 03 – Criação da Chave no PuTTYGen

A chave pública será exibida na tela, podemos atribuir uma **passphrass** em **Key** **passphrase** e **Confirm passphrasse,** então podemos copiar e colocá-la em um arquivo, ou salvar usando os botões **Save public key** e **Save private key**. Não se esquecendo de remover o conteúdo de cabeçalho e rodapé do arquivo salvo. Agora vamos transferir a chave pública para o servidor. No meu caso eu salvei o arquivo com a chave pública na pasta onde se encontra o conjunto de softwares do **PuTTY** em ***C:\PuTTY\***. Lembrando que ela deve ser copiada para o diretório ***.ssh*** dentro do diretório home do usuário, com o nome **authorized\_keys**.

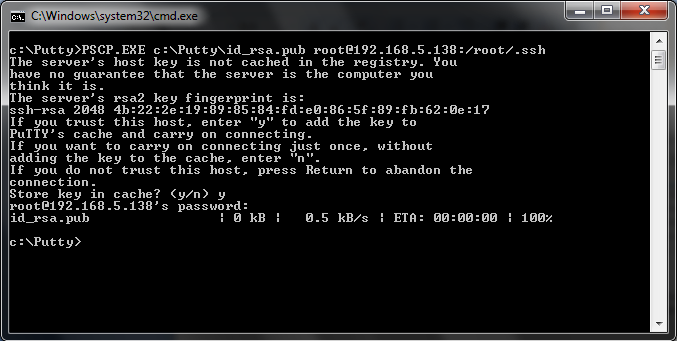


Figura 04 – Transferência do Certificado Publico do SSH

**Observação:** Caso já exista o arquivo ***/home/<User>/.ssh/authorized\_keys*** com chaves de outras máquinas, não mova por cima a chave gerada. Em vez disso, redirecione a saída do arquivo gerado para o arquivo ***authorized\_keys*** com o utilitário **cat** e o redirecionamento de saída **>>**.

root@fusion:~/.ssh# cat authorized\_keys >> /home/W3ll/.ssh/authorized\_keys

Não transporte seu certificado de máquina em máquina, nem em dispositivos de armazenamento móvel pelo risco de perdemos ou alguém capturá-lo, e assim comprometendo toda estrutura de certificados.

Para usarmos o **PuTTY** sem a necessidade de passar senhas (seja para fazer backups automáticos, ou outro propósito que não precise da intervenção) podemos usar o **Pageant.exe**. Execute-o, que ele irá aparecer no **SysTray** da máquina (próximo do relógio do Windows).



Figura 05 – Pageant

De um duplo clique no ícone. Vamos adicionar nossa chave privada, clicando no botão **Add Key** e apontando o arquivo que salvamos anteriormente. Será solicitado a **passphrasse** e então clique em **OK**.

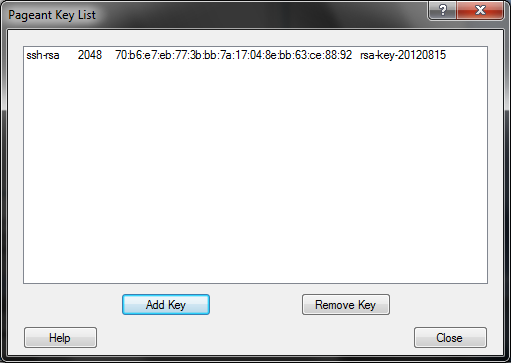


Figura 06 – A chave inserida no Pageant

Agora basta executamos sempre o **Pageant** antes do **PuTTY** que entraremos sem senha, apenas usando o certificado.

Vamos configurar o **PuTTY** para utilizar o certificado para autenticação.

De um duplo-clique no **PuTTY.exe**, em **Host Name (or IP address)** digite o nome ou o IP da máquina que iremos nos conectar (192.168.5.138) e em **Saved Session** atribua um nome para conexão e salve clicando no botão **Save**.

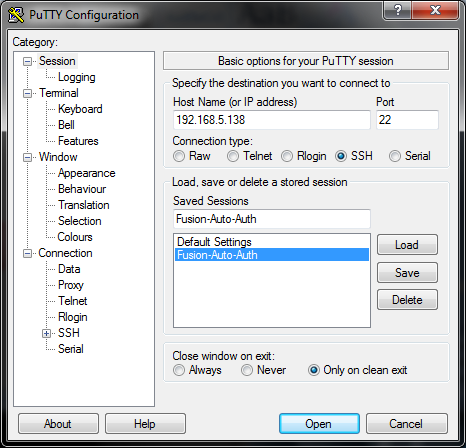


Figura 07 – Session do PuTTY

No Painel da esquerda expanda **Connection** e clique em **Data**. Em **Login datails, Auto-login username** digite o nome do usuário que irá logar automaticamente (**W3ll**).

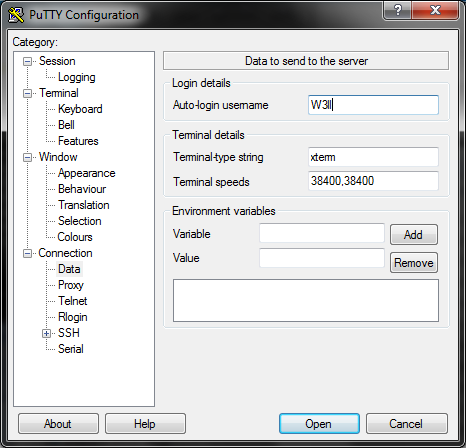


Figura 08 – Connection/Data do PuTTY

Expanda **SSH** em **Connection**, e clique em **Auth**, aponte o certificado privado em **Private key file for authentication**, clicando em **Browser** e selecionando o arquivo.

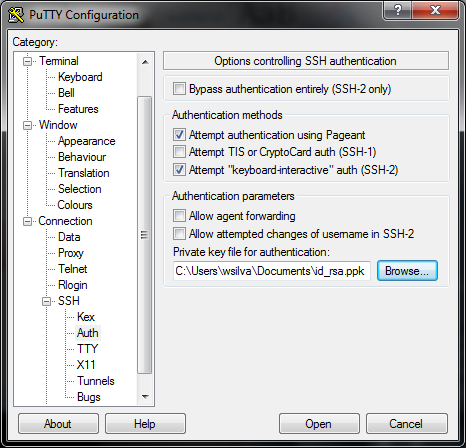


Figura 09 – Connection/SSH/Auth do PuTTY

Volte em **Session** e **Salve**. Agora vamos testar.

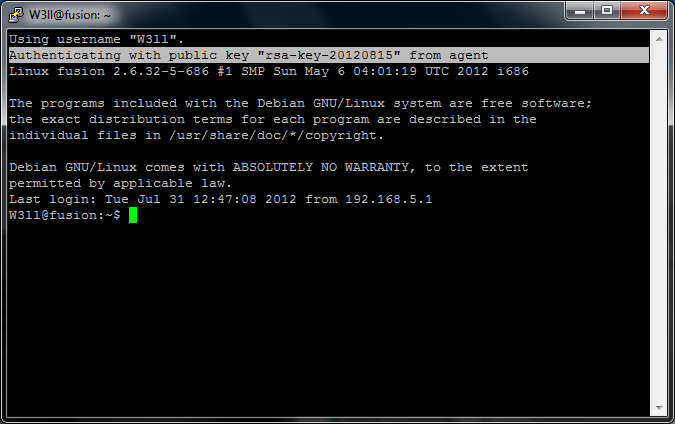


Figura 10 – Acesso Sem Senha Através do PuTTY

Como podemos observar, o **login** foi feito com sucesso através do certificado.

**Observação:** Para finalizar e garantir que os ataques de **Brute-Force** não terão sucesso, devemos mudar a variável **PasswordAuthentication yes** para **PasswordAuthentication no,** no arquivo ***/etc/ssh/sshd\_config***, barrando o uso de usuários e senhas.

**3.4 – Restringindo o Acesso ao SSH Por Grupo**

Aqui iremos restringir por grupo os usuários que poderão usar o **SSH** para administração remota.

Para isso, usaremos o **PAM**. Entre no arquivo ***/etc/pam.d/sshd*** e crie a seguinte entrada:

root@fusion:~# vi /etc/pam.d/sshd

--==[ Resumo ]==--

# RESTRIGINDO POR GRUPO

auth required pam\_listfile.so item=group sense=allow file=/etc/ssh.allow onerr=fail

--==[ Resumido ]==--

"/etc/pam.d/sshd" 42L, 1381C written

root@fusion:~#

Agora vamos criar o arquivo ***/etc/ssh.allow*** e colocar os nomes dos grupos que podem fazer login:

root@fusion:~# > /etc/ssh.allow

root@fusion:~# vi /etc/ssh.allow

# Grupos que podem fazer login

supgrp

"/etc/ssh.allow" 2L, 38C written

root@fusion:~#

Vamos à máquina cliente para testarmos a nossa regra.

W3ll@corola:~$ ssh w3ll@192.168.5.138

Password:

Password:

Password:

w3ll@192.168.5.138's password:

Permission denied, please try again.

w3ll@192.168.5.138's password:

W3ll@corola:~$ ssh W3ll@192.168.5.138

Password:

Linux fusion 2.6.32-5-686 #1 SMP Sun May 6 04:01:19 UTC 2012 i686

W3ll@fusion:~$

Pronto, se o usuário não participar de algum grupo registrado em ***/etc/ssh.allow*** ele não poderá fazer login.

**3.5 – Limitar Logins Consecutivos no SSH**

Para evitar que um único usuário seja usado por diversas pessoas, podemos restringir o número de **logins** consecutivos que este usuário pode fazer.

Consulte no arquivo ***/etc/pam.d/sshd*** é veja se o **módulo pam\_limits.so** está ativado (descomentado).

root@fusion:~# vi /etc/pam.d/sshd

--==[ Resumido ]==--

**# Sets up user limits according to /etc/security/limits.conf**

**# (Replaces the use of /etc/limits in old login)**

**account required pam\_limits.so**

"/etc/pam.d/sshd" 109L, 4665C written

root@fusion:~#

Agora vamos inserir a seguinte linha no arquivo ***/etc/security/limits.conf*** para limitar o número de sessões abertas.

root@fusion:~# vi /etc/security/limits.conf

--==[ Resumido ]==--

**#@student - maxlogins 4**

**\* hard maxlogins 2**

**# End of file**

"/etc/security/limits.conf" 58L, 2172C written

root@fusion:~#

Desta forma, limitamos o usuário a utilizar dois terminais **ssh** consecutivos.

Observação: Se você vem seguindo a série de artigos sobre hardening, o arquivo ***/etc/security/limits.conf*** já está configurado.

**4 – Proteção de Acesso Remoto**

**Referencias Bibliograficas**

**[1]** Red Hat. Disponível em: < http://docs.redhat.com/docs/pt-BR/Red\_Hat\_Enterprise\_Linux/5/html/Deployment\_Guide/s1-tcpwrappers-xinetd.html>. Acessado em: 06/08/2012.

**[2]** Red Hat. Disponível em: <http://docs.redhat.com/docs/pt-BR/Red\_Hat\_Enterprise\_Linux/5/html/Deployment\_Guide/s1-tcpwrappers-xinetd-config.html>. Acessado em: 06/08/2012.

**[3]** Mugu. Disponível em: <http://linuxnextgen.blogspot.com.br/2011/02/add-ssh-service-to-xinetd-in-rhel.html>. Acessado em: 06/08/2012.

**[4]** Ribeiro, Uira – Certificação Linux, 1º Ed, São Paulo, 2004, Axcel Books

**[5]** Manual do GNU GRUB v2. Disponível em: <http://www.gnu.org/software/grub/manual/html\_node/Security.html#Security>. Acessado em: 24/07/2012.

**[6]** Pereira, Pedro. Disponível em: < http://www.pedropereira.net/ssh-sem-autenticacao-atraves-de-certificados-rsa/>. Acessado em 14/08/2012.