دليل أحتراف خدمة SSH وطرُق حمايتها

بسم الله الرحمن الرحيم

أقرأ بأسم ربك الذي خلق، خلق الإنسان من علق،
أقـــرأ وربـــك الأكـرم، الـــذي عـلـم بـالقـلـــم،
عـــلـــــم الإنـســـــان مـــا لـــم يـعـــلم.

صدق الله العظيم

الـمـحـتـوى

عن الكاتب:

والصلاة والسلام على أفضل المرسلين سيدنا محمد وعلى اله وصحبه أجمعين

هذا العمل إهداء الى كل أبناء وطننا العربي من مشرقه الى مغربه والى كل المهتمين بالأنظمة الحرة/مفتوحة المصدر، والى المهتمين بنظام جنو/لينوكس على وجه الخصوص.

الاسم: علي الشمري

الجنسية: عراقي

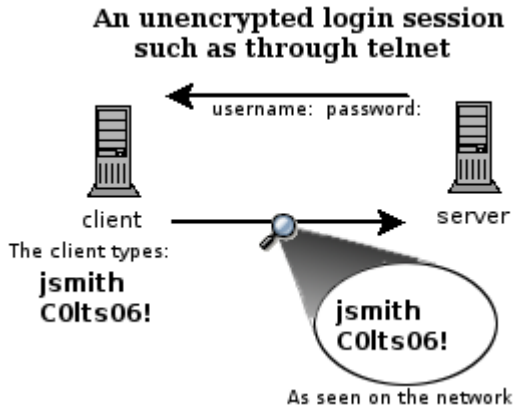
بيتي الثاني: مجتمع لينوكس العربي www.linuxac.org

اسم العضوية: B!n@ry

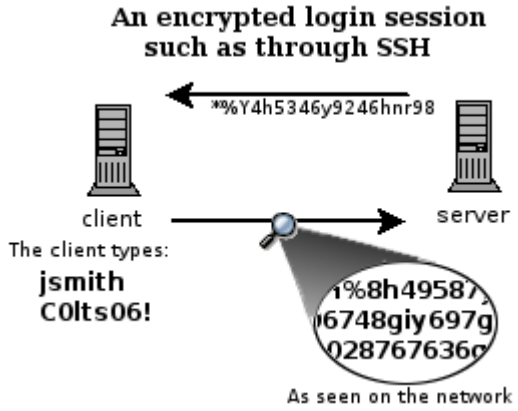
السلام عليكم ورحمة الله وبركاته

مقدمة:

أعلم يوجد ربما مواضيع كثيرة تحدثت عن الـ **SSH** لكن إن شاء الله بعد قراءة الموضوع هذا أنا متأكد ستجدون أمور أخرى تضاف لمعلوماتكم حول هذه الخدمة الرائعة بكل معنى الكلمة ... بالبداية لمن لا يعرف ما هو الـ **SSH** فهو إختصار لكلمة Secure SHell ... تم عمل هذه الخدمة من أجل إستبدال الخدمة السابقة telnet والتي يتم إرسال البيانات بين المستخدم والسيرفر بشكل مكشوف أي *Clear Text* ... يعني الكلمة السرية والبيانات المارة بينك وبين الجهة المتصل بها كلها عبارة عن كتابة مكشوفة يمكن لأي شخص أن يعمل مراقبة على هذه البيانات من خلال برامج الـ *SNIFFING* وبرامج مراقبة البيانات مثل *Wireshark* ... خدمة **SSH** أستبدلت هذه الطريقة بطريقة أكثر قوة من خلال تشفير جميع البيانات المارة بينك وبين السيرفر وبطرق تشفير قوية للغاية منها: *AES (Advanced Encryption Scheme), Triple DES, Blowfish* وغيرها من الطرق القوية ...



أنظر الصورة رقم 1 (الاتصال بالطريقة السابقة، طريقة telnet)



أنظر الصورة رقم 2 (الاتصال بالطريقة الجديدة المشفرة، طريقة ssh)

خدمة SSH تدعم طرق توثيق Authentication مختلفة:

اولاً: Host-Key Authentication
وهذه هي الطريقة السائدة عند الجميع وتعمل من خلال إستخدامك أسم مستخدم وكلمته السرية على السيرفر للإتصال بذلك السيرفر. أي ستقوم بالإتصال من خلال إستعمال أسم مستخدم مثلاً *binary* وتستعمل الكلمة السرية للمستخدم هذا للدخول الى السيرفر. ويقوم المفتاح الخاص بجهازك *Host-Key* بتشفير الخط بينك وبين السيرفر. طبعاً أعلم بإن الكثيرين من أصحاب السيرفرات خاصة يستعملون هذه الطريقة رغم إنها طريقة عادية ويوجد الأقوى منها (سنرى من خلال الشرح) وأيضاً المستخدمين يستعملونها ربما لعدم معرفتهم بغيرها.

ثانياً: Public-Key Authentication
بدل من إستخدامك لمستخدم وكلمته السرية على السيرفر للإتصال به، نقوم بإستعمال مفتاح خاص بك للإتصال بالسيرفر من خلال هذا المفتاح. ويكون للمفتاح كلمة مرور *Passphrase*. ستقوم بعمل مفتاحين واحد عام **Public** والآخر خاص **Private** كل ما عليك هو وضع المفتاح العام على السيرفر المراد الإتصال به مع الإحتفاظ بالمفتاح الخاص وعدم جعل احد يصل له. هذه الطريقة هي أفضل طريقة بصراحة ولكن تحتاج الى القليل من العمل من طرفك ليتم عملها بالشكل الصحيح، على العموم لا تقلق موضوعي يستهدف هذه الطريقة بالتحديد فتابع القراءة فقط.

ثالثاً: Passphrase-Less Authentication
هذه الطريقة هي نفس الطريقة السابقة ولكن فقط لا نقوم بوضع *Passphrase* على المفتاح الذي نقوم بعمله والسبب في ذلك هو لكي يتم إستخدام في العمليات الأوتوماتيكية *Automated* أو في السكربتات أو في وظائف الـ *cron*. لكن عيب هذه الطريقة هي في حالة حصل احدهم على المفتاح الخاص بك، يصبح بإمكانه عمل كل ما يريد في السيرفر.

الآن ربما تتسائل كيف أقوم بتركيب **SSH** ؟ أقول لا تقلق حسب ما قرأة بإنه من بعد عام 2002 أصبحت الخدمة تاتي جاهزة مع جميع التوزيعات تقريباً خاصة المعروفة منها **CentOS، Fedora، Debian، Ubuntu، Slackware** الى آخره.

طيب لنقوم بتشغيل الخدمة في البداية:

على توزيعات مثل CentOS و Fedora وأعجوبة طبعاً :) نفذ:

```
/etc/init.d/sshd start
```

الخيارات المتاحة لك هي:

```
/etc/init.d/sshd {start|stop|restart|reload|condrestart|status}
```

أما بخصوص توزيعات الدبيان وأوبنتو نفذ:

```
/etc/init.d/ssh start
```

والخيارات المتاحة مختلفة بعض الشيء:

```
/etc/init.d/ssh {start|stop|reload|force-reload|restart|try-restart}
```

ذكرت الخيارات لكم لاننا سنحتاج في الكثير من الأحيان في هذا الموضوع الى عمل إما *restart* أو الطريقة المفضلة لي هي *reload* وسنرى الفرق قريباً. تابع معي الله يسعدك عزيزي القاريء

الآن الخدمة تم تشغيلها والأمور كلها تمام على السيرفر ... الحين من جهاز آخر قم بعمل إتصال على السيرفر. الطريقة العامة هي:

```
ssh username@ip/domain
```

لنعمل تجربة:

```
ssh binary@5.5.5.5
```

سيقوم بسؤالك هل تريد إضافة هذا السيرفر الى قائمة الاجهزة المسموحة؟ قم بالإجابة حيث سيتم تسجيل المفتاح للسيرفر هذا في الملف:

```
~/.ssh/known_hosts
```

بعد ذلك سيطلب مني الكلمة السرية للمستخدم *binary*، نقوم بإدخالها وبعد أن تتحقق الخدمة من صحتها ستسمح لي بدخول السيرفر. طبعاً ظروري يا شباب تركزون على المستخدم الذي أنت تستعمله على جهازك حين تريد تتصل. مثلاً لو كان المستخدم على الجهاز الأول هو *binary* ولك مستخدم على السيرفر أيضاً أسمه binary وكنت داخل في النظام الذي عندك بالمستخدم *binary* لا ضرر من الإتصال بهذه الطريقة:

```
ssh 5.5.5.5
```

لانه سيعرف بإنك تريد تتصل من خلال المستخدم المستعمل نفسه والذي هو *binary*. طيب لو كنت بمستخدم آخر وحاولت الإتصال بالسيرفر بهذه الطريقة؟ مثلا كنت تعمل بمستخدم *mohamed* وقمت بعمل إتصال على السيرفر من خلال الأمر:

```
ssh 5.5.5.5
```

في حالة لم يكن على السيرفر مستخدم إسمه *mohamed* ؟ سيقول لك المستخدم غير موجود ويمنعك من الدخول. ولهذا التركيز الله يرضى عليكم حين يتم التطبيق ظروري جداً.

ملاحظة مهمة جداً جداً: الى الآن ما قمنا بشرحه هو الطريقة البسيطة والمعروفة Host-Key Authentication. جميع الإعدادات الإفتراضية للخدمة تسمح بهذه الطريقة ولا حاجة لك للتعديل على ملف الخدمة.

الآن لنقم بعمل إعداد للخدمة لكي تعمل على الطريقة الثانية والتي هي محل إهتمامي في هذا الموضوع، أي Public-Key Authentication. سنحتاج الى التعديل على الملف الخاص بالخدمة، ولكن قبل ذلك لنقوم بعمل إنشاء للمفتاح الذي سنقوم بإستعماله ومن ثم نسخه وبعد ذلك نرجع للتعديل.

الآن قم بالدخول بالمستخدم الذي ستقوم بإستعماله (في الغالب هو المستخدم الذي أعتدت على إستعماله) لكي نقوم بعمل المفتاح، مثلاً بالنسبة لي سأقوم بإستعمال الأسم *binary*. أفتح الطرفية على جهازك *CLIENT* ومن ثم نفذ عليه الأمر:

```
ssh-keygen -t rsa
```

أو

```
ssh-keygen -t dsa
```

ملاحظة: بإمكانك إستعمال اي طريقة تشاء من هذه الطرق، كلاهما يعمل مفاتيح قوية جداً لكن الفرق بينهما هو الأول يعمل

المفاتيح الى حد أقصاه *2048bit* والثاني الى حد أقصاه *1024bit*.

أنا سأكمل مع المفتاح الأول أو النوع الأول *RSA*. أضغط على *ENTER* وأكمل سيظهر لك كلام يشبه التالي:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/binary/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/binary/.ssh/id_rsa.
Your public key has been saved in /home/binary/.ssh/id_rsa.
The key fingerprint is:
02:09:09:09:ee:cc:dd:4d:3d:3a:66:ff:ab:df:34:11 binary@binary-zone.com
```

طبعاً الـ *fingerprint* بدون شك لك سيكون أرقام مختلفة، ومكان التخزين للمفتاح سيختلف في حالة كان المستخدم المستعمل عندك غير *binary* سيكون مثلا بالنسبة لـ *mohamed* التالي

```
/home/mohamed/.ssh/id_rsa
```

وهكذا. عند وصولك الى طلب إدخال الـ *passphrase* قم بوضع عبارة مرور للمفتاح هذا ولا تنسى ولا أريد أن أوصيك بأن تستعمل عبارة معقدة

هكذا قمنا بعمل المفتاحين العام والخاص للمستخدم *binary* طيب ماذا علينا عمل الحين؟ الأمر بسيط جداً قم بتنفيذ التالي:

```
scp ~/.ssh/id_rsa.pub binary@5.5.5.5:.ssh/authorized_keys
```

هذا سيقوم بعمل نسخ للمفتاح العام الخاص بك الى السيرفر في الملف *authorized_keys* الموجود في المجلد

```
/home/binary/.ssh/
```

الآن قم بتنفيذ الأمر التالي:

```
chmod 600 /home/binary/.ssh/id_rsa.pub
```

وهذا:

```
chmod 400 /home/binary/.ssh/id_rsa
```

هكذا منعنا أي شخص غير المستخدم *binary* من قراءة الملف *id_rsa.pub* والذي هو المفتاح العام والملف *id_rsa* الذي هو المفتاح الخاص.

الآن تم عمل المفتاح وتم نسخه ماذا تبقى يا **B!n@ry؟** لم يبقى الكثير. الآن هل أنت جاهز للتعديل على ملف إعدادات الخدمة؟ أذن توكل على الله وأبدأ معي

```
vim /etc/ssh/sshd_config
```

أذهب الى السطر الموجود فيه العبارة هذه **PasswordAuthentication** وقم بتعديلها الى التالي:

```
PasswordAuthentication no
```

الآن ما قمنا به هو منعنا الدخول الى السيرفر من خلال إستعمال اسم المستخدم وكلمته السرية الموجودة على السيرفر وسمحنا

بإستعمال الطريقة الثانية Public-Key Authentication. الآن أضغط على *Escape* ومن ثم : ومن ثم *x*

هكذا خرجنا من المحرر *vim* مع تخزين التغييرات وعدنا الى الطرفية، الآن نأتي الى سبب ذكري للخيارات المتاحة في طريقة التشغيل/أعادة التشغيل/إيقاف الخدمة. حيث الآن بإمكانك عمل إعادة تشغيل للخدمة. على توزيعات CentOS و Fedora وأعجوبة طبعاً :) نفذ:

```
/etc/init.d/sshd restart
```

أما على توزيعات الدبيان وأوبنتو نفذ:

```
/etc/init.d/ssh restart
```

لكن بالنسبة لي أفضل طريقة الـ *reload* وذلك لاني لا أحتاج الى عمل إيقاف للخدمة بشكل كامل وإنما أقوم بإخبارها بإني أريدها أن تعيد قراءة ملف الإعدادات الخاص بها، وتحميله الى الذاكرة بدل الاعدادات التي تم تحميلها وقت تشغيل الخدمة. ولهذا سيكون حلي دائماً بعد التعديل على الإعدادات هو *reload*. على توزيعات CentOS و Fedora وأعجوبة طبعاً :) نفذ:

```
/etc/init.d/sshd reload
```

أما على توزيعات الدبيان وأوبنتو نفذ:

```
/etc/init.d/ssh reload
```

الآن أنتهينا من عمل المفاتيح، نسخ المفتاح العام الى السيرفر الثاني وقمنا بعمل إعدادات الخدمة وتشغيلها، الآن لنقوم بالتجربة في الدخول بالطريقة الجديدة. نفذ ما يلي:

```
ssh binary@5.5.5.5
```

ستظهر لك عبارة شبيهة بالتالي:

```
Enter passphrase for key 'id_rsa':
```

هنا ما يحصل بالحقيقة، هو جهازك يقوم بطلب معرفة هوية مستخدم هذا المفتاح وذلك من خلال طلب عبارة المرور التي أدخلتها حين قمت بعمل المفتاح. قم بإدخال هذه العبارة ومن ثم أضغط على *enter* وهكذا تكون قد وصلت الى السيرفر ولكن من خلال المفتاح العام الخاص بك.

سؤال: الآن ربما يسأل سائل كيف اعمل الطريقة الثالثة Passphrase-Less Authentication؟

الجواب: اعمل نفس الطريقة الثانية فقط لا تضع كلمة مرور على المفتاح حين تقوم بعملهم وانتهى الموضوع باقي الأمور جميعها نفس الشيء.

سؤال: طيب يا **B!n@ry** ماذا لديك أكثر حول الخدمة هذه؟
الجواب: بالحقيقة يوجد الكثير، لكن أحب أن أضيف أمر أخر وهو إستعمال مفاتيح متعددة.
ربما هنا يتفاجيء أحدكم ويتسائل كيف يعني مفاتيح متعددة؟
الجواب: هو أن أستعمل مفتاح معيين للوصول الى السيرفر الأول ومفتاح آخر للوصول الى السيرفر الثاني وهكذا.

سؤال: وهل هذا ممكن يا **B!n@ry**؟
الجواب: نعم ممكن تابع معي

الآن لنقم بعمل مفتاح جديد لنفرض للإتصال بسيرفر ثاني لك وعلى سبيل المثال يحمل رقم *IP* هو *6.6.6.6* ولكن هذه المرة بالطريقة التالية:

```
ssh-keygen -t rsa -f id_server2
```

وأكمل باقي خطوات عمل المفتاح من وضع عبارة المرور الى آخره.

سؤال: طيب الحين صار عندنا في المجلد

```
/home/binary/.ssh
```

مفاتيح أثنين، كيف أقوم بإخبار السيرفر الثاني أنا أريد أتصل عليك من خلال المفتاح *id_server2* وليس المفتاح *id_rsa* ؟

الجواب: قم بتنفيذ الأمر التالي:

```
ssh -i id_server2 binary@6.6.6.6
```

هكذا قمت بإستعمال المفتاح الثاني للوصول الى سيرفرك الثاني *6.6.6.6* وسيطلب منك عبارة المرور *passphrase* للمفتاح الثاني التي أدخلتها حين عملت المفتاح.

سؤال: طيب **B!n@ry** أريد تغيير عبارة المرور للمفتاح شلون؟

الجواب: بسيط يا عزيزي، لو كنت تريد تغيير العبارة للمفتاح *id_rsa* نفذ التالي:

```
ssh-keygen -p -f /home/binary/.ssh/id_rsa
```

أما لو كانت للمفتاح الجديد *id_server2* نفذ التالي:

```
ssh-keygen -p -f /home/binary/.ssh/id_server2
```

سيطلب منك وضع عبارة المرور القديمة ومن ثم وضع عبارة جديدة للمفتاح.

شايفين شلون الأمر بغاية البساطة ؟؟؟

الآن لا اريد أن أقوم بذكر جميع التفاصيل هناك أمرين سأتركهما لكما للتفكير وربما لكي يقوم أحدكم بإضافتهم الى الموضوع.

الأول: كيف أستطيع أن أعرف ما هو الـ *fingerprint* الخاص بالمفتاح الذي عندك؟

الثاني: كيف أستطيع تنفيذ أمر معيين على السيرفر من دون الحاجة الى تسجيل الدخول على السيرفر؟

الجواب: القراءة + التجربة

آخر حاجة في موضوعي هذا. الآن نأتي الى أمور أخرى أحب أضيفها لعمل لنقول **Hardening** لخدمة الـ **SSH** بصورة أكثر وتعقيد أكثر (حال المخترق) ما سأقوم به هو السماح لجهاز معيين بالدخول الى الخدمة سواءاً من خلال *iptables* أو من خلال *xinetd*.

طريقة iptables:

```
/sbin/iptables -A INPUT -p tcp -s 5.5.5.5 --dport 22 -j ACCEPT;
```

هكذا سمحت لمن معه المفتاح ويعمل من جهاز رقم الـ *IP* له هو *5.5.5.5* بالوصول الى خدمة **ssh** التي على سيرفري.

طريقة xinetd:

```
vim /etc/hosts.allow
```

بعد ذلك ضع التالي:

```
sshd: 5.5.5.5: ALLOW
```

بعد ذلك أعمل:

```
vim /etc/hosts.deny
```

وضع بداخله التالي:

```
sshd: ALL : DENY
```

هكذا سمحنا فقط للشخص الذي يتصل من الـ *IP* رقم *5.5.5.5* بالإتصال بالخدمة وبما إننا نستعمل المفاتيح للإتصال، يجب أن يكون معه المفتاح للإتصال. بهذا عقدنا الأمور أكثر على المخترق (مع العلم بالحقيقة يمكنك عمل *spoof* للـ *IP* لكن مع هذا التعقيد جيد).

الآن لزيادة الحماية أكثر لنقم بتحرير ملف الخدمة

```
vim /etc/ssh/sshd_config
```

وتستطيع وضع الخيارات التالية:

```
ListenAddress 5.5.5.5
PermitRootLogin no
Protocol 2
AllowUsers binary mohamed
AllowGroups admins
Port 5858
```

ما عملناه هو:

- في السطر الأول حددنا بإن الخدمة تستمع للطلبات القادمة من العنوان 5.5.5.5 فقط. أي تضع رقم الأي بي لسيرفرك نفسه، وليس للجهاز صاحب هذا الـ *IP* الذي تريد السماح له بالدخول.
- في السطر الثاني منعنا الدخول مباشرة الى الحساب root، إذا كنت تريد تستعمل root على السيرفر؟ أدخل بالمستخدم العادي ثم حول الى المستخدم root. أكيد تعرفون كيف؟ :)
- السطر الثالث بالغالب هو الأساسي أحببت تأكيده وهو إستعمال الخدمة SSH-2 وليس SSH-1.
- السطر الرابع سمحنا للمستخدمين binary و mohamed فقط بالدخول الى الخدمة.
- السطر الخامس سمحنا فقط للمستخدمين الذين يقعون في مجموعة إسمها admins من إستعمال الخدمة.
- في السطر السادس زيادة خير أننا قمنا بتغيير المنفذ الأصلي للخدمة ولكن لو وضعت هذا الرقم 5858؟ لا تنسى تقوم بتغييره في الأمر الذي كتبناه بالاعلى الخاص بـ iptables من 22 الى 5858 أوك؟

الآن لكي تتأكد من عدم وجود مشاكل لديك في ملف الإعدادات؟ قم بتنفيذ التالي:

```
`which sshd` -t
```

حيث سيقوم بفحصل الاعدادات، وإذا كانت لديك مشاكل فيه، سيقوم بكتابتها لك على الطرفية، وإن لم يكن لن يظهر لك شيء.
بهذه الحماية للخدمة؟ أنت مش بس عملت **Harden** للخدمة لديك وإنما كرهت المخترق بحياته وكرهته بنفسه لو يفكر يصل اليك.

سؤال: طبعاً تريد أكثر من هذه الحماية؟
الجواب: موجود يا عزيزي ، في شي إسمه **Port-Knocking** وفي شيء إسمه **SPA** وفي شي سأشرحه قريباً (أو أي متبرع) وهو **DenyHosts**.

سؤال: تريد أكثر؟ خبرني؟
الجواب: أيضاً موجود ولكن هنا أقول لك اسئل Google

أتمنى تكونوا أستفدتم من الموضوع، وأن يكف الناس عن إستعمال طريقة Host-Key Authentication ويقوم بإستعمال طريقة Public-Key Authentication الرائعة والممتازة. ورمضان مبارك وإفطار شهي إن شاء الله ...

مشكلة قد تواجهك إذا عملت أعدادات خاطئة على ملفات

```
/etc/hosts.allow
```

و

```
/etc/hosts.deny
```

وهي ربما لم تقم بوضع رقم الـ IP لجهازك أو الجهاز الذي تريد السماح له بالوصول الى خدمة SSH وبالتالي ستظهر لك الرسالة التالية عند محاولتك الإتصال بالسيرفر:

```
ssh_exchange_identification: Connection closed by remote host
```

هذه الرسالة لتلافي الحصول عليها؟
قم بالتأكد من إعداداتك في الملفات التي ذكرتهم بالأعلى

سؤال: عندك أكثر من سيرفر وعندك لكل سيرفر مستخدم مختلف، وكل سيرفر تستعمل أعدادات مختلفة للدخول عليه، يعني تريد تستعمل المستخدم *binary* للإتصال بسيرفر إسمه **serv1** ورقم الـ *IP* له هو **5.5.5.5** وتريد تستعمل مثلاً *mohamed* للإتصال بسيرفر إسمه **serv2** ورقم الـ *IP* له **6.6.6.6** وكل سيرفر له إعدادات مختلفة مثلاً الأول يستعمل منفذ port **22** بينما الثاني يستعمل منفذ **2222**، شلون تسهل العمل عليك؟ خاصة وإنه من المعروف لو حاولت الإتصال بالسيرفر بدون خيارات سيقوم بإستعمال المنفذ **22** والذي هو المنفذ الأساسي، أذن الحل شلون ؟؟؟؟

الجواب: قم بوضع ملف الأعدادات الخاصة بكل سيرفر داخل مجلد **ssh**. أي:

```
~/.ssh/
```

مثلا للإتصال بالسيرفر الأول ومنفذه **22** بواسطة المستخدم *binary* أضع ملف أعداداته في المجلد، وأعطيه أسم، مثلاً **serv1** لاحظ:

```
~/.ssh/serv1
```

ولكي أضع خيارات هذا السيرفر، قم بتحرير الملف **serv1**:

```
vim ~/.ssh/serv1
```

وضع فيه التالي:

```
IdentityFile ~/.ssh/serv1
Port 22
User binary
```

الآن ونعمل ملف أعدادات للمستخدم *mohamed* كالتالي:

```
vim ~/.ssh/serv2
```

ونضع فيه التالي:

```
IdentityFile ~/.ssh/serv2
Port 2222
User mohamed
```

الآن للدخول الى السيرفر الأول بواسطة المستخدم *binary* ننفذ الأمر:

```
ssh -F serv1 5.5.5.5
```

أي خبرنا **ssh** بأن يستعمل الإعدادات الموجودة في الملف **serv1** وهكذا ...

وللإتصال بالسيرفر الثاني بواسطة المستخدم *mohamed* نفذ الأمر:

```
ssh -F serv2 6.6.6.6
```

وهنا خبرنا **ssh** بأن يستعمل الإعدادات الموجودة في الملف **serv2** وهكذا ...

ملاحظة: طبعاً لا تنسى تفتح منفذ بالجدار الناري للمنفذ الجديد **2222**.

تستطيع التأكد من خلال ناتج امر **netstat** التالي:

```
netstat -a --tcp -p | grep ssh
```

سؤال: الآن ربما يتساءل البعض لنفرض عندي مجموعة مستخدمين على الجهاز وكلنا على هذا الجهاز نستعمل إعدادات ثابتة للجميع، هل أقوم بعمل ملف لكل مستخدم وأضعه في المجلد **ssh**. ؟
الجواب: بدون شك لا ...

سؤال: طيب والحل ؟

الجواب: قم بوضع الخيارات العامة التي تريدها في الملف

```
/etc/ssh/ssh_config
```

سؤال: لماذا هذا الملف يا **B!n@ry** ؟

الجواب: لأن هذا هو الملف الخاص بالإعدادات العامة للـ **Client** ... أما الملف

```
/etc/ssh/sshd_config
```

فهو الملف الخاص بالأعدادات العامة للخدمة **SSH** نفسها، أي حين تعمل هي كسيرفر.

إن شاء الله تفيدكم هذه الإضافة ...

طريقة تنفيذ الأوامر من دون الدخول الى السيرفر:

بعض الأحيان تريد أنت كمدير للسيرفر أن تنفذ أمر ما على السيرفر (backup، إعادة تشغيل، فحص، الى آخره) ولكن من دون الحاجة الى الدخول الى السيرفر ومن ثم تنفيذ الأمر. هذه المسألة مع **OpenSSH** ممكنة، كل ما عليك فعله هو:

```
ssh binary@5.5.5.5 sudo /etc/init.d/httpd restart
```

في المثال أعلاه فرضنا إنه هناك خلل في خدمة الأباتشي *httpd* ونريد إعادة تشغيلها من دون أن أدخل ومن ثم أعيد تشغيلها ...

مثال آخر:

```
ssh binary@5.5.5.5 tar cvf binary-backup.tar /home/binary
```

هنا سيقوم بأخذ نسخة إحتياطية للمجلد الخاص بالمستخدم *binary* ويضعه في ملف إسمه *binary-backup.tar* ... يعني كما تلاحظون هذه الأوامر هي نفسها التي ننفذها على السيرفر أو على اجهزتنا في حالة وجودنا على هذا السيرفر ... وبسبب كوننا ننفذ أمر واحد، فلا ضرر من تنفيذه عن بعد

سؤال: أنا مدير نظام لأكثر من خادم، وعندي العديد من المفاتيح، كل خادم مفتاح خاص. ألا يوجد طريقة لكي أستطيع تمييز مفتاح عن آخر؟

الجواب: أكيد موجود ... كل ما عليك فعله هو، عندما تقوم بعمل المفتاح قم بوضع تعليق *comment* عليه من خلال تنفيذك لأمر إنشاء المفتاح بالطريقة التالية:

```
ssh-keygen -t rsa -C "binary on serv1"
```

بعدما نفتذت عملية الإنشاء بهذه الطريقة ... ستجد في آخر المفتاح المكون لديك التعليق:

```
binary on serv1
```

سؤال: يا أخ **B!n@ry** أنا عندي ملفات كثيرة أود تحريرها على السيرفر، أو عندي شغل كثير أود القيام به، ولا اريد أستعمال تمرير **X** من خلال **SSH**، ولا اريد الإتصال بالسيرفر والعمل عليه مباشرة. ألا يوجد لديك طريقة أخرى؟

الجواب: نعم، كله موجود، نقوم بعمل *mount* للمجلد أو نظام الملفات *filesystem* الموجود على السيرفر ونربطه عندنا على جهازنا. أنا متأكد ليس الكل يعرف عن هذه الميزة.

سؤال: وكيف نقوم بعمل ذلك؟

الجواب: تابع معي. أول حاجة لنقوم بعمل مجلد جديد على جهازنا لنستعمله في عملية ربط أو *mount* مجلد على السيرفر **serv1** بجهازنا:

```
mkdir /mnt/serv1
```

الآن قم بتركيب كل من **fuse-utils** و **sshfs** الذي سنستعمله في عملية *mount* للمجلد الخارجي على الجهاز الداخلي عندنا. لتركيبهم على دبيان أو أوبنتو:

```
sudo apt-get install fuse-utils sshfs
```

بعد ذلك لنتأكد من عمل fuse بداخل النواة. نفذ التالي:

```
lsmod | grep fuse
```

إذا لم تجده في النواة نفذ الأمر التالي:

```
modprobe fuse
```

سؤال: الآن لنفرض لدينا على السيرفر المجلد الخاص بالمستخدم *binary* ونريد أن نعمل له mount على جهازنا، ماذا نفعل؟

الجواب: نفذ الأمر التالي:

```
sshfs binary@5.5.5.5: /mnt/serv1
```

أو نفذ الأمر (كلاهما نفس الشيء، فقط لمن لم يفهم ماذا تعني الرمز :):
كود:

```
sshfs binary@5.5.5.5:/home/binary/ /mnt/serv1
```

الحين أذهب الى المجلد:
كود:

```
mnt/serv1/
```

وهوووووبا ستجد الملفات التي هي على السيرفر موجودة عندك على جهازك

سؤال: كيف أعمل *umount* للمجلد؟

الجواب: نفذ الأمر:

```
sudo umount /mnt/serv1
```

تمرير برامج X من خلال SSH

بعض الأحيان تكون أنت في مكان والسيرفر المراد الإتصال عليه في مكان آخر ... أمر طبيعي ... وأمر طبيعي إنك ستستخدم أحد برامج الإتصال بالسيرفر بشكل remotely ... مثل rdesktop و vnc و rlogin و telnet وأخيراً ssh ... طيب ماذا لو كنت تريد أن تعرض الواجهة الرسومية لبرنامج معيين على اللاب توب الخاص بك، ومن دون أن تشبك من خلال rdesktop أو vnc مثلاً ؟ حيث هذه البرامج تسحب لك الشاشة الخاصة بالسيرفر كلها وليس برنامج معيين ... أيضاً تريد تتأكد من أمان الإتصال في نفس الوقت وأمان عرض الواجهة ؟ ما هو العمل ؟

الجواب بسيط جداً نستخدم خاصية إسمها: SSH Tunneling ... إي إننا من خلال النفق أو المسار الذي يستعمله برنامج SSH نقوم بسحب الواجهة لبرنامج معيين ... طيب كيف يا B!n@ry ؟ الحل هنا:

```
ssh -X user@domain.com
```

هنا نريد أن نشبك على سيرفر يمكننا الوصول له بإسم domain.com وللدخول عليه يوجد مستخدم إسمه user ... الآن لسحب الواجهة من داخل هذا السيرفر وضعنا الخيار X في البداية والذي يشير الى إنني أريد أستعمل خاصية الـ X Forwarding ... الآن بعد دخولك الى السيرفر قم بتشغيل أي برنامج ذات واجهة رسومية كالتالي:

```
gedit &
```

سيظهر البرنامج gedit على الشاشة الخاصة بجهازك ولكن البرنامج فعلياً يعمل على السيرفر

طريقة أخرى للشبك هي كالتالي:

```
ssh -X user@IP-Address
```

حيث هنا أستعملنا الـ IP Address وليس دومين معيين للوصول الى السيرفر ... أيضاً بعد أن تتم عملية الدخول الى السيرفر شغل أي برنامج تريد كالتالي:

```
gcalctool &
```

سيظهر على سطح مكتبك برنامج الـ gcalctool أي الآلة الحاسبة والتي هي فعلياً تعمل على السيرفر ... إن شاء الله يكون موضوع مفيد للجميع وخاصة الـ SysAdmin ...

إستعمال sshguard للتصدي لهجمات Brute Force على خدمة SSH

بسبب كثرت محاولات الإختراق من نوع Brute Force التي تتم على خدمة SSH فإن *sshguard* وجد ليتصدى لهذه النوعية من الهجمات ... يقوم *sshguard* بمراقبة الـ LOG وحين يرى محاولات متكررة من IP معيين للدخول أو الشبك على خدمة الـ SSH يقوم بعمل BLOCK للـ IP الخاص بذلك الشخص الذي يحاول ويكرر محاولة الدخول الغير شرعية للخدمة ...

لتركيب البرنامج على توزيعة أوبنتو:

```
sudo apt-get install sshguard
```

بالنسبة للتوزيعات الأخرى قم بتحميل البرنامج من هنا --> أضغط
بعد التحميل قم بتركيبه بالطريقة المعتادة

ما سأقوم بشرحه هو طريقة ربطه مع خدمة التسجيل **syslog-ng** بحيث يصير يعتمد على سجلاته ولو تريد ربطه مع نوع آخر أذهب الى الرابط هذا --> دوس هنا

الآن لكي نقوم بتمرير السجلات **LOGS** من **syslog-ng** الى ***sshguard*** قم بتحرير الملف التالي:

```
vim /etc/syslog-ng/syslog-ng.conf
```

الآن قم بوضع الكود التالي في الملف الذي قمت بتحريره بالأعلى:

```
# pass only entries with auth+authpriv facilities that contain sshd
        filter sshlogs { facility(auth, authpriv) and match("sshd"); };
        # pass to this process with this template (avoids <ID> prefixes)
        destination sshguardproc {
           program("/usr/local/sbin/sshguard"
              template("$DATE $FULLHOST $MESSAGE\n"));
        };
        log { source(src); filter(sshlogs); destination(sshguardproc); };
```

الآن قم بحفظ الملف وغلقه ونريد من خدمة **syslog-ng** بقراءة التغييرات التي حصلت على ملف الإعداد الخاص بها، قم بعمل التالي:

```
killall -HUP syslog-ng
```

أو قم بعمل:

```
sudo /etc/init.d/syslog-ng reload
```

الآن لكي تتأكد بإن ***sshgaurd*** جالس يعمل قم بتنفيذ الأمر:

```
ps ax | grep sshguard
```

جميل الآن كل شي جاهز باقي نضيف الرووولز/القوانين الخاصة بالـ **netfilter** أي **iptables**
أول حاجة نعملها هي عمل **CHAIN** جديدة خاصة بـ ***sshgaurd*** لكي نستعملها في تمرير جميع الباكتس المارة الى **SSH** من خلالها... نفذ عزيزي القاريء :

```
iptables -N sshguard
```

بعد ذلك نريد أن نقوم بتمرير جميع البيانات المتجهة الى **SSH** أي المنفذ **Port** رقم **22** الى السلسلة **CHAIN** التي عملناها بالأعلى ***sshguard***، نفذ يا طيب معي:

```
iptables -A INPUT -p tcp --dport 22 -j sshguard
```

أهم شيء تأكد بإن القوانين الأساسية للسلاسل عندك هي **DROP** وليست **ACCEPT** أوك ؟

الآن أي محاولة متكررة للدخول الى نظامك من خلال خدمة **SSH** سيتم عمل منع **BLOCK** لها من خلال ***sshgaurd***

حماية خدمة SSH بواسطة DenyHosts

وذكرت في آخر الموضوع بإني سأقوم بشرح طرق حماية هذه الخدمة بالتحديد. في نفس الموضوع ذكرت طرق عديدة لحماية الخدمة واليوم سنتكلم عن إضافة أخرى نضيفها لزيادة الحماية على خدمة SSH. الموضوع هذا يتحدث عن حماية SSH بواسطة DenyHosts.

قد يستغرب البعض حين أقول له بإن مجرد وصول الـ *BOX* (سيرفر) الخاص بك على النت *Online*، ستبدأ المشاكل وتبدأ محاولات الإختراق من قبل المخترقين، ولكن هذه هي الحقيقة. لو قمت بعمل خادم SSH مثلاً يستطيع الناس أو العالم الخارجي الوصول له من خلال الأنترنت (عن بُعد)، أنصحك بمتابعة ومراقبة السجلات الخاصة بالوصول وعمل *Access* على خادم SSH التي في *RHEL*، *Fedora*، *CentOS* و أعجوبة مثلاً هي:

```
/var/log/secure
```

وعلى دبيان وأوبنتو:

```
/var/log/auth.log
```

قم بمراقبتهم بالأمر:

```
sudo tail -f /var/log/secure
```

أو

```
sudo tail -f /var/log/auth.log
```

ستجد هناك العديد من السطور تبيين محاولات متعددة للوصول الى هذه الخدمة. إذا لم تكن أنت صاحب هذه الـ *IP's* ولا هذه المحاولات فمن هو؟ إنهم المخترقون :)

الحين ربما تتسائل مع نفسك وتقول لي: معقولة وصلوا هؤلاء الى الـ *BOX* الخاص بي بهذه السرعة؟ يا **B!n@ry** لم يمضي على وجود الـ *BOX* على النت سوى ساعات أو أيام معدودة، معقولة؟

الجواب: يا عزيزي للأسف نعم معقولة ونص وثلاث أرباع
هؤلاء يستعملون أدوات تنفذ بشكل أوتوماتيكي أو تلقائي ويتم توجيهها على *IP Range* معيين مثلاً وهي تقوم بمحاولات دخول عشوائية إستناداً الى قواعد بيانات لديها بأسماء مستخدمين وكلمات سرية. الطريقة هذه الأغلب يعرفها وهي ما يسمى بالـ *Brute Force* ولكن بشكل *Automated Brute Force Attack*.

سؤال: عندك حلول من تقليل المخاطر؟
الجواب: نعم، تابع معي يا عزيزي القاريء.

في البداية هذا الموضوع سأشرح فيه كيفية تركيب وإعداد خدمة **DenyHosts** من خلال ملفات السورس لها، وليس من خلال *rpm* أو *yum* أو *apt-get* و *dpkg*. وهي سهلة جداً فقط خلك مركز معي الله يرضى عليك \

الخطوة الأولى: تحميل ملفات DenyHosts والتأكد من وجود Python.
قم بتحميل الملفات من خلال زيارة الموقع التالي: DenyHosts-2.6.tar.gz

الآن لنتأكد من *Python*، نفذ التالي:

```
python -V
```

سيعطيك رقم النسخة المستعملة من البايثون. إن كنت لم تركب البايثون فقم بذلك، لأنه حاجة أساسية لعمل الخدمة هذه.

الخطوة الثانية: فك الضغط مع تغيير الأسماء وتهيئة ملف الإعدادات.
أنقل الملف الذي قمت بتحميله الى المجلد المراد تشغيل الخدمة منه، وليكن:
كود:

```
mv /path2/DenyHosts-2.6.tar.gz /usr/share/
```

بعد ذلك:

```
cd /usr/share/
```

بعد ذلك لنقم بفك الضغط:

```
tar xvfz DenyHosts-2.6.tar.gz
```

بعد ذلك:

```
cd DenyHosts-2.6/
```

ومن ثم نفذ الأوامر التالية:

```
mv denyhosts.cfg-dist denyhosts.cfg
```

الخطوة الثالثة: عمل الإعدادات اللازمة.
الآن لنقم بتحرير الملف الخاص بالاعدادات والذي اسمه denyhosts.cfg:

```
vim denyhosts.cfg
```

الآن تأكد من وضع الاعدادات التالية (سأشرح كل واحدة لا تقلق):

```
WORK_DIR = /usr/share/denyhosts/
HOSTS_DENY = /etc/hosts.deny
BLOCK_SERVICE = sshd
SECURE_LOG = /var/log/secure
DENY_THRESHOLD_INVALID = 2
DENY_THRESHOLD_VALID = 5
DENY_THRESHOLD_ROOT = 2
LOCK_FILE = /var/lock/subsys/denyhosts
HOSTNAME_LOOKUP=YES
AGE_RESET_VALID=5d
AGE_RESET_INVALID=
AGE_RESET_ROOT=10d
DAEMON_PURGE = 10d
DAEMON_SLEEP = 10m
DAEMON_LOG = /var/log/denyhosts
DAEMON_LOG_TIME_FORMAT = %b %d %H:%M:%S
SUSPICIOUS_LOGIN_REPORT_ALLOWED_HOSTS=YES
ADMIN_EMAIL = root@localhost
```

الآن لنقوم بتوضيح كل سطر، بالرغم من كونهم موضحين في ملف الاعداد، إلا إنه سأوضحهم للفائدة العامة ولمن لا يجيد اللغة الأنجليزية.

- **السطر الأول** يتم تحديد المسار الموجود بداخله ملفات الخدمة/السكربت DenyHosts والذي في مثالنا هذا هو

```
/usr/share/denyhosts/
```

- **السطر الثاني** يتم تحديد المسار الخاص بملف **hosts.deny** الذي سيتم تسجيل الـ *IP's* الممنوعة فيه.

- **السطر الثالث** حددنا الخدمة التي نريد نعمل حماية عليها.

- **السطر الرابع** حددنا مكان تسجيل الدخول لخدمة **SSH** كما ذكرت بالأعلى هذا المسار هو لتوزيعات *RHEL,Fedora,CentOS* و أعجوبة فقط. إذا كنت تريد تحديد المسار لتوزيعة أوبنتو أو دبيان أقرأ المسار كتبته بالاعلى :)

((الآن ركز معي أكثر الله يرضى عليك))

- **السطر الخامس** نقوم بتحديد بعد كم محاولة فاشلة للدخول بواسطة مستخدم غير موجود تريد تسجيل منع ذلك الـ *IP*؟ هنا حددنا 2 وذلك لأنه إذا المستخدم ليس موجود ليش أخله يعيد محاولات عديدة.

- **السطر السادس** نقوم بتحديد بعد كم محاولة فاشلة للدخول بواسطة مستخدم موجود على الـ *BOX* لديك، تقوم بمنعه؟ أعتقد 5 قيمة عادلة، يعني مو معقولة الإنسان ناسي كلمة المرور له وخمس محاولات لا يتذكرها، لو صحيح عليه طلب الكلمة من مدير الـ *BOX* أحسن له.

- **السطر السابع** نقوم بتحديد بعد كم محاولة فاشلة للدخول بواسطة المستخدم *root* على الـ *BOX* لديك تريد منعه؟ بالنسبة لي وضعت هذا الخيار لمن يسمح بدخول *root* بشكل مباشر. لمن لا يسمح بدخول *root* بشكل مباشر لا تضع شيء.

- **السطر الثامن** حددنا فيه مسار الـ *Lock File* للخدمة، في الغالب سيكون في المسار الذي وضعته، إن لم يكن راجع ملف الإعدادات أو عليك أن تكون على دراية بمكانه في توزيعتك، لكن بالغالب هذا هو.

- **السطر التاسع** نحدد فيه هل نريد أن يتعمل عمل *Hostname Lookups* أم لا، لا أنصح به لانه ربما يسبب بطأ في الشبكة عندك.

- **السطر العاشر** في حالة تم منع مستخدم موجود من الدخول بسبب محاولاته المتكررة الفاشلة، فقم برفع الحجب عنه بعد مدة معينة. في الاعدادات التي عملتها أنا وضعت يوم واحد. ضع ما تشاء.

- **السطر الحادي عشر** أتركه كما عملت أنا، وذلك يعني بإنه لن يتم رفع الحجب عن الـ *IP's* تلك ابداً.

- **السطر الثاني عشر** يخص المستخدم *root* ومتى تريد رفع الحجب عن محاولات دخوله الفاشلة. أنا وضعتها هنا 10 أيام، كما قلت لك لو كنت لا تسمح بوصول *root* لخدمة **SSH** اصلا فلا حاجة لوضع هذا الخيار.

- **السطر الثالث عشر** تحدد فيه بعد كم من الزمن تريد أن يتم مسح جميع السطور الخاصة بمنع *IP's* من ملف **hosts.deny**، وذلك لأنه ممكن يصبح الملف كبير جداً مع المدة.

- **السطر الرابع عشر** تحدد فيه متى تريد الخدمة/السكربت أن يتم تنفيذها. أنا قمت بتحديد عملها بعد كل 10 دقائق. لا تقلق السكربت خفيف كما لاحظت ولهذا كل 10 دقائق عملها لن يؤثر على أداء باقي الخدمات التي عندك على الـ *BOX*.

- **السطر الخامس عشر** حددنا أين يتم تسجيل الأمور التي تتعلق بالخدمة نفسها.

- **السطر السادس عشر** حددنا في طريقة تسجيل التواريخ الخاصة بالخدمة.

- **السطر السابع عشر** أخبرنا الخدمة بإننا نريد تقرير حول العمليات المشبوهة التي حصلت.

- **السطر الثامن عشر** حددنا اسم البريد الذي نرسل له التقارير.

الخطوة الرابعة: تنصيب وتشغيل السكربت.
الآن لتشغيل السكربت هذه علينا بعمل تنصيب لها. لهذا قم بتنفيذ الأمر (بصلاحيات *root*):

```
python setup.py install
```

هذه الخطوة ستقوم بعمل نسخ وتنصيب للمودويلات الخاصة بالخدمة في مجلد *site-packages* الخاص بالبايثون.

خطوة مهمة قبل التشغيل، قم بإضافة رقم الـ *IP* أو الدومين الذي تريد السماح له بالوصول الى خدمة **ssh** في ملف **hosts.allow** قبل التنفيذ. مثلاً:

```
echo "sshd: 5.5.5.5" >> /etc/hosts.allow
```

حيث 5.5.5.5 هو رقم الجهاز الذي أريد السماح له بالوصول للخدمة فقط. الآن لنقوم بتجربة عمل السكربت:

```
python denyhosts.py
```

إذا وجد في ملفات الأكسس لديك محاولات دخول مطابقة للخيارات والأعدادات التي وضعناها؟ سيقوم بتسجيلها في ملف **hosts.deny** وذلك لمنعها بالمرات القادمة من الدخول.

الخطوة الخامسة: تحويل السكربت DenyHosts الى خدمة لتعمل بشكل تلقائي حتى بعد إعادة التشغيل للجهاز.
الآن لعمل السكربت على شكل خدمة، قم بمتابعة الخطوات التالية معي. أول شيء:

```
mv daemon-control-dist denyhosts
```

بعد ذلك نفذ:

```
ln -s /usr/share/denyhosts/denyhosts /etc/init.d/denyhosts
```

الآن قم بتحرير الملف **denyhosts**:

```
vim /usr/share/denyhosts/denyhosts
```

وتأكد من وضع الخيارات التالية:

```
DENYHOSTS_BIN = "/usr/share/denyhosts/denyhosts.py"
DENYHOSTS_LOCK = "/var/lock/subsys/denyhosts"
DENYHOSTS_CFG = "/usr/share/denyhosts/denyhosts.cfg"
```

- السطر الأول حددنا مكان وجود الخدمة **DenyHosts**.
- السطر الثاني حددنا مكان الـ Lock File للخدمة.
- السطر الثالث حددنا مكان وجود ملف الإعدادات للخدمة.

وأعجوبة: (*RHEL,Fedora,CentOS*) الآن قم علينا أن نضيف الخدمة للعمل بعد أعادة التشغيل، بالبداية نفذ التالي-

```
chkconfig denyhosts --add
```

بعدها نفذ:

```
chkconfig denyhosts on
```

على توزيعات دبيان وأوبنتو، راجع الأمر **update-rc.d** أو راجع موضوع الأخ **أبو عبد الرحمن (Exp1r3d)**:
HowTo : Control Startup Services

الآن الخدمة جاهزة، لنقوم بتشغيلها:

```
/etc/init.d/denyhosts start
```

وهكذا ستقوم الخدمة بالعمل وفقاً للأعدادات التي عملناها، والتي قلنا لها نريدك أن تعملي وتنفذي خدماتك كل 10 دقائق.

الى هنا أنتهي من شرح إضافة الخدمة **DenyHosts** على الـ *BOX* عندك لحماية خدمة الـ **SSH** التي لديك، **ومع هذه الإضافة اقول لك لازال هناك لدي المزيد**

دق دق: من هناك ؟ (Port Knocking)

أكيد البعض أستغرب من عنوان الموضوع ولكن تابع معي الموضوع لتفهم الهدف منه ...
اليوم في حياتنا الواقعية حين ترجع الى منزلك بعد العودة من الخارج تقوم بالدق على الباب لمنزلك لكي يقوم من في المنزل من أهلك (**حفظهم الله لك**) بفتح الباب لك بعد سؤالهم: من أنت؟ وتجيب أنا فلان وتدخل ...

حين تذهب لزيارة صديق تدق على الباب الخاص ببيته ... وربما يرد عليك سائل: من أنت؟ ... وستقول أنا فلان الفلاني ... ومن ثم يفتحون لك الباب وتدخل ...

بعض الأحيان تكون بينك وبين أخوك/صديقك/زميلك بالعمل إشارة معينة بينكما ... وعلى ضوء هذه الإشارة تقوم بعمل تصرف معيين ... يعني مثلاً:
تقوم بالدق على الباب ثلاث مرات متتالية وبسرعة ... هنا يفهم الطرف الآخر بإن الذي على الباب هو أنت وليس شخص آخر ...

أو
تقوم بالدق على الخشب دقة معينة ... هنا يعرف الطرف الآخر بإن على سبيل المثال هناك من جاء ولا تريدوه أن يسمع كلامكم ...

والأمثلة كثيرا ...

هذه الإشارة التي بينك وبين الآخرين هي إشارة لا يفهم ما الغرض منها سواكم ... أي لن يفهمها سواك أنت والطرف الآخر (أخوك/صديقك/زميلك بالعمل) من الهدف منها ...

من هنا جائت فكرة الـ Port Knocking ... جميع السيناريوهات التي أخذناها هي من واقع حياتنا ... الآن لنأخذ كيف تم تطبيق هذه الفكرة على الحياة التكنلوجية وبالتحديد الحواسيب ...

لنفرض بإن هناك مدير شبكة/مدير أنظمة (**Admin**) وهو خارج العمل وأحتاج الى الوصول الى جهازه بداخل العمل ... والمشكلة بإن جميع المنافذ من الخارج مغلقة ... يعني جميع الـ **packets** التي ستصل الى الجدار الناري الخاص بالشركة/العمل يتم عمل **BLOCK** لها من ثم **DROP** ... طيب حتقول لي مهو يعمل **DROP** لكل شيء بإستثناء الـ **IP** الذي يدخل منه المدير هذا ؟
الجواب وماذا لو كان المدير هذا في مكان آخر (مسافر) غير الذي يتصل من خلاله كل مرة ... كيف سيدخل الى الداخل ؟

هنا يأتي دور الـ Port Knock ... حيث يتم تركيب برنامج **daemon** على احد الأجهزة التي تريد أن تتصل بها من الخارج ... لنفرض هو نفسه الـ **GW** ... الآن هذا البرنامج يقوم بقراءة الـ **LOG** وحين يرى بإن هناك مثلا عملية مسح **SCAN** على منافذ معينة يقوم بإضافة rule الى الـ **iptables** تفتح منفذ وممر لهذا الشخص للدخول الى النظام ... مثلا يقوم بالدق على المنافذ 3000 5000 7000 والتي سيقوم الجدار الناري بستجيلها في الـ **LOG** والبرنامج بما إنه يقرا هذه الـ **LOG** سيقوم على ضوئها بإضافة الأمر الذي يتيح للشخص الذي عمل هذه الدقات بالدخول الى النظام ... طبعا هذه هي الطريقة التقليدية للـ Port Knock هناك طرق أكثر متقدمة من هذه لكنها خارج نطاق هذا الشرح ...

الآن لنقم بعمل تجربة بسيطة أوضح لكم الفكرة بطريقة عملية ... حيث سنقوم بعمل دق على السيرفر الذي عليه برنامج الـ **PK** وحين يتعرف على الدقات الخاصة بنا يسمح لنا بإستعمال الـ **SSH** أي سيفتح لنا ممر أو منفذ للوصول الى خدمة الـ **SSH** من خلال المنفذ رقم 22 ... اول شيء سنقوم بتركيب برنامج الـ Knockd على النظام جنو/لينوكس (توزيعة فيدورا/أوبنتو)

لتركيبه على فيدورا قمت بعمل بناء للحزم وذلك لأنه الموقع الرسمي لا يقدمها بشكل جاهز ولهذا قم بتحميلهم من الروابط التالية:

Server
Client
Debug

لتركيبه على توزيعة أوبنتو قم بعمل التالي:

```
apt-get install knockd
```

الآن لنقم بعمل إعدادات للجدار الناري عندك لعمل التجربة:

```
iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -t nat -F
iptables -t nat -X
```

الآن تأكد من إعداداتك الجدار الناري:

```
iptables -L -n
```

الآن لنفتح ملف الإعدادات الخاص بالخدمة:

```
vi /etc/knockd.conf
```

وتأكد من وجود الإعدادات التالية:

```
[options]
        UseSyslog

[opencloseSSH]
        sequence      = 2222:tcp,3333:tcp,4444:tcp
        seq_timeout   = 15
        tcpflags      = syn,ack
        start_command = /sbin/iptables -I INPUT 1 -s %IP% -p tcp --dport ssh -j ACCEPT
        cmd_timeout   = 10
        stop_command  = /sbin/iptables -D INPUT -s %IP% -p tcp --dport ssh -j ACCEPT
```

لمستخدمي أوبنتو عليك تحرير ملف آخر ولهذا قم بفتح الملف التالي:

```
vi /etc/default/knockd
```

وتأكد من وضع رقم 1 بدل من 0 في المتغيير START_KNOCKD كما يلي:

```
START_KNOCKD=1
```

الآن قم بتشغيل السيرفر Knockd هكذا:

```
/etc/init.d/knockd start
```

الآن لنقم بتشغيل خدمة SSH هكذا:

```
/etc/init.d/sshd start
```

الحين كل شيء جاهز لعمل التجربة ...

الآن قم بمحاولة الإتصال بالسيرفر الـ SSH:

```
ssh 192.168.0.44
```

لن تستطيع وذلك لأنه جميع المنافذ للوصول للخدمة SSH مغلقة ... الآن من جهاز لينوكس آخر قم بتركيب برنامج الـ Client عليه (يوجد واحد للويندوز لكني لم اقم بتجربته) وقم بتنفيذ التالي:

```
knock -v 192.168.0.44 2222 3333  4444
```

ما قمنا به هو عمل دق على المنافذ 2222 و 3333 و 4444 التي حددناها في إعدادات السيرفر knockd ...

الآن قم بالإتصال مرة أخرى بخدمة SSH كالتالي:

```
ssh 192.168.0.44
```

ستلاحظ إنه أصبح الآن بإمكانك الإتصال بالخدمة وكل الأمور تمام ...

الآن لنرى ماذا حدث على الجدار الناري:

```
iptables -L -n
```

سترى هناك شي شبيه بالتالي:

```
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.0.44         0.0.0.0/0           tcp dpt:22
```

أي تم إضافة **rule** جديدة الى الجدار الناري يفتح منفذ 22 لكي يتم إستعماله بالإتصال بخدمة SSH ...

آخر حاجة هي ان نغلق الإتصال هذا وتقوم خدمة knockd بغلق المنفذ هي بعمل التالي:

```
knock -v 4444 3333 2222
```

وهنا قمنا بعكس الدق على الخدمة والتي يفهمها الـ knockd على إنها طلب بغلق المنفذ كما هو محدد في ملف الإعدادات الخاص بخدمة knockd ... طبعا تستطيع تغيير كل من المنافذ الخاصة بفتح المنفذ أو غلقها وذلك من ملف الإعدادات التي ذكرناه بالاعلى ...

المصادر:
المصادر التي أعتمدت عليها، وانصح بزيارتها ومتابعتها هي:

طريقة التركيب
الربط مع الـ syslog-ng
الربط مع الـ iptables

الموقع الرسمي للبرنامج
Documentation

موقعي الشخصي لرسالة الدكتوراه
موقع الـ Knockd الرسمي
موقع PortKnocking.org العالمي

الى هنا نأتي الى نهاية هذا الكتييب الصغير ... أتمنى أن ينال رضاكم وإعجابكم ويكون محل فائدة لكم ...
وإن شاء الله إذا سنحت الفرصة أضيف أمور وخفايا جديدة لن أبخل بها عليكم بإذن الله ...

أخوكم B!n@ry ...