İSİM

ssh – OpenSSH SSH istemcisi (uzaktan oturum açma aracı)

KULLANIM

ssh [**-l** *kullanıcı*] *konakismi* | *kullanıcı@konakismi* [*komut*]

ssh [**-afgknqstvxACNTX1246**] [**-b** *bağlantı_adresi*]
[**-c** *şifre_belirtimi*] [**-e** *önceleme_krk*] [**-i** *kimlik_dosyası*]
[**-l** *kullanıcı*] [**-m** *mac_belirtimi*] [**-o** *seçenek*] [**-p** *port*]
[**-F** *yapılandırma_dosyası*] [**-L** *yerelport:uzakkonak:uzakkonakportu*]
[**-R** *uzakport:yerelkonak:yerelkonakportu*] [**-D** *port*]
konakismi | *kullanıcı@konakismi* [*komut*]

AÇIKLAMA

ssh (SSH istemci) uzaktaki bir makinada komut çalıştırmak için uzaktaki makinada bir kullanıcı oturumu açmayı sağlayan bir uygulamadır. **rlogin** ve **rsh**'ın yerini alması amaçlanmıştır. **ssh** güvenli olmayan bir ağ üzerindeki güvenilir olmayan iki sistemin şifreli dolayısı ile güvenli iletişim kurmalarını sağlar. X11 bağlantıları ve çeşitli TCP/IP bağlantı portları da güvenli kanal üzerinden iletilebilirler.

ssh, *konakismi* ile belirtilen makinaya bağlanır ve oturum açar. Şayet kullanıcı bir *komut* belirtmiş ise, kullanıcıya oturum kabuğu yerine komutun çalıştırıldıktan sonraki çıktısı döndürülür.

Kullanıcı, uzaktaki sisteme kimliğini kanıtlamak zorundadır ve bunu kullanılan protokole bağlı olarak farklı yöntemlerle yapabilir:

SSH protokolünün 1. sürümü

İlk olarak eğer kullanıcının oturum açtığı sistemin ismi uzaktaki sistemin `/etc/hosts.equiv` veya `/etc/ssh/shosts.equiv` dosyalarında bulunuyorsa ve her iki taraftaki kullanıcı isimleri aynı ise, kullanıcın uzak sisteme erişmesine hemen izin verilir. İkinci olarak, eğer `.rhosts` veya `.shosts` dosyaları kullanıcının uzaktaki ev dizininde mevcut ise ve istemci makinanın adı ile bu makinadaki kullanıcı adı bu dosyalardan birinde var ise, yine kullanıcın erişmesine izin verilir. Güvenli olmamaması nedeni ile bu çeşit kimlik denetiminin kullanmasına normalde sunucu tarafından izin verilmez.

İkinci kimlik denetim yöntemi ise `rhosts` veya `hosts.equiv` yöntemlerinin RSA tabanlı konak kimlik denetimi ile birlikte kullanılmasıdır. Bu şu anlama gelir: eğer erişime `$HOME/.rhosts`, `$HOME/.shosts`, `/etc/hosts.equiv` veya `/etc/ssh/shosts.equiv` tarafından izin verilmiş ve buna ek olarak sunucu da istemci anahtarını doğrulayabiliyorsa (DOSYALAR bölümündeki `/etc/ssh/ssh_known_hosts` ve `$HOME/.ssh/known_hosts` dosyalarının açıklamalarına bakınız), ancak o zaman oturum açılmasına izin verilir. Bu kimlik denetimi yöntemi sayesinde IP kandırmaca, DNS kandırmaca ve yönlendirme kandırmacanın sebep olduğu güvenlik açıkları engellenmiş olur.

Sistem yöneticilerine bilgi

`/etc/hosts.equiv`, `$HOME/.rhosts` ve **rlogin**/**rsh** protokolleri genel olarak güvenli değillerdir ve güvenliğin gerektiği durumlarda kullanılmamaları gerekir.

Üçüncü kimlik kanıtlama yöntemi olarak, **ssh** RSA tabanlı kimlik denetimini destekler. Bu senaryo açık anahtarlı kriptografiye dayanır: Şifrelemenin ve şifre çözmenin farklı anahtarlarla yapıldığı ve şifre çözme anahtarını şifreleme anahtarından türetmeniz mümkün olmadığı kripto sistemleri vardır. RSA böyle bir sistemdir. Burada amaç her kullanıcının kimlik denetimi için bir çift genel ve özel anahtar

oluşturmasıdır. Sunucu genel anahtara sahip iken, özel anahtar sadece kullanıcının kendisinde bulunur. `$HOME/.ssh/authorized_keys` dosyası erişim hakkı olan genel anahtarları listeler. Kullanıcı oturum açmak istediği zaman, **ssh** kimlik denetimi için hangi anahtar çiftini kullanmak istediğini sunucuya bildirir. Sunucu bu anahtarın geçerli olup olmadığına bakar, şayet geçerli ise kullanıcıya (aslında kullanıcı adına çalışan **ssh**'a) kullanıcının genel anahtarı ile şifrelenmiş bir kimlik sorgusu, rasgele bir sayı gönderir. Şifreli kimlik sorgusu sadece uygun özel anahtar ile çözülebilir. Kullanıcının istemcisi bu kimlik sorgusunu kullanıcının özel anahtarı ile çözer, böylece uygun özel anahtarın varlığını ispatlar (özel anahtarı sunucuya göstermeden).

ssh RSA kimlik denetim protokolünü özdevinimli olarak gerçekleştirir. Kullanıcı kendi RSA anahtar çiftini **ssh-keygen(1)** komutunu çalıştırarak oluşturur. Bu işlem sayesinde özel anahtar `$HOME/.ssh/identity` dosyasında ve genel anahtar da `$HOME/.ssh/identity.pub` doyasında saklanır. Daha sonra kullanıcı `identity.pub` dosyasını uzak makinenin kullanıcının ev dizinideki `$HOME/.ssh/authorized_keys` dosyasına kopyalamalıdır. `authorized_keys` dosyası geleneksel `$HOME/.rhosts` dosyasına karşılık gelir ve her satırı bir anahtar içerir (satırlar çok uzun olabilmektedir). Sonuçta kullanıcı parola girmeden oturum açmaya hak kazanmaktadır. RSA kimlik denetimi **rhosts** kimlik denetimine göre daha güvenlidir.

RSA kimlik denetiminin en uygun kullanımı bir kimlik denetimi ajanı ile sağlanabilir. Daha fazla bilgi için **ssh-agent(1)**'a bakınız.

Şayet bahsedilen kimlik denetimi yöntemleri hata verirler ise, bu sefer **ssh** kullanıcıya parola sorar. Parola ana makineye denetim için gönderilir. Fakat bütün iletişim şifrelendiği için parola ağı dinlemekte olan biri tarafından görülemez.

SSH protokolünün 2. sürümü

Kullanıcı protokolün 2. sürümünü kullanarak bağlantı kurduğunda benzer kimlik denetimi yöntemleri kullanılır. **PreferredAuthentications** (tercih edilen kimlik kanıtlamaları) için öntanımlı değerler kullanılarak, istemci öncelikle konak tabanlı kimlik denetimi ile uzaktaki makineye erişmeye çalışır; bu yöntemin başarısız olması durumunda, genel anahtarlı kimlik denetimi yöntemi denenir, bu yöntemin de başarısız olması durumunda son olarak klavye etkileşimli ve parolalı kimlik denetimi yöntemine başvurulur.

Genel anahtar yöntemi, bir önceki bölümde anlatılan RSA kimlik denetimi ile benzerdir ve RSA veya DSA algoritmalarının kullanılmasına izin verir: İstemci oturum tanıtıcısını özel anahtarı (`$HOME/.ssh/id_dsa` veya `$HOME/.ssh/id_rsa`) ile imzalar ve sonucu sunucuya gönderir. Sunucu, bu imza ile eşleşen genel anahtarın `$HOME/.ssh/authorized_keys` dosyasında olup olmadığına bakar. Hem anahtarın listede bulunması hem de imzanın doğru olması durumunda istemciye erişim izni verilir. Oturum tanıtıcı sadece sunucu ve istemci tarafından bilinen ortak Diffie–Hellman değerinden türetilir.

Şayet genel anahtarlı kimlik denetimi başarısız olursa ya da desteklenmiyorsa, kullanıcının kimliğini ispatlamak için kullanıcının parolası şifrelenerek sunucuya gönderilebilir.

ssh bunlara ek olarak konak tabanlı ve kimlik sorma/yanıt verme kimlik denetimi yöntemlerini de destekler.

Protokol 2 gizlilik ve bütünlük ilkeleri için ek mekanizmalar sağlar. Gizlilik için veri trafiği 3DES, Blowfish, CAST128 veya Arcfour algoritmaları ile şifrelenir ve bütünlük ilkesi için ise hmac–md5 ya da hmac–sha1 algoritmaları kullanılır. Protokol 1'in bağlantının bütünlüğünü kesinlikle garanti eden bir mekanizmaya gereksinimi olduğu unutulmamalıdır.

Oturum ve Uzaktan Yürütme

Sunucu, kullanıcının kimliğini onayladığı takdirde ya kullanıcının belirttiği komutu yürütür ya da sistemde oturum açar ve uzak sistemde kullanıcının normal bir kabuğa erişmesini sağlar. Uzakta çalıştırılacak komut ya da kabukla gerçekleştirilen bütün iletişim şifrelenir.

Kullanıcı, sözde uçbirim (normal oturum) sağlanması durumunda aşağıdaki önceleme karakterlerini kullanabilir.

Şayet bir uçbirim sağlanmazsa, oturum saydam olur ve ikili veri güvenilir şekilde aktarılabilir. Birçok sistemde önceleme karakterini "none" şeklinde ayarlamak tty kullanılsa bile oturumu saydam kılar.

Uzak sistemdeki komut ya da kabuk sonlandığında oturum sonlanır ve bütünl X11 ve TCP/IP bağlantıları kapatılır. Erişilen makinada çalıştırılan uygulamanın çıkış durumu **ssh**'ın çıkış durumu olarak döndürülür.

Önceleme Karakterleri

Sözde uçbirim istenmesi durumunda **ssh** birçok işlevi bir önceleme karakteri aracılığı ile desteker.

Tek bir yaklaşık işareti (tilde) **~~** şeklinde ya da yaklaşık işaretini aşağıda belirtilen karakterlerin haricindeki bir karakterin takip etmesi şeklinde gönderilebilir. Önceleme karakterinin önceleme karakteri olarak yorumlanabilmesi karakterden hemen sonra bir satırsonu karakteri gelmelidir. Önceleme karakteri yapılandırma dosyasındaki **EscapeChar** yapılandırma seçeneği ile ya da komut satırında **-e** seçeneği ile değiştirilebilir.

Desteklenen öncelemler (öntanımlının '~' olduğu kabulüyle) şunlardır:

~.
: Bağlantıyı kes

~^Z
: **ssh**'ı artalanda çalıştır

~#
: İletilen bağlantıları listele

~&
: İletilen bağlantının ya da X11 oturumlarının sonlandırılmasını beklerken **ssh**'ı artalanda çalıştır

~?
: Önceleme karakterlerinin listesini göster

~B
: Uzak sisteme sonlandırma iletisi gönder (sadece uzak sistemin de desteklemesi durumunda SSH protokolünün 2. sürümü için geçerlidir)

~C
: Komut satırı aç (sadece **-L** ve **-R** seçeneklerini kullanarak port iletimi eklemek için yararlıdır)

~R
: Bağlantının anahtarlarının yenilenmesini iste (sadece uzak sistemin de desteklemesi durumunda SSH protokolünün 2. sürümü için geçerlidir)

X11 ve TCP İletimi

Şayet **ForwardX11** değişkeni "yes" şeklinde ayarlı ise (ya da aşağıdaki **-X** ve **-x** seçeneklerinin tanımlarına bakınız) ve kullanıcı X11 (DISPLAY çevre değişkeni atanmış ise) kullanıyorsa, X11 görüntüye olan bağlantı özdevinimli biçimde uzaktaki sisteme iletilir. Bu sayede kabuktan (ya da komut ile) çalıştırılan herhangi bir X11 programı şifreli kanal boyunca iletilir ve yerel sistemin gerçek X sunucusu ile bağlantısı gerçekleştirilmiş olur. Kullanıcı DISPLAY değişkenini el ile ayarlamamalıdır. X11 bağlantılarının iletilmesi komut satırında ya da yapılandırma dosyaları aracalığı ile ayarlanabilir.

ssh tarafından ayarlanan DISPLAY değeri (sıfırdan büyük bir sayı) sunucu sistemi işaret eder. Bu normaldir ve sebebi de **ssh**'nın bağlantıları şifreli kanal üzerinden iletmek için sunucu sistemde 'vekil' X sunucusu oluşturmasıdır.

ssh yine sunucu makinede özdevinimli biçimde Xauthority verisini ayarlar. Bu amaçla rasgele yetkilendime çerezi üretilip sunucuda 'Xauthority' dosyasında saklanır ve iletilen bağlantıların bu çerezi taşıyıp taşımadıkları kontrol edilir ve bağlantı açıldığında bu çerez gerçek çerez ile değiştirilir. Gerçek kimlik denetimi çerezi sunucu sisteme hiçbir zaman gönderilmez (ve hiçbir çerez düz metin olarak gönderilmez).

Şayet **ForwardAgent** değişkeni "yes" şeklinde ayarlandı ise (aşağıda açıklanan **-A** ve **-a** seçeneklerine bakınız) ve kullanıcı kimlik denetimi ajanı kullanılıyorsa, ajana olan bağlantı özdevinimli biçimde uzak tarafa iletilir.

Keyfi TCP/IP bağlantılarının güvenli kanal üzerinden iletimi ya komut satırında ya da yapılandırma dosyasında belirtilebilir. TCP/IP iletimi uygulamalarına örnek olarak elektronik para çantasına güvenli bağlantı ya da güvenlik duvarı üzerinden iletişim verilebilir.

Sunucu kimlik denetimi

ssh kullanılmış olan bütün ana sistemleri içeren bir veritabanını özdevinimli olarak oluşturur ve denetler. Ana sistem anahtarları kullanıcının ev dizinindeki $HOME/.ssh/known_hosts dosyasında tutulur. Buna ek olarak /etc/ssh/ssh_known_hosts dosyasına da bilinen sistemleri kontrol için özdevinimli olarak başvurulur.

Yeni sistemler özdevinimli olarak kullacının dosyasına eklenir. Şayet bir sistemin kimliği değişirse, **ssh** bu konuda uyarır ve truva atlarının kullanıcın parolasını çalmasını engellemek için parola kimlik denetimini edilgenleştirir. Bu mekanizmanın diğer bir amacı şifrelemeyi es geçebilen araya girme saldırılarına engel olmaktır. **StrictHostKeyChecking** seçeneği anahtarı bilinmeyen ya da değişmiş olan sistemlerde oturum açmayı engellemek için kullanılabilir.

Seçenekler şunlardır:

-1
: **ssh**'yı sadece protokolün 1. sürümünü kullanmaya zorlar.

-2
: **ssh**'yı sadece protokolün 2. sürümünü kullanmaya zorlar.

-4
: **ssh**'yı sadece IPv4 adreslerini kullanmaya zorlar.

-6
: **ssh**'yı sadece IPv6 adreslerini kullanmaya zorlar.

-a
: Kimlik denetimi ajanı bağlantı iletimini iptal eder.

-A
: Kimlik denetimi ajanı bağlantı iletimini etkinleştirir. Bu ayrıca yapılandırma dosyasında da her bir konak için ayrı ayrı belirtilebilir.

  Bu seçenek etkinleştirilirken dikkat edilmelidir. Uzak konaktaki (ajanın Unix–alan soketi için) dosya izinlerini atlayabilen kullanıcılar iletilen bağlantılar sayesinde yerel ajana erişebilirler. Saldırgan, ajandan anahtarları alamaz ancak ajanda yüklü olan kimlikleri kullanarak kimlik denetimini geçmeyi başarabilir.

-b *bağlantı_adresi*
Çoklu arayüzü olan ya da takma isimli adresler kullanan sistemlerde iletişimin yapılacağı arayüzü belirler.

-c `blowfish | 3des | des`
Oturumu şifrelemekte kullanılacak şifreyi seçer. Öntanımlı değer **3des**'dir. Güvenli olduğuna inanılır. **3des** (üçlü–des) şifreleme–şifre çözme–şifreleme üçlüsünü 3 farklı anahtarla gerçekleştirir. **blowfish** hızlı bir blok şifresidir; çok güvenilirdir ve **3des**'ten çok daha hızlıdır. **des** ise **3des** şifrelemeyi desteklemeyen eski protokol 1 gerçeklemeleri ile işbirliktelik adına desteklenir. Şifreleme ile ilgili zayıflıklarından dolayı **des**'in kullanılmaması önemle tavsiye edilir.

-c *şifre_belirtimi*
Ek olarak, protokol 2'de tercih sırasını belirtmek için virgülle ayrılmış şifreleme listesi verilebilir. Daha fazla bilgi için yapılandıma dosyaındaki **Ciphers** (Şifreler) satırına bakınız.

-C
Bütün verilerin (stdin, stdout, stderr, X11 verileri ve TCP/IP bağlantı verileri) sıkıştırılmasını sağlar. Sıkıştırma algoritması **gzip(1)**'in kullandığı ile aynıdır. **CompressionLevel**(Sıkıştırma Seviyesi) seçeneği protokolün 1. sürümü için "seviye" yi belirler. Modem hatları ve diğer yavaş bağlantılar için sıkıştırma kullanılmalıdır, ancak hızlı ağlar için bu sadece yavaşlamaya neden olacaktır. Öntanımlı değer yapılandırma dosyalarında her konak için ayrı ayrı belirtilebilir. (**Compression** [Sıkıştırma] seçeneğine bakınız).

-D *port*
Yerel "özdevimli" uygulama seviyesinde port iletimini belirtir. Yerel tarafta portu dinlemek üzere bir soket ayrılır. Bu port ile ne zaman bir bağlantı kurulsa, bağlantı güvenli kanal üzerinden iletilir ve uzak sistemde nereye bağlanılacağı uygulama protokolü kullanılarak belirlenir. Şu anda SOCKS4 ve SOCKS5 protokolleri desteklenmektedir. **ssh** bir SOCKS sunucusu olarak davranır. Sadece yetkili kullanıcı (root) ayrıcalıklı portları iletebilir. Özdevimli port iletimleri yapılandırma dosyasında da belirtilebilir.

-e *krk* | ^*krk* | none
pty'li bir oturum için önceleme karakterini ayarlar (öntanımlı: '~'). Önceleme karakteri sadece bir satırının başında ise tanınabilir. Önceleme karakterini nokta ('.') takip ederse bağlantı sonlandırılır; control–Z takip ederse bağlantı askıya alınır; '~' takip ederse önceleme karakteri bir kez gönderilir. Karakteri "none" şeklinde ayarlamak öncelemleri iptal eder ve oturumu tamamen saydam yapar.

-f
Komut yürütülmeden hemen önce **ssh**'nın arkaplanda çalışmasını sağlar. Bu **ssh**'nın kullanıcı ya da anahtar parolası sorması gerektiği ancak kullanıcının bu işlemin arkaplanda yapılmasını istediği durumlarda yararlıdır. Bu seçenek **-n** seçeneğinin de uygulanmasını sağlar. X11 programları uzak konakta çalıştırırken komutun şöyle çağrılması tavsiye edilir: **ssh -f konakismi xterm**.

-F *yapılandırma_dosyası*
Kullanıcının kendine özgü yapılandırma dosyasını belirtmek içindir. Şayet komut satırında bir *yapılandırma_dosyası* verilirse, sistemin yapılandırma dosyası (`/etc/ssh/ssh_config`) görmezden gelinir. `$HOME/.ssh/config` dosyası kullanıcının öntanımlı yapılandırma dosyasıdır.

-g
Uzak konakların iletilen yerel portlara bağlanmasına izin verir.

-i *kimlik_dosyası*

RSA ya da DSA kimlik doğrulamaları için kullanılacak olan kimlik dosyasını (dolayısıyla gizli anahtarı) belirtir. Protokolün 1. sürümü için varsayılan dosya `$HOME/.ssh/identity` dosyasıdır. `$HOME/.ssh/id_rsa` ve `$HOME/.ssh/id_dsa` ise protokolün 2. sürümü için öntanımlı dosyalardır. Kimlik dosyaları her bir konak için ayrı ayrı yapılandırma dosyasında da belirtilebilir. **-i** seçeneğinin birden fazla kullanılması mümkündür (ve yapılandırma dosyalarında birden fazla kimlik tanımlamak da).

-I *akıllıkart_aygıtı*
Kullanılacak olan akıllıkart aygıtını belirtir. Belirtilen aygıt kullanıcının RSA gizli anahtarının bulunduğu akıllıkart ile iletişim için kullanılan aygıt olmalıdır.

-k
Kerberos biletlerinin ve AFS dizgeciklerinin iletimini iptal eder. Bu ayrıca yapılandırma dosyasında her konak için ayrı ayrı belirtilebilir.

-l *kullanıcı_ismi*
Uzak sistemde oturum açmak için kullanılacak kullanıcı adını belirtir. Bu yine yapılandırma dosyasında her bir konak için ayrı ayrı belirtilebilir.

-L *yerelport:uzakkonak:uzakkonakportu*
Belirtilen yerel portun uzak konaktaki porta iletilmesi için kullanılır. Bunu sağlamak için yerel sistemde bir soket *yerelport* portunu dinlemeye başlar ve bu porta bağlantı isteği geldiğinde, bağlantı güvenli kanaldan iletilerek *uzakkonak* üzerindeki *uzakkonakportu*na bir bağlantı gerçekleştirilir. Port iletimi, yapılandırma dosyasında da belirtilebilir. Sadece yetkili kullanıcı (root) ayrıcaklı portları iletebilir. IPv6 adresler için farklı bir sözdizimi kullanılır: *yerelport/uzakkonak/uzakkonakportu*.

-m *mac_belirtimi*
Protokolün 2. sürümünde ayrıca tercih sırasına göre virgüllerle ayrılmış MAC (message authentication code – ileti kimlik denetimi kodu) algoritmaları belirtilebilir. Daha fazla bilgi için **MACs** anahtar sözcüğüne bakınız.

-n
/dev/null'u standart girdiye yöneltir (yani standart girdiden okuma yapılması engellenir). **ssh** artalanda çalışıyorsa bu seçenek kullanılmak zorundadır. Uzak sistemde X11 programları çalıştırılırken bu seçenek çok kullanılır. Örneğin, **ssh -n shadows.cs.hut.fi emacs &** komutu shadows.cs.hut.fi konağında **emacs** uygulamasını başlatacaktır ve X11 bağlantısı özdevinimli olarak şifreli kanal üzerinden iletilecektir. **ssh** artalana yerleştirilecektir. (Ancak **ssh**'nın kullanıcı ya da anahtar parolası gerektirmesi durumunda bu çalışmayacaktır; ayrıca **-f** seçeneğine de bakınız.)

-N
Bir uzak komut çalıştırmaz. Bu sadece port iletimi için yararlıdır (sadece protokolün 2. sürümünde).

-o *seçenek*
Yapılandırma dosyasındaki biçime uygun seçenekler belirtmek için kullanılabilir. Kendisine özel komut satırı seçeneği olmayan yapılandırma seçeneklerini belirtmek için kullanılabilir. Aşağıda sıralanan seçeneklere ait tüm ayrıntılar ve alabilecekleri olası değerler için **ssh_config(5)**'e başvurunuz.

```
AddressFamily (Adres Ailesi)
BatchMode (Toplu İş Kipi)
BindAddress (Bağlantı Adresi)
ChallengeResponseAuthentication (Kimlik Sorma/Yanıtlama Yöntemi)
CheckHostIP (Konak IP Denetimi)
Cipher (Şifre)
```

```
Ciphers (Şifreler)
ClearAllForwardings (Bütün İletimleri Temizle)
Compression (Sıkıştırma)
CompressionLevel (Sıkıştırma Seviyesi)
ConnectionAttempts (Bağlantı Denemeleri)
ConnectionTimeout (Bağlantı Zaman Aşımı)
DynamicForward (Özdevimli İletim)
EscapeChar (Önceleme Karakteri)
ForwardAgent (İletim Ajanı)
ForwardX11 (X11 İletimi)
ForwardX11Trusted (Güvenilir X11 İletimi)
GatewayPorts (Ağ Geçidi Portları)
GlobalKnownHostsFile (Genel Bilinen Konaklar Dosyası)
GSSAPIAuthentication (GSSAPI Kimlik Denetimi)
GSSAPIDelegateCredentials (GSSAPI Yetkilendirme Tanıtımları)
Host (Konak)
HostbasedAuthentication (Konak Tabanlı Kimlik Denetimi)
HostKeyAlgorithms (Konak Anahtarı Algoritmaları)
HostKeyAlias (Konak Anahtarı Takma Adları)
HostName (Konak Adı)
IdentityFile (Kimlik Dosyası)
LocalForward (Yerel İletim)
LogLevel (Günlükleme Düzeyi)
MACs (İleti Kimlik Denetimi Kodları)
NoHostAuthenticationForLocalhost (Yerel Konak İçin Kimlik Denetimi Yapma)
NumberOfPasswordPrompts (Parola İsteme Adedi)
PasswordAuthentication (Parolalı Kimlik Denetimi)
Port
PreferredAuthentications (Tercih Edilen Kimlik Denetimi Yöntemleri)
Protocol (Protokol)
ProxyCommand (Vekil Komutu)
PubkeyAuthentication (Genel Anahtarlı Kimlik Denetimi)
RemoteForward (Uzak İletim)
RhostsRSAAuthentication (RSA tabanlı Rhosts Kimlik Denetimi)
RSAAuthentication (RSA Kimlik Denetimi)
ServerAliveInterval (Sunucu Canlılık Aralığı)
ServerAliveCountMax (Canlı Sunucu En Çok Mesaj Sayısı)
SmartcardDevice (Akıllı Kart Aygıtı)
StrictHostKeyChecking (Mutlak Konak Anahtarı Denetimi)
TCPKeepAlive (TCP Canlı Tutma)
UsePrivilegedPort (Ayrıcalıklı Port Kullan)
User (Kullanıcı)
UserKnownHostsFile (Kullanıcının Bilinen Konakları Dosyası)
VerifyHostKeyDNS (Uzak Konak Anahtarının Doğruluğunun DNS ile Sağlanması)
XAuthLocation (XAuth'un Tam Yolu)
```

-p *port*
Uzak konaktaki bağlantı portu. Yapılandırma dosyasında her bir konak için ayrı ayrı belirtilebilir.

-q
Sessizlik kipi. Bütün uyarı ve tanı iletilerinin gizlenmesini sağlar. Sadece onarılamaz hatalar gösterilir. Şayet ikinci bir **-q** verilirse onarılamaz hatalar da gösterilmez.

-R *uzakport:yerelkonak:yerelkonakportu*
Belirtilen uzak portun yerel konaktaki porta iletilmesi için kullanılır. Bunu sağlamak için uzak sistemde bir soket *uzakport* portunu dinlemeye başlar ve bu porta bağlantı isteği geldiğinde, bağlantı

güvenli kanaldan iletilerek *yerelkonak* üzerindeki *yerelkonakportu*na bir bağlantı gerçekleştirilir. Port iletimi, yapılandırma dosyasında da belirtilebilir. Sadece yetkili kullanıcı (root) ayrıcaklı portları iletebilir. IPv6 adresler için farklı bir sözdizimi kullanılır: *uzakport/yerelkonak/yerelkonakportu*.

-s

Uzak konakta bir altsistemi çağırmak amacıyla kullanılabilir. Altsistemler, diğer uygulamaların (örn. sftp) güvenli taşınmasını sağlayan SSH2 protokolünün bir özelliğidir. Altsistem uzak komut olarak belirtilir.

-t

Sözde–tty ayırmayı zorlar. Bu uzak sistemde ekran tabanlı herhangi bir uygulamaların çalıştırılmasında kullanılabilir. Örnek olarak menü hizmetlerinin gerçekleştirilmesinde bu çok yararlı olabilir. Çok sayıda **-t** seçeneği **ssh**'nın yerel tty'si olmasa bile bir tty ayrılmasını sağlar.

-T

Sözde–tty ayırmayı iptal eder.

-v

Ayrıntı kipi. **ssh**'nın çalışması esnasındaki hata ayıklama iletilerinin gösterilmesini sağlar. Bağlantı, kimlik denetimi ve yapılandırma sorunlarındaki hataları ayıklamada bu seçenek çok faydalıdır. Çok sayıda **-v** seçeneği ayrıntı seviyesini artırır. En fazla üç tane olabilir.

-V

Sürüm numarasını gösterir ve çıkar.

-x

X11 iletimini iptal eder.

-X

X11 iletimini etkinleştirir. Bu her bir konak için ayrı ayrı yapılandırma dosyasında belirtilebilir.

Bu seçenek etkinleştirilirken dikkat edilmelidir. Uzak sistemdeki dosya izinlerini atlayabilen kullanıcılar (kullanıcının X yetkilendirme veritabanı için) iletilen bağlantılar sayesinde yerel X11 görüntüye erişebilirler. Saldırgan, tuş vuruşlarını izlenmek gibi etkinliklerde bulunabilir.

-Y

Güvenilir X11 iletimini etkinleştirir.

YAPILANDIRMA DOSYALARI

ssh yapılandırma verilerini her kullanıcıya özel ayrı birer yapılandırma dosyasından ve de sistemin yapılandırma dosyasından alabilir. Dosya biçimi ve yapılandırma seçenekleri **ssh_config(5)**'de açıklanmaktadır.

ORTAM DEĞİŞKENLERİ

ssh normalde aşağıdaki çevre değişkenlerini ayarlar:

DISPLAY

DISPLAY değişkeni X11 sunucusunun konumunu gösterir. **ssh** bu değişkene özdevinimli biçimde "konak_adı:n" biçiminde değer atar. Burada konak_adı kabuğun çalıştığı sistemi işaret ederken n de n >= 1 şartını sağlayan bir tamsayıya karşılık gelir. **ssh** bu özel değeri X11 bağlantılarını güvenli kanal üzerinden iletmede kullanır. Kullanıcı DISPLAY değişkenini doğrudan kendisi ayarlamamalıdır, çünkü bu X11 bağlantılarını güvensiz hale getirir (ve ayrıca kullanıcının gerekli olan yetkilendirme çerezlerini el ile kopyalamasını gerektirir).

HOME

Kullanıcının ev dizininin yolu bu değişkene atanır.

`LOGNAME`
`USER` değişkeni ile aynıdır; bu değişkeni kullanan sistemlerle uyumluluk için tanımlanır.

`MAIL`
Kullanıcının posta kutusunun yolu bu değişkene atanır.

`PATH`
ssh derlenirken belirtilen gibi öntanımlı `PATH`'a ayarlanır.

`SSH_ASKPASS`
Şayet **ssh** bir anahtar parolası gerektiriyorsa, bunu mevcut uçbirimden okur (eğer uçbirimden çalıştırılıyorsa). Diğer taraftan **ssh** bir uçbirimden çalıştırılmıyor fakat `DISPLAY` ve `SSH_ASKPASS` değişkenleri ayarlanmış iseler, **ssh**, `SSH_ASKPASS` tarafından belirtilen uygulamayı çalıştırır ve anahtar parolayı okumak için bir X11 penceresi açar. Bu özellikle **ssh** bir `.Xsession` ya da ilgili betik tarafından çağırılıyorsa yararlıdır. (Burada dikkat edilmesi gereken bu yöntemin çalışması için bazı sistemlerde girdilerin `/dev/null`'dan yönlendirilmesi gerektiğidir.)

`SSH_AUTH_SOCK`
Ajanla iletişimde kullanılacak Unix–alan soketinin yolunu belirtir.

`SSH_CONNECTION`
Bağlantının kurulduğu istemci ve sunucuyu belirtir. Değişken boşluk ile birbirinden ayrılmış 4 değerden oluşur: istemci ip–adresi, istemci portu, sunucu ip–adresi ve sunucu portu.

`SSH_ORIGINAL_COMMAND`
Zorunlu bir komutun çalıştırılması durumunda özgün komut satırını içerir. Bu özgün argümanların özütlenmesinde kullanılabilir.

`SSH_TTY`
Yürürlükteki kabuk ya da komutla ilişkili olan tty'nin adı (aygıt yolu) bu değişkene atanır. Yürürlükteki oturum tty'ye sahip değilse, bu değişkene bir atama yapılmaz.

`TZ`
Zaman Dilimi değişkeni; o anki zaman dilimini işaret etmesi için ayarlanır (şayet artalan süreci başlatıldığında ayarlandı ise). Yani, artalan süreci yeni bağlantılara bu değeri aktarır.

`USER`
Oturum açan kullanıcı adı olarak ayarlanır.

Ayrıca **ssh**, `$HOME/.ssh/environment` dosyasını okur ve eğer bu dosya mevcut ise ve de kullanıcılar değişkenlerini değiştirme hakkına sahip iseler "DEĞİŞKEN=değer" biçimindeki satırları ortama ekler. Daha fazla bilgi için, **sshd_config(5)**'deki **PermitUserEnvironment** (Kullanıcı Ortamına İzin Ver) seçeneğine bakınız.

İLGİLİ DOSYALAR

`$HOME/.ssh/known_hosts`
Kullanıcının oturum açtığı `/etc/ssh/ssh_known_hosts` dosyasında kayıtlı olmayan bütün konakların anahtarlarını kaydeder. Bakınız, **sshd(8)**.

`$HOME/.ssh/identity`, `$HOME/.ssh/id_dsa`, `$HOME/.ssh/id_rsa`

Kullanıcının kimlik denetim kimliğini içerirler. Sırasıyla protokol 1 RSA, protokol 2 DSA ve protokol 2 RSA'ye karşılık gelirler. Bu dosyalar duyarlı veri içerirler ve bu sebeple kimlik sahibi haricindekilerin erişim hakkı olmamalıdır. Şayet özel anahtar dosyası diğer kullanıcıların erişimine açıksa, **ssh**'nın bu dosyayı kullanmayı reddedeceğini unutmayın. Gizli anahtarı oluştururken bir anahtar parolası belirtilebilir ve anahtar parolası bu dosyanın duyarlı kısmını 3DES algoritması ile şifrelemede kullanılır.

`$HOME/.ssh/identity.pub`, `$HOME/.ssh/id_dsa.pub`, `$HOME/.ssh/id_rsa.pub`
Kimlik denetiminde kullanılacak genel anahtarı içerirler (kimlik dosyasının okunabilir biçimdeki genele açık olan kısmı). `$HOME/.ssh/identity.pub`'un içeriği kullanıcının protokol sürüm 1 RSA kimlik denetimini kullanmak istediği bütün sistemlerdeki `$HOME/.ssh/authorized_keys` dosyasına eklenmelidir. `$HOME/.ssh/id_dsa.pub` ve `$HOME/.ssh/id_rsa.pub` dosyalarının içerikleri kullanıcının protokol sürüm 2 DSA/RSA kimlik denetimini kullanmak istediği bütün sistemlerdeki `$HOME/.ssh/authorized_keys` dosyasına eklenmelidir. Bu dosyalar duyarlı değillerdir ve herkes tarafından okunabilirler (fakat gerekmemektedir). Bu dosyalar hiçbir zaman özdevinimli biçimde kullanılmamalıdır ve kullanılmaları da gerekmez. Bu dosyalar sadece kullanıcının rahatlığı için oluşturulur.

`$HOME/.ssh/config`
Kullanıcıya özel yapılandırma dosyası. Dosyanın biçimi ve yapılandırma seçenekleri **ssh_config(5)**'de açıklanmaktadır.

`$HOME/.ssh/authorized_keys`
Ev dizini sahibinin oturum açması için kullanılan açık anahtarları (RSA/DSA) listeler. Dosya biçimi **sshd(8)** kılavuz sayfasında açıklanmaktadır. En basit biçimiyle dosya `.pub` kimlik dosyaları ile aynı biçimdedir. Bu dosya çok duyarlı değildir, ancak tavsiye edilen izinler kullanıcı için okuma/yazma hakları ve diğer kullanıcılar için ise erişim hakkının olmamasıdır.

`/etc/ssh/ssh_known_hosts`
Bilinen konakların anahtarlarını listeleyen sistem dosyası. Bu dosya sistem yöneticisi tarafından hazırlanmalıdır ve örgütleşimdeki bütün konakların genel anahtarları dosyaya eklenmelidir. Dosya izinleri herkes tarafından okunabilecek şekilde ayarlanmalıdır. Dosya her satırda belirtilecek şekilde genel anahtarları şu biçimde içerir (alanlar birbirinden boşluk ile ayrılırlar): sistem adı, açık anahtar ve yorum alanı (zorunlu değil). Şayet aynı makina için farklı isimler kullanılırsa, bütün bu isimler virgülle ayrılacak şekilde listelenmelidir. Biçim hakkında detaylı bilgiyi **sshd(8)** kılavuz sayfasında bulabilirsiniz.

Kurallı sistem adı (isim sunucularından dönene ad) istemci için oturum açılırken denetimde **sshd**(8) tarafından kullanılır; diğer isimlere de gerek vardır çünkü **ssh**, anahtarı denetlemeden önce kullanıcının belirttiği ismi kurallı isme çevirmez. Bunun nedeni isim sunucularına erişebilen birinin sistem kimlik denetimini yanıltma olasılığıdır.

`/etc/ssh/ssh_config`
Sistem yapılandırma dosyası. Dosyanın biçimi ve yapılandırma seçenekleri **ssh_config(5)**'de açıklanmaktadır.

`/etc/ssh/ssh_host_key`, `/etc/ssh/ssh_host_dsa_key`, `/etc/ssh/ssh_host_rsa_key`
Bu üç dosya sistem anahtarlarının özel kısmını içerirler ve **RhostsRSAAuthentication** ve **HostbasedAuthentication** için kullanılırlar. Şayet **RhostsRSAAuthentication** yöntemi protokolün 1. sürümü için kullanılıyor ise, **ssh** `root` yetkileriyle (`setuid root`) çalışmalıdır, zira sistem anahtarı sadece `root` tarafından okunabilir. Protokolün 2. sürümü için ise, **ssh**, **HostbasedAuthentication** için sistem anahtarlarına erişimde **ssh-keysign(8)**'ı kullanır. Bu sayede bu kimlik denetimi yöntemi kullanıldığında ssh'nın `root` yetkileriyle çalışması zorunluluğu ortadan kalkar. Öntanımlı olarak, **ssh** `root` yetkileriyle çalışmaz.

`$HOME/.rhosts`
: Bu dosya `.rhosts` kimlik denetiminde erişim izni olan konak/kullanıcı çiftlerini listeler. (Dikkat: Bu dosya ayrıca **rlogin** ve **rsh** tarafından da kullanıldığından kullanımı güvenli değildir.) Dosyanın her satırında birbirinden boşluk ile ayrılmış bir konak adı (isim sunucuları tarafından sağlanan kurallı ad biçiminde) ve bu konakta geçerli kullanıcı adı bulunur. Bazı makinalarda bu dosyanın erişim haklarının herkes tarafından okunacak şekilde düzenlenmiş olması gerekmektedir (şayet kullanıcının ana dizini NFS bölümünde ise; çünkü **sshd(8)** bu dosyayı `root` olarak okur). Ayrıca bu dosyanın sahibi kullanıcı olmalı ve hiçbir kimsenin bu dosyaya yazma hakkı bulunmamalıdır. Birçok makina için tavsiye edilen erişim yetkileri, kullanıcı için okuma/yazma hakkı ve diğerleri için ise erişimin olmamasıdır.

  sshd(8) öntanımlı olarak `.rhosts` kimlik kanıtlama yönteminden önce RSA konak kimlik denetimini gerektirecek şekilde kurulur. Şayet sunucu makinanın `/etc/ssh/ssh_known_hosts` dosyasında istemcinin konak anahtarı bulunmuyorsa, bu anahtar `$HOME/.ssh/known_hosts` dosyasında bulunabilir. Bunu yapmanın en basit yolu sunucu konağa **ssh** ile oturum açtıktan sonra istemciye **ssh** kullanarak bağlanmaktır. Böylece konak anahtarı kendiliğinden `$HOME/.ssh/known_hosts` dosyasına eklenir.

`$HOME/.shosts`
: Aynı `.rhosts`'un kullanıldığı şekilde kullanılır. Bu dosyanın bulunmasının amacı `.rhosts` kimlik denetiminin sadece **ssh** ile kullanılmasını, **rlogin** ya da **rsh(1)** ile kullanılmamasını sağlamaktır.

`/etc/hosts.equiv`
: Bu dosya `.rhosts` kimlik denetiminde kullanılır. Her satırda kurallı sistem adlarını içerecek şekilde düzenlenir (Biçim hakkında ayrıntılı bilgi **sshd(8)** kılavuz sayfasında bulunabilir). Şayet istemci konak bu dosyada bulunuyorsa, istemci ve sunucu kullanıcı adları aynı olması şartı ile erişime izin verilir. Ayrıca RSA sistem kimlik denetimi normalde gereklidir. Bu dosyaya sadece `root` yazabilmelidir.

`/etc/ssh/shosts.equiv`
: Aynı `/etc/hosts.equiv` gibi işlem görür. **ssh** kullanarak erişimin sağlandığı ancak **rsh**/**rlogin**'in kullanılmaması gerektiği durumlarda faydalıdır.

`/etc/ssh/sshrc`
: Bu dosyadaki komutlar kullanıcı oturum açtığında kullanıcının kabuğu (ya da komut) çalıştırılmadan hemen önce **ssh** tarafından çalıştırılır. Daha arıntılı bilgi için **sshd(8)** kılavuz sayfasına bakınız.

`$HOME/.ssh/rc`
: Bu dosyadaki komutlar kullanıcı oturum açtığında kullanıcının kabuğu (ya da komut) çalıştırılmadan hemen önce **ssh** tarafından çalıştırılır. Daha arıntılı bilgi için **sshd(8)** kılavuz sayfasına bakınız.

`$HOME/.ssh/environment`
: Ortam değişkenlerinin ek tanımlarını içerir. Yukarıdaki *ORTAM DEĞİŞKENLERİ* (sayfa: 8) bölümüne bakınız.

ÇIKIŞ DURUMU

ssh uzak komutun çıkış durumu ile ya da hata olmuş ise 255 ile sonlanır.

İLGİLİ BELGELER

gzip(1), **rsh(1)**, **scp(1)**, **sftp(1)**, **ssh-add(1)**, **ssh-agent(1)**, **ssh-keygen(1)**, **telnet(1)**, **hosts.equiv(5)**, **ssh_config(5)**, **sshd_config(5)**, **ssh-keysign(8)**, **sshd(8)**.

T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, ve S. Lehtinen, SSH Protocol Architecture (SSH Protokol Mimarisi), draft–ietf–secsh–architecture–12.txt, Ocak 2002 (halen geliştiriliyor).

YAZARLAR

OpenSSH, Tatu Ylonen'in özgün ve özgür ssh 1.2.12 sürümünün bir türevidir. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt ve Dug Song birçok yazılım hatasını ortadan kaldırmışlar, yeni özellikler eklemişler ve OpenSSH'ı oluşturmuşlardır. Markus Friedl SSH protokolünün 1.5 ve 2.0 sürümü desteği için katkıda bulunmuştur.

ÇEVİREN

Emin İslam Tatlı <eminislam (at) web.de>, Ağustos 2004

YASAL UYARI

Özgün belgedeki telif hakkı beyanı

–*– nroff –*–

Bu dosya (man1-ssh.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

19 Ocak 2007