

# *SSH ile Sistem Yönetimi*

Can Erkin Acar

## *5. Linux ve Özgür Yazılım Şenliği*

can.acar@pro-g.com.tr  
canacar@openbsd.org

---

---

# *İçindekiler*

- Neden SSH?
  - SSH Protokol Mimarisi
  - Temel SSH Kullanımı
  - Kimlik Doğrulama Yöntemleri
  - Açık Anahtar Kullanımı
  - Ajan, TCP ve X11 Yönlendirme
  - SSH Kullanan Uygulamalar
  - Diğer OpenSSH Özellikleri
  - Son Sözler
- 
-

# Neden SSH – Eski Protokoller

- Sunucu ve sistemlere uzaktan erişim ihtiyacı
    - Komut çalıştırma, dosya transferi, otomatik görevler
    - Telnet, FTP, rsh, rexec, rlogin, rcp
  - Ağlar eskisi kadar güvenli değil
    - Parola ve erişim bilgilerinin açık iletimi
    - Sistemler arası güven ilişkileri
  - Bugün nereye bağlanmak istiyorsunuz?
    - Kandırma saldırıları (spoofing)
    - Aktif bağlantı izleme saldırıları (MITM)
- 
-

# Neden SSH – Gereksinimler

- Eski araçları aratmamalı
  - r\* araçları ile aynı kullanım
  - Hem interaktif hem otamatikleştirilmiş görevler
  - Dosya Transferleri
- Kurulum ve kullanım kolay olmalı
  - Ayrı bir altyapı (örn. Kerberos veya PKI) olmadan çalışabilmeli
  - Farklı yönetim bölgeleri arasında çalışabilmeli
- Eski araçlardaki güvenlik problemlerini çözmeli

# Neden SSH – Tarihçe

- 1995: Tatu Ylönen r\* komutlarının yerini doldurabililen güvenli bir uygulama yazdı.
    - **rsh** yerine **ssh**; **rcp** yerine **scp**
    - Yıl sonunda dünya çapında 20,000 kullanıcıya ulaştı
    - Şubat 1995'te *SSH Communications Inc.* Kuruldu
  - 1996: SSH-v2 tasarlandı.
    - SSH-v1 ile uyumlu değil
    - Çeşitli protokol zafiyetleri giderildi
  - 1999: Son özgür SSH sürümü: 1.2.12
    - OSSH (Björn Grönvall), ardından **OpenSSH** doğdu
- 
-

# Neden SSH – Sonuç

- 2000 yılında yaklaşık **2,000,000** SSH kullanıcısı olduğu tahmin ediliyor.
  - 2006 yılında SSH-v2 protokolü standart önerisi **RFC 4251** (secsh) olarak yayınlandı.
  - Günümüzde işletim sistemlerinin büyük çoğunluğu ve pek çok ağ cihazı *OpenSSH* tabanlı kod ile SSH desteği vermekte.
  - Pek çok farklı işletim sistemi için özgür ve ticari SSH istemci ve sunucu desteği bulunmakta.
  - İnternet biraz daha güvenli ...
- 
-

# SSH Protokol Mimarisi

- Sunucu ile bağlantı açık anahtar kriptografik yöntemleri kullanılarak şifreli olarak oluşturulur.
  - Her sunucunun desteklenen SSH sürümü ve açık anahtar algoritmaları için ayrı birer anahtar çifti vardır.
- Kimlik doğrulama için çeşitli yöntemler kullanılır.
  - Parola, Açık-Anahtar, İnteraktif, GSSAPI, ...
  - Tek bağlantıda farklı yöntemler denenebilir (v2)
- Hedef sistem ile şifreli bir bağlantı oluşturulur
  - Bağlantı üzerinde çok sayıda farklı *kanal* tanımlanabilir (v2).

# Temel SSH Kullanımı

- İlerleyen bölümler *OpenSSH* tabanlı anlatılacaktır.
- Sunuş boyunca çeşitli renk kodları kullanılmıştır:
  - Komut isimleri ve referanslar: `ssh-keygen(1)`
  - Komut satırı anahtarları: `-i identity_file`
  - Komut örnekleri:
    - `ssh -D 8080 canacar@curie.metu.edu.tr`
  - Dosyalar: `/etc/ssh/sshd_config`
  - Yapılandırma Dosyası Seçenekleri
    - `sshd_config`: `Port 22` *Varsayılan ayarr*
    - `ssh_config`: `Compression yes` *Değiştirilmiş değer*

# Temel SSH Kullanımı

- SSH Bağlantısı: `ssh(1)`
    - `ssh kullanıcı_adi@uzak_sistem`
    - `ssh -l kullanıcı_adi uzak_sistem`
  - Dosya Transferi: `scp(1)`, `sftp(1)`
    - `scp kullanıcı_adi@uzak_sistem:dosya hedef_dosya_dizin`
    - `scp dosya kullanıcı_adi@uzak_sistem:hedef_dosya_dizin`
    - `sftp kullanıcı_adi@uzak_sistem`
  - Sunucu yapılandırma dosyaları
    - Sunucu `sshd_config(5)`: `/etc/ssh/sshd_config`
    - İstemci `ssh_config(5)`: `/etc/ssh/ssh_config`
    - Kullanıcı: `$HOME/.ssh/config`
- 
-

# Kimlik Doğrulama Yöntemleri

- Parola Tabanlı Kimlik Doğrulama:
    - sshd\_config: PasswordAuthentication **yes**
    - sshd\_config: KerberosAuthentication **no**
    - sshd\_config: KerberosOrLocalPasswd **yes**
  - Açık Anahtar Kimlik Doğrulama
    - v1: sshd\_config: RSAAuthentication **yes**
    - v2: sshd\_config: PubkeyAuthentication **yes**
  - Soru-cevap (interaktif) kimlik doğrulama
    - sshd\_config: ChallengeResponseAuthentication **yes**
    - sshd\_config: UsePAM **no**
- 
-

# Kimlik Doğrulama Yöntemleri

- GSSAPI ile kimlik doğrulama
  - sshd\_config: GSSAPIAuthentication no
- Host tabanlı kimlik doğrulama
  - v1: sshd\_config: RhostsRSAAuthentication no
  - v2: sshd\_config: HostbasedAuthentication no
  - sshd\_config: IgnoreRhosts yes
  - sshd\_config: IgnoreUserKnownHosts no

# Açık Anahtar Kullanımı

- Açık anahtar kriptografisi
  - Bir açık (public) bir de gizli (private) anahtar çiftinden oluşur
  - Açık anahtar ile şifrelenen bilgi sadece gizli anahtar ile okunabilir. Bu sayede
- SSH protokolünde her sunucunun desteklenen her protokol için bir anahtar çifti vardır.
  - v1: sshd\_config: HostKey /etc/ssh/ssh\_host\_key
  - v2: sshd\_config: HostKey /etc/ssh/ssh\_host\_rsa\_key
  - v2: sshd\_config: HostKey /etc/ssh/ssh\_host\_dsa\_key

# Açık Anahtar Kullanımı

- Anahtar yönetimi için `ssh-keygen(1)`
  - Yaratma: `ssh-keygen -t rsa -f dosya_ismi`
  - Parmakizi: `ssh-keygen -l -f dosya_ismi`
- Her istemci tanıdığı sunucuların açık anahtarlarını tutar ve anahtarlar bağlantı sırasında kontrol edilir.
  - `ssh_config`: `CheckHostIP yes`
  - `ssh_config`: `GlobalKnownHostsFile /etc/ssh/ssh_known_hosts`
  - `ssh_config`: `UserKnownHostsFile $HOME/.ssh/known_hosts`
  - `ssh_config`: `StrictHostKeyChecking ask`
  - `v2: ssh_config`: `VerifyHostKeyDNS no`

# Açık Anahtar Kullanımı

- Örnek gizli anahtar

```
-----BEGIN RSA PRIVATE KEY-----
MIICWQIBAAKBgQDZADyp3kjSBacN01Yzck6BzbSknxqBa/n1CLYgwnM9HIIAv8i+
ztrVcJYR0iNftAajixy3BtPQF2z6h2qUoxSE2NgNA3Y+NigdWEnbXVa0s131Kotd
umVi12GoNjibiVBDydQvHDXPiTv0d1BLuyEZWTMhI37Z3hac4VuWFXg1aQIBIwKB
gAxmad3SMAwAUrBMefRPrLb1z87khSv3ip9f1WD1KygQQfFqC3ieGyIjsM3IztJE
zS3qsS8HtFUIpyRCQJrHfYPUJd9aiPQvtqe1+WyXMkM5bB+mg9xUXnCMmLPoOm1R
Q0kwvpqhU36kYyF7LyiT1TC0YBdDMLL7xpADunfy0hZrAkEA+mi1Z0ubbfSJJC1+
2j/KFmvVh82Kg+FMrWcmifzHMAyFYx2VuBsNBWN16gXwGmYJ4YSDNr1r9cW/4tvX
V3zaRQJBAN3Yku4t8YUwnUOPcC0nLDAN5cC6Q3IPZI2KNUw9oUtTs5GvB1z1mkcR
xKCRFYoIOyMdBtTtMbI+BL1Ai68C0tUCQHmgkp671KMiKQLxhsHHNkVnk5m7mwzb
JT5IC2eYCPoUwnH4bUrLT3e1TsIgjCoUTfExvBNGD93GcyULsbzHnTcCQB1a1HL9
77dzRSxK6D+stJfHEvF0X3q/7jwPy50vRaI1c5ue+Yb1qztSfN8mhh5nVzc2hHAb
G59Xi4NX1XMWQ/sCQEtxyhZ9sRgdBxvJYJjNDBzTUmvAVzAteK2KJ1a0DnuPjVYo
7b4AfWbhhr8QBNfBSW/ks4GLsUpGA0n5t0b0E0Y=
-----END RSA PRIVATE KEY-----
```

- Örnek açık anahtar (tek satır)

```
ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAABIwAAAIEA2QA8qd5I0gWnDTpWM3J0gc20pJ8agWv55Qi2IMJzPRyCAL/Iv
s7a1XCWETojX7QGo4sctwbT0Bds+odq1KMUhNjYDQN2PjYoHVhJ211WjrNd5SqLXbp1YtdhqDY4m4
1QQ8nULxw1z4k79HZQS7shGVkzISN+2d4Wn0Fb1hV4Nwk= canacar@light
```

# Açık Anahtar Kullanımı

- Örnek `.ssh/known_hosts` (her anahtar tek satır)

192.168.10.2 ssh-rsa

```
AAAAB3NzaC1yc2EAAAABIwAAAIEAqHaPX0qYexD00CHelh70SiSTCUtcIUXvCTgd1zcB  
z1iF5jxYztuVqamrbThBY3d2ePEwVmKD1cdI/OY0GBzhjCpeuVHY5hmjjajtavY13NmE  
idMSWPBKn2Bfy00+y9Gb6+RPu4NXGE19xPChw9VByBOAW3uvmWg/EchJzgNu81k=
```

192.168.10.73 ssh-rsa

```
AAAAB3NzaC1yc2EAAAABIwAAAQEAsrrpBZfVZURXj5L95SJQ7UazPbYMOBvN8mDn2Lva  
pczy4SUb1w+0j5QQ3rYqQt+vQjiZhc+E9GogEK6Wpj1iz2pocX7eAdXJVNViCscEbbFJ  
pH/EjvpyPBkDLRUh5mrDP1puPSqU0eGYtPqHHDaMbLtpiJKPXQ1CAAPNSPJPNC0qcXgy  
1Fd+HpIotr6m8e0zwrtM501aPwAosp8FI6cf8f6Bjgc5oIX2Dkk6Rfz1jbvfPkVopMe  
MFI0ZY1m55T3/gLRiP0p07ejwSPJ9QxitrmQ2cG63eS2goqnW7p06a139Ej/fPcW+CRx  
roDAsnhidFf/cq6TP3r7oH0nfEW6yQ==
```

anoncvs.ca.openbsd.org,129.128.5.191 ssh-rsa

```
AAAAB3NzaC1yc2EAAAABIwAAAIEAsQpVyGYI7vjnNUfWBSQe2jq9FdgV/S4/yvBSIcRh  
PpuyPeU1NXLf9Vey9paxbowhcCyu+xk/Mwz+L15UPg9If2PYN0NG7+ayNqTpS+eP6bE6  
rbqtCdFSBEM9zRuZU1n14kGwSgJYQqcT/qDt80Ro8Z+zSh9MCQuLbIrspSKYx88=
```

# Açık Anahtar Kimlik Doğrulama

- Her kullanıcı kendi açık anahtar çift(ler)ini kendisi yönetir. Açık ve özel anahtarlar kullanıcının ev dizininde bulunur:
    - v1: `$HOME/.ssh/identity` ve `$HOME/.ssh/identity.pub`
    - v2: `$HOME/.ssh/id_rsa` ve `$HOME/.ssh/id_rsa.pub`
    - v2: `$HOME/.ssh/id_dsa` ve `$HOME/.ssh/id_dsa.pub`
  - Kullanıcı hedef sistemde, hedef ev dizininin altındaki `.ssh/authorized_keys` dosyasına açık anahtarını ekler.
  - Kullanıcı anahtarları için uzun ve tahmini zor bir *passphrase* seçilmelidir.
- 
-

# Açık Anahtar Kimlik Doğrulama

- Örnek `.ssh/authorized_keys` (her anahtar tek satır)

ssh-rsa

```
AAAAB3NzaC1yc2EAAAABIwAAAIEA2QA8qd5I0gWnDTpWM3J0gc20pJ8agWv55Qi2IMJz  
PRyCAL/Ivs7a1XCWETojX7QGo4sctwbT0Bds+odq1KMUhNjYDQN2PjYoHVhJ211WjrNd  
5SqLXbp1YtdhqDY4m41QQ8nULxw1z4k79HZQS7shGVkzISN+2d4Wn0Fb1hV4Nwk=  
canacar@light
```

ssh-dss

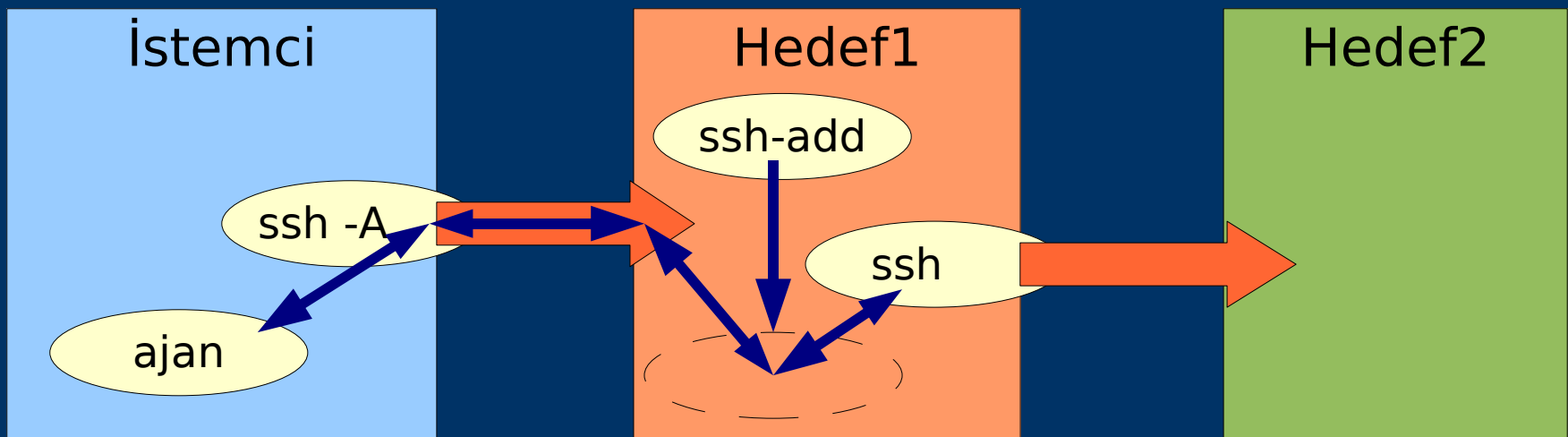
```
AAAAB3NzaC1kc3MAAACBALLstc7xqhX66BFIUj68gtm83e4c0C1wzknKvVKhL08Rhz2K  
BStCa/53vFAZC83vkCw01k/duxA+AU1IQg0J6rA/zn4ZmtwJSZaRYyFyUg/D+0IGUJFe  
R1biEU32MZT3IXikyjqbG69W6AnkmFmVbeQimLWK2ESD0adANp8vku/bAAAAFQCIrGpk  
5GQJo1Vwb4WvKaCn0UV1pQAAAIbB8P3mCKvx+eAKpPg9XMPHrYm0xofwDHxQaMPIjqbg  
WB04hyJXHwwpD1+nAfftU3SNVdaCwqdKiQn0oBvng6qBAZf1C1SB4Pdxsq3d69T5kLzf  
mUwsqKj1c6AmgqUF85xI90xbT+5HTUN5aVLAZvVr5Ww39zxuExwX2tgabXfgaQAAAIAX  
ekiDePcKc/Fh1JMXqAEFbgNDmm+QbpkRDsC7XYLJjCptw48vFAhkkQhW1L1V3tbjXCXk  
s3IntJkYI70umFUEod+//u3v0bwBuq9yCovxIs2b9C1bVWLNbnHCZDm+gz0iiec7dCcq  
1j70pJ+pViUfIy1ce671meQU8ieMj6PpBQ== canacar@curie
```

# Ajan Kullanımı

- Kullanıcının gizli anahtarlarını oturum boyunca saklamak için `ssh-agent(1)` ajanı kullanılır
    - Passphrase sadece anahtarı ajana yüklerken kullanılır.
    - Ajanda yüklü gizli anahtarlar için ssh bağlantısı sorgusuz gerçekleşir.
  - Ajan arka planda çalışır, ssh komutu ajana erişmek için `SSH_AUTH_SOCK` çevre değişkenini kullanır.
    - `eval `ssh-agent``
  - Ajan üzerindeki anahtarların yönetimi `ssh-add(1)` komutu ile gerçekleştirilir.
- 
-

# Ajan Yönlendirme

- SSH bağlantılarında istemciye ajana olan bağlantı da uzak sistemden erişilecek şekilde yönlendirilebilir: **-A**
  - **ssh -A uzak\_sistem**
- Uzak sistem de istemciye ajanı kullanacaktır:

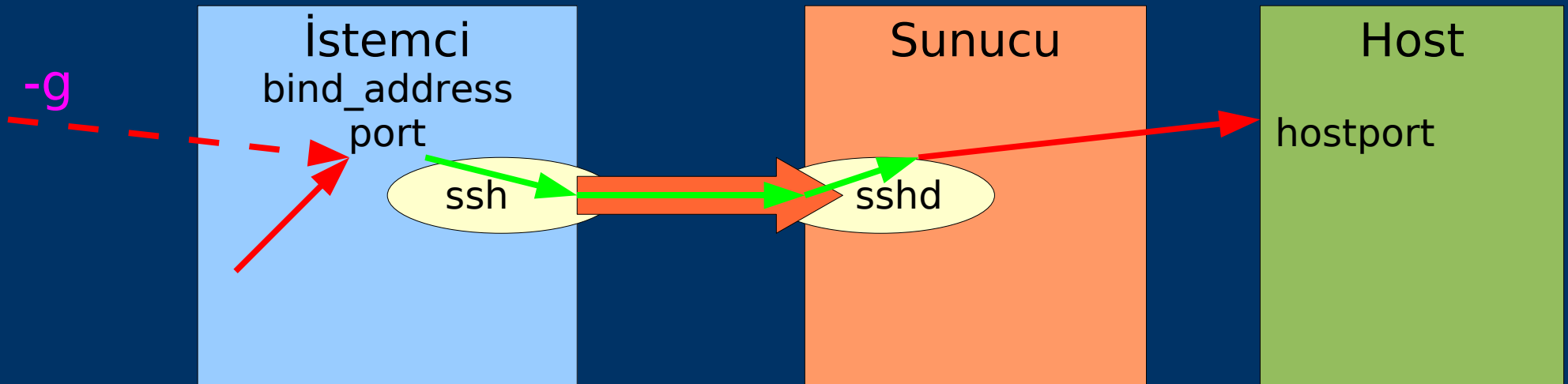


# TCP Yönlendirme

- Kurulan SSH bağlantısı üzerinden TCP bağlantılarını her iki yönde de aktarır.
  - Sunucu üzerinde izin vermek gerekebilir
    - sshd\_config: `AllowTcpForwarding yes`
  - Üç farklı şekilde yönlendirme yapılabilir
    - Yerel: `-L [bind_address:]port:host:hostport`
    - Uzak: `-R [bind_address:]port:host:hostport`
    - Dinamik: `-D [bind_address:]port`
- 
-

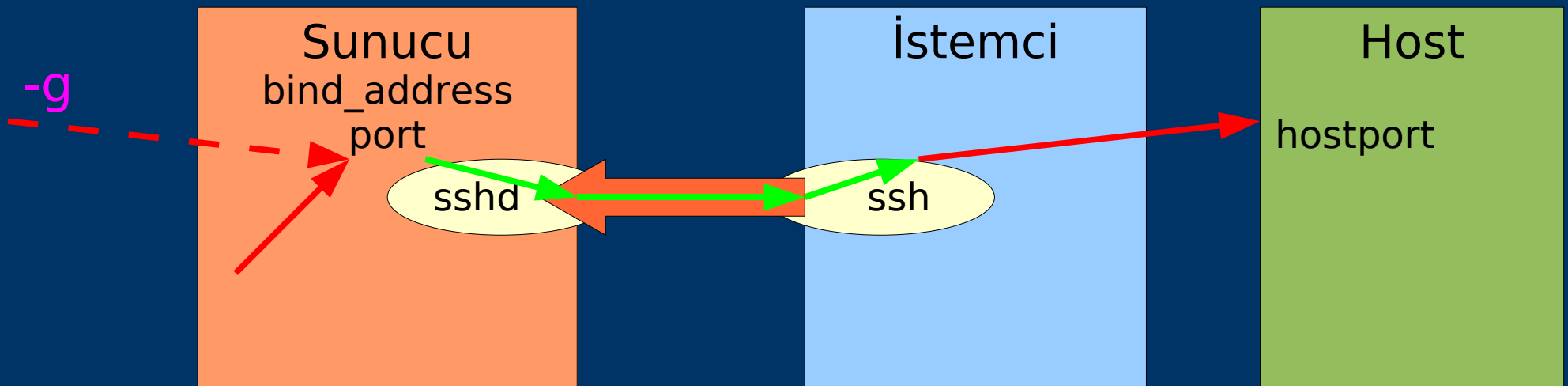
# TCP Yönlendirme – Yerel

- -L [bind\_address:]port:host:hostport
  - bind\_address: istemci dinleme adresi
  - port: istemcideki yerel port numarası
  - host: TCP bağlantısının hedef adresi
  - hostport: hedef port numarası



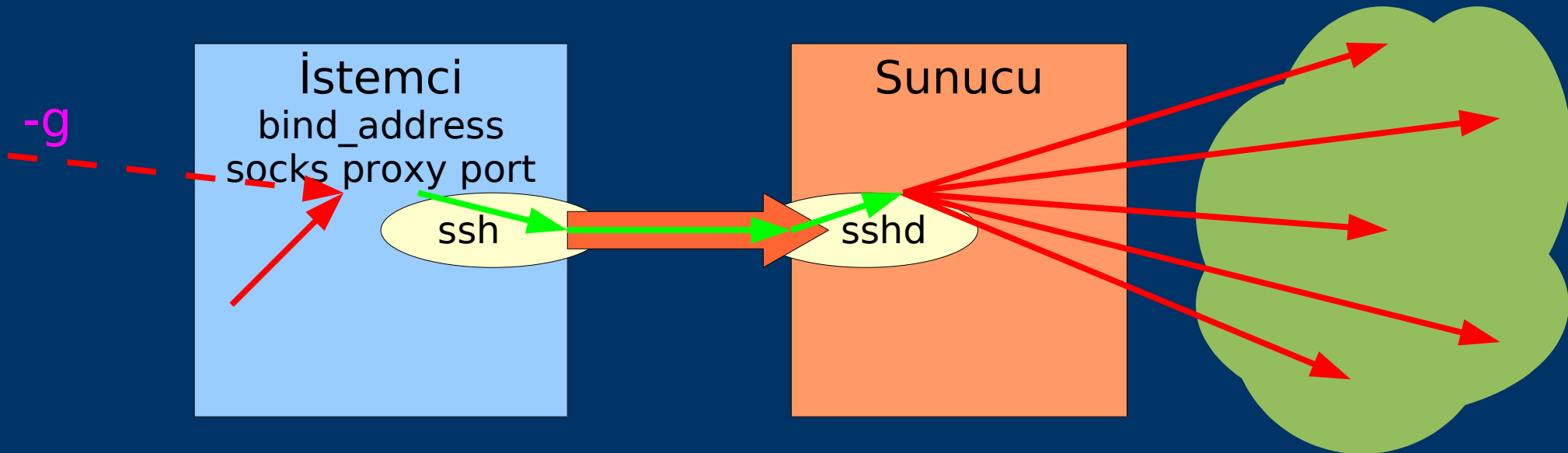
# TCP Yönlendirme – Uzak

- `-R [bind_address:]port:host:hostport`
  - `bind_address`: sunucu dinleme adresi
  - `port`: sunucudaki yerel port numarası
  - `host`: TCP bağlantısının hedef adresi
  - `hostport`: hedef port numarası



# TCP Yönlendirme – Dinamik

- **-D [bind\_address:]port**
  - **bind\_address**: istemci dinleme adresi
  - **port**: istemcideki socks\_proxy servisi verilecek olan yerel port numarası
- SOCKS protokolü ile farklı hedefler ...



# TCP Yönlendirme – Kullanım

- Çeşitli bağlantıların güvenliğini sağlama: **-L**, **-R**
- Sadece uzak sistemden erişilebilecek kaynaklara erişim: **-D**, **-L**, **-R**
- TCP yönlendirmenin varsayılan kurulumda açık olması bir zafiyet yaratmamaktadır. Kabuk erişimi olan kullanıcılar başka yöntemlerle de port yönlendirmesi yapabilirler.
- **Kullanıcılara kısıtlı bir kabuk ile erişim verdiğiniz durumlarda TCP yönlendirme kapatılmalıdır**
  - anoncvs, authpf, sconly ...

# X11 Yönlendirme

- Uzaktaki bir grafik uygulamaya erişim
  - **Sunucu**: Sizin Bilgisayarınız
  - **İstemci**: Uzak sistem
- Klasik X11 erişimi:
  - sunucu üzerinde erişim ayarı
    - .Xauthority, xhost, xauth, ...
  - istemci üzerinden uygulamayı çalıştırma
    - **DISPLAY=sunucu:0 xterm**
    - TCP port 6000 üzerinden şifresiz bağlantı

# *X11 Yönlendirme – SSH üzerinden*

- Hedef sshd ile X11 tüneli kurulur
  - sshd\_config: **X11Forwarding yes**
- Farklı *X11 Kimlik doğrulama anahtarı*
  - sshd\_config: **XAuthLocation ...**
- Uzak sistemde: **DISPLAY=localhost:10.0**
  - sshd\_config: **X11UseLocalhost yes**
  - sshd\_config: **X11DisplayOffset 10**
- Uygulama çalıştırılır
  - X11 protokolü mesajları SSH üzerinden aktarılır

# X11 Yönlendirme – Güvenlik

- X11 istemcileri diğer istemcilere erişebilir
  - Pencere içeriğine erişim
  - Klavye ve fare izleme
  - Uzak sistemde dosyalarınızı okuyabilen birisi (örn. root) X11 sunucunuza da erişebilir
- X11 protokolü SECURITY eklentileri
  - İstemcileri Güvenilir ve Güvenilmez diye ayırır
  - Güvenilmeyen istemciler Güvenilir istemcilerin verilerine erişemez.

# X11 Yönlendirme – Kullanım

- SSH üzerinden X11 erişimi:
  - `ssh -X istemci`
- İstemcide X11 SECURITY desteği varsa:
  - Güvenilmez: `ssh -f -X istemci xterm`
  - Güvenilir: `ssh -f -Y istemci xterm`
  - Güvenilmez bağlantılar: 20 dakika
  - Yapılandırma Dosyası Ayarları
    - `ssh_config`: `ForwardX11Trusted no`
    - `ssh_config`: `ForwardX11 no`

# SSH Kullanan Uygulamalar – CVS

- CVS popüler bir sürüm yönetim sistemidir. Pek çok özgür yazılım projesi CVS üzerinde geliştirilmektedir.
- Uzak CVS sunucularına bağlantı SSH üzerinden gerçekleştirilebilir:
  - `cvscvs -d user@sunucu:/repo checkout`
- Bazı CVS kurulumlarında `CVS_RSH` çevre değişkenini SSH kullanacak şekilde değiştirmek gerekebilir.
  - `CVS_RSH=/usr/bin/ssh`

# SSH Kullanan Uygulamalar – rsync

- rsync gelişmiş bir dosya senkronizasyon aracıdır.
    - <http://rsync.samba.org/>
  - Sadece değişiklikleri aktarabildiği için iki dizin arası senkronizasyon çok hızlı gerçekleşir.
  - `scp(1)` benzeri bir şekilde SSH üzerinden `rsync(1)` kullanmak mümkündür:
    - `rsync -azP user@sunucu:kaynak_dizin .`
  - Eski sürümlerde `RSYNC_RSH` çevre değişkenini SSH kullanacak şekilde değiştirmek gerekebilir.
    - `RSYNC_RSH=/usr/bin/ssh`
- 
-

# SSH Kullanan Uygulamalar – VNC

- VNC uzaktan masaüstü erişimi için yaygın kullanılan bir protokoldür.
  - Protokol erişim bilgilerini ve oturum içeriğini şifrelemeden aktarır. Şifreleme için modern VNC istemcilerinde SSH desteği bulunmaktadır.
    - **vncviewer -via <sunucu> <hedef>**
  - Bu sayede VNC bağlantısının sunucuya kadar şifreli gitmesini sağlar.
    - SSH'ın TCP yönlendirme özelliği kullanılır
    - VNC *uzak sistem modunda* çalışır
    - **Sunucu – hedef arası bağlantı şifreli olmayacaktır**
- 
-

# Diğer OpenSSH Özellikleri

- SmartCard desteği
    - SSH RSA özel anahtarları için bir smart-card okuyucu kullanabilmektedir.
    - SmartCard desteği derleme sırasında aktif hale getirilmelidir.
  - VPN özelliği
    - Yeni sürümde tun(4) arabirimi kullanılarak iki sistem arasında ağ bağlantısı kurma özelliği sağlanmıştır
  - Yüksek Güvenlik
    - **Privilege Separation** tekniği ile `sshd(4)` sunucusunun root yetkileri ile çalışan bölümü küçültülmüştür.
- 
-

# Son Sözler

- OpenSSH, OpenBSD projesi tarafından geliştirilmektedir.
    - milyonlarca kullanıcı
    - ticari ve ücretsiz pek çok işletim sistemi ve ağ cihazları
  - Geçtiğimiz aylarda projen maddi destek isteğinde bulunmuştur.
    - Projenin sunucu, ağ ve elektrik ihtiyaçları
    - Yıllık OpenBSD geliştirici toplantıları (hackathon)
    - Kullanıcılar ve çeşitli projelerden çok miktarda destek ve bağış gelmiştir. **Çok teşekkürler.**
    - **Hiçbir Linux/UNIX sisteminden destek gelmemiştir.**
- 
-

*Teşekkürler ...*

<http://www.openssh.com/>

