Server Hardening

Αυξημένη προστασία με χρήση ssh και fail2ban

Credit to Daniel Waymel and Corrin Thompson

Outline

- Introduction
- Securing Your Access
- Restrict Unwanted Access
- Monitoring and Alerts

- Note: Slides provide a good basic overview of material covered, but in-person demos will be important to a full understanding.

Systems

- Linux (Ubuntu 14.04) Server
  - Always on
  - Always connected to the internet
- Client used to administer server
  - Could be anything, we will use Ubuntu 14.04

Objectives

- Understand how to:
  - Configure secure remote access
  - Defend against basic network based attacks
  - Configure remote alerts and monitoring
  - Apply concepts we talk about in new and exciting ways

Securing Your Access

Objectives

- Access your system securely and reliably via a command shell

- Provide basic authentication measures to prevent others from accessing your server

- We will look at telnet (bad) and SSH (good)
- Other relevant protocols:
  - SFTP (Secure File Transfer Protocol)
  - HTTPS (Secure Hypertext Transfer Protocol)
  - RDP (Remote Desktop Protocol)

First Thing’s First

- Basic server access
  - Maybe physical access available
  - Maybe not!

- In any case, we need to have a user on our server:

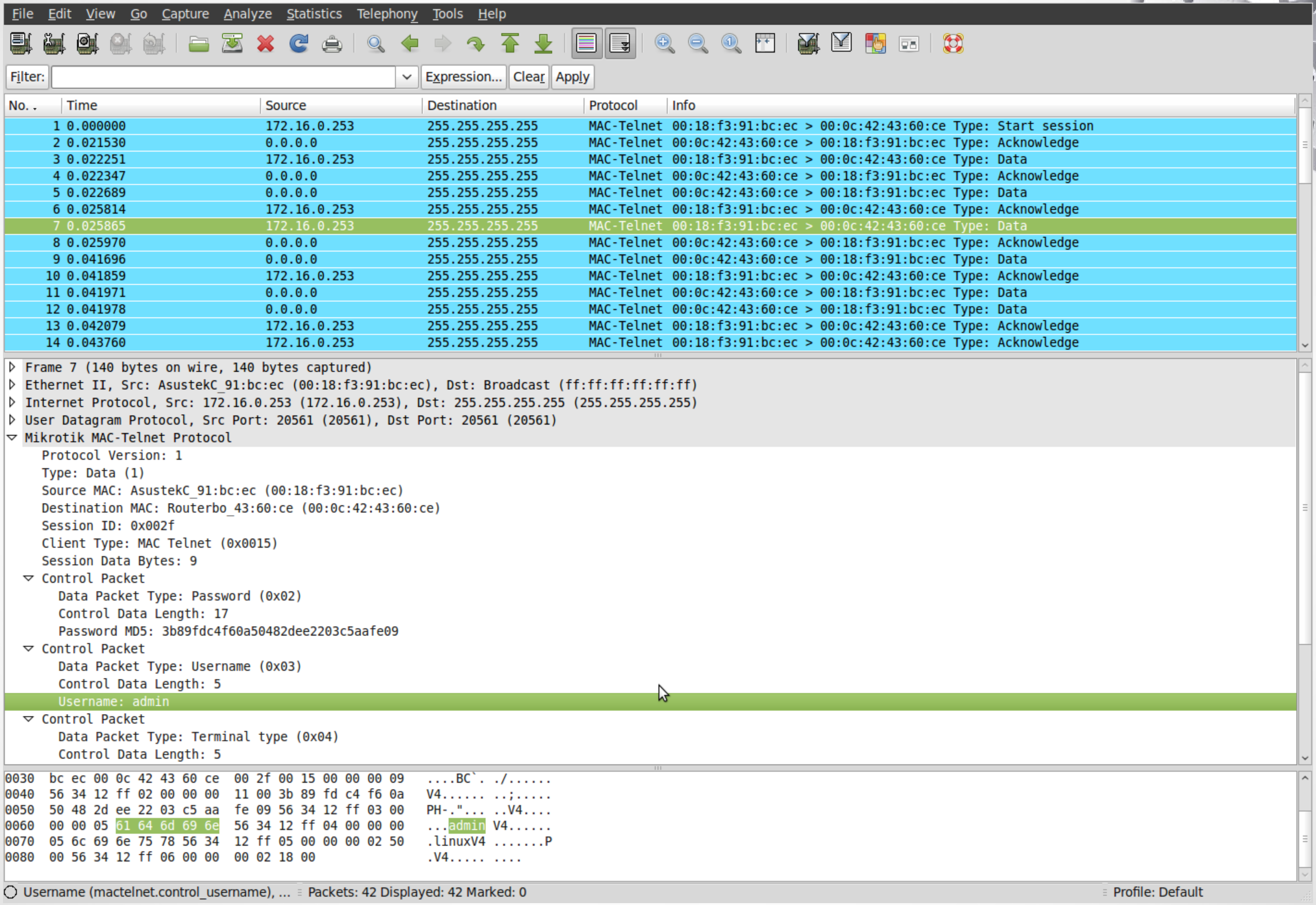
```
# adduser <username>
# adduser <username> sudo
```

Telnet – The “Old School” Solution

- Username/Password authentication
- Grants shell on remote computer

*** TELNET IS COMPLETELY UNENCRYPTED! ***

- Why do we care?


File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply

```
No. . Time         Source        Destination      Protocol   Info
    1 0.000000     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Start session
    2 0.021530     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Acknowledge
    3 0.022251     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Data
    4 0.022347     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Acknowledge
    5 0.022689     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Data
    6 0.025814     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Acknowledge
    7 0.025865     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Data
    8 0.025970     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Acknowledge
    9 0.041696     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Data
   10 0.041859     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Acknowledge
   11 0.041971     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Data
   12 0.041978     0.0.0.0       255.255.255.255  MAC-Telnet 00:0c:42:43:60:ce > 00:18:f3:91:bc:ec Type: Data
   13 0.042079     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Acknowledge
   14 0.043760     172.16.0.253  255.255.255.255  MAC-Telnet 00:18:f3:91:bc:ec > 00:0c:42:43:60:ce Type: Acknowledge
```

```
▷ Frame 7 (140 bytes on wire, 140 bytes captured)
▷ Ethernet II, Src: AsustekC_91:bc:ec (00:18:f3:91:bc:ec), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Internet Protocol, Src: 172.16.0.253 (172.16.0.253), Dst: 255.255.255.255 (255.255.255.255)
▷ User Datagram Protocol, Src Port: 20561 (20561), Dst Port: 20561 (20561)
▽ Mikrotik MAC-Telnet Protocol
    Protocol Version: 1
    Type: Data (1)
    Source MAC: AsustekC_91:bc:ec (00:18:f3:91:bc:ec)
    Destination MAC: Routerbo_43:60:ce (00:0c:42:43:60:ce)
    Session ID: 0x002f
    Client Type: MAC Telnet (0x0015)
    Session Data Bytes: 9
  ▽ Control Packet
      Data Packet Type: Password (0x02)
      Control Data Length: 17
      Password MD5: 3b89fdc4f60a50482dee2203c5aafe09
  ▽ Control Packet
      Data Packet Type: Username (0x03)
      Control Data Length: 5
      Username: admin
  ▽ Control Packet
      Data Packet Type: Terminal type (0x04)
      Control Data Length: 5
```

```
0030  bc ec 00 0c 42 43 60 ce  00 2f 00 15 00 00 00 09   ....BC`. ./......
0040  56 34 12 ff 02 00 00 00  11 00 3b 89 fd c4 f6 0a   V4...... ..;.....
0050  50 48 2d ee 22 03 c5 aa  fe 09 56 34 12 ff 03 00   PH-."... ..V4....
0060  00 00 05 61 64 6d 69 6e  56 34 12 ff 04 00 00 00   ...admin V4......
0070  05 6c 69 6e 75 78 56 34  12 ff 05 00 00 00 02 50   .linuxV4 .......P
0080  00 56 34 12 ff 06 00 00  00 02 18 00              .V4..... ....
```

Username (mactelnet.control_username), ... Packets: 42 Displayed: 42 Marked: 0 Profile: Default

SSH – A Better Solution

- SSH (Secure Shell) provides telnet functionality through an encrypted tunnel
- You can authenticate with a password, crypto keys, or both (more on this in a minute)
- Highly configurable based on your needs

ssh overview

- Purpose
- Protocol specifics
- Configuration
- Security considerations
- Other uses

purpose

- A network protocol
  - Uses public-key cryptography to establish a secure connection between hosts
- Intended to replace the clear-text telnet protocol
- Client-server model
- Commonly used to connect to a *shell* remotely
  - Also supports
    - port forwarding
    - tunneling
    - file transfers
    - …

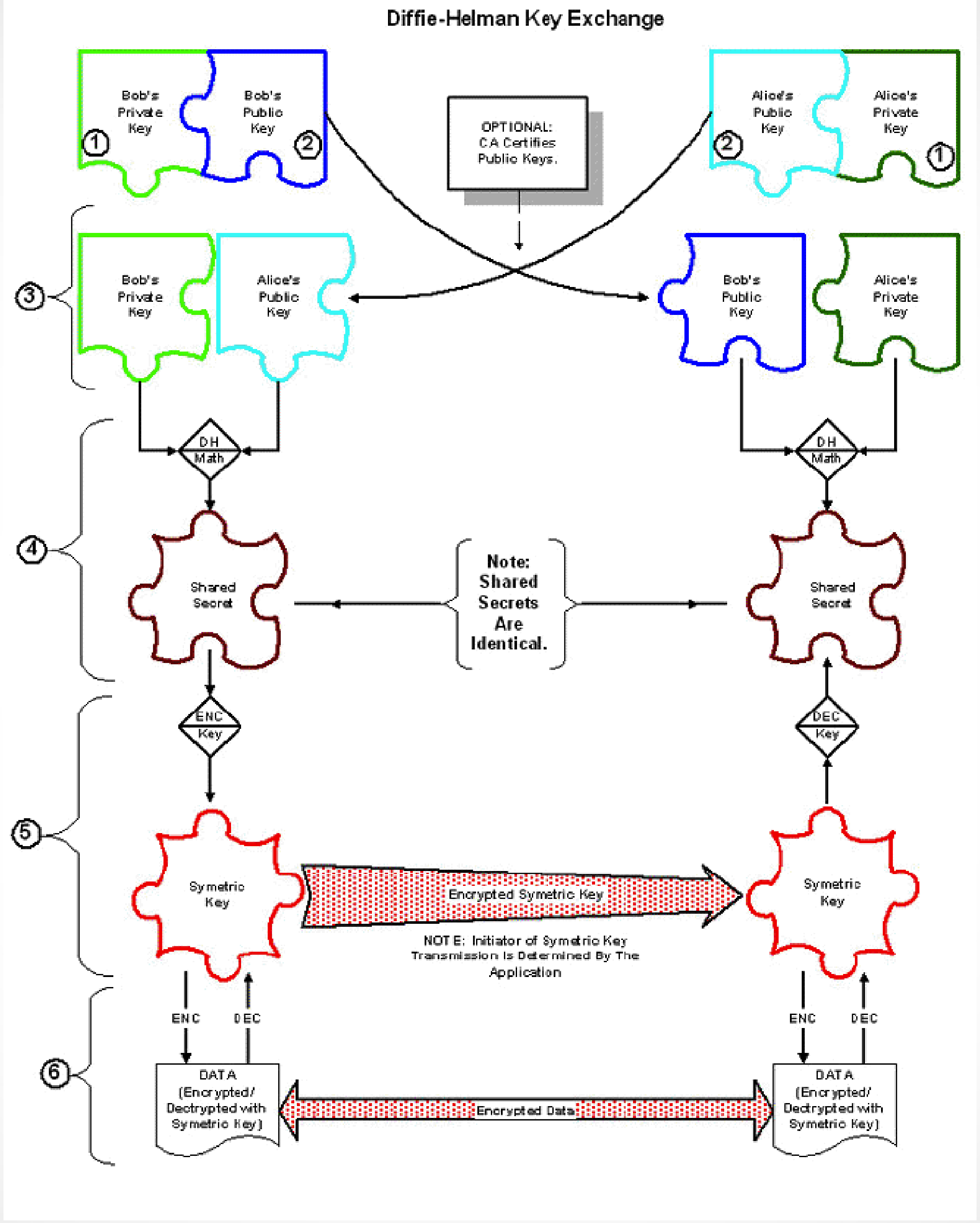
Important Note

- Two programs needed to make a secure connection
  - SSH client
    - Typically ssh
    - Typically comes enabled on a lot of systems
  - SSH server
    - Typically sshd
    - Typically needs to be installed and configured
- Allows a one-way connection from client to server
- If another connection is required back another client and server pair is required for the other direction

protocol

- 2 versions (SSH-1, SSH-2)
    - Version 1 is deprecated
        - Did not use *Diffie-Hellman* key exchange
- Server listens on TCP port 22 (default)
- Authentication based on:
    - Passwords
    - RSA/DSA key pairs
    - GSSAPI (Kerberos, NTLM)

diffie-hellman



keys

- For public cryptography each party has 2 keys:
    - 1 public, 1 private
        - Keeps *private*, distributes *public*
        - For each ID or client
- Stored on **client**
    - Server public key(s): ~/.ssh/known_hosts
        - One for each server you want to go to
    - Client private key: ~/.ssh/id_rsa
- Stored on **server**
    - Server private key: /etc/ssh/ssh_host_rsa_key
    - Client public key(s): ~/.ssh/authorized_keys
        - One for each person allowed
- Note: this is for one-way authentication
    - Another key pair is needed if want to connect from the other direction

Sample keys:

Public Key:

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC8VXxhszucLV0jOu8GlNt6T/eJYW5VICh1GomcsGfhDRS0nNKMHy9IJvCrT9toj+06pdFt4ODVEcqp40IClZnPxq
1duFu/6TrExkDO+LsaZ4M+GWB1sUGIDiPtkafLieN9trUT417RND9/y1nVZ8AutLx3rT15fpi5rN/TJC6A6BIRNeOn/
ETTkGB3NT3cytULBh2fWkaBuzZMpUNXgWxgsxMDOpc2NTZQGd59Kn/
qUlB94lmB1UCytpkINTn9b9H1bM9kIxfMiuomV1QMJ5A15qMeZiOgTl5qLi63Kotb7WYW7nt6BKTbBVMc9w1kdLNCcRPrYXaMFYCz3+K2ZN1GPAvGdccNe
Qy9z925UgYnuPbvpj1cMNSEgylZlPXNRjpQKwz6/JcUDTD9brB+eOf7uM1MwSOSLim2baZ+e3bfu/
VfmBEnSPAl15Ba9CyOByMDuHlAOsflmeOfmxMM86Jz03xUy+DDmBCBfSztwOaxJSTJMDhq6hY6AUytRzLErOeMFGM0+yTTux3dJX4i2xe4ieZWAlsqCR0
yjGTaINAY2iY6IKo8NFjKTLWn8TxgIdCxRKMBbrFOJq7zfomLbZwDcWXl4Wo+7s1mov+SmtwiWuGxLkLMPGDci/
W1CGhbcwwLMIYGGHzUQ6wevcwb7GClIlAbnf5dAJi3feTzZGDsIQ== tkombol@tkombol-Latitude-E6320
-----BEGIN RSA PRIVATE KEY-----MIIJKQIBAAKCAgEAvFV8YbM7nC1dIzrvBpTbek/3iWFuVSAodRqJnLBn4Q0UtJzSjB8vSCbwq0/baI/
tOqXRbeDg1RHKqeNCApWZz8atXbhbv+k6xMZAzvi7GmeDPhlgdbFBiA4j7ZGny4njfba1E+Ne0TQ/f8tZ1WfALrS8d609eX6Yuazf0yQugOgSETXjp/
xE05BgdzU93MrVCwYdn1pGgbs2TKVDV4FsYLMTAzqXNjU2UBnefSp/6lIQfeJZgdVAsraZCDU5/W/
R9WzPZCMXzIrqJldUDCeQNeajHmYjoE5eai4utyqLW+1mFu57egSk2wVTHPcNZHSzQnET62F2jBWAs9/itmTdRjwLxnXHDXkMvc/
duVIGJ7j276Y9XDDUhIMpWZT1zUYxkCsM+vyXFA0w/
W6wfnjn+7jNTMEjki4ptm2mfnt237v1X5gRJ0jwJdeQWvQsjgcjA7h5QDrH5Znjn5sTDPOic9N8VMvgw5gQgX0s7cDmsSUkyTA4auoWOgFMrUcyxKznjBRjNPsk07
sd3SV+ItsXuInmVgJbKgkdMoxk2iDQGNomOiCqPDRYyky1p/E8YCHQsUSjAW6xTiau836Ji22cA3Fl5eFqPu7NZqL/
kprcIlrhsS5CzDxg3Iv1tQhoW3MMCzCGBhh81EOsHr3MG+xgpSJQG53+XQCYt33k82Rg7CECAwEAAQKCAgEAqQRTBt8yLPvtLRPTtWVb/
s3LSchdmxmsFUQGoc8Sur7hiSGANu45oZgIvsWBE7qu3MY5SFHblHxOE972u5kEm5oitgwgkv89laCSQuyoBY9GEjH2BklYlUCTb74bBygtOAIDSeDwk/
E+13JooYNlzsS2qvSXSfSaHXAOws8iyN78b+Ob9oMIRZG5cOIgLYj+XtFTPlJnGkAn/
+sEn4BwAexTsL8hOy3QG1zL9ipw95pEYKUFTOZUFM6YUexqqY5zr7zB9o0j65Xzgws2S14qJqVgWISzjkcmpkXh+NG+lXZc+1F1ENEgHcsOht0UcMXmpkcS6F
fkat1VTpgrPyMQDFDqgHRJHkrNcW8donE9NE3cmI1vcMgyFo2QSnNS9qFcnINocr1tvR5N1mxxUGeOL9aMURB/Q2A5MP/
oZNE7t6OS6ZPL2yBxFP3s6FILUIDxOwTTuO01BHX4urpDrJRmYHQIpir3IkdU9crfg4c7gcXeh0WuW3rmRtVIkTC/
hNTA0hYKqCz+aAMnmWofZfETHetOiSYuJYa1FRcIoRsCrJYJhqQs/ktk+k/JAzZXVe2XM/VJVj8/954bdvP1Fn2PcIt4cu8WNUFrjwzjUa6yuBQmF7wr0GO5befYG/
LJd5W8JowLwMRW3Yei4Bj9yzr6kop8dzzAzVC7mhw15c1Hlm0CxxEBAN6PlRh2Co6vd7ARQbgUJAIwokF6wuCmdYGaSQ3Y24/VXFtlXD+ugzd46d/
g314N2aAW43MddJmT9B+hBt0PuDS4vdXrJaMORo1nqTHRYgw/
S3KPygYaF79CB3AA29MOW5WMiauUanUR6ovCLDE9U9p5TzBEmKMgcWbN5z3bll2Hp0AFvENTAD8mVsAmLF7QEF/bWBg/cnbxei42exbDifGK33FH/
VI4b8h6GhfpAg4L7Io8zKVAPevJJDMjTdIYxLGJqp79C88m6vkmK1qz7DeA7DuoIdEXL/
SZRB2f8x+P31qR4vyVppQTX49PKv2Sz9wPp0XlI2Osy9vbaEKfYdMCggEBANihbHKz6ejzbhH6knq5uVKmx1zmOoO0+U9LDN+NaF0YaSO2g54BzeXQtYkTTR
1tH8pxXyEPTfQUSf0pKzJnM+16Tbtr+JEtCk+hcsbfFItawXhOzznhbGTxp1ZSR912C3X41lyq/
IPZI5g9LY4vliPHG0A2QUFOnP8UE1IH35mplsZAKy+nW4KekVpHsCUXk+e93rUVTpK2U4Vyi8BEwKFGg591hpdZ8cwwXQ3swMceQ89Mnn4JJjXlavFaGoOEU
y5aSCvhGrus8Lif81F2Hae8hfbFhykSbqS3BBElJtpbWMjWfx1QnOFieHya7gfWPXjIDICg1/T9HBL/
KZNYTbsCggEAfA+lmMEUGX2ORkMYUzhG6kGZ8M4xm3Cux9PtLR7ZJVBV70yNI6Jv2pg4Jmf/
mzo1OZwIpb6hpIpo5sioPsnocNsaVwiBLmdixKgoFHEXKqSNtgqZHtWkryRraO/RmdDDFJYGl/
JfdWrLR6SxZbE98Ob2UX2raCNJk3jrkfu50eEwRevsicrWtFz2tp2Q1jk9J3HppXqYn9zzspcD/ih52H8FFux+NTrodOQ7b2CfmJzk+hnyKZup6Kly2F6xno/
X9O88gOuljY+wI7o3KJRq9HWVOZv7XcaDIOHeqnTi3ZEhfCceVJZHCPvTpNsIp9kSrSS8paXZweIssR2Y/
KpDqQKCAQEApY0Xd8EOrTv7jjnT3343pnZWPSSk6ypOrM5KFD3Y1+xjzSsaApKWa17InOznLenLNcbWUEmF5VXsBVCE9ovwHzgsV2L4Htow2xIiyOCKrsS4v
dxceXtQfwQ+QbW3vgMMVyfHiiIRwCEdFqcKPXMYZlcu+C9+Rw5w5G7PJQ1nT+NOmktHta9MO9I6eqf2cSJHof50SCb0WSKFSaJ0ModYPufIhwAlz1ypcMY1Fw
MrgAAdCisflGohiWa6B6A4SyHBL9dG+GGbMHnFrRJvyH1rxFhKeIAT/

configuration client

- Example: open-ssh
  - Install the *openssh* package
  - Create keys
    - `ssh-keygen -t rsa -b 4096`
      - Creates id_rsa and id_rsa.pub keys in `~/.ssh/`
  - Give your public key to the ssh *server*
    - in `~/.ssh/authorized_keys`
  - Your private key will be used from `~/.ssh/id_rsa`

Side question: why does the ssh directory begin with a period?

configuration server

- Install the *openssh-server* package
- Configuration file is located in /etc/ssh/sshd_conf
  - listening port
  - protocol (1 or 2)
  - authentication specifics

  - Show config files on laptop
  - /etc/ssh

configuration

Users can transfer files from client to server with

```
  user@computer:/path/to/file
```

So, to copy a file (/home/johnny/file.txt) from Debian to CentOS (/home/alice/file.txt) what is the values for the command?

```
scp username@<source_comp>:/path/to/file
username@<dest_comp>:/path/to/file
```

Note: this can copy from any computer to any computer (remote or local that has an appropriate key pair for each user on the computers

security considerations

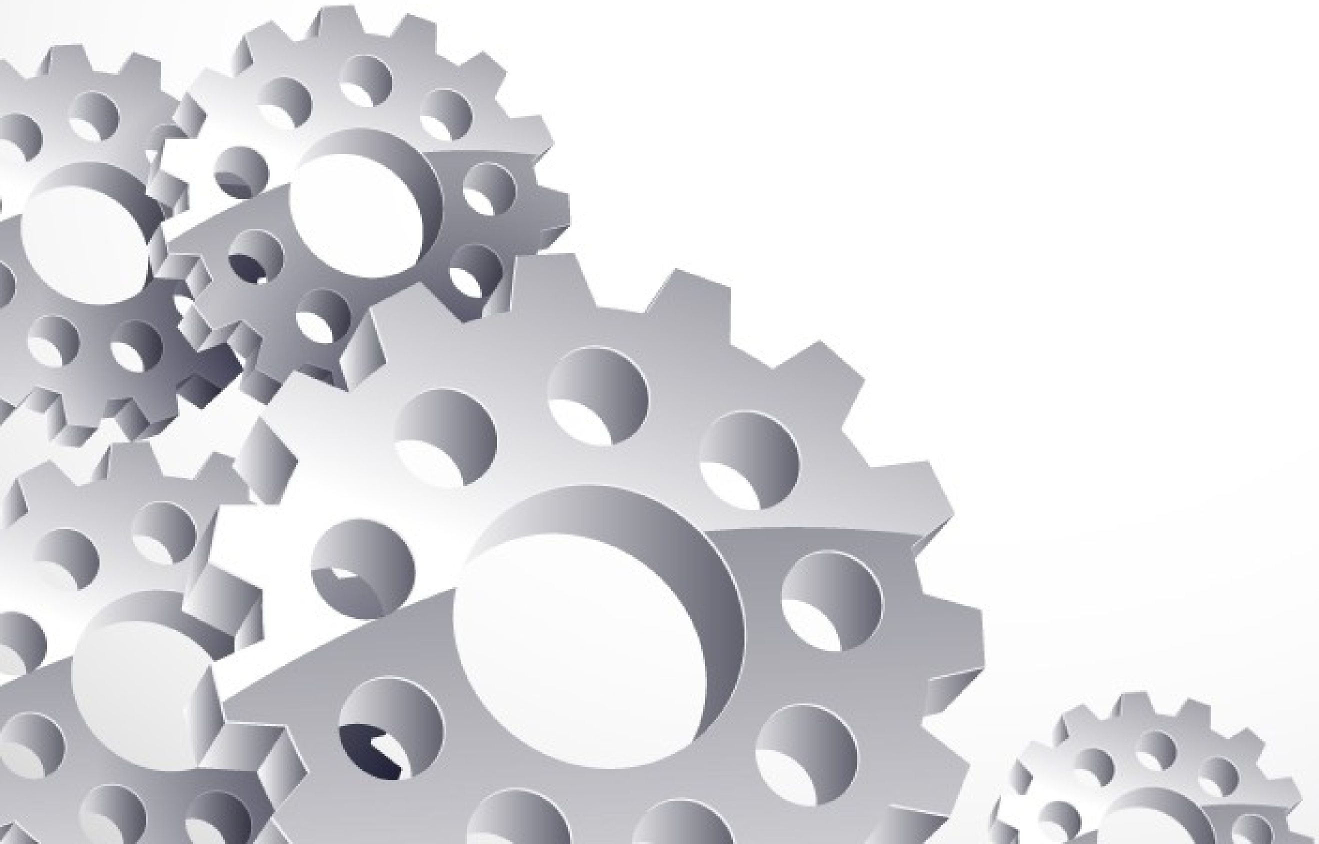
- ssh is often a target of (automated) attacks as outside parties try to gain access to a system
- Brute-force attacks
  - target port 22 (default)
  - try to login using common usernames and passwords
    - (only effective against password-based authentication)
- Prevention:
  - use key-based authentication (with >2048 bit length)
  - change the port sshd listens on
  - use tools to block access after *n* failed login attempts (denyhosts, fail2ban)

denyhosts

- One of many utilities to detect and block brute-force attacks against secure shell
- Blocks attacker IP addresses using TCP wrappers
- Capable of downloading and sharing attacker information with other users
- Default configuration usually acceptable

tcp wrappers

- Host-based Network ACL
- Filters network access to network services
- Services must be compiled against it
    - Most are these days
- */etc/hosts.allow* and */etc/hosts.deny*



other uses

- X11 forwarding
  - ssh -X user@host
- SOCKS Proxy
  - ssh -ND 9999 user@host
- File transfers (SSH FTP, Secure Copy)
  - sftp user@host (put/get/ls, etc.)
  - scp srcFile user@host:dir/destFile

More...

- SSH tutorial
  - Video:
    - https://www.youtube.com/watch?v=hQWRp-FdTpc

Securing SSH

- Restrict access via ssh:

```
# nano /etc/ssh/sshd_config
```

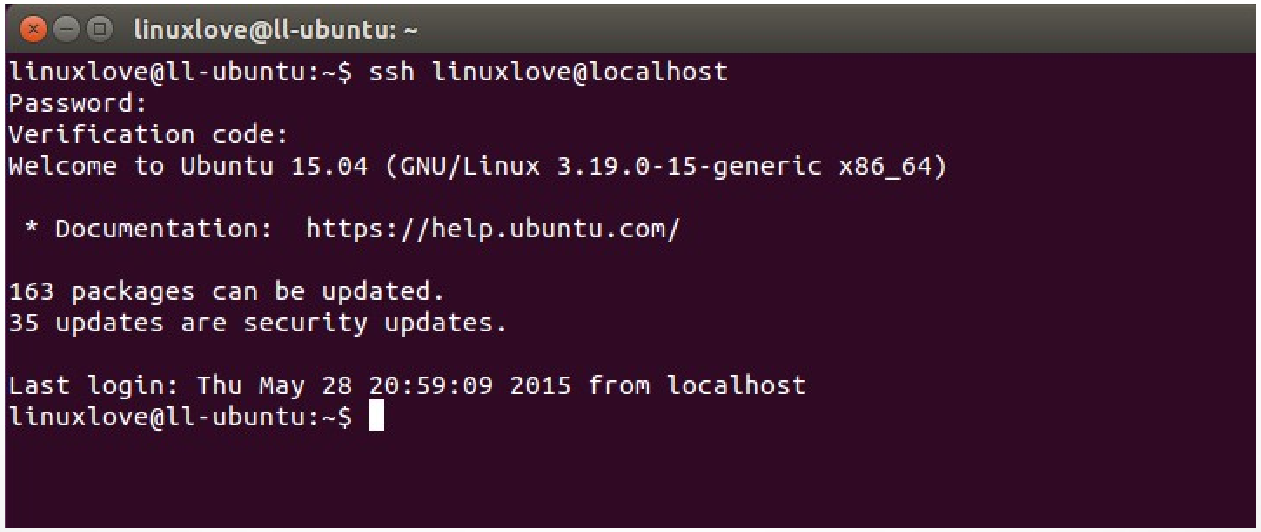
- Additional lines:

```
# PermitRootLogin no
```

  - Disallows ssh access for root account

Securing SSH – Password

- SSH must be installed on server
  - “*sudo apt-get install ssh*” (while connected to internet)
- Client must know username, password, and IP of server



linuxlove@ll-ubuntu: ~

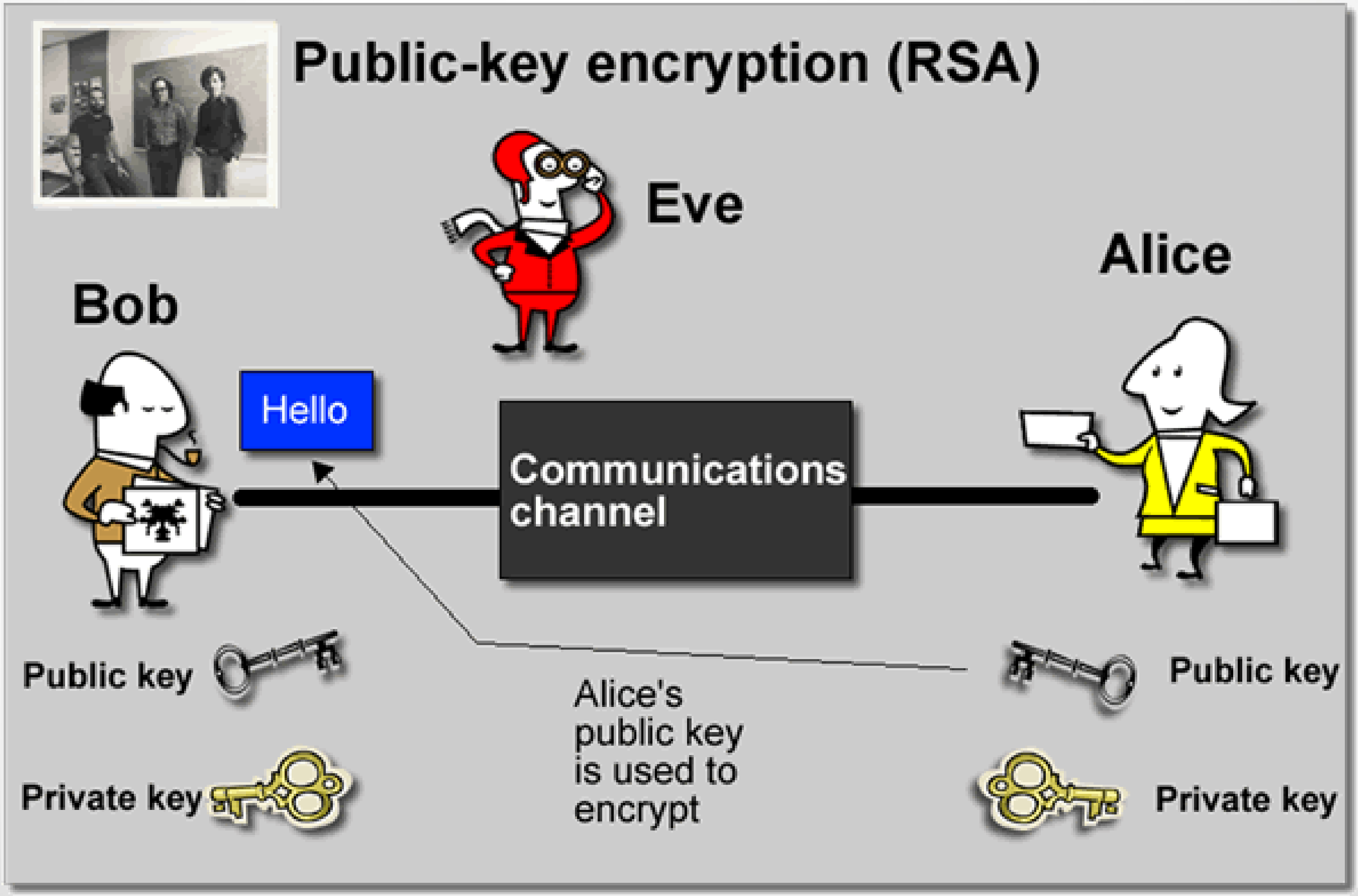
```
linuxlove@ll-ubuntu:~$ ssh linuxlove@localhost
Password:
Verification code:
Welcome to Ubuntu 15.04 (GNU/Linux 3.19.0-15-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

163 packages can be updated.
35 updates are security updates.

Last login: Thu May 28 20:59:09 2015 from localhost
linuxlove@ll-ubuntu:~$
```

Securing SSH – RSA Keys



Securing SSH – RSA Keys

- On client system:

  ```
  $ ssh-keygen –t rsa
  ```

  Now hit enter a few times…

  ```
  $ cd ~/.ssh
  $ ls
  ```

  You should see at least two important files:

  – id_rsa (private key file)

  – Id_rsa.pub (public key file)

Securing SSH – RSA Keys

- How do we let the server know it should trust the client?
  - By giving the server the public key of the client!
  - Trusted public keys should go in :

  ~/.ssh/authorized_keys

  How do we do this?

Restricting Unwanted Access

Objectives

- Be reasonably confident that unauthorized access will be unsuccessful
- We will look at:
  - Lockout of IP addresses following failed access attempts
  - Basic firewall configuration (iptables)
  - Security by Obscurity: using knockd

Other related topics:

Network Intrusion Detection and Prevention Systems (IDS/IPS)

FAIL2BAN
Block bad SSH attempts

- Fail2ban allows easy lockouts following failed connection attempts. Uses Iptables.
  - `Sudo cp  /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`
  - `Sudo services fail2ban restart`
- Can edit jail.local to make changes
  - Enable for more services besides SSH
  - Change ban time
  - Change allowed attempts
  - Whitelist IPs
  - Send alerts by email (or SMS via email-to-SMS gateway)

More...

- https://www.digitalocean.com/community/tutorials/how-fail2ban-works-to-protect-services-on-a-linux-server
- https://regex101.com/
- https://www.digitalocean.com/community/questions/how-to-configure-sendmail-to-send-mail-using-an-external-gmail-smtp-server

/etc/fail2ban/jail.local

```
[nextcloud]
enabled = true
filter  = nextcloud
action =iptables-allports[name=nextcloud, bantime="%(bantime)s", port="%(port)s", protocol="%(protocol)s", chain=INPUT]
logpath = /var/log/joomla/nextcloud.log
findtime = 600
bantime = 600
maxretry = 5

[joomla]
enabled = true
filter  = joomla
action =iptables-allports[name=joomla, bantime="%(bantime)s", port="%(port)s", protocol="%(protocol)s", chain=INPUT]
logpath = /var/log/joomla/error.php
findtime = 600
bantime = 600
maxretry = 5
```

/etc/fail2ban/filter.d/filter.conf

- #[INCLUDES]
- #before = common.conf

- [Definition]
- failregex = ^ \[error\] \d+#\d+: *\d+ user "(?:[^"]+|.*?)":? (?:password mismatch|was not found in "[^\"]*"), client: <HOST>, server: \S*, request: "\S+ \S+ HTTP/\d+\.\d+", host: "\S+"(?:, referrer: "\$
- ignoreregex =
- datepattern = {^LN-BEG}

- [Definition]
- failregex = ^ sogod \[\d+\]: SOGoRootPage Login from '<HOST>' for user '.*' might not have worked(- password policy: \d* grace: -?\d* expire: -?\d* bound: -?\d*)?\s*$
- ignoreregex = "^<ADDR>"

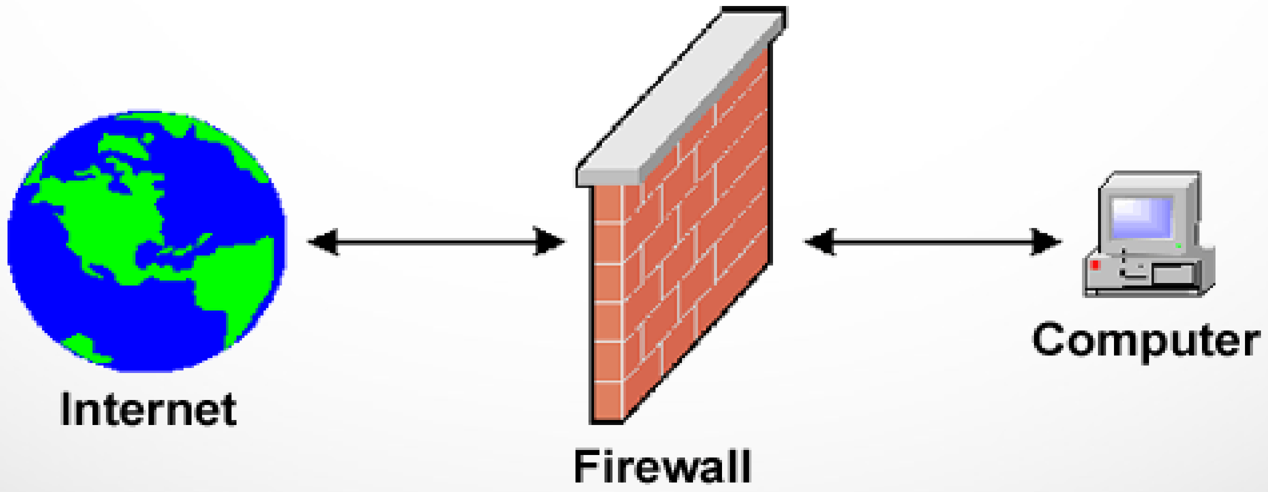
- datepattern = {^LN-BEG}%%ExY(?P<_sep>[-/.])%%m(?P=_sep)%%d[T]%%H:%%M:%%S(?:[.,]%%f)?(?:\s*%%z)?
-             {^LN-BEG}(?:%%a)?%%b %%d %%H:%%M:%%S(?:\.%%f)?(?: %%ExY)?
-             ^[^\[]*\[({DATE})
-             {^LN-BEG}

- service fai2ban stop | start | status | enable

Firewalling Concept

- Two main approaches:
  - Specify what to allow (**Whitelisting**)
    - Allow only these IP addresses
  - Specify what to not allow (**Blacklisting**)
    - Allow all except these IP addresses

In this scenario, whitelist is easy and more effective



IPTABLES

- Firewall rules that runs with kernel privileges
- Firewall sits between your machine and the external world
- Rules are evaluated top-down.
- The first rule that fits is applied, and the rest rules are ignored
- This means that ordering of rules is important

IPTABLES RULES

ALLOW SSH CONNECTIONS

- `iptables –A INPUT –i eth0 –p tcp --dport 22 –m state --state NEW,ESTABLISHED –j ACCEPT`
- `iptables –A OUTPUT –o eth0 –p tcp --sport 22 –m state --state ESTABLISHED –j ACCEPT`

ALLOW SSH CONNECTION FROM A SPECIFIC NETWORK

- `iptables –A INPUT –i eth0 –p tcp –s 192.168.100.0/24 --dport 22 –m state --state NEW,ESTABLISHED –j ACCEPT`
- `iptables –A OUTPUT –o eth0 –p tcp --sport 22 –m state --state ESTABLISHED –j ACCEPT`

Rule ordering is important !

<u>ALLOW only HTTP and SSH</u>

- `iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT`
- `Iptables -A INPUT -I eth0 -p tcp -j DENY`
- `iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT`
- `iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT`
- `iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT`

More...

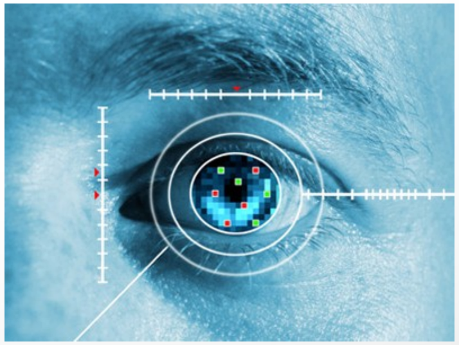
- https://www.digitalocean.com/community/tutorials/how-fail2ban-works-to-protect-services-on-a-linux-server
- https://regex101.com/
- https://www.digitalocean.com/community/questions/how-to-configure-sendmail-to-send-mail-using-an-external-gmail-smtp-server

Authentication

- Three possible ways to authenticate
  - Something you know. (e.g. password)
  - Something you have. (e.g. crypto card)
  - Something you are. (e.g. physical identifiers, fingerprints)

Single-Factor Authentication: choose one type of the above

Two-Factor Authentication: choose two types above



PORT KNOCKING
ADDING ANOTHER LAYER OF SECURITY

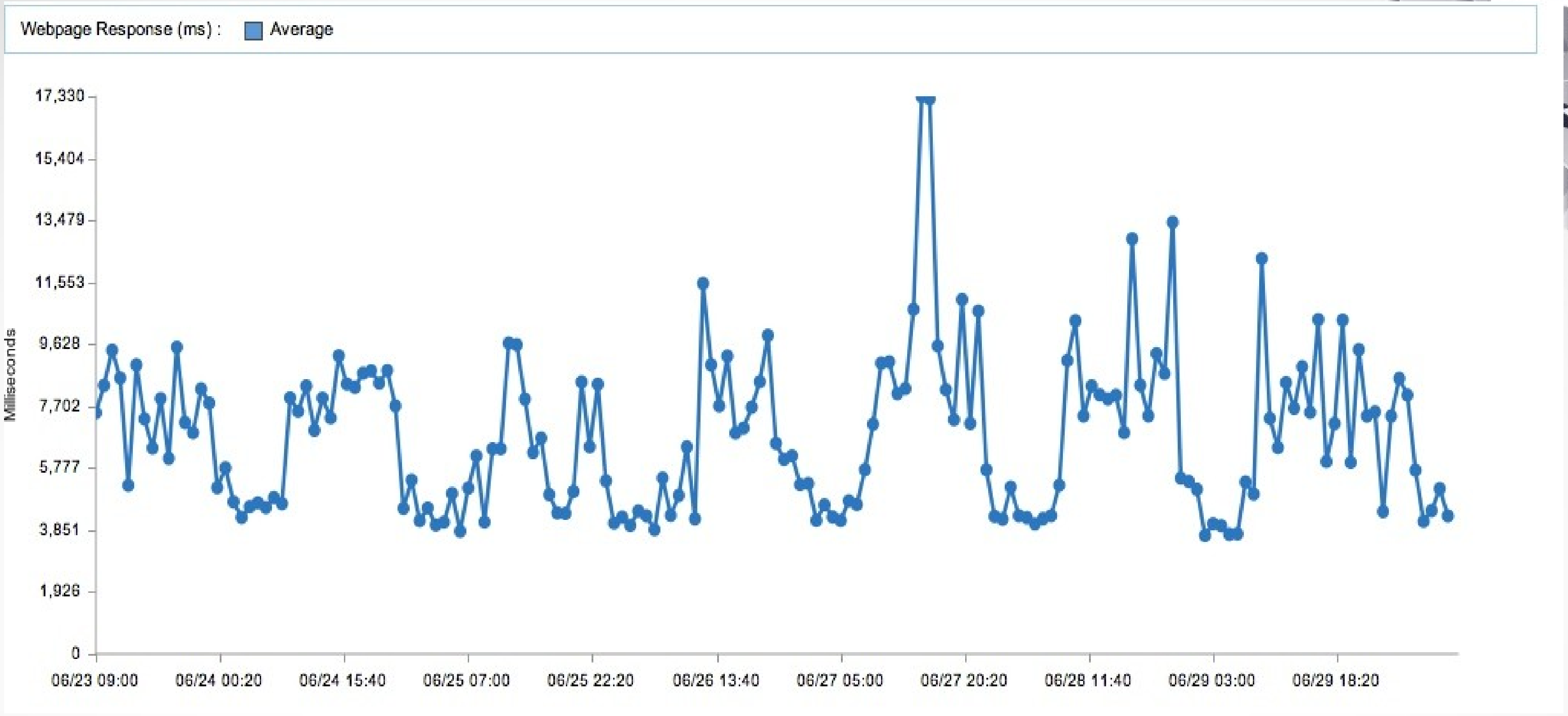
- Port Knocking is similar to two-factor authentication
- Our example case
  - SSH is the only service we are running
  - All ports are closed
  - Requesting a connection (which will be refused as all ports are closed) on a pre-determined sequence of ports within a specified time period will open the port for SSH
  - The port closes automatically after the allowed window has passed

PORT KNOCKING
ADDING ANOTHER LAYER OF SECURITY

- For our scenario
  - Must “Knock” on three ports in sequence within a 10 second period.
  - Standard SSH port 22 will open for 10 seconds receiving connection attempts before closing automatically

PORT KNOCKING
SETTING UP PORT KNOCKING

- `sudo apt-get install knockd`
- This will install both the knockd daemon, as well as the knock utility
- Default configuration is to have one knock sequence to open a port, and another sequence to close the port
  - Problem: What if you forget to close it?



Monitoring and Alerts

Objectives

- Increase awareness of important events on the remote system
- We will look at
  - Automated email/SMS alerts
  - System Logs
- Other related topics
  - Anti-Virus
  - Host-based IDS

Sending Email Notification
SMTP CONFIGURATION

- Monitoring for events and logging is good, but only if those logs and events are known
  - Failed access attempts (SSH in our case)
  - Unexpected system changes (flagged by IDS, such as tripwire)
  - Benign events
    - Task has completed
    - Message received (ex. IRC)
    - ..

Sending Email Notifications
SSMTP CONFIGURATION

- Ssmtp can be used to easily send email notifications

- For this scenario:
  - Create a gmail account to use for sending
  - Configure ssmtp on the system to use that account
  - Create a script to streamline sending notifications

SYSTEM LOGS
TRACKING EVENTS

- Some logs related to topics covered
  - `/var/log/auth.log`
  - `/var/log/syslog`
  - `/var/log/fail2ban.log`
  - `/var/log/mail.log`
  - `~/portknock.log`

- Useful tool to determine what has happened on a given system.
- Acts as timeline of events, unauthorized access, etc

Synopsis

- Covered the following
  - Securing remote access
    - Root Login, Public-key Login
  - Restricting unauthorized access

    Configure Lockouts after bad access attempts, basic firewall rules, and a means of adding more layers of defense
  - Setting up notification of system events
    - Setup email/SMS alerts
  - Brief Look at system logs
- What about a Windows system? Tablets, notebooks, etc

Synopsis

Covered the following for a Linux system:

- Securing (individual) remote access
  - Login using on-root account using keyfiles only; no remote root access permitted
- Restricting unwanted access
  - Configure lockouts after bad access attempts, basic firewall rules, and a means of adding more layers of defense
- Setting up notification of system events
  - Setup email/SMS alerts; discussed means of system changes triggering alerts
- Brief look at system logs

- What about a Windows system? Portable systems (tablets, notebooks, etc)?