AttributeError: module has no attribute in Python

Python

One very common mistake almost everyone makes in **Python** is this: you import a module for some additional functionality, but **Python** won't interpret your code and instead will return you an **AttributeError** message.

AttributeError: module 'csv' has no attribute 'reader'

I needed to parse a CSV file, so I created a new file for the Python code using vim editor:

```
greys@mcfly:~ $cd ~/proj/python
greys@mcfly:~/proj/python $ vim csv.py
```

with the following contents:

```
!/usr/local/bin/python3

import csv

with open('input-data/sample.csv', newline='') as csvfile:
     sample_report = csv.reader(csvfile, delimiter=' ',
quotechar='|')

    for row in sample_report:
        print(', '.join(row))
```

When I attempted to run this code, I received an error:

```
greys@mcfly:~/proj/python $ chmod a+rx csv.py
greys@mcfly:~/proj/python $ ./csv.py
Traceback (most recent call last):
File "./csv.py", line 3, in
import csv
File "/Users/greys/Documents/proj/python/csv.py", line 6, in
face_report = csv.reader(csvfile, delimiter=' ',
quotechar='|')
AttributeError: module 'csv' has no attribute 'reader'
```

csv is a standard module in the Python 3 library, so this puzzled me a bit. It definitely has the **reader** method, as examples on many websites show.

So Why This AttributeError Message?

Having double-checked for typos and indentation, I realised that this mistake is probably related to my filename.

You see, what's happening is that my code tries to import Python module named csv. It's a standard module, but purist approach insists on not making such assumptions. That's why the Python interpreter checks current directory for any modules named csv before it goes searching further in its standard locations.

Because my own example code was saved in the csv.py file, I was making it import itself instead of the standard (global) Python 3 csv module.

Solution for this Type of AttributeError

The fix is simple: rename your file to something else and it will no longer be importing itself. Python will not find any modules with specified name in your current directory and will then assume you're definitely talking about a module from standard library.

Have a look, once I renamed the file it started working:

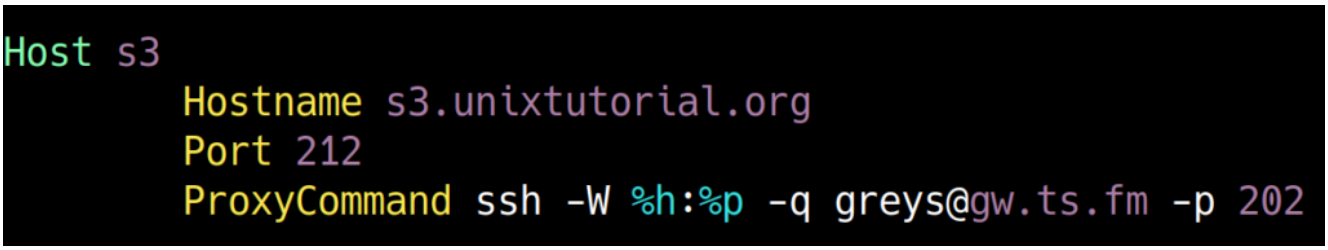
```
greys@mcfly:~/proj/python $ mv csv.py csv-test.py
greys@mcfly:~/proj/python $ ./csv-test.py
username,userid,fullname,homedir,password_hash
```

Hope this saves you time some day, enjoy!

See Also

- Check Python Version
- Converting Epoch Time with Python
- Book Review: Practical Programming Python
- Setting Alternatives Path for Python in RHEL8
- Book review: Introduction to Computer Science with Python
- Book Review: Text Processing with Python

SSH client Config for Using a Jumphost



SSH jumphost configuration in .ssh/config

I needed to use a mobile 4G hotspot today and realised there's another very common reason for using SSH jumphosts. You see, when I'm on a 4G hotspot I tend to use VPN client for securing Internet connection. And jumphosts may help in VPN scenarios.

Sometimes I then switch to home WiFi and need to manually disable VPN, otherwise my connection keeps getting protected. Using VPN on home WiFi allows me to use SSH client on laptop

to use one of my Raspberry Pi servers as jumphost for connecting to external servers.

The reason this works is because laptop is the only system running VPN, but it usually still has access to local networks in my home network. So if I ssh onto one of my Raspberry Pi servers, any connection I make from there on will use my home's broadband IP (because Raspberry doesn't have VPN configured).

How To Specify Jumphost in SSH Client Config

I have my dedicated servers configured like this in /Users/greys/.ssh/config file:

```
Host s3
     HostName s3.unixtutorial.org
     Port 212
```

If I need to use jumphost to access s3, I'll update this configuration setting:

```
Host s3
      HostName s3.unixtutorial.org
      Port 212
      ProxyCommand ssh -W %h:%p greys@gw.ts.fm -p 202
```

Just to remind you, gw.ts.fm is my SSH jumphost name and it's

listening to SSH on port 202.

In case you need to edit this config file in Linux, this is in /home/greys/.ssh/config file for me – so /home/$USER/.ssh/config format.

See Also

- **SSH reference**
- Ansible via SSH jumphost
- SSH port
- **SSH port forwarding**
- Ansible non-standard port

How To Check HTTPS Connection with OpenSSL

OpenSSL

Before I forget about this little addition, I want to write a follow up to the Check SSL Connection with OpenSSL – specifically, show you how to check HTTPS connection to a typical website.

I have migrated UnixTutorial.RU to **Jekyll** CMS and wanted to make sure it has a proper certificate generated by hosting platform of Netlify.

The only difference from previous **SSL connection check with OpenSSL** example is that we use standard HTTPS port – 443 – for connection:

```
greys@maverick:~ $ openssl s_client -connect www.unixtutorial.ru:443
 CONNECTED(00000005)
 depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
 verify return:1
 depth=1 C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA
 verify return:1
 depth=0 C = US, ST = ca, L = San Francisco, O = "Netlify, Inc", CN = *.netlify.com
 verify return:1
 Certificate chain
      0 s:/C=US/ST=ca/L=San Francisco/O=Netlify, Inc/CN=*.netlify.com
    i:/C=US/O=DigiCert Inc/CN=DigiCert SHA2 Secure Server CA
  1 s:/C=US/O=DigiCert Inc/CN=DigiCert SHA2 Secure Server CA
     i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Global Root CA
 Server certificate
 -----BEGIN CERTIFICATE-----
MIIGIzCCBQugAwIBAgIQC1W/C9syOFclqIEumW8/STANBgkqhkiG9w0BAQsFADBN
```

MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMScwJQYDVQQDEx
5E
aWdpQ2VydCBTSEEyIFNlY3VyZSBTZXJ2ZXIgQ0EwHhcNMTkwNzAzMDAwMDAwWh
cN
MjAwNzA3MTIwMDAwWjBhMQswCQYDVQQGEwJVUzELMAkGA1UECBMCY2ExFjAUBg
NV
BAcTDVNhbiBGcmFuY2lzY28xFTATBgNVBAoTDE5ldGxpZnksIEluYzEWMBQGA1
UE
AwwNKi5uZXRsaWZ5LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCgg
EB
AOwJHctI1oxqg0wlz9CZYvbhlVZ6sB0COANCfC9qIjTUpw+TxVnG9/tuRDNkfX
es
f7nvjmPlZcfQQKgpmGahX5ndFZmXvCXGMtvO5ZWnqzfv0mxkzTQb6RdDMtPA9X
n/
LvzaFRlxVpeg6xGMweUpLDTJAJWkcMnGAR2mNUc6nuQgteQsg/7I6YpspYZKas
R/
2xDj5FasQf2+9HqbJgdSSzQKF46VXL2QRBK5UF1lFeBFWMAiUk42Su+YZy/w5p
VF
jlHaVVE42/uG1dyudWAaT65AIIT91eK7GghHv5ppuMZUUmEMV0FUdktMXECqCl
rt
kknMNie30oRFj1ADXJGGHd0CAwEAAaOCAukwggLlMB8GA1UdIwQYMBaAFA+AYR
yC
MWHVLyjnjUY4tCzhxtniMB0GA1UdDgQWBBSJl47dfzdwOfakpcJRog3jMU/JUz
Al
BgNVHREEHjAcgg0qLm5ldGxpZnkuY29tggtuZXRsaWZ5LmNvbTAOBgNVHQ8BAf
8E
BAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMGsGA1UdHwRkMG
Iw
L6AtoCuGKWh0dHA6Ly9jcmwzLmRpZ2ljZXJ0LmNvbS9zc2NhLXNoYTItZzYuY3
Js
MC+gLaArhilodHRwOi8vY3JsNC5kaWdpY2VydC5jb20vc3NjYS1zaGEyLWc2Lm
Ny
bDBMBgNVHSAERTBDMDcGCWCGSAGG/WwBATAqMCgGCCsGAQUFBwIBFhxodHRwcz
ov
L3d3dy5kaWdpY2VydC5jb20vQ1BTMAgGBmeBDAECAjB8BggrBgEFBQcBAQRwMG
4w
JAYIKwYBBQUHMAGGGGh0dHA6Ly9vY3NwLmRpZ2ljZXJ0LmNvbTBGBggrBgEFBQ
cw
AoY6aHR0cDovL2NhY2VydHMuZGlnaWNlcnQuY29tL0RpZ2lDZXJ0U0hBMlNlY3
Vy
ZVNlcnZlckNBLmNydDAMBgNVHRMBAf8EAjAAMIIBBAYKKwYBBAHWeQIEAgSB9Q

```
SB
8gDwAHUA7ku9t3XOYLrhQmkfq+GeZqMPfl+wctiDAMR7iXqo/csAAAFruE2Z1A
AA
BAMARjBEAiBaMAQ2lhU9A/zF2waHg1jyKezSioWSrngikJo9ur9sTgIgSJlL/l
pv
paD9QVDLMjZi3GCE6uIIIJbb0GM3hliV+MsAdwCHdb/nWXz4jEOZX73zbv9WjU
dW
Nv9KtWDBtOr/XqCDDwAAAWu4TZogAAAEAwBIMEYCIQCNuzKN42qTaCdQvNu5P2
b1
Nr/bTj37qaPUfP9iiTLNRgIhALHf05TguQDpldNVWwI5HZjJXybMS4M1XwBhBe
2j
voCgMA0GCSqGSIb3DQEBCwUAA4IBAQA7Va9atxzWXysR5FtM0kDftriiBWvFfD
1I
e1zMJ40ZvUHh7+St+YB2okv0SqrUsJchQ2mQsvyyX3bKn1d7bmfSBSFdeAeE8h
aK
hCFcf4RFsgm76X2rECIr8qZeCuUguaAFEATN82FWYfPpdCH7k30OIjC4xXpz9X
Pz
ZlIn5tB7pkfDfc9LJ1h68hFsdu1hP8nOKUQnVuEUCvz9raikjg3J1Mz3kiDx7W
8l
UEnuc4q2odt6E4jeGcu0pbc/2v5cmc4JTTJlvVD7qk2Q7Y9v39vRCT0LK/JSY+
8x
 kZAJNiddue8vgageQpXojzNT8NN7INbdepRjp4wlAGc8ieDgTfEU
 -----END CERTIFICATE-----
    subject=/C=US/ST=ca/L=San    Francisco/O=Netlify,
Inc/CN=*.netlify.com
 issuer=/C=US/O=DigiCert Inc/CN=DigiCert SHA2 Secure Server CA
 No client certificate CA names sent
 Server Temp Key: ECDH, X25519, 253 bits
 SSL handshake has read 3407 bytes and written 289 bytes
 New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
 Server public key is 2048 bit
 Secure Renegotiation IS supported
 Compression: NONE
 Expansion: NONE
 No ALPN negotiated
 SSL-Session:
     Protocol  : TLSv1.2
     Cipher    : ECDHE-RSA-AES128-GCM-SHA256
                                                 Session-ID:
DA5D473C1896A1F1A2714F88E3D6AA70D3B3E90B1F15AECDC20F2AB1B0A89F
FF
```

```
    Session-ID-ctx:
                                                    Master-Key:
862C7889A4F0A1C44E165D592F004D11C7A5E3AAEE6B897F32DB4789683988
656920A5D2CCF8326477408F85DC9F299B
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
    0000 - a7 ec 5a e8 a3 3d 35 21-68 b9 2a 9c 5c f0 5e f2
..Z..=5!h...^.     0010 - d3 dd de 9f d8 d8 f1 6c-fc 8c 77 5d
ba 40 fd 24   …….l..w].@.$     0020 - 58 05 da d8 df 9f eb
e0-41 6c 2c 0d 6c 51 ca 1e   X…….Al,.lQ..     0030 - ae db 9d
ac 72 aa fa d2-2e 70 08 e6 0f bb a7 45   ….r….p…..E     0040 -
4d d2 d4 bb 62 84 81 b5-d5 9b 8d 7e a6 2a 30 80   M…b……~.0.
    0050 - af b2 4b 8f 41 eb a0 98-b8 92 59 90 a8 dd 67 7d
..K.A…..Y…g}
    0060 - 89 ff 61 eb 37 a1 d8 e6-f8 05 ea d4 de 04 46 24
..a.7……F$
    0070 - 69 fc a9 6a ad 94 02 c4-11 19 d4 c6 d4 03 3b 33
i..j…….;3
    0080 - 24 2b 30 d2 af f3 86 3e-ec 4b f7 c4 87 9a b2 24
$+0….>.K…..$
    0090 - 08 cb e4 83 75 35 1e 34-30 9a 82 75 92 e9 42 d7
….u5.40..u..B.
    00a0 - 03 ab 09 1b a2 fe 7f 8d-9c cb 55 a7 a5 99 03 42
…….U….B
    00b0 - 30 00 d2 80 64 d9 cb 5b-fa 56 af fc 66 65 06 19
0…d..[.V..fe..
 Start Time: 1584575518 Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
 ^C
```

See Also

- curl vs wget
- test SSL with OpenSSL
- Cloudflare Crypto Week
- Test TCP connectivity with curl
- Migrate to Jekyll from WordPress

- Jekyll 4 – install in macOS

How To: Filter RSyslog Messages by String

I think I'm pretty much done with the Centralised RSyslog server project – it's finally in the stage I'm happy enough with. One of the final touches I needed on the server side was to filter more important messages into a separate file, so that I can tail that file instead of looking at all the RSyslog messages arriving from all the servers and appliances.

Why You Might Want to Filter Syslog Messages

Most of the reasons are around focusing on specific subset of the logs.

IMPORTANT: I want to highlight that there's filtering where you drop (ignore) messages and filtering where you just forward certain logs to separate files. In my opinion, you should **never discard logs** – the risk of missing something important is just too high.

So collect everything, but filter it into separate files and only inspect the more useful elements. The rest is stored and quite possibly rotated very often – meaning you're not really wasting that much space. But in a case of some security incident you'll expand research surface and will look into all the logs and not just the usually important ones.

Specific reasons for filtering Syslog Messages

1. **You have multiple teams or team members with different responsibilities** – someone needs to look at application logs, someone else at OS services logs
2. **You have specific processes that run regularly and need monitoring** – for instance, auto-deployments or cron-based automations – they use the same common tools like HTTPS for git hooks or SSH for remote access or SUDO for privilege escalation – but will be lost among other standard traffic about HTTPS, SSH and SUDO
3. **You have parts of logging providing additional, non-essential information** – like, when you ssh onto a server, you have pam_unix session management messages that you may not be interested in. It's important to know when someone's unsuccessfully trying to SSH, but not nearly as important to track when someone logs in or

out.

How To Filter Syslog Messages by Expression

Relatively new approach is using **expression-based filters** in **RSyslog** – they're using a common and readable enough format of if-then:

```
 if $msg contains 'ssh' then /logs/security.log
```

How To Filter Syslog Messages by Application/Service

It's also super useful to filter by program name – this is essentially a process name.

For instance, in this syslog line highlighted **sudo** is the program name and not the text of the logged message:

Mar 29 09:45:38 s7 **sudo**: greys : TTY=pts/0 ; PWD=/home/greys ; USER=root ; COMMAND=/bin/grep -E ^pi: /etc/shadow

The following RSyslog example helps me extract only sudo lines with COMMAND into /logs/security.log file. Any other sudo lines will not be captured in /logs/security.log, but will

still be captured elsewhere.

```
if $programname == 'sudo' and $msg contains 'COMMAND' then
/logs/security.log
```

See Also

- Centralised RSyslog
- RSyslog – log into files by hostname
- RSyslog startup failure: reading fork pipe

---

rsyslog parent startup failure: error reading "fork pipe"

RSyslog

I've been configuring my OpenMediaVault NAS server for RSyslog shipping logs to my centralised RSyslog setver, when I experienced a cryptic error:

```
rsyslog startup failure: error reading "fork pipe": No such file or directory
```

rsyslog didn't start, so it took me a bit to investigate.

Turns out, the issue was mismatch of **RSyslog** config syntax: OMV used one version, my templates used another.

Specifically, I'm using old-syntax multi-line way of describing global TLS settings for configuring client side of RSyslog:

```
global(
        DefaultNetstreamDriver="gtls"
        DefaultNetstreamDriverCAFile="/etc/rsyslog.d/ca.crt"
DefaultNetstreamDriverCertFile="/etc/rsyslog.d/helios4.crt"
DefaultNetstreamDriverKeyFile="/etc/rsyslog.d/helios4.key"
 )
```

But earlier in the file I used a more recent way of configuring RSyslog modules:

```
module(load="imtcp")
input(type="imtcp" port="514")
```

It seems RSyslog doesn't suppot this kind of mixing config

styles – so one of these config stanzas needs rewriting. In my case, I actually only needed imtcp for debug purposes – so I just commented it out and RSyslog restarted just fine.

See Also

- Centralised RSyslog
- RSyslog by Hostname

Migrate iptables to nftables in CentOS 8

iptables to nftables

Although **Ansible** provides support for managing firewall rules via module, I still find initial setup is best done with a tested batch of firewall rules instead of adding them one-by-one. Since I'm migrating CentOS 7 servers to CentOS 8 now, I decided to convert iptables into nftables.

Will probably post a **Unix Tutorial Project** about this, but today I'm just capturing notes.

What is nftables?

nftables is the next (current) generation of NetFilter based firewall solutions, replacing **iptables** and providing backward compatible tools with iptables syntax.

If all you used before is **iptables**, you can continue using familiar commands – but in CentOS 8 this means that on the firewall level there's no longer iptables running, all the functionality is provided by NFT.

How To Save iptables rules/chains into a file

 # iptables-save > /etc/sysconfig/iptables.current

How to Convert iptables rules into nftables rules

 # iptables-restore-translate -f /etc/sysconfig/iptables.current > nft-rules.txt

IMPORTANT: make sure you put this into some **nft-rules.txt** file outside of the **/etc/sysconfig** location – if things go wrong, you'll just reboot server via hosting console and regain access.

Try/Check NFT Ruleset

Now comes the moment to disable iptables and try NFT tables in their place.

I did the following: flushed IPtables (removed any rules) and then applied NFT rules.

Flush iptables

 # iptables -F

Apply NFT rules from nft-rules.txt file

 # nft -f nft-rules.txt

We can now have a look at the list of active NFT rules:

```
# nft list ruleset
```

Configure nftables Rules to Apply upon Reboot

Assuming everything works as expected, we can now move the nfs-rules.txt file into default location that will be used by NFT upon reboot:

```
# mv nft-rules.txt /etc/sysconfig/nftables.conf
```

Make sure it belongs to root and has correct permissions (it’s not a script so needs no execution bits):

```
root@s1:~ # ls -lad /etc/sysconfig/nftables.conf
-rw-------.  1  root  root  5227  Mar  12  01:48
/etc/sysconfig/nftables.conf
```

See Also

- Migrate to nftables
- Using nftables
- keep iptables after reboot
- Book review: iptables pocket reference
- Protect SSH with fail2ban

- Unix Tutorial Projects
- Ansible: getting started

How To Check SSL Connection with OpenSSL

OpenSSL

I'm tidying up Centralised RSyslog setup on the newly reinstalled becky Raspberry Pi system. One of the tasks at hand was to configure TLS based encrypted log shipping from my dedicated servers to home RSyslog server, this post shows the command and technique I use.

How To Check SSL Connection

What we do is run openssl command with the s_client option and specify remote server we're testing connection to. It can be an HTTPS connection (port 443) to a website (will do a post

about it some other time), but in my case I’m connecting to home office server becky.ts.fm with port 6514 (TLS encrypted port for Syslog):

```
root@s2:/ # openssl s_client -connect becky.ts.fm:6514
 CONNECTED(00000003)
 depth=0 CN = becky.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
 verify error:num=20:unable to get local issuer certificate
 verify return:1
 depth=0 CN = becky.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
 verify error:num=21:unable to verify the first certificate
 verify return:1
 Certificate chain
  0 s:/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
    i:/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 Server certificate
 -----BEGIN CERTIFICATE-----
MIIEJDCCAoygAwIBAgIUJGqZcuyXa7ekrK+U8yfB2Cu54FYwDQYJKoZIhvcNAQEL
jMNHiZ0zdzolHWzkV6iKc20MxOv3ftQ1TsE7vg+/Z2fTSv2f0uirPZUPegSzwr75
 ABRIDGED
9n7UHknn7/mV+lLcloOA8oyXB5zeVf+lxLufVRyhEIpLFVtRiu0Go6PW0gjwMoPM
QB/0E6WgtSDMf43f9qzSdtKNgHFw1MpxVQdULSabnI6n0gpfuUIvKDBmBazgh6lR
 RtZqUqzO9pE=
 -----END CERTIFICATE-----
 subject=/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 issuer=/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 Acceptable client certificate CA names
 /CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 Client Certificate Types: RSA sign, DSA sign, ECDSA sign
        Requested        Signature        Algorithms:
RSA+SHA256:0x09+0x08:0x04+0x08:ECDSA+SHA256:0x07+0x08:
RSA+SHA384:0x0A+0x08:0x05+0x08:ECDSA+SHA384:RSA+SHA512:
0x0B+0x08:0x06+0x08:ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
    Shared    Requested    Signature    Algorithms:
```

```
RSA+SHA256:ECDSA+SHA256:RSA+SHA384:ECDSA+SHA384:RSA+SHA512:
ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
 Peer signing digest: SHA512
 Server Temp Key: ECDH, P-256, 256 bits
 SSL handshake has read 1704 bytes and written 427 bytes
 New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
 Server public key is 2048 bit
 Secure Renegotiation IS supported
 Compression: NONE
 Expansion: NONE
 No ALPN negotiated
 SSL-Session:
     Protocol  : TLSv1.2
     Cipher    : ECDHE-RSA-AES256-GCM-SHA384
                                                 Session-ID:
224B0D3C5183426D7DDAD5A5FB361BC9C5175EC9CB0AA6A2F396DAAEE71780
80
     Session-ID-ctx:
                                                 Master-Key:
6BE67A8AD4E22029DE1B3D0DE1F4351FD0488AB1D8ABC7E25187
     Key-Arg   : None
     Krb5 Principal: None
     PSK identity: None
     PSK identity hint: None
     Start Time: 1583790681
     Timeout   : 300 (sec)
      Verify return code: 21 (unable to verify the first
certificate)
```

As you can see, ther is a problem. I’m checking the last line in the output, which should return code **ok**, but it tells me that my s2 server can’t verify the first (only) certificate presented by my RSyslog server becky.ts.fm.

From experience, I konw that’s because s2 somehow needs to acknowledge the certificate authority (CA) that issued certificate for becky.ts.fm.

How Successful TLS Connection Looks in OpenSSL

So if I specify this CA cert as a command line option, openssl will establish TLS connection and confirm code **ok**:

Now let’s specify the CA certificate I used for RSyslog, the connection and certificates verification should work just fine now:

```
root@s2:/ # openssl s_client -CAfile /etc/rsyslog.d/ca.crt -connect becky.ts.fm:6514
 CONNECTED(00000003)
 depth=1 CN = syslog.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
 verify return:1
 depth=0 CN = becky.ts.fm, O = Tech Stack Solutions, L = Dublin, C = IE
 verify return:1
 Certificate chain
  0 s:/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
    i:/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 Server certificate
 -----BEGIN CERTIFICATE-----
MIIEJDCCAoygAwIBAgIUJGqZcuyXa7ekrK+U8yfB2Cu54FYwDQYJKoZIhvcNAQEL
BQAwVDEVMBMGA1UEAxMMc3lzbG9nLnRzLmZtMR0wGwYDVQQKExRUZWNoIFN0YWNr
 ABRIDGED
jMNHiZ0zdzolHWzkV6iKc20MxOv3ftQ1TsE7vg+/Z2fTSv2f0uirPZUPegSzwr75
9n7UHknn7/mV+lLcloOA8oyXB5zeVf+lxLufVRyhEIpLFVtRiu0Go6PW0gjwMoPM
QB/0E6WgtSDMf43f9qzSdtKNgHFw1MpxVQdULSabnI6n0gpfuUIvKDBmBazgh6lR
 RtZqUqzO9pE=
```

```
 -----END CERTIFICATE-----
 subject=/CN=becky.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 issuer=/CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 Acceptable client certificate CA names
 /CN=syslog.ts.fm/O=Tech Stack Solutions/L=Dublin/C=IE
 Client Certificate Types: RSA sign, DSA sign, ECDSA sign
         Requested         Signature         Algorithms:
RSA+SHA256:0x09+0x08:0x04+0x08:ECDSA+SHA256:0x07+0x08:RSA+SHA3
84:
0x0A+0x08:0x05+0x08:ECDSA+SHA384:RSA+SHA512:0x0B+0x08:
0x06+0x08:ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
     Shared     Requested     Signature     Algorithms:
RSA+SHA256:ECDSA+SHA256:RSA+SHA384:ECDSA+SHA384:RSA+SHA512:
ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
 Peer signing digest: SHA512
 Server Temp Key: ECDH, P-256, 256 bits
 SSL handshake has read 1704 bytes and written 427 bytes
 New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
 Server public key is 2048 bit
 Secure Renegotiation IS supported
 Compression: NONE
 Expansion: NONE
 No ALPN negotiated
 SSL-Session:
     Protocol  : TLSv1.2
     Cipher    : ECDHE-RSA-AES256-GCM-SHA384
                                                 Session-ID:
C6797515EEA312D7A9EC6685F895AE004798550FF70619E85F24AB5ACF80F0
A9
     Session-ID-ctx:
                                                 Master-Key:
4B84DF3CFE9697EEC634DC271B2A490D94B7A7AB1CA218F016B1ED141FA147
9C
     Key-Arg   : None
     Krb5 Principal: None
     PSK identity: None
     PSK identity hint: None
     Start Time: 1583790782
     Timeout   : 300 (sec)
     Verify return code: 0 (ok)
 ^C
```

That’s it – this means secure connection establishes successfully, so I can enjoy secure log shipping from s2 to becky.ts.fm.

See Also

- Centralised RSyslog Server
- RSyslog with grc

Ban Specific IP Manually with fail2ban

fail2ban

Now that I’m monitoring my logs using cetralised RSyslog, I regularly notice SSH attacks right when and as they happen. When it becomes obvious that someone’s trying to brute-force SSH, I don’t always wait to let fail2ban fix the issue – sometimes I ban the offending IP myself.

How To Ban Specific IP with fail2ban

Assuming a standard install, we’ll use the **fail2ban-client command** to notify **sshd** jail module to ban a specific IP.

Here’s how it works:

```
root@s1:/etc/fail2ban # fail2ban-client -vvv set sshd banip 202.70.66.228
30 7F0B121F6640 fail2ban.configreader      INFO   Loading configs for fail2ban under /etc/fail2ban
30 7F0B121F6640 fail2ban.configreader      DEBUG Reading configs for fail2ban under /etc/fail2ban
31 7F0B121F6640 fail2ban.configreader    DEBUG Reading config files: /etc/fail2ban/fail2ban.conf
31 7F0B121F6640 fail2ban.configparserinc  INFO    Loading files: ['/etc/fail2ban/fail2ban.conf']
31 7F0B121F6640 fail2ban.configparserinc  TRACE     Reading file: /etc/fail2ban/fail2ban.conf
31 7F0B121F6640 fail2ban.configparserinc  INFO    Loading files: ['/etc/fail2ban/fail2ban.conf']
31 7F0B121F6640 fail2ban.configparserinc  TRACE     Shared file: /etc/fail2ban/fail2ban.conf
32 7F0B121F6640 fail2ban                  INFO  Using socket file /var/run/fail2ban/fail2ban.sock
32 7F0B121F6640 fail2ban                  INFO  Using pid file /var/run/fail2ban/fail2ban.pid, [INFO] logging to SYSLOG
32 7F0B121F6640 fail2ban                  HEAVY CMD: ['set', 'sshd', 'banip', '202.70.66.228']
48 7F0B121F6640 fail2ban                HEAVY OK : 1
48 7F0B121F6640 fail2ban.beautifier       HEAVY Beautify 1 with ['set', 'sshd', 'banip', '202.70.66.228']
1
48 7F0B121F6640 fail2ban                DEBUG Exit with code 0
```

Once you become comfortable, you can omit the -vvv option and skip all this verbose output:

```
root@s1:/etc/fail2ban # fail2ban-client set sshd banip 202.70.66.229
1
```

That’s it for today! Have fun!

See Also

- fail2ban how-to
- Log fail2ban to syslog
- Centralised RSyslog
- RSyslog file by hostname
- SSH server
- SSH reference

---

git push Asks for Username and Password

I’ve been refreshing my gleb.reys.net website recently and experienced a weird error: pushing latest changes to GitHub resulted in my username and password prompted. Figured I should write down what the issue was and how easy it was to fix it.

git Repo Asks for Username/Password

```
greys@mcfly:~/proj/gleb.reys.net $ git push
Username for ‘https://github.com’: greys
Password for ‘https://greys@github.com’:
remote: Invalid username or password.
fatal: Authentication failed for
‘https://github.com/greys/greys.github.io/’
greys@mcfly:~/proj/gleb.reys.net $ git remote
origin
greys@mcfly:~/proj/gleb.reys.net $ git remote show
origin
greys@mcfly:~/proj/gleb.reys.net $ git remote -v
origin https://github.com/greys/greys.github.io (fetch)
origin https://github.com/greys/greys.github.io (push)
```

At first I couldn’t see it

Update The Origin in git Repository

```
greys@mcfly:~/proj/gleb.reys.net $ git remote rm origin
greys@mcfly:~/proj/gleb.reys.net $ git remote -v
```

```
greys@mcfly:~/proj/gleb.reys.net $ git remote add origin
git@github.com:greys/greys.github.io.git
greys@mcfly:~/proj/gleb.reys.net $ git remote -v
origin git@github.com:greys/greys.github.io.git (fetch)
origin git@github.com:greys/greys.github.io.git (push)
```

Push (with set-upstream)

```
greys@mcfly:~/proj/gleb.reys.net $ git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use
```

```
git push --set-upstream origin master
```

```
greys@mcfly:~/proj/gleb.reys.net $ git push –set-upstream
origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 568 bytes | 568.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local
objects.
To github.com:greys/greys.github.io.git
018a8c0..e4f79d4 master -> master
Branch ‘master’ set up to track remote branch ‘master’ from
‘origin’.
```

That’s it! I’m using Netlify for automatic build and hosting of my Jekyll website, so a minute or two after the git push shown above the website got refreshed to the latest version. Nice!

See Also

- gleb.reys.net – my technical website and latest CV
- GlebReys.com – random notes on gadgets and technology
- Project: Static websites with Jekyll
- **Unix Tutorial**
- Free private repos on GitHub

NEVER edit /etc/sudoers Directly

But if you do have to edit **/etc/sudoers**, at least follow this advice to avoid locking yourself out.

How to Edit SUDOERS file Correctly

The proper way of updating **sudo** configuration is to use **visudo command**:

- it creates a temporary copy of the **/etc/sudoers** file and only commits changes if they are syntaxically correct
- visudo carries out basic sanity checks
- this approach even prevents multiple simultaneous edits of the /etc/sudoers file

Common Reasons for Editing /etc/sudoers Directly

There are some valid scenarios when using visudo is not easily possible – for instance, when deploying **sudoers** file using script or post-configuration system (although you should really use a specialised plugin, if possible).

Most commonly though, we edit **/etc/sudoers** directly simply because changes are not meant to be complicated – we're fixing a typo or adding a user.

When Editing /etc/sudoers Goes Wrong

Hardly any such scenario is intentional, but it's still useful

to know why people regularly end up locking themselves out of sudo privileges.

Scenario 1: You make a typo in username/privilege

It’s VERY easy to make a typo and end up with wrong username or wrong command added to **/etc/sudoers** file.

This scenario is bad, but maybe not too bad: you could be editing someone else’s privileges so while that other user ends up without sudo access, you yourself still have a valid **sudoers** privilege and can work on fixing the situation.

Equally, some other sysadmin on your system might still have working sudo privilege, so they can fix your access for you.

Scenario 2: You lose connection in a middle of editing sudoers file

Depending on your habits, this may not be too bad. If you were using **visudo**, there’d be no issue at all: you were editing a copy of **/etc/sudoers** and not the actual file – so no changes were made and this means sudo setup is still solid.

If you were editing manually, there may still be a chance sudo config is okay. But if you have the habit of saving your work in progress (invoking save file in your editor), effectively

saving live **/etc/sudoers** config before you truly finish working on it – you might have a problem because broken connection will mean only last saved changes are on your disk – and they may contain broken syntax or incomplete **sudo** privilege definitions.

Scenario 3: You make a typo and add or remove character in /etc/sudoers

Equally dangerous is just accidentally adding an extra character where it's not expected – this means you end up with broken syntax of the sudoers file.

This scenario is really bad – because it means nobody on your system can use **sudo** to become root and fix the problem. You'll probably need some sort of break-glass procedure where root user password is dug up and local login is necessary from server console to manually fix **sudo**.

How To Minimize Risks When Editing /etc/sudoers Directly

Step 1: Open another root session to the same system

Step 2: Edit file from interactive

session

Step 3: Use visudo to check

Just like I explained in a previous post: run visudo -c to confirm all sudoers config files are valid.

That’s all for now. Stay safe editing your SUDO files!

See Also

- what sudo means
- sudo command
- How To use visudo
- visudo tutorial