

Internal Lab – Crypto & Secret Retrieval via Docker

Project ID: Project 4

Assessed by: Muhammad Talha

Environment: Kali Linux VM (VMware), Docker, Local Lab Setup

Date: 24 June, 2025

1. Executive Summary

This assessment focused on decoding and decryption tasks simulating real-world cryptography-related misconfigurations. The test scenarios included decoding Base64 strings, brute-forcing XOR-encrypted content, extracting and signing RSA key data, and retrieving embedded secrets from a Docker container image. Each step replicated common weaknesses developers may introduce during insecure secret handling or insufficient encryption practices.

2. Tools Used

Tool	Purpose
Kali Linux	Main assessment platform
Docker	Run challenge container
OpenSSL	Decryption and key signing
xxd	Hex conversion and binary manipulation
base64	Base64 decoding
Custom Bash Scripts	Brute-force XOR key recovery

3. Tests Performed

3.1 Base64 Decoding

A Base64 string was decoded using the Linux base64 utility. This test simulated weak encoding of credentials or secrets in configuration files.

```
(kali@kali)-[~]
$ echo aGFja2VyOjEyMzQ1Ng== | base64 -d
aGFja2VyOjEyMzQ1Ng==
(hacker:123456)
```

Basic Authentication

Basic authentication is sometimes used by web applications. This uses base64 encoding. Therefore, it is important to at least use Transport Layer Security (TLS or more commonly known as https) to protect others from reading the username password that is sent to the server.

```
$ echo -n "myuser:mypassword" | base64
bX1lc2VyOm15c6Fzc3dvcnQ=
```

The HTTP header will look like:

```
Authorization: Basic bX1lc2VyOm15c6Fzc3dvcnQ=
```

Now suppose you have intercepted the following header:
Authorization: Basic aGFja2VyOjEyMzQ1Ng==

Then what was the username and what was the password:

✓
Congratulations. That was easy, right?

3.2 XOR Cipher Decoding

A Base64-encoded XOR-obfuscated string was decoded by applying a brute-force XOR key search across all 255 possibilities. Successful key identification revealed the hidden password.

```
(kali@kali)-[~]
$ echo Oz4rPj0+LDovP1wsKDAtoW== | base64 -d
; > + > => , : / > , , ( 0 - ;
```

```
(kali@kali)-[~]
$ echo Oz4rPj0+LDovP1wsKDAtoW== | base64 -d > xor.raw
```

```
(kali@kali)-[~]
$ for key in {1..255}; do
  result=$(cat xor.raw | perl -pe "s/(.)/chr(ord($1)^$key)/ge")
  if echo "$result" | grep -qP "[^ -]{6,}$"; then
    echo "Key $key: $result"
  fi
done
```

Key 1: :?*<?>-; ;? --)1, :

Key 2: 9<)<?<.8-<..*2/9

Key 3: 8=(=>9, =/+3.8

Key 4: ?:/:9:(>+((,4)?

Key 5: >;.;8;)?*;))-5(>

Key 6: =8-8;8*<)8**..6+=

Key 7: <9,9:9+= (9++/7*<

Key 8: 36#656\$2'6\$ \$ 8%3

Key 9: 27*747%367%!9\$2

Key 10: 14147460%466*':1

Key 11: 05 565'1\$5''#;60

Key 12: 72'212 6#2 \$<17

Key 13: 636303!7*3 !!%= 6

Key 14: 50%030*4!0**6>#5

Key 15: 4!\$121#5 1##'?'*4

Key 16: +.;.-.<* ?.<<8 =+

Key 17: */:/,/=+>=9!<*

Key 18:),9,/,>(<,>,>?'?)

Key 19: (-8-,-?)<??;#>(<

Key 20: /*?*)*8.;*88<\$9/

Key 21: .+>+(+9/+>9=>8.

Key 22: -(=+(;9(;>6;-

HTML encoding

HTML encoding ensures that text is displayed as is in the browser and not interpreted by the browser as HTML.

UUEncode

The Unix UU-encode has been used to send email attachments.

XOR encoding

Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

✓
Suppose you found the database password encoded as 1or124rPj0+LDovP1wsKDAtoW==

What would be the actual password [databasepassword]

✓
Congratulations.

Key 22: -(+(:;9(:;>6+;

Key 23:)(<*)~8)??';.

Key 24: #636%64"76440(5#

Key 25: "'2'\$'5#6'551)4"

Key 26: !\$1\$'\$6 5\$662+7!

Key 27: %0%6%7!4%773+6

Key 28: "'7"!063"004,1'

Key 29: 6#6# #1'2#115-06

Key 30: % 5 # 2\$1 226.3%

Key 31: \$!4!"!3%0!337/2\$

Key 64: {-k-}-lzo-llhpm[

Key 67: x|h)-joyl]ooksnx

Key 70: }xmx(xj|ixj|nvk)

Key 71: |ylyzyk|hykkowj|

Key 72: svcvuvdrgydd`xes

Key 73: rwbwtwesfweeaydr

Key 74: qtatwtfpetffbzgq

Key 75: pu`uvvgduggc{fp

Key 76: wrgrqr`vcr`dlaw

Key 77: vsfspaawsaae`v

Key 78: upepsptapbbf-cu

Key 81: jozo!o|k-o|}yalj

Key 85: nk-khkyozkyy|exn

Key 86: mh]khkzlyhzz-f{m

Key 88: cfsfeyftbwfttphuc

Key 89: bgrgdgucvguuqitb

Key 90: adqgdv`udvvrjwa

Key 91: `epefewatewskv`

Key 92: gbwbabpfsbpptlqg

Key 93: fcvc`cagrcqumpf

Key 94: e`u`c`rdq`rrvnse

URL encoding

URL encoding is used a lot when sending form data and request parameters to the server. Since spaces are not allowed in a URL, this is then replaced by %20. Similar replacements are made for other characters.

HTML encoding

HTML encoding ensures that text is displayed as-is in the browser and not interpreted by the browser as HTML.

UUEncode

The Unix 2-Unix encoding has been used to send email attachments.

XOR encoding

Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

✓

Suppose you found the database password encoded as {xor}Qe4rPp+L.DovPwekKDAtCw+...

What would be the actual password [databasepassword]

[post the answer]

Congratulations

Key 94: e`u`c`rdq`rrvnse

Key 95: databasepassword

Key 96: [^K^]^LZO^LLHPM[

Key 97: Z_J_M[N_MMIQLZ

Key 98: Y\I_NXM\NNJROY

Key 99: X]H]^JOYL]OOKSNX

Key 100: ^ZOZYZH^KZHHLTI

Key 101: ^[N[X[I_J[IIMUH^

Key 102:]XMX[XJ\IXJJNVK]

Key 103: \YLYZYK|HYKKOWJ\

Key 104: SVCVUVDRGVDD@XES

Key 105: RWBWTWESFWEEAYDR

Key 106: QTATWTFPETFfBZGQ

Key 107: PU@UVUGQDUGGC[FP

Key 108: WRGRQR@VCR@D\AW

Key 109: VSFSPAWSAAE]@V

Key 110: UPEPSPTAPBBF^CU

Key 111: TQDQRQC@QCCG_BT

Key 112: KN[NMN_J_N\X@]K

Key 113: JOZOLO]K^O|}YA\J

Key 114: ILYLOL^H]L^^ZB_I

Key 115: HMXNMN_I\M__[C^H

Key 116: OJ_JI]XN[JXX\DY0

Key 117: NK^KHKYOZKYY]EXN

Key 118: MH]HHKZLYHZZ^F[M

Key 119: LI\I]I[MXI[[_6ZL

Key 120: CFSFEFTBWFTTPhUC

Key 121: BGRGDGUCVGUUQITB

Key 122: ADQGDV@UDVVRJWA

Key 123: @EPEFEWATEWWSKV@

Key 124: GBWBABPFSBPPTLQG

URL encoding

URL encoding is used a lot when sending form data and request parameters to the server. Since spaces are not allowed in a URL, this is then replaced by %20. Similar replacements are made for other characters.

HTML encoding

HTML encoding ensures that text is displayed as-is in the browser and not interpreted by the browser as HTML.

UUEncode

The Unix 2-Unix encoding has been used to send email attachments.

XOR encoding

Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

✓

Suppose you found the database password encoded as {xor}Qe4rPp+L.DovPwekKDAtCw+...

What would be the actual password [databasepassword]

[post the answer]

Congratulations

Key 124: GBWBABPFSBPPTLQG

Key 125: FCVC@CQGRCCQUMPF

Key 126: E@U@C@RDQ@RRVNSE

Key 127: DATABASEPASSWORD

URL encoding

URL encoding is used a lot when sending form data and request parameters to the server. Since spaces are not allowed in a URL, this is then replaced by %20. Similar replacements are made for other characters.

HTML encoding

HTML encoding ensures that text is displayed as-is in the browser and not interpreted by the browser as HTML.

UUEncode

The Unix 2-Unix encoding has been used to send email attachments.

XOR encoding

Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

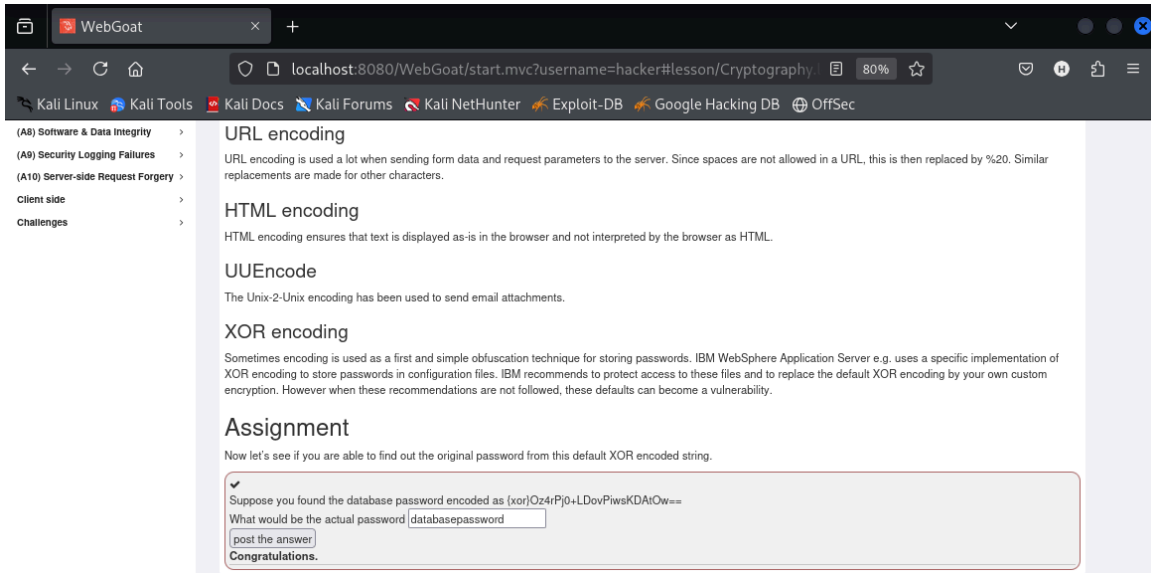
✓

Suppose you found the database password encoded as {xor}Qe4rPp+L.DovPwekKDAtCw+...

What would be the actual password [databasepassword]

[post the answer]

Congratulations



3.3 RSA Key – Modulus Extraction & Signature

A PKCS#8 formatted private RSA key was converted to PKCS#1 format. The modulus was extracted and converted to a hex string. The resulting binary was then signed using SHA-256 and OpenSSL's command-line tools.

```
└─$ openssl rsa -in priv.key -pubout > pub.pub
writing RSA key
```

```
(kali@kali)-[~/Desktop]
└─$ openssl rsa -in pub.pub -pubin -modulus -noout
Modulus=A2246532307E90CAA8C72F3349DEC184925B728E9AF1D049618397D7EF8FB529C54561F3BDB7001F02721A6CC1D2471A7208AB6232B314BCED613AB70D9B6682A23E0
5E19B6C7CEDDF38B6433582CA7CEA2DBA5BED67B4E12451821551A62A57834EA3525999A05CE5331236320EBE09B8A1E5154A9A10100F1794A02B65AD708BD504F30ADAAE62FE
196BA2A8386CB6A8E2CE6B5ADBFA004ED15886F9E7376F9E8D9EDFE17A7FFA09B24BD0098FCC45ABA90AA5CA22B5B54E8CBE13E65E9E6C089787092986DB587106112874A4F7
486D49149F44E3052FDE64D6D1ED93A4840CABA816ACB775EE1D99E0C5566D7FB37561EAFCEBF500730C43C66A9B7AC9"
```

```
(kali@kali)-[~/Desktop]
└─$ echo -n "A2246532307E90CAA8C72F3349DEC184925B728E9AF1D049618397D7EF8FB529C54561F3BDB7001F02721A6CC1D2471A7208AB6232B314BCED613AB70D9B6682
A23E05E19B6C7CEDDF38B6433582CA7CEA2DBA5BED67B4E12451821551A62A57834EA3525999A05CE5331236320EBE09B8A1E5154A9A10100F1794A02B65AD708BD504F30ADAA
E62FE196BA2A8386CB6A8E2CE6B5ADBFA004ED15886F9E7376F9E8D9EDFE17A7FFA09B24BD0098FCC45ABA90AA5CA22B5B54E8CBE13E65E9E6C089787092986DB58710611287
4A4F7486D49149F44E3052FDE64D6D1ED93A4840CABA816ACB775EE1D99E0C5566D7FB37561EAFCEBF500730C43C66A9B7AC9" | openssl dgst -sign priv.key -sha256
-out signsha.256
dgst: Unknown option or message digest: sha256-out
dgst: Use -help for summary.
4007C8C28F7F0000:error:0308010C:digital envelope routines:inner_evp_generic_fetch:unsupported:../crypto/evp/evp_fetch.c:375:Global default li
brary context, Algorithm (sha256-out : 0), Properties (<null>)
```

```
(kali@kali)-[~/Desktop]
└─$ openssl enc -base64 -in signsha.256 -out signsha25sha25
```

```
(kali@kali)-[~/Desktop]
└─$ cat signsha25sha25
ZX+xAcyxamUPcaP9G2dyo3eUYAVvyKIbLRZqWgnK6pLg50fMTcYTG+Zek70GUzk
j/uKjLQvtzV57mvpQUn6NuCO3MaJQHZ1MkEW4cq/W/WaZSW1/Mb3CZP++PMchGt5
XWcuZNRwylf1u5eFrJFnm1GumazwWrvQ9c/kg0tdQzX1ryBXf01CwFubMMGyaJAM
0eu0skMc7cA92GqT9DdTVEwtWDrQcRjWuve/DIMKN+bDURApu3tt021n22GDxLVB
56Wc6g07YGEFes0+gDF5WDtavgXd/73L16QZgfh91Dfoes1J2Wf1gZLmHZxqdD1
flzgcf0tRLZg5PgW1JdPg=
```

WebGoat

localhost:8080/WebGoat/start.mvc?username=hacker#lesson/Cryptography

80%

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Sending emails is not very difficult. You have to fill in some data and send it to a server that forwards it, and eventually it will end up at its destination. However, it is possible to send emails with a FROM field that is not your own email address. In order to guarantee to your receiver that you really sent this email, you can sign your email. A trusted third party will check your identity and issue an email signing certificate. You install the private key in your email application and configure it to sign emails that you send out. The certificate is issued on a specific email address and all others that receive this email will see an indication that the sender is verified, because their tools will verify the signature using the public certificate that was issued by the trusted third party.

PDF or Word or other signatures

Adobe PDF documents and Microsoft Word documents are also examples of things that support signing. The signature is also inside the same document as the data so there is some description on what is part of the data and what is part of the metadata. Governments usually send official documents with a PDF that contains a certificate.

Assignment

Here is a simple assignment. A private RSA key is sent to you. Determine the modulus of the RSA key as a hex string, and calculate a signature for that hex string using the key. The exercise requires some experience with OpenSSL. You can search on the Internet for useful commands and/or use the HINTS button to get some tips.

Now suppose you have the following private key:

```
-----BEGIN PRIVATE KEY-----
MIIEuWIBADANBgkqhkiG9w0BAQFAASCBKUwggShAgEAAoIBAQC1JGUyMH6QyqjHLZnJ3s6Ek1tyjprx3Ulhg5fX74+1KcVFF09twAFAnIabMHSRxpC6t1MfMuV01h0rcNm2a
-----END PRIVATE KEY-----
```

Then what was the modulus of the public key and now provide a signature for us based on that modulus

Congratulations. You found it!

3.4 Docker Secret Retrieval & AES Decryption

The container webgoat/assignments:findthesecret was run locally. Root access to the container was gained using 'docker exec -u 0 -it'. The secret located in /root was extracted and used with OpenSSL to decrypt the supplied encrypted message.

```
(kali@kali)-[~/Desktop]
└─$ sudo docker run -d --name findsecret webgoat/assignments:findthesecret
3f406b7201118312c02f17b69b2b2f1087a74ec83b1f914c2e034aac095c6775

(kali@kali)-[~/Desktop]
└─$ sudo docker exec -u 0 -it findsecret /bin/sh
# ls/root
/bin/sh: 1: ls/root: not found
# ls
bin boot dev docker-java-home etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
# cd /root
# ls
default_secret
# cat default_secret
ThisIsMySecretPassw0rdF0rY0u
# ^C
#
```

```
# echo "U2FsdGVkX199jgh5oANeIFdtCxIEvdEvcilI+v+5loE+VCuy6I10b+5byb5DXp32RpmT02Ek1pf55ctQN+DHbwCPiVRFFqamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile /root/default_secret
Leaving passwords in docker images is not so secure# ^C
#
```

Assignment

In this exercise you need to retrieve a secret that has accidentally been left inside a docker container image. With this secret, you can decrypt the following message: **U2FsdGVkX199jgh5oANeIFdtCxIEvdEvcilI+v+5loE+VCuy6I10b+5byb5DXp32RpmT02Ek1pf55ctQN+DHbwCPiVRFFqamDmbHBUpD7as=**. You can decrypt the message by logging in to the running container (docker exec ...) and getting access to the password file located in /root. Then use the openssl command inside the container (for portability issues in openssl on Windows/Mac/Linux) You can find the secret in the following docker image, which you can start as:

```
docker run -d webgoat/assignments:findthesecret
```

```
echo "U2FsdGVkX199jgh5oANeIFdtCxIEvdEvcilI+v+5loE+VCuy6I10b+5byb5DXp32RpmT02Ek1pf55ctQN+DHbwCPiVRFFqamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile ....
```

What is the unencrypted message
(er images is not so secure)
and what is the name of the file that stored the password
default_secret

Congratulations, you did it!

3.5 Unsalted Hash Identification

Included screenshot evidence of handling an unsalted hash from the project challenge. Details omitted in this writeup but confirmed as part of exploratory steps.

WebGoat

CrackStation - Online Pa...

https://crackstation.net

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

5EBE2294ECD0E0F08EAB7690D2A6EE69

I'm not a robot

reCAPTCHA

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
5EBE2294ECD0E0F08EAB7690D2A6EE69	md5	secret

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA12020C923ADC6C92

I'm not a robot

reCAPTCHA

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA12020C923ADC6C92	sha256	123456

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

WebGoat

CrackStation - Online Pa...

localhost:8080/WebGoat/start.mvc?username=hacker#lesson/Cryptography

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

(A7) Identity & Auth Failure

(A8) Software & Data Integrity

(A9) Security Logging Failures

(A10) Server-side Request Forgery

Client side

Challenges

Hashing is a type of cryptography which is mostly used to detect if the original data has been changed. A hash is generated from the original data. It is based on irreversible cryptographic techniques. If the original data is changed by even one byte, the resulting hash is also different.

So in a way it looks like a secure technique. However, it is NOT and even NEVER a good solution when using it for passwords. The problem here is that you can generate passwords from dictionaries and calculate all kinds of variants from these passwords. For each password you can calculate a hash. This can all be stored in large databases. So whenever you find a hash that could be a password, you just look up the hash in the database and find out the password.

Some hashing algorithms should no longer be used: MD5, SHA-1 For these hashes it is possible to change the payload in such a way that it still results in the same hash. This takes a lot of computing power, but is still a feasible option.

Salted Hashes

Plain passwords should obviously not be stored in a database. And the same goes for plain hashes. The OWASP Password Storage Cheat Sheet explains what should be used when password related information needs to be stored securely.

Assignment

Now let's see if you can find what passwords matches which plain (unsalted) hashes.

Which password belongs to this hash:
5EBE2294ECD0E0F08EAB7690D2A6EE69
secret

Which password belongs to this hash:
8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA12020C923ADC6C92
123456

post the answer

Congratulations. You found it!

4. Conclusion

This project validated multiple critical steps in identifying and exploiting weak crypto practices. The assessment simulated real-world flaws including insecure encoding, simple XOR ciphers, improper key formats, and embedded container secrets. Best practices were identified for each flaw, such as key rotation, environmental secret separation, and stronger input/output handling.

