Phòng thí nghiệm An toàn Thông tin InsecLab
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO THỰC HÀNH

Môn học: Nhập môn mạng

Tên chủ đề: Lab 5

GVHD: Tô Trọng Nghĩa

Nhóm: Mệt mỏi

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)
Lớp: ATTT2021

STT	Họ và tên	MSSV	Email
1	Nguyễn Thị Minh Châu	21520645	21520645@gm.uit.edu.vn
2	Lưu Thị Huỳnh Như	21521242	21521242@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

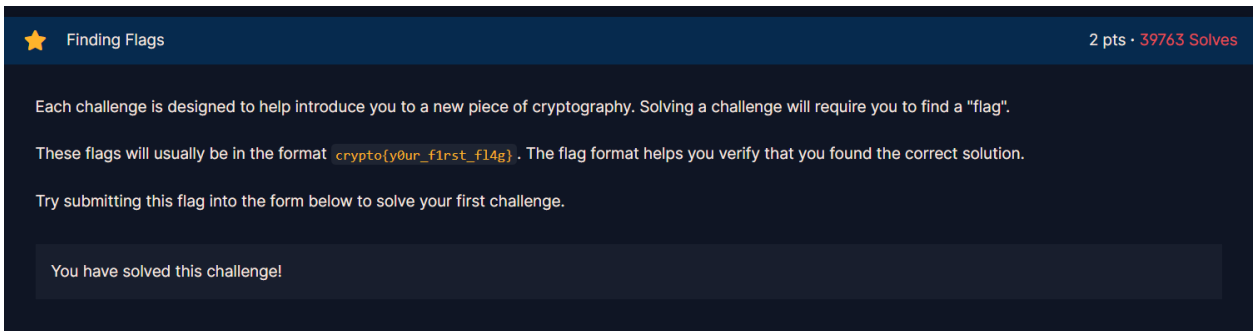
STT	Nội dung	Tình trạng	Trang
1	Yêu cầu 1	100%	2 – 12
2	Yêu cầu 2	100%	12 – 18
Điểm tự đánh giá			**10/10**

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

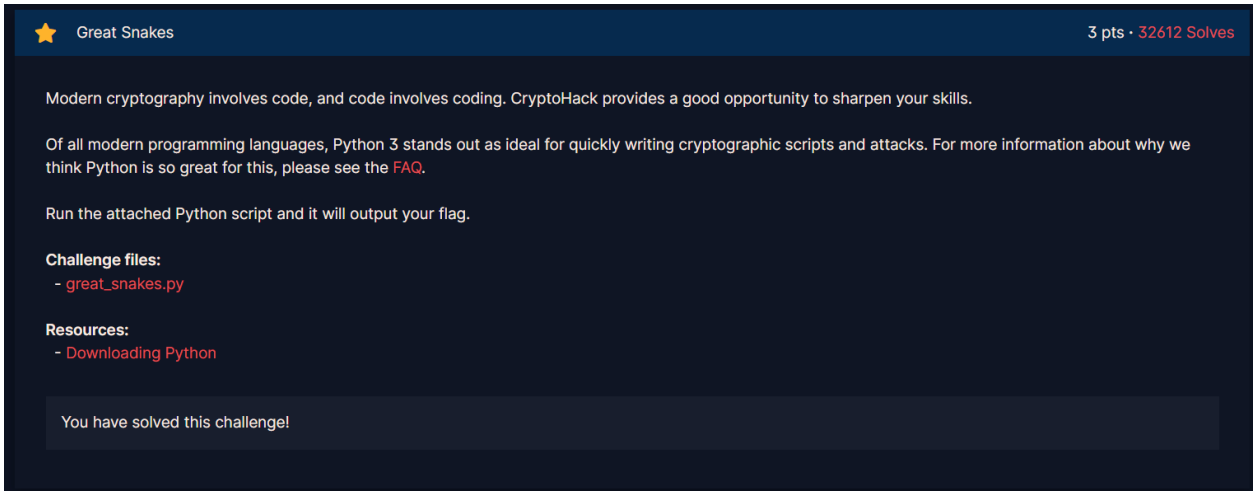
Task 1: 10 bài đầu tiên

1.



Bài này là bài mở đầu nên chỉ cần nhập flag như trong hướng dẫn là được

2.



- Đầu tiên ta download file great_snakes.py về. Sau đó ta chạy file thì sẽ ra được flag

great_snakes_35381fca29d68d8f3f25c9fa0a9026fb.py × telnetlib_example_dbc6ff5dc4dcfac568d7978a801d3ead.py

C: > Users > admin > Downloads > great_snakes_35381fca29d68d8f3f25c9fa0a9026fb.py > ...

```python
#!/usr/bin/env python3

import sys
# import this

if sys.version_info.major == 2:
    print("You are running Python 2, which is no longer supported. Please update to Python 3

ords = [81, 64, 75, 66, 70, 93, 73, 72, 1, 92, 109, 2, 84, 109, 66, 75, 70, 90, 2, 92, 79]

print("Here is your flag:")
print("".join(chr(o ^ 0x32) for o in ords))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PowerShell 7.3.4
PS C:\Users\admin\Downloads>  & 'C:\Users\admin\AppData\Local\Programs\Python\Python310\python.exe' 'c:\U
.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '55443' '--' 'C:\Users\admin\Downloads\
Here is your flag:
crypto{z3n_0f_pyth0n}
PS C:\Users\admin\Downloads>
```

3.

Network Attacks 5 pts · 19283 Solves

Several of the challenges are dynamic and require you to talk to our challenge servers over the network. This allows you to perform man-in-the-middle attacks on people trying to communicate, or directly attack a vulnerable service. To keep things consistent, our interactive servers always send and receive JSON objects.

Python makes such network communication easy with the `telnetlib` module. Conveniently, it's part of Python's standard library, so let's use it for now.

For this challenge, connect to `socket.cryptohack.org` on port `11112`. Send a JSON object with the key `buy` and value `flag`.

> The example script below contains the beginnings of a solution for you to modify, and you can reuse it for later challenges.

Connect at `nc socket.cryptohack.org 11112`

Challenge files:
- telnetlib_example.py

You have solved this challenge!

- Đầu tiên ta tải file telnetlib_example.py về. Ta thấy trong đề bài có hướng dẫn gửi 1 JSON object với khóa “buy” và value “flag” nên ta thay value thành “flag” rồi chạy để nhận được flag

C: > Users > admin > Downloads > telnetlib_example_dbc6ff5dc4dcfac56

```python
print(readline())
print(readline())
print(readline())
print(readline())


request = {
    "buy": "flag"
}
json_send(request)

response = json_recv()

print(response)
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
crypto{z3n_0f_pyth0n}
PS C:\Users\admin\Downloads>  c:; cd 'c:\Users\admin\Download
\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\
dbc6ff5dc4dcfac568d7978a801d3ead.py'
b"Welcome to netcat's flag shop!\n"
b'What would you like to buy?\n'
b"I only speak JSON, I hope that's ok.\n"
b'\n'
{'flag': 'crypto{sh0pp1ng_f0r_fl4g5}'}
PS C:\Users\admin\Downloads>
```

4.

Privacy-Enhanced Mail? 25 pts · 4620 Solves · 22 Solutions

As we've seen in the encoding section, cryptography involves dealing with data in a wide variety of formats: big integers, raw bytes, hex strings and more. A few structured formats have been standardised to help send and receive cryptographic data. It helps to be able to recognise and manipulate these common data formats.

PEM is a popular format for sending keys, certificates, and other cryptographic material. It looks like:

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKC... (a whole bunch of base64)
-----END RSA PUBLIC KEY-----
```

It wraps base64-encoded data by a one-line header and footer to indicate how to parse the data within. Perhaps unexpectedly, it's important for there to be the correct number of hyphens in the header and footer, otherwise cryptographic tools won't be able to recognise the file.

The data that gets base64-encoded is DER-encoded ASN.1 values. Confused? The resources linked below have more information about what these acronyms mean but the complexity is there for historical reasons and going too deep into the details may drive you insane.

Extract the private key d as a decimal integer from this PEM-formatted RSA key.

> There are two main approaches for solving this challenge. The data in the certificate can be read with the openssl command line tool, or in Python using PyCryptodome. We recommend using PyCryptodome: first import the RSA module with `from Crypto.PublicKey import RSA` and you can read the

- Bài này sử dụng kiến thức đã được học ở lab 4. Ta mở file .pem ra xem

```
zeri@zeri: ~
zeri@zeri:~$ openssl asn1parse -in privacy_enhanced_mail_1f696c053d76a78c2c531bb013a92d4a.pem

    0:d=0  hl=4 l=1187 cons: SEQUENCE
    4:d=1  hl=2 l=   1 prim: INTEGER           :00
    7:d=1  hl=4 l= 257 prim: INTEGER           :CEF283B7E10EF80EA81352C8B52BA791627CBCDE9381C
BBC0A4D3BEE3A060DF1B636DC43E2FC90C464D09E0AFA8C4289B95CF6308C0371D5ED14B205F88497F477C02D0DF2
89E74D4BCD27FD347504D7094A5CC8BAA2E617175D9A399F52F1C5B852EFC605D181ADE6837AFBA254D6FB57F1392
D1955E0C9A8509A39146FA0060E80B82E776FC4D1A69A262F5C37B0A399D27B4379BAAF21AE0B0A84BD9D4A815596
6BB7CA705A299E5FDF72C49E7306C35798567E6BF1880509CB598D48FC247F9621190FBCDE0F4DE7CF11A4B2CC5E6
82DE8A8A6B625A726CAAD0CF465638063C5DA6E57743251DA36CA6AA7AAE66713FBE553F867EBFB3CF9CB5D4EA7D2
0B
  268:d=1  hl=2 l=   3 prim: INTEGER           :010001
  273:d=1  hl=4 l= 256 prim: INTEGER           :7C3B1D534F299B43C1260876303C0A95BE17BF91A5DF2
F1CACDA7C75A0236E4F81E1210D27C0126FB34D80F27A41A4D7E48CA7C5B0E78878B19FD0D6C0BF6830FB8A4401B1
6D938AD54C4D0B356862056CB0554EB2AB8390AD1825B31DAFBF2FC05D194F38C2F22420D3210ADA0230242640CAE
005EB85CBC8DCCA1825EA7496D9B170C5CBFE354FE19A63102B82F38D5D7C251735208B83A54240927F899848C16A
5FE70CE950DAFF7BF9F4B71B598101A52048CD30C16CB994330B10592D2C95D4D0E579F5287FF74A88268D0389698
C8F7B9AE813F39246893D02661CF08D9CBCEC9F722CF76D0E96F1E17737E29ECE8676767CB6E1DF0DBD2D731ED848
B1
  533:d=1  hl=3 l= 129 prim: INTEGER           :EDFB4715EBA93BC4C2CBE712C8081027CC86A8D28D2C7
8C9720E6DE6F68031E0E34FFA5EEF0FD1D085AE49C0A800388BF7EE98A94A77E1181E603924B3B3BB9DCE97B80062
F2830C8F11983DFADD55F1F9CE5362992E14C25F776EF7DACEEB719E1CF9F2F62F4BA6D003DE4D427EEB5A4D98156
44FCE1255931BDF2BA37FC7A7
  665:d=1  hl=3 l= 129 prim: INTEGER           :DE9DB5C35D2562F1CD3622342818C7BEBA0333207EDFD
BC3F2648E6D1410B8914974A5AE32AFA8E4EAE40B42ADA5867E1B0E332FD0D0A2C8A9DE1ADBEDBD81F9BAB4C8FEC8
CE3E660155E2CD04C6925B93FD88AFBE05DCC552A836E353A931209B23A13E7EB0F8FA919C44AC485CE37D6ADA853
0AB56899C6669D44C5874AEFD
  797:d=1  hl=3 l= 128 prim: INTEGER           :444CDEBCFAD2AA35B15685EE0CFCCB6E30B3E115F4B07
3C614F6F131DD43338D808FBEA2AA67D6E6CAC717A1B455C3E4DFF6595814E84CF0F81ED3A7A5EF8A8422FBC6324E
339DCAE7F0BBC9E60ACA14D58612C67482163126E70731195A53965B33A3C4C84510A8428129B6F0C3AE564F78BB8
2FBA87FF8916CE96303DCB377
  928:d=1  hl=3 l= 129 prim: INTEGER           :D3F4F73E16EEE4E173510A89FC6F73A79E3633B4C9F85
CA7999FB2981AD5BCD5E049A70250123E4E0F73A7610A32A2F668CE4160528283AB694926EBA5D59CEE689D7F0E4F
A5477619E96B73670BA60879C49923335B2393E11A76804584BF58DB3DB665E97C98E30246F67FCEBA5A836C7CB8F
9D8F92136FFAFDDC9FF22C205
 1060:d=1  hl=3 l= 128 prim: INTEGER           :76BC5D830BCC7EB721E87AF55645FFB8CEDDDDE56782E
5304613D0117BB329DF7BDABAC7BB3489AF5B7FAFD00A498EC4F0BCEBCAA138C8124B8F0BF9330A9903504A6F5BF6
8CB620B94B034283B17E4EFC5A328B3D6C730AFB9E1EAD67EB5540246F16F88810691A5DD12204DE1E4DB7237DCE6
677FBBD780E4DDB53F381DFC6
zeri@zeri:~$
```

- Ta có format như bên dưới. Theo đề bài ta cần phải tìm *private key d* nên ta nhận thấy cần phải mã hóa dòng thứ 4 để có thể tìm ra key

[–] PKCS#1 RSA Private Keys as defined in RFC 8017 Appendix 1.2.

ASN.1 structure containing a serialized RSA private key:

```
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

- Tiến hành mã hóa giá trị ở trên ta thu được flag

```
zeri@zeri:~$ python3 -q
>>> en = "7C3B1D534F299B43C1260876303C0A95BE17BF91A5DF2F1CACDA7C75A0236E4F81E1210D27C0126FB34
D80F27A41A4D7E48CA7C5B0E78878B19FD0D6C0BF6830FB8A4401B16D938AD54C4D0B356862056CB0554EB2AB8390
AD1825B31DAFBF2FC05D194F38C2F22420D3210ADA0230242640CAE005EB85CBC8DCCA1825EA7496D9B170C5CBFE3
54FE19A63102B82F38D5D7C251735208B83A54240927F899848C16A5FE70CE950DAFF7BF9F4B71B598101A52048CD
30C16CB994330B10592D2C95D4D0E579F5287FF74A88268D0389698C8F7B9AE813F39246893D02661CF08D9CBCEC9
F722CF76D0E96F1E17737E29ECE8676767CB6E1DF0DBD2D731ED848B1"
>>> de = bytes.fromhex(en)
>>> from Crypto.Util import number
>>> d = number.bytes_to_long(de)
>>> print(d)
15682700288056331364787171045819973654991149949197959929860861228180021707316851924456205543 6
65565810892674190059831330231436970914474774562714945620519144389785158908994181951348846017 4
32506464163564960993784254153395406799101314760033445065193429592512349952020982932218524462 3
41002102063435489318813316464511621736943938440710470694912336237680219746204595128959161800 5
95216366237538296447335375818871952520026993102148328897083547184286493241191505953601668858 9
41129790966909236941127851370202421135897091086763569884760099112291072056970636380417349019 5
797687480547601048387904247089882604439269066737959751046 89
>>>
```

5.

CERTainly not 30 pts · 4137 Solves · 20 Solutions

As mentioned in the previous challenge, PEM is just a nice wrapper above DER encoded ASN.1. In some cases you may come across DER files directly; for instance many Windows utilities prefer to work with DER files by default. However, other tools expect PEM format and have difficulty importing a DER file, so it's good to know how to convert one format to another.

An SSL certificate is a crucial part of the modern web, binding a cryptographic key to details about an organisation. We'll cover more about these and PKI in the TLS category. Presented here is a DER-encoded x509 RSA certificate. Find the modulus of the certificate, giving your answer as a decimal.

Challenge files:
- 2048b-rsa-example-cert.der

You have solved this challenge! View solutions

- Bài này ta cần đổi từ đuôi .der sang đuôi .pem rồi ta đổi từ file chứng nhận sang file chữ kí.

```
zeri@zeri:~$ openssl x509 -in bai5.der -inform der -outform pem -out key.pem
zeri@zeri:~$ openssl x509 -pubkey -in key.pem -out key1.pem
zeri@zeri:~$ cat key1.pen
cat: key1.pen: No such file or directory
zeri@zeri:~$ cat key1.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtM/RXjMp7AvPrnb1/i3I
mcZ4ebkY+AvUurTXngJSBgn0GJNM1HDRQqApE5JzUHf2BImsAyzW8QarrWzA2dWm
q8rNWtJWJlHlSwiKr8wZDyU0kLAqKUEPVfFrk9uds8zc7OvHVRjXQiXeSTUUMpKc
HsZp4zz79Jr4+4vF4Bt+/U8luj/llleaJHlJFyfXiUtqLg2HUdkjPQaFVvhYMQ7u
gZl4aM1uRH7J2oxaexy/JEApSNEDnO/cripd+Pdqx+m8xbBZ9pX8FsvYnO3D/BKQ
k3hadbRWg/r8QYT2ZHk0NRyseoUOc3hyAeckiSWe2n9lvK+HkxmM23UVtuAwxwj4
WQIDAQAB
-----END PUBLIC KEY-----
-----BEGIN CERTIFICATE-----
MIIC2jCCAkMCAg38MA0GCSqGSIb3DQEBBQUAMIGbMQswCQYDVQQGEwJKUDEOMAwG
A1UECBMFVG9reW8xEDAOBgNVBAcTB0NodW8ta3UxETAPBgNVBAoTCEZyYW5rNERE
MRgwFgYDVQQLEw9XZWJDZXJ0IFN1cHBvcnQxGDAWBgNVBAMTD0ZyYW5rNEREIFdl
YiBDQTEjMCEGCSqGSIb3DQEJARYUc3VwcG9ydEBmcmFuazRkZC5jb20wHhcNMTIw
ODIyMDUyNzQxWhcNMTcwODIxMDUyNzQxWjBKMQswCQYDVQQGEwJKUDEOMAwGA1UE
CAwFVG9reW8xETAPBgNVBAoMCEZyYW5rNEREMRgwFgYDVQQDDA93d3cuZXhhbXBs
ZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC0z9FeMynsC8+u
dvX+LciZxnh5uRj4C9S6tNeeAlIGCfQYk0zUcNFCoCkTknNQd/YEiawDLNbxBqut
bMDZ1aarys1a0lYmUeVLCIqvzBkPJTSQsCopQQ9V8WuT252zzNzs68dVGNdCJd5J
NRQykpwexmnjPPv0mvj7i8XgG379TyW6P+WWV5okeUkXJ9eJS2ouDYdR2SM9BoVW
+FgxDu6BmXhozW5EfsnajFp7HL8kQClI0QOc79yuKl3492rH6bzFsFn2lfwWy9ic
7cP8EpCTeFp1tFaD+vxBhPZkeTQ1HKx6hQ5zeHIB5ySJJZ7af2W8r4eTGYzbdRW2
4DDHCPhZAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAQMv+BFvGdMVzkQaQ3/+2noVz
/uAKbzpEL8xTcxYyP3lkOeh4FoxiSWqy5pGFALdPONoDuYFpLhjJSZaEwuvjI/Tr
rGhLV1pRG9frwDFshqD2Vaj4ENBCBh6UpeBop5+285zQ4SI7q4U9oSebUDJiuOx6
+tZ9KynmrbJpTSi0+Ao=
-----END CERTIFICATE-----
zeri@zeri:~$
```

- Ta thấy file key1.pem có cấu trúc giống bên trên, ta cần tìm modulus nên ta sẽ xét đến dòng 19

```
zeri@zeri:~$ openssl asn1parse -i -in key1.pem
    0:d=0  hl=4 l= 290 cons: SEQUENCE
    4:d=1  hl=2 l=  13 cons:  SEQUENCE
    6:d=2  hl=2 l=   9 prim:   OBJECT            :rsaEncryption
   17:d=2  hl=2 l=   0 prim:   NULL
   19:d=1  hl=4 l= 271 prim:  BIT STRING
zeri@zeri:~$ openssl asn1parse -i -in key1.pem -strparse 19
    0:d=0  hl=4 l= 266 cons: SEQUENCE
    4:d=1  hl=4 l= 257 prim:  INTEGER           :B4CFD15E3329EC0BCFAE76F5FE2DC899C67879B918F8
0BD4BAB4D79E02520609F418934CD470D142A0291392735077F60489AC032CD6F106ABAD6CC0D9D5A6ABCACD5AD25
62651E54B088AAFCC190F253490B02A29410F55F16B93DB9DB3CCDCECEBC75518D74225DE49351432929C1EC669E3
3CFBF49AF8FB8BC5E01B7EFD4F25BA3FE596579A2479491727D7894B6A2E0D8751D9233D068556F858310EEE81997
868CD6E447EC9DA8C5A7B1CBF24402948D1039CEFDCAE2A5DF8F76AC7E9BCC5B059F695FC16CBD89CEDC3FC129093
785A75B45683FAFC4184F6647934351CAC7A850E73787201E72489259EDA7F65BCAF8793198CDB7515B6E030C708F
859
  265:d=1  hl=2 l=   3 prim:  INTEGER           :010001
```

- Ta giải mã đoạn hex thì thu được flag như bên dưới

```
zeri@zeri:~$ python3 -q
>>> from Crypto.Util import number
>>> en = "
  File "<stdin>", line 1
    en = "
         ^
SyntaxError: unterminated string literal (detected at line 1)
>>> en = "B4CFD15E3329EC0BCFAE76F5FE2DC899C67879B918F80BD4BAB4D79E02520609F41893
4CD470D142A0291392735077F60489AC032CD6F106ABAD6CC0D9D5A6ABCACD5AD2562651E54B088A
AFCC190F253490B02A29410F55F16B93DB9DB3CCDCECEBC75518D74225DE49351432929C1EC669E3
3CFBF49AF8FB8BC5E01B7EFD4F25BA3FE596579A2479491727D7894B6A2E0D8751D9233D068556F8
58310EEE81997868CD6E447EC9DA8C5A7B1CBF24402948D1039CEFDCAE2A5DF8F76AC7E9BCC5B059
F695FC16CBD89CEDC3FC129093785A75B45683FAFC4184F6647934351CAC7A850E73787201E72489
259EDA7F65BCAF8793198CDB7515B6E030C708F859"
>>> de = bytes.fromhex(en)
>>> s = number.bytes_to_long(de)
>>> print(s)
22825373692019530804306212864609512775374171823993708516509897631547513634635856
37562400373706803454904767799931094183745437882935139830238262965826407877545683
86262075077254940306005168728523061912554929264959655363792718753104573191079360
20730050476235278671528265817571433919561175665096171189758406136453987966255236
96378266606696265467846495007592306032735869135663290860649823175596356738233901
09852226232055869234664058092174266703334100144299051469416522933662129037336300
83016398810887356019977409467374742266276267137547021576874204809506045914964491
06339380049916741647194902199544772241595997978595956949 7
>>> SS
```

6.

SSH Keys 35 pts · 2516 Solves · 14 Solutions

Secure Shell Protocol (SSH) is a network protocol that uses cryptography to establish a secure channel over an insecure network (i.e. the internet). SSH enables developers and system administrators to run commands on servers from the other side of the world, without their password being sniffed or data being stolen. It is therefore critical to the security of the web.

In the old days, system administrators used to logon to their servers using telnet. This works similarly to our interactive challenges that involve connecting to `socket.cryptohack.org` - data is sent to a remote server, which performs actions based on what is sent. There is no transport encryption, so anyone listening in on the network (such as the WiFi access point owner, your ISP, or the NSA) can see all the telnet traffic.

As the internet became increasingly hostile, people realised the need for both authentication and encryption for administrative network traffic. SSH, first released in 1995, achieves these goals and much more, with advanced functionality built into the software like port forwarding, X11 forwarding, and SFTP (Secure File Transfer Protocol). SSH uses a client-server architecture, meaning the server runs SSH as a service daemon which is always online and waiting to receive connections, and the user runs an SSH client to make a connection to it.

Most commonly, SSH is configured to use public-private key pairs for authentication. On the server, a copy of the user's public key is stored. The user's private

```
zeri@zeri:~$ ssh-keygen -f bruce_rsa_6e7ecd53b443a97013397b1a1ea30e14.pub -e -m
PKCS8
-----BEGIN PUBLIC KEY-----
MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEArTy6m2vhhbwx3RVbNVb3
ZOenCqqsOXHaJpbtN+OuulLKBSKpIoPB+ZDbDXn0qWkf4lOxtGSgolkUbgG07Lhz
fgs+dul4UL84CkwZExmF3Rf1nRv+v7pqLt2dPsCb02YLxJnhHJb4rQaz2ZM4QCtT
OcqYDUeKfLHCaZU4Ekm/OApKrpfw4/0ofn8KOrFN0t4/dqnNuwVRgoaUIhsI47re
ApB2rs0AP4CggSIi8s6BXCxB4YzgThBK5760T1giACYQC5MFdq1Gw+INSFmu0CNq
t5wdJ5Z4z5448Gke06R+IMtjUiGDQ3QtT2fK3gWhZxk14M4UNrdETgTW/mQ4B/Bc
vikxvoBGpKbttG0agfOjTen6wyzpGfcd8N9rSbaqqyUwC8uDotzFtFzzutVAU9d9
1TagGzWBhNoMfplwVTns27GOOgv1dn5sQSSSmP0hTbPMDlThysKkR9BiOVbBtWGQ
pV936pPBgyWERGqMqC9xykLdVHv2Vu05T0WMwKCAetgtAgMBAAE=
-----END PUBLIC KEY-----
```

- Bài này cũng tương tự những bài trên. Ta mở file public key rồi conver sang file .pem sau đó làm các bước tương tự để thu được flag

```
zeri@zeri:~$ ssh-keygen -f bruce_rsa_6e7ecd53b443a97013397b1a1ea30e14.pub -e -m pem
-----BEGIN RSA PUBLIC KEY-----
MIIBigKCAYEArTy6m2vhhbwx3RVbNVb3ZOenCqqsOXHaJpbtN+OuulLKBSKpIoPB
+ZDbDXn0qWkf4lOxtGSgolkUbgG07Lhzfgs+dul4UL84CkwZExmF3Rf1nRv+v7pq
Lt2dPsCb02YLxJnhHJb4rQaz2ZM4QCtTOcqYDUeKfLHCaZU4Ekm/OApKrpfw4/0o
fn8KOrFN0t4/dqnNuwVRgoaUIhsI47reApB2rs0AP4CggSIi8s6BXCxB4YzgThBK
5760T1giACYQC5MFdq1Gw+INSFmu0CNqt5wdJ5Z4z5448Gke06R+IMtjUiGDQ3Qt
T2fK3gWhZxk14M4UNrdETgTW/mQ4B/BcvikxvoBGpKbttG0agfOjTen6wyzpGfcd
8N9rSbaqqyUwC8uDotzFtFzzutVAU9d91TagGzWBhNoMfplwVTns27GOOgv1dn5s
QSSSmP0hTbPMDlThysKkR9BiOVbBtWGQpV936pPBgyWERGqMqC9xykLdVHv2Vu05
T0WMwKCAetgtAgMBAAE=
-----END RSA PUBLIC KEY-----
zeri@zeri:~$ gedit bruce.pem
zeri@zeri:~$ openssl asn1parse -i -in bruce.pem
    0:d=0  hl=4 l= 394 cons: SEQUENCE
    4:d=1  hl=4 l= 385 prim:  INTEGER           :AD3CBA9B6BE185BC31DD155B3556F764E7A70AAAAC3971DA2696ED37E3AE
BA52CA0522A92283C1F990DB0D79F4A9691FE253B1B464A0A259146E01B4ECB8737E0B3E76E97850BF380A4C19131985DD17F59D1BFEB
FBA6A2EDD9D3EC09BD3660BC499E11C96F8AD06B3D99338402B5339CA980D478A7CB1C26995381249BF380A4AAE97F0E3FD287E7F0A3A
B14DD2DE3F76A9CDBB0551828694221B08E3BADE029076AECD003F80A0812222F2CE815C2C41E18CE04E104AE7BEB44F58220026100B9
30576AD46C3E20D4859AED0236AB79C1D279678CF9E38F0691ED3A47E20CB6352218343742D4F67CADE05A1671935E0CE1436B7444E04
D6FE643807F05CBE2931BE8046A4A6EDB46D1A81F3A34DE9FAC32CE919F71DF0DF6B49B6AAAB25300BCB83A2DCC5B45CF3BAD54053D77
DD536A01B358184DA0C7E99705539ECDBB18E3A0BF5767E6C41249298FD214DB3CC0E54E1CAC2A447D0623956C1B56190A55F77EA93C1
832584446A8CA82F71CA42DD547BF656ED394F458CC0A0807AD82D
  393:d=1  hl=2 l=   3 prim:  INTEGER           :010001
```

```
zeri@zeri:~$ python3 -q
>>> from Crypto.Util from number
  File "<stdin>", line 1
    from Crypto.Util from number
                     ^^^^
SyntaxError: invalid syntax
>>> from Crypto.Util import number
>>> en = "AD3CBA9B6BE185BC31DD155B3556F764E7A70AAAAC3971DA2696ED37E3AEBA52CA0522A92283C1F990DB0D79F4A9691FE25
3B1B464A0A259146E01B4ECB8737E0B3E76E97850BF380A4C19131985DD17F59D1BFEBFBA6A2EDD9D3EC09BD3660BC499E11C96F8AD06
B3D99338402B5339CA980D478A7CB1C26995381249BF380A4AAE97F0E3FD287E7F0A3AB14DD2DE3F76A9CDBB0551828694221B08E3BAD
E029076AECD003F80A0812222F2CE815C2C41E18CE04E104AE7BEB44F58220026100B930576AD46C3E20D4859AED0236AB79C1D279678
CF9E38F0691ED3A47E20CB6352218343742D4F67CADE05A1671935E0CE1436B7444E04D6FE643807F05CBE2931BE8046A4A6EDB46D1A8
1F3A34DE9FAC32CE919F71DF0DF6B49B6AAAB25300BCB83A2DCC5B45CF3BAD54053D77DD536A01B358184DA0C7E99705539ECDBB18E3A
0BF5767E6C41249298FD214DB3CC0E54E1CAC2A447D0623956C1B56190A55F77EA93C1832584446A8CA82F71CA42DD547BF656ED394F4
58CC0A0807AD82D"
>>> de = bytes.fromhex(en)
>>> f = number.bytes_to_long(de)
>>> print(f)
39314062729225234484361945998200930162414726581518015528450945185795078159906004596692596036452615329276111 52
98494284088989875653206089485704517530014576580063349900545173887208138126700406986555739563855004111420614 30
85403607234109293286336393552756893984605214352988705258638979454736514997314223669075900783806715398880310 69
59459451477551329190379738890751917859777978615572286781595388821535447177971004010964350623594741297556254 53
83188249060356013447704323543320270894861523453698471587211334381276010281232318039154449603016365304693141 47
23851374554873036582282389904838597668286543337426581680817796038711228401443244655162199302352017964997866 67
73171610140831167305358755212866318581027689610988512094009738993939649316050678560054109986318426730309010 78
00840864961353814379995980368504156696451448980921196298453432234839442801090898431894041169896115073120431 66
70646676976361958828528229837610795843145048243492909
>>>
```

7.

Transparency 50 pts · 3058 Solves · 10 Solutions

When you connect to a website over HTTPS, the first TLS message sent by the server is the ServerHello containing the server TLS certificate. Your browser verifies that the TLS certificate is valid, and if not, will terminate the TLS handshake. Verification includes ensuring that:

- the name on the certificate matches the domain

- the certificate has not expired

- the certificate is ultimately signed (via a "chain of trust") by a root key of a Certificate Authority (CA) that's trusted by your browser or operating system

- Bài này sau khi mở thử file .pem mà chẳng thấy gì, em đã lên search subdomain của cryptohack.org. Vào thử thetransparencyflagishere.cryptohack.org thì nó ra flag!

Subdomains Lookup Pricing Related products

Subdomains matching the domain name: 9

tls2.cryptohack.org	First seen at: December 2, 2022	Date of the last update: February 15, 2023
blog.cryptohack.org	First seen at: August 12, 2020	Date of the last update: April 3, 2023
aes.cryptohack.org	First seen at: April 4, 2020	Date of the last update: February 6, 2023
tls1.cryptohack.org	First seen at: February 16, 2022	Date of the last update: February 12, 2023
www.cryptohack.org	First seen at: July 23, 2020	Date of the last update: April 3, 2023
matrix.cryptohack.org	First seen at: February 12, 2022	Date of the last update: April 22, 2023
web.cryptohack.org	First seen at: January 11, 2021	Date of the last update: March 6, 2023
thetransparencyflagishere.cryptohack.org	First seen at: September 7, 2020	Date of the last update: March 2, 2023
tls3.cryptohack.org	First seen at: December 2, 2022	Date of the last update: February 15, 2023

thetransparencyflagishere.cryptohack.org

```
crypto{thx_redpwn_for_inspiration}
```

8.

RSA Starter 1 10 pts · 6793 Solves

All operations in RSA involve modular exponentiation.

Modular exponentiation is an operation that is used extensively in cryptography and is normally written like: $2^{10} \bmod 17$

You can think of this as raising some number to a certain power ($2^{10} = 1024$), and then taking the remainder of the division by some other number ($1024 \bmod 17 = 4$). In Python there's a built-in operator for performing this operation: `pow(base, exponent, modulus)`

In RSA, modular exponentiation, together with the problem of prime factorisation, helps us to build a "trapdoor function". This is a function that is easy to compute in one direction, but hard to do in reverse unless you have the right information. It allows us to encrypt a message, and only the person with the key can perform the inverse operation to decrypt it.

Find the solution to $101^{17} \bmod 22663$

You have solved this challenge!

- Bài này chỉ đơn giản là ta chạy hàm pow sẽ ra được flag

```
f = pow(101,17,22663)
print(f)
```

```
PROBLEMS   OUTPUT   TERMINAL   DE
[Running] python -u "c:\Users\
19906
```

9.

RSA Starter 2 — 15 pts · 6378 Solves · 3 Solutions

RSA encryption is modular exponentiation of a message with an exponent `e` and a modulus `N` which is normally a product of two primes: `N = p * q`.

Together the exponent and modulus form an RSA "public key" `(N, e)`. The most common value for `e` is `0x10001` or `65537`.

"Encrypt" the number `12` using the exponent `e = 65537` and the primes `p = 17` and `q = 23`. What number do you get as the ciphertext?

- Bài này ta cũng làm các phép tính đơn giản là ra được flag

```
n = 17*23

f = pow(12,65537,n)

print(f)
```

```
[Running] python
"c:\Users\admin\
301
```

10.

RSA Starter 3 — 20 pts · 6021 Solves · 6 Solutions

RSA relies on the difficulty of the factorisation of the modulus `N`. If the primes can be found then we can calculate the Euler totient of `N` and thus decrypt the ciphertext.

Given `N = p*q` and two primes:

`p = 85750408333971275248999381077 7`

`q = 102922494794299807508034864721 9`

- Để tính giá trị hàm phi euler ta có n = (p-1) * (q-1)

```
print((857504083339712752489993810777-1) *
    (1029224947942998075080348647219-1))
```

```
882564595536224140639625987657529300394956519977044270821168
```

Task 2: 10 bài tiếp theo

1. ASCII — 5 pts · 29381 Solves

ASCII is a 7-bit encoding standard which allows the representation of text using the integers 0-127.

Using the below integer array, convert the numbers to their corresponding ASCII characters to obtain a flag.

`[99, 114, 121, 112, 116, 111, 123, 65, 83, 67, 73, 73, 95, 112, 114, 49, 110, 116, 52, 98, 108, 51, 125]`

In Python, the `chr()` function can be used to convert an ASCII ordinal number to a character (the `ord()` function does the opposite).

You have solved this challenge!

Cách giải:

```python
nums = [99, 114, 121, 112, 116, 111, 123, 65, 83, 67, 73, 73, 95, 112, 114, 49, 110, 116, 52, 98, 108, 51, 125]
for x in nums:
        print(chr(x), end ='')
```

Flag:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
crypto{ASCII_pr1nt4bl3}
```

2.

Hex — 5 pts · 28075 Solves

When we encrypt something the resulting ciphertext commonly has bytes which are not printable ASCII characters. If we want to share our encrypted data, it's common to encode it into something more user-friendly and portable across different systems.

Hexadecimal can be used in such a way to represent ASCII strings. First each letter is converted to an ordinal number according to the ASCII table (as in the previous challenge). Then the decimal numbers are converted to base-16 numbers, otherwise known as hexadecimal. The numbers can be combined together, into one long hex string.

Included below is a flag encoded as a hex string. Decode this back into bytes to get the flag.

`63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d`

> In Python, the `bytes.fromhex()` function can be used to convert hex to bytes. The `.hex()` instance method can be called on byte strings to get the hex representation.

Resources:
- ASCII table
- Wikipedia: Hexadecimal

You have solved this challenge!

Cách giải:

```
hex_input = "63727970746f7b596f755f77696c6c5f62655f776f726b696e675f776974685f6865785f737472696e67735f615f6c6f747d"
print(bytes.fromhex(hex_input))
```

Flag:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
b'crypto{You_will_be_working_with_hex_strings_a_lot}'
```

3.

Base64 — 10 pts · 25069 Solves

Another common encoding scheme is Base64, which allows us to represent binary data as an ASCII string using an alphabet of 64 characters. One character of a Base64 string encodes 6 binary digits (bits), and so 4 characters of Base64 encode three 8-bit bytes.

Base64 is most commonly used online, so binary data such as images can be easily included into HTML or CSS files.

Take the below hex string, *decode* it into bytes and then *encode* it into Base64.

`72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf`

> In Python, after importing the base64 module with `import base64`, you can use the `base64.b64encode()` function. Remember to decode the hex first as the challenge description states.

You have solved this challenge!

Cách giải:

```python
import base64
str_input = '72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf'

#decode str_input sang bytes
byte_input = bytes.fromhex(str_input)

#encode chuoi bytes vua co duoc sang base64
out = base64.b64encode(byte_input).decode()

print(out)
```

Flag:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
crypto/Base+64+Encoding+is+Web+Safe/
```

4.

Bytes and Big Integers — 10 pts · 20587 Solves

Cryptosystems like RSA works on numbers, but messages are made up of characters. How should we convert our messages into numbers so that mathematical operations can be applied?

The most common way is to take the ordinal bytes of the message, convert them into hexadecimal, and concatenate. This can be interpreted as a base-16/hexadecimal number, and also represented in base-10/decimal.

To illustrate:

```
message: HELLO
ascii bytes: [72, 69, 76, 76, 79]
hex bytes: [0x48, 0x45, 0x4c, 0x4c, 0x4f]
base-16: 0x48454c4c4f
base-10: 310400273487
```

Python's PyCryptodome library implements this with the methods `bytes_to_long()` and `long_to_bytes()`. You will first have to install PyCryptodome and import it with `from Crypto.Util.number import *`. For more details check the FAQ.

Convert the following integer back into a message:

`11515195063862318899931685488813747395775516287289682636499965282714637259206269`

You have solved this challenge!

Cách giải:

```python
from Crypto.Util.number import *
out = long_to_bytes(11515195063862318899931685488813747395775516287289682636499965282714637259206269).decode()
print (out)
```

Flag:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
crypto{3nc0d1n6_4ll_7h3_w4y_d0wn}
```

5.

XOR Starter 10 pts · 17302 Solves

XOR is a bitwise operator which returns 0 if the bits are the same, and 1 otherwise. In textbooks the XOR operator is denoted by ⊕, but in most challenges and programming languages you will see the caret ^ used instead.

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

For longer binary numbers we XOR bit by bit: `0110 ^ 1010 = 1100`. We can XOR integers by first converting the integer from decimal to binary. We can XOR strings by first converting each character to the integer representing the Unicode character.

Given the string `label`, XOR each character with the integer `13`. Convert these integers back to a string and submit the flag as `crypto{new_string}`.

The Python `pwntools` library has a convenient `xor()` function that can XOR together data of different types and lengths. But first, you may want to implement your own function to solve this.

You have solved this challenge!

Cách giải:

```
s = "label"
for x in s:
  print(chr(ord(x)^13), end="")
```

Lấy mỗi ký tự trong chuỗi s xor với 13 rồi chuyển kết quả vừa xor được thành ký tự.

New string:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
aloha
```

➔ Flag :crypto{aloha}

6.

XOR Properties 15 pts · 14223 Solves · 81 Solutions

In the last challenge, you saw how XOR worked at the level of bits. In this one, we're going to cover the properties of the XOR operation and then use them to undo a chain of operations that have encrypted a flag. Gaining an intuition for how this works will help greatly when you come to attacking real cryptosystems later, especially in the block ciphers category.

There are four main properties we should consider when we solve challenges using the XOR operator

```
Commutative: A ⊕ B = B ⊕ A
Associative: A ⊕ (B ⊕ C) = (A ⊕ B) ⊕ C
Identity: A ⊕ 0 = A
Self-Inverse: A ⊕ A = 0
```

Let's break this down. Commutative means that the order of the XOR operations is not important. Associative means that a chain of operations can be carried out without order (we do not need to worry about brackets). The identity is 0, so XOR with 0 "does nothing", and lastly something XOR'd with itself returns zero.

Let's put this into practice! Below is a series of outputs where three random keys have been XOR'd together and with the flag. Use the above properties to undo the encryption in the final line to obtain the flag.

```
KEY1 = a6c8b6733c9b22de7bc0253266a3867df55acde8635e19c73313
KEY2 ^ KEY1 = 37dcb292030faa90d07eec17e3b1c6d8daf94c35d4c9191a5e1e
KEY2 ^ KEY3 = c1545756687e7573db23aa1c3452a098b71a7fbf0fdddddde5fc1
FLAG ^ KEY1 ^ KEY3 ^ KEY2 = 04ee9855208a2cd59091d04767ae47963170d1660df7f56f5faf
```

Before you XOR these objects, be sure to decode from hex to bytes.

You have solved this challenge! View solutions

Cách giải:

```
*~/Documents/lab_mmh/lab.py - Mousepad
File Edit Search View Document Help
from pwn import *

k1 = "a6c8b6733c9b22de7bc0253266a3867df55acde8635e19c73313"
k2_k1 = "37dcb292030faa90d07eec17e3b1c6d8daf94c35d4c9191a5e1e"
k2_k3 = "c1545756687e7573db23aa1c3452a098b71a7fbf0fdddde5fc1"
flag_k1_k3_k2 = "04ee9855208a2cd59091d04767ae47963170d1660df7f56f5faf"

#dua cac du lieu de bai cho ve bytes
key1 = bytes.fromhex(k1)
key2_1 = bytes.fromhex(k2_k1)
key2_3 = bytes.fromhex(k2_k3)
flag1_3_2 = bytes.fromhex(flag_k1_k3_k2)

#tim key2 va key3
key2 = xor(key2_1, key1)
key3 = xor(key2, key2_3)

#tim lai flag nho cac key da giai duoc
flag1_3 = xor(flag1_3_2, key2)
flag1 = xor(flag1_3, key3)
flag = xor(flag1, key1)

print (flag)
```

Flag:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
b'crypto{x0r_i5_ass0c1at1v3}'
```

7.

Keyed Permutations 5 pts · 6531 Solves

AES, like all good block ciphers, performs a "keyed permutation". This means that it maps every possible input block to a unique output block, with a key determining which permutation to perform.

> A "block" just refers to a fixed number of bits or bytes, which may represent any kind of data. AES processes a block and outputs another block. We'll be specifically talking the variant of AES which works on 128 bit (16 byte) blocks and a 128 bit key, known as AES-128.

Using the same key, the permutation can be performed in reverse, mapping the output block back to the original input block. It is important that there is a one-to-one correspondence between input and output blocks, otherwise we wouldn't be able to rely on the ciphertext to decrypt back to the same plaintext we started with.

What is the mathematical term for a one-to-one correspondence?

Đáp án: bijection

8.

RSA Starter 1 — 10 pts · 6890 Solves

All operations in RSA involve modular exponentiation.

Modular exponentiation is an operation that is used extensively in cryptography and is normally written like: $2^{10} \bmod 17$

You can think of this as raising some number to a certain power ($2^{10} = 1024$), and then taking the remainder of the division by some other number ($1024 \bmod 17 = 4$). In Python there's a built-in operator for performing this operation: `pow(base, exponent, modulus)`

In RSA, modular exponentiation, together with the problem of prime factorisation, helps us to build a "trapdoor function". This is a function that is easy to compute in one direction, but hard to do in reverse unless you have the right information. It allows us to encrypt a message, and only the person with the key can perform the inverse operation to decrypt it.

Find the solution to $101^{17} \bmod 22663$

Cách giải:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python
Python 3.10.5 (main, Jun  8 2022, 09:26:22) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> pow(101, 17, 22663)
19906
```

Kết quả: 19906

9.

RSA Starter 2 — 15 pts · 6474 Solves · 4 Solutions

RSA encryption is modular exponentiation of a message with an exponent `e` and a modulus `N` which is normally a product of two primes: `N = p * q`.

Together the exponent and modulus form an RSA "public key" `(N, e)`. The most common value for `e` is `0x10001` or `65537`.

"Encrypt" the number `12` using the exponent `e = 65537` and the primes `p = 17` and `q = 23`. What number do you get as the ciphertext?

Cách giải:

```
n = 17 *23
out = pow (12, 65537, n)
print (out)
```

Kết quả:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
301
```

10.

RSA Starter 3 — 20 pts · 6111 Solves · 7 Solutions

RSA relies on the difficulty of the factorisation of the modulus `N`. If the primes can be found then we can calculate the Euler totient of `N` and thus decrypt the ciphertext.

Given `N = p*q` and two primes:

`p = 857504083339712752489993810777`

`q = 1029224947942998075080348647219`

What is the totient of `N`?

Cách giải:

```python
p = 857504083339712752489993810777
q = 1029224947942998075080348647219
phi = (p - 1) * (q - 1)
print(phi)
```

Kết quả:

```
┌──(kali㉿kali)-[~/Documents/lab_mmh]
└─$ python lab.py
882564595536224140639625987657529300394956519977044270821168
```