

Titulo

La importancia del entorno

Descripción

En una empresa se hace uso de un programa principal para ingresar a una base de datos, pero para proteger la privacidad de las credenciales se ha optado por hacer uso de un script adicional para cargar un nuevo entorno bash con variables de entorno. Pero la única persona que sabía la llave era el becario y solamente ha dejado un pequeño programa para obtener la llave de nuevo.

¿Podrás recuperar la llave y descubrir que hay dentro del programa principal?

Titulo

The importance of the enviroment

Description

A company uses a main program to access a database, but to protect the privacy of the credentials, they decided to use an additional script to load a new Bash enviroment with enviroment variables. However, the only person who knew the key was the intern, and he only left a small program to retrive the key again.

Can you recover the key and discover what's inside the main program?

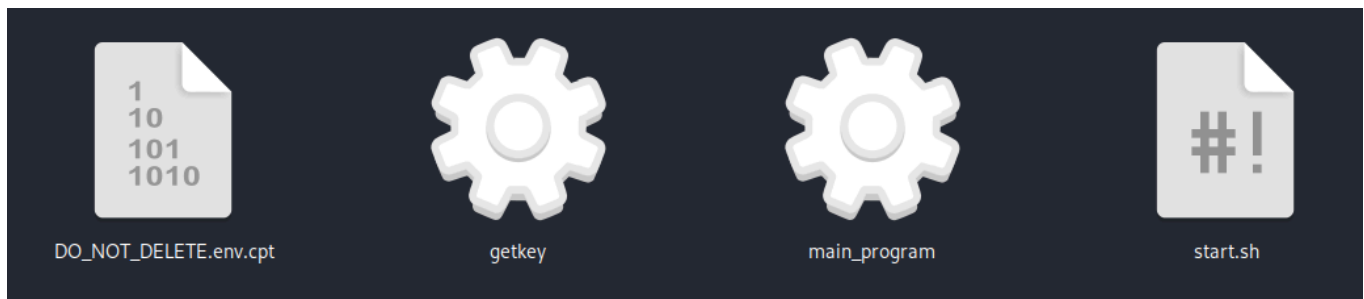
Write Up

Requisitos para resolver el reto

- Un sistema operativo Linux, con Bash
- ccrypt
- [radare2](#)
- [pyinstxtractor](#)

Pasos para resolver el reto

Contenido del reto



- **DO_NOT_DELETE.env.cpt**, archivo cifrado con variables de entorno.
- **getkey**, ejecutable para recuperar la llave de cifrado de DO_NOT_DELETE.env.cpt
- **main_program**, ejecutable con la flag, solo mostrará la flag si existen las variables de entorno correctas
- **start.sh**, script que permite generar un entorno nuevo de Bash con las variables de DO_NO_DELETE.env.cpt

Paso 1. Obtener la llave de "getkey"

- Al ejecutar el comando "strings" en getkey podemos obtener las siguientes cadenas que indican que fue programado con Python 3.11:

```
blib-dynload/_bz2.cpython-311-x86_64-linux-gnu.so
blib-dynload/_codecs_cn.cpython-311-x86_64-linux-gnu.so
blib-dynload/_codecs_hk.cpython-311-x86_64-linux-gnu.so
blib-dynload/_codecs_iso2022.cpython-311-x86_64-linux-gnu.so
blib-dynload/_codecs_jp.cpython-311-x86_64-linux-gnu.so
blib-dynload/_codecs_kr.cpython-311-x86_64-linux-gnu.so
blib-dynload/_codecs_tw.cpython-311-x86_64-linux-gnu.so
blib-dynload/_contextvars.cpython-311-x86_64-linux-gnu.so
blib-dynload/_decimal.cpython-311-x86_64-linux-gnu.so
blib-dynload/_hashlib.cpython-311-x86_64-linux-gnu.so
blib-dynload/_lzma.cpython-311-x86_64-linux-gnu.so
blib-dynload/_multibytecodec.cpython-311-x86_64-linux-gnu.so
blib-dynload/_typing.cpython-311-x86_64-linux-gnu.so
blib-dynload/resource.cpython-311-x86_64-linux-gnu.so
```

- Con ello se puede hacer uso de [pyinstxtractor](#) para recuperar el los contenidos de un archivo ejecutable generado por Pyinstaller, con ello poder recuperar el archivo .pyc
- Posteriormente se hará uso de una herramienta para recuperar el código fuente del archivo .pyc a .py, por ejemplo [PyLingual](#)

- Con el código fuente que se encuentra en el anexo del documento se podrá obtener la contraseña llevando a cabo las funciones del programa.
- La clave que está en base64 es: bs3ci2NGUDlcBflsFICV y la llave final es la que te muestra el programa "bs3ci2NGUDlcBflsFICV_4m_1_c0rr3ct?"

```
DO_NOT_DELETE.env.cpt getkey main_program start.sh
:challengeEnv$ ./getkey bs3ci2NGUDlcBflsFICV
Password obtained...
Congrats!, now you will be able to open the gate to the main program.
Take this: bs3ci2NGUDlcBflsFICV_4m_1_c0rr3ct?
Insert the right key: █
```

- Ingresando la clave completa el programa principal podrá funcionar con normalidad:

```
Insert the right key:
Connecting to db_production using root in localhost...
SELECT key FROM db_production.keys;
db
Something may happen in the background... █
```

Paso 2. Depurar (hacer Debugging) de "main_program" con las variables de entorno cargadas

- Para este paso se hará uso de la herramienta [radare2](#) la cual también es un programa para hacer debugging, pero "main_program" no funcionará si no tiene las variables de entorno cargadas, por ello es necesario ejecutar radare2 en modo de depuración con las variables de entorno.
- La forma de logra esto es modificando el script.sh para que en vez de solo ejecutar el ./main_program ejecute radare2 en modo de depuración:

```
#!/usr/bin/env/ bash

activated=''
while read line; do
    activated='1'
    export "$line"
done <<(ccdecrypt -c ./DO_NOT_DELETE.env.cpt --prompt "Insert the right key: ")

test "$activated" || { echo "Something went wrong..."; exit 1;}

r2 -d main_program
```

- Al ejecutar de nuevo el script iniciará radare2 en modo de depuración:

```
Insert the right key:
WARN: Relocs has not been applied. Please use '-e bin.relocs.apply=true' or '-e bin.cache=true' next time
-- Put some sugar before the release, attracts the bugs and makes it easier to squash them
[0x7f6dcc928a50]> █
```

- A continuación se ejecutará el comando "aaa" para realizar un análisis del código decompilado.

```
[0x7f6dcc928a50]> aaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods (af @@ method.*)
INFO: Recovering local variables (afva@@@F)
INFO: Skipping type matching analysis in debugger mode (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
[0x7f6dcc928a50]> █
```

- Con el comando "afl" se podrán visualizar las funciones del programa.

```
[0x7f6dcc928a50]> afl
0x5626c5313030 1 6 sym.imp.__cxa_atexit
0x5626c5313040 1 6 sym.imp.__isoc99_sscanf
0x5626c5313060 1 6 sym.imp.getenv
0x5626c5313080 1 6 sym.imp.__cxa_finalize
0x5626c5313090 1 33 entry0
0x5626c5315fc0 3 8207 reloc.__libc_start_main
0x5626c53130c0 4 34 sym.deregister_tm_clones
0x5626c53130f0 4 51 sym.register_tm_clones
0x5626c5313130 5 54 entry.fini0
0x5626c5313170 1 9 entry.init0
0x5626c531342d 4 82 sym.__static_initialization_and_destruction_0_int__int__
0x5626c5313070 1 6 sym.imp.std::ios_base::Init::Init__
0x5626c531347f 1 21 entry.init1
0x5626c5313251 4 476 main
0x5626c5313494 1 9 sym._fini
0x5626c5313179 4 216 sym.comp_str_int__
0x5626c5313000 3 23 sym._init
```

- Con el comando "s <función a seleccionar>" permitirá moverte a través del código a la función deseada, en este caso si se desea moverse a la función main se ejecutará **s main**:

```
[0x7f6dcc928a50]> s main
[0x5626c5313251]> pdf
; DATA XREF from entry0 @ 0x5626c53130a4(r)
476: int main (int argc, char **argv, char **envp);
afv: vars(8:sp[0x10..0x44])
0x5626c5313251 55 push rbp
0x5626c5313252 4889e5 mov rbp, rsp
0x5626c5313255 4883ec40 sub rsp, 0x40
0x5626c5313259 488d05d30d.. lea rax, str.00000000 ; 0x5626c5314033 ; "00000000"
0x5626c5313260 4889c7 mov rdi, rax
0x5626c5313263 e8f8fdffff call sym.imp.getenv ; char *getenv(const char *name)
0x5626c5313268 488945f8 mov qword [var_8h], rax
0x5626c531326c 488d05c90d.. lea rax, str.00000000 ; 0x5626c531403c ; "00000000"
0x5626c5313273 4889c7 mov rdi, rax
0x5626c5313276 e8e5fdffff call sym.imp.getenv ; char *getenv(const char *name)
0x5626c531327b 488945f0 mov qword [var_10h], rax
0x5626c531327f 488d05bf0d.. lea rax, str.00000000 ; 0x5626c5314045 ; "00000000"
0x5626c5313286 4889c7 mov rdi, rax
0x5626c5313289 e8d2fdffff call sym.imp.getenv ; char *getenv(const char *name)
0x5626c531328e 488945e8 mov qword [var_18h], rax
0x5626c5313292 488d05b50d.. lea rax, str.00000000 ; 0x5626c531404e ; "00000000"
0x5626c5313299 4889c7 mov rdi, rax
0x5626c531329c e8bffdffff call sym.imp.getenv ; char *getenv(const char *name)
0x5626c53132a1 488945e0 mov qword [var_20h], rax
0x5626c53132a5 488d05ab0d.. lea rax, str.00000000 ; 0x5626c5314057 ; "00000000"
0x5626c53132ac 4889c7 mov rdi, rax
0x5626c53132af e8acfdffff call sym.imp.getenv ; char *getenv(const char *name)
0x5626c53132b4 488945d8 mov qword [var_28h], rax
0x5626c53132b8 488d05a10d.. lea rax, str.00000000 ; 0x5626c5314060 ; "00000000"
0x5626c53132bf 4889c7 mov rdi, rax
0x5626c53132c2 e899fdffff call sym.imp.getenv ; char *getenv(const char *name)
```

- La flag se encuentra en la funcion `compstr`, entonces se ejecutará `**s sym.comp_str_int`, posteriormente se cambiará la vista a modo de debugging, con el comando **V** y luego

presionar p** dos veces, hasta ver la siguiente pantalla:

```
[0x5626c5313179 [xaDvc]0 195 /home/ctf/bank-scripts/challengeEnv/main_program]> diq;?t0;f .. @ sym.comp_str_int_
stopped at 0x00000000
- offset - 7071 7273 7475 7677 7879 7A7B 7C7D 7E7F 0123456789ABCDEF
0x7ffdcfd89d70 0100 0000 0000 0000 63b5 d8cf fd7f 0000 .....C.....
0x7ffdcfd89d80 0000 0000 0000 0000 72b5 d8cf fd7f 0000 .....R.....
0x7ffdcfd89d90 82b5 d8cf fd7f 0000 90b5 d8cf fd7f 0000 .....
0x7ffdcfd89da0 a2b5 d8cf fd7f 0000 b9b5 d8cf fd7f 0000 .....
s:0 z:0 c:0 o:0 p:0
rax 0x00000000 rbx 0x00000000 rcx 0x00000000
rdx 0x00000000 r8 0x00000000 r9 0x00000000
r10 0x00000000 r11 0x00000000 r12 0x00000000
r13 0x00000000 r14 0x00000000 r15 0x00000000
rsi 0x00000000 rdi 0x00000000 rsp 0x7ffdcfd89d70
rbp 0x00000000 rip 0x7f6dcc928a50 rflags 0x00000200
orax 0x0000003b
; CALL XREF from main @ 0x5626c5313406(x)
216: sym.comp_str_int_ (int64_t arg1);
'- args(rdi) vars(12:sp[0xc..0xac])
0x5626c5313179 55 push rbp ; comp_str(int)
0x5626c531317a 4889e5 mov rbp, rsp
0x5626c531317d 4881ecb000.. sub rsp, 0xb0
0x5626c5313184 89bd5cffffff mov dword [var_a4h], edi ; arg1
0x5626c531318a 48b8bdb7ba.. movabs rax, 0xaa0a3b6bcbab7bd
0x5626c5313194 48baa8b2b5.. movabs rdx, 0xad5be84bcb5b2a8
0x5626c531319e 488945b0 mov qword [var_50h], rax
0x5626c53131a2 488955b8 mov qword [var_48h], rdx
0x5626c53131a6 48b8b2a9b4.. movabs rax, 0xafb5beb6b5b4a9b2
0x5626c53131b0 48ba84adba.. movabs rdx, 0xb7b9bab2a9baad84
```

- Con **F2** podremos establecer un breakpoint en la línea de código actual, el cuál se verá de la siguiente forma:

```
[0x55e654d8d179 [xaDvc]0 215 /home/ctf/bank-scripts/challengeEnv/main_program]> diq;?t0;f .. @ sym.comp_str_int_
stopped at 0x00000000
- offset - 5051 5253 5455 5657 5859 5A5B 5C5D 5E5F 0123456789ABCDEF
0x7ff9b85d450 0100 0000 0000 0000 54e6 859b fd7f 0000 .....T.....
0x7ff9b85d460 0000 0000 0000 0000 63e6 859b fd7f 0000 .....C.....
0x7ff9b85d470 73e6 859b fd7f 0000 81e6 859b fd7f 0000 s.....
0x7ff9b85d480 93e6 859b fd7f 0000 aae6 859b fd7f 0000 .....
s:0 z:0 c:0 o:0 p:0
rax 0x00000000 rbx 0x00000000 rcx 0x00000000
rdx 0x00000000 r8 0x00000000 r9 0x00000000
r10 0x00000000 r11 0x00000000 r12 0x00000000
r13 0x00000000 r14 0x00000000 r15 0x00000000
rsi 0x00000000 rdi 0x00000000 rsp 0x7ff9b85d450
rbp 0x00000000 rip 0x7f6e6bea2a50 rflags 0x00000200
orax 0x0000003b
; CALL XREF from main @ 0x55e654d8d406(x)
216: sym.comp_str_int_ (int64_t arg1);
'- args(rdi) vars(12:sp[0xc..0xac])
0x55e654d8d179 b 55 push rbp ; comp_str(int)
0x55e654d8d17a 4889e5 mov rbp, rsp
0x55e654d8d17d 4881ecb000.. sub rsp, 0xb0
0x55e654d8d184 89bd5cffffff mov dword [var_a4h], edi ; arg1
```

- Con **F9** se podrá saltar hasta el siguiente breakpoint, en este caso al breakpoint anteriormente establecido.

```
[0x55e654d8d179 [xavc]0 215 /home/ctf/bank-scripts/challengeEnv/main_program]> diq;?t0;f .. @ sym.comp_str_int_
breakpoint at 0x00000000
- offset -      F8F9 FAFB FCFD FEFF  0 1  2 3  4 5  6 7  89ABCDEF01234567
0x7ffd9b85d2f8  0bd4 d854 e655 0000 0000 0000 db00 0000  ...T.U.....
0x7ffd9b85d308  5eef 859b fd7f 0000 b3ef 859b fd7f 0000  ^.....
0x7ffd9b85d318  0000 0000 0000 0000 cbe6 859b fd7f 0000  .....
0x7ffd9b85d328  b3e6 859b fd7f 0000 dee6 859b fd7f 0000  .....
s:0 z:0 c:0 o:0 p:1
rax 0x000000db      rbx 0x7ffd9b85d458      rcx 0x00000400
rdx 0x7f6e6be11390  r8 0x1999999999999999  r9 0x7ffd9b85d130
r10 0x00001000      r11 0x00000202      r12 0x00000000
r13 0x7ffd9b85d468  r14 0x55e654d8fdcd  r15 0x7f6e6beba020
rsi 0x55e6564c2eb0  rdi 0x000000db      rsp 0x7ffd9b85d2f8
rbp 0x7ffd9b85d340  rip 0x55e654d8d179  rflags 0x00000206
orax 0xfffffffffffff
;-- rip:
; CALL XREF from main @ 0x55e654d8d406(x)
216: sym.comp_str_int_ (int64_t arg1);
'- args(rdi) vars(l2:sp[0xc..0xac])
0x55e654d8d179 b 55 push rbp ; comp_str(int)
0x55e654d8d17a 4889e5 mov rbp, rsp
0x55e654d8d17d 4881ecb000.. sub rsp, 0xb0
0x55e654d8d184 89bd5cffffff mov dword [var_4h], edi ; arg1
0x55e654d8d18a 48b8bcb7ba.. movabs rax, 0xaa0a3b6bcbab7bd
0x55e654d8d194 48baa8b2b5.. movabs rdx, 0xad5be84bcb5b2a8
```

- Una vez ubicados en el breakpoint se usará **F7** para ir avanzando instrucción por instrucción, mientras que con **F8** se utiliza también para ir avanzando instrucción por instrucción pero sin entrar a las funciones call.
- Con ello se irá avanzando hasta encontrar el **for** donde se realiza una operación XOR:

```
[0x55e654d8d200 [xavc]0 215 /home/ctf/bank-scripts/challengeEnv/main_program]> diq;?t0;f .. @ sym.comp_str_int_+135 # 0x55e654d8d200
step at 0x55e654d8d20a
- offset -      4041 4243 4445 4647 4849 4A4B 4C4D 4E4F 0123456789ABCDEF
0x7ffd9b85d240  4000 d954 e655 0000 4800 d954 db00 0000  @..T.U..H..T....
0x7ffd9b85d250  0100 0000 0000 0000 c0fd d854 e655 0000  .....T.U..
0x7ffd9b85d260  90d2 859b fd7f 0000 8d08 d36b 6e7f 0000  .....kn....
0x7ffd9b85d270  afe0 d854 e655 0000 0000 0000 0000 0000  ...T.U.....
s:0 z:0 c:0 o:0 p:0
rax 0x000000db      rbx 0x7ffd9b85d458      rcx 0x00000400
rdx 0xa8baefb8bf84b7ba  r8 0x1999999999999999  r9 0x7ffd9b85d130
r10 0x00001000      r11 0x00000202      r12 0x00000000
r13 0x7ffd9b85d468  r14 0x55e654d8fdcd  r15 0x7f6e6beba020
rsi 0x55e6564c2eb0  rdi 0x000000db      rsp 0x7ffd9b85d240
rbp 0x7ffd9b85d2f0  rip 0x55e654d8d20a  rflags 0x00000202
orax 0xfffffffffffff
0x55e654d8d200 8845fb mov byte [var_5h], al
0x55e654d8d203 c745fc0000.. mov dword [var_4h], 0
;-- rip:
0x55e654d8d20c eblf jmp 0x55e654d8d22b
0x55e654d8d20e 8b45fc mov eax, dword [var_4h]
0x55e654d8d20f 4898 cdqe
0x55e654d8d211 0fb64405b0 movzx eax, byte [rbp + rax - 0x50]
0x55e654d8d216 3245fb xor al, byte [var_5h]
0x55e654d8d219 89c2 mov edx, eax
0x55e654d8d21b 8b45fc mov eax, dword [var_4h]
0x55e654d8d21e 4898 cdqe
0x55e654d8d220 88940560ff.. mov byte [rbp + rax - 0xa0], dl
0x55e654d8d227 8345fc01 add dword [var_4h], 1
; CODE XREF from comp_str(int) @ 0x55e654d8d20a(x)
0x55e654d8d22b 837dfc3f cmp dword [var_4h], 0x3f ; '?'
0x55e654d8d22f 7edb jle 0x55e654d8d20c
0x55e654d8d231 c645a100 mov byte [var_5h], 0
0x55e654d8d235 488d05cc0d.. lea rax, str.Something may happen in the background... n ; 0x55e654d8e008 ; "Something may happen in the background..."
```

- Con ello se avanzará en el for y en la parte superior empezará a mostrar el contenido de la cadena que se va generando al descifrarla con XOR:

```
[0x55e654d8d200 [xaDvc]0 215 /home/ctf/bank-scripts/challengeEnv/main_program> diq;?t0:f .. @ sym.comp_str_int_+135 # 0x55e654d8d200
step at 0x55e654d8d22f
- offset - 4041 4243 4445 4647 4849 4A4B 4C4D 4E4F 0123456789ABCDEF
0x7ffd9b85d240 4000 d954 e655 0000 4800 d954 db00 0000 @..T.U..H..T....
0x7ffd9b85d250 666c 6167 6d78 7b75 7369 6e67 5f65 6e76 flagmx{using_env
0x7ffd9b85d260 6972 6f6e 6d65 6e74 5f76 6172 6961 626c ironment_variabl
0x7ffd9b85d270 6573 5f63 616e 5f73 6563 7572 655f 636f es_can_secure_co
s:l z:0 c:l o:0 p:0
rax 0x0000002f rbx 0x7ffd9b85d458 rcx 0x00000400
rdx 0x0000006f r8 0x1999999999999999 r9 0x7ffd9b85d130
r10 0x00001000 r11 0x00000202 r12 0x00000000
r13 0x7ffd9b85d468 r14 0x55e654d8fdc0 r15 0x7f6e6beba020
rsi 0x55e6564c2eb0 rdi 0x000000db rsp 0x7ffd9b85d240
rbp 0x7ffd9b85d2f0 rip 0x55e654d8d22f rflags 0x00000293
orax 0xffffffffffffffff
0x55e654d8d200 8845fb mov byte [var_5h], al
0x55e654d8d203 c745fc0000.. mov dword [var_4h], 0
0x55e654d8d20a eblf jmp 0x55e654d8d22b
0x55e654d8d20c 8b45fc mov eax, dword [var_4h]
0x55e654d8d20f 4898 cdqe
0x55e654d8d211 0fb64405b0 movzx eax, byte [rbp + rax - 0x50]
0x55e654d8d216 3245fb xor al, byte [var_5h]
0x55e654d8d219 89c2 mov edx, eax
0x55e654d8d21b 8b45fc mov eax, dword [var_4h]
0x55e654d8d21e 4898 cdqe
0x55e654d8d220 88940560ff.. mov byte [rbp + rax - 0xa0], dl
0x55e654d8d227 8345fc01 add dword [var_4h], 1
; CODE XREF from comp_str(int) @ 0x55e654d8d20a(x)
0x55e654d8d22b 837dfc3f cmp dword [var_4h], 0x3f ; '?'
;-- rip:
0x55e654d8d22f 7edb jle 0x55e654d8d20c
```

- Como se puede apreciar, la flag está apareciendo en la parte superior pero ya no alcanza el espacio, en ese caso presionando **C** podremos mover la parte de arriba haciendo uso de la flechas direccionales, presionando **C** de nuevo hará que nos salgamos de este modo.

```
[0x55e654d8d200 *0x55e654d8d200 [xaDvc]0 ($$+0x0)]> diq;?t0:f .. @ sym.comp_str_int_+135 # 0x55e654d8d200
step at 0x55e654d8d22f
- offset - |5051 5253 5455 5657 5859 5A5B 5C5D 5E5F| 0123456789ABCDEF
0x7ffd9b85d250 |666c 6167 6d78 7b75 7369 6e67 5f65 6e76| |flagmx{using_env
0x7ffd9b85d260 |6972 6f6e 6d65 6e74 5f76 6172 6961 626c| ironment_variabl
0x7ffd9b85d270 |6573 5f63 616e 5f73 6563 7572 655f 636f| es_can_secure_co
0x7ffd9b85d280 |0000 0000 0000 0000 0000 0000 0000 0000| .....
s:l z:0 c:l o:0 p:0
rax 0x0000002f rbx 0x7ffd9b85d458 rcx 0x00000400
rdx 0x0000006f r8 0x1999999999999999 r9 0x7ffd9b85d130
r10 0x00001000 r11 0x00000202 r12 0x00000000
r13 0x7ffd9b85d468 r14 0x55e654d8fdc0 r15 0x7f6e6beba020
rsi 0x55e6564c2eb0 rdi 0x000000db rsp 0x7ffd9b85d240
rbp 0x7ffd9b85d2f0 rip 0x55e654d8d22f rflags 0x00000293
orax 0xffffffffffffffff
0x55e654d8d200 8845fb mov byte [var_5h], al
0x55e654d8d203 c745fc0000.. mov dword [var_4h], 0
0x55e654d8d20a eblf jmp 0x55e654d8d22b
0x55e654d8d20c 8b45fc mov eax, dword [var_4h]
0x55e654d8d20f 4898 cdqe
0x55e654d8d211 0fb64405b0 movzx eax, byte [rbp + rax - 0x50]
```

- Obteniendo como flag la siguiente:
"flagmx{using_environment_variables_can_secure_confidential_data}"

```
[0x55e654d8d200 [xaDvc]0 215 /home/ctf/bank-scripts/challengeEnv/main_program]> diq;?t0;f .. @ sym.comp_str_int_+135 # 0x55e654d8d200
step at 0x55e654d8d227
- offset - 5051 5253 5455 5657 5859 5A5B 5C5D 5E5F 0123456789ABCDEF
0x7ffd9b85d250 666c 6167 6d78 7b75 7369 6e67 5f65 6e76 flagmx(using_env
0x7ffd9b85d260 6972 6f6e 6d65 6e74 5f76 6172 6961 626c ironment_variab1
0x7ffd9b85d270 6573 5f63 616e 5f73 6563 7572 655f 636f es_can_secure_co
0x7ffd9b85d280 6e66 6964 656e 7469 616c 5f64 6174 617d nfidential_data}
s:0 z:0 c:0 o:0 p:1
rax 0x0000003f rbx 0x7ffd9b85d458 rcx 0x000000400
rdx 0x0000007d r8 0x1999999999999999 r9 0x7ffd9b85d130
r10 0x00001000 r11 0x00000202 r12 0x00000000
r13 0x7ffd9b85d468 r14 0x55e654d8fdc0 r15 0x7f6e6beba020
rsi 0x55e6564c2eb0 rdi 0x000000db rsp 0x7ffd9b85d240
rbp 0x7ffd9b85d2f0 rip 0x55e654d8d227 rflags 0x00000206
orax 0xfffffffffffff
0x55e654d8d200 8845fb mov byte [var_5h], al
0x55e654d8d203 c745fc0000.. mov dword [var_4h], 0
0x55e654d8d20a eblf jmp 0x55e654d8d22b
0x55e654d8d20c 8b45fc mov eax, dword [var_4h]
0x55e654d8d20f 4898 cdqe
0x55e654d8d211 0fb64405b0 movzx eax, byte [rbp + rax - 0x50]
0x55e654d8d216 3245fb xor al, byte [var_5h]
0x55e654d8d219 89c2 mov edx, eax
0x55e654d8d21b 8b45fc mov eax, dword [var_4h]
0x55e654d8d21e 4898 cdqe
0x55e654d8d220 88940560ff.. mov byte [rbp + rax - 0xa0], dl
;-- rip:
0x55e654d8d227 8345fc01 add dword [var_4h], 1
; CODE XREF from comp_str(int) @ 0x55e654d8d20a(x)
0x55e654d8d22b 837dfc3f cmp dword [var_4h], 0x3f
0x55e654d8d22f 7edb jle 0x55e654d8d20c
```

Anexos

Código fuente de getkey

```
p = print

import subprocess
import base64
import sys

def d(s):
    pd = base64.b64decode(s).decode('utf-8')
    return pd

def dc(s):
    n = 0
    a = ""
    for c in s:
        if n == 0:
            n += 1
            a += c
        else:
            n += 1
            if n == 5:
                n = 0
    return a

def es():
    subprocess.run(['bash', './start.sh'])
```



```

def gp(s):
    p =
    b'VmFiY2RDZWZnaGxpamtsRm1vcHFzcnN0dU12eHl6ZjEyMzRCNTY3OGM5MEFCbENERUZER0hJS1VLTE1OR
    09QUVJOU1RVVjJjYWVphaWJjZGVjZmdoaTNqa2xtc25vcHFi'
    if s == dc(rs(d(p))):
        return True
    else:
        return False

def main():
    if va():
        p("Password obtained...")
        c = sys.argv[1]
        if gp(c) == True:
            p("Congrats!, now you will be able to open the gate to the main
program. \nTake this: " + c + "_4m_1_c0rr3ct?")
            es()
        else:
            p("That's not the password :( ")

def rs(s):
    rs = s[::-1]
    return rs

def va():
    if len(sys.argv) != 2:
        p("Usage: getkey password")
        return False
    else:
        return True

if __name__ == '__main__':
    main()

```

Código fuente de main_program

```

#include <cstdlib>
#include <iostream>
#include <cstring>

using namespace std;
void comp_str(int key){
    unsigned char data[64] = {0xbd, 0xb7, 0xba, 0xbc, 0xb6, 0xa3 ,0xa0, 0xae,
0xa8, 0xb2, 0xb5, 0xbc, 0x84, 0xbe, 0xb5, 0xad, 0xb2, 0xa9, 0xb4, 0xb5, 0xb6, 0xbe,

```

```

0xb5, 0xaf, 0x84, 0xad, 0xba, 0xa9, 0xb2, 0xba, 0xb9, 0xb7, 0xbe, 0xa8, 0x84, 0xb8,
0xba, 0xb5, 0x84, 0xa8, 0xbe, 0xb8, 0xae, 0xa9, 0xbe, 0x84, 0xb8, 0xb4, 0xb5, 0xbd,
0xb2, 0xbf, 0xbe, 0xb5, 0xaf, 0xb2, 0xba, 0xb7, 0x84, 0xbf, 0xba, 0xaf, 0xba,
0xa6};

    char flag[65];
    unsigned char x_key;
    x_key = (unsigned char)key;
    for (int i=0; i<64; i++){
        flag[i] = data[i] ^ x_key;
    }
    flag[65] = 0x00;
    cout << "Something may happen in the background...\n";
}

int main(){
    char* database = getenv("00000000");
    char* user_db = getenv("00000000");
    char* pass_db = getenv("00000000");
    char* host_db = getenv("00000000");
    char* port_db = getenv("00000000");
    char* key = getenv("00000000");
    char* hex_key = getenv("00000000");
    int int_key;
    if(key != NULL){
        sscanf(key, "%d", &int_key);
        cout << "Connecting to " << database << " using " << user_db << "
in " << host_db << "... \n";
        cout << "SELECT key FROM " << database << ".keys;\n";
        cout << hex_key << "\n";
        comp_str(int_key);
    }
    else{
        cout << "Enviroment variable not found, please use getkey
first!\n";
    }

    return 0;
}

```

Código fuente de start.sh

```

#!/usr/bin/env/ bash

activated=' '

```

```
while read line; do
    activated='1'
    export "$line"
done < <(ccdecrypt -c ./DO_NOT_DELETE.env.cpt --prompt "Insert the right key: ")

test "$activated" || { echo "Something went wrong..."; exit 1;}

./main_program
```