Liliane Owens
U00846594
Undergraduate
Meilin Liu
3/31/2025

Lab/Project 2 OF CEG 4424/6624 Security Attacks and Defenses

(90 Points)

1. The objective

   The objective of this lab is for students to learn more about Buffer Overflow, and exhaustive key search. Students will gain hands-on experience on Buffer overflow and exhaustive key search.

2. Submission

   A team can have up to 3 students. All students in the same team will receive the same grade.

   a. Each team only submits one report or a copy of answers/files. (See the section of tasks).
   b. **Each team member needs to submit the list of the names of all team members**.

3. Tools

You will use the cs unix server, fry.cs.wright.edu for this project, you need to connect to this unix server remotely using a secure shell client, putty. You can remotely connect to this unix server, fry.cs.wright.edu, on campus from a Wright State computer or use your own laptop connecting to the WSU wifi network named "WSU-Secure". Note that you cannot remotely connect to this computer using ssh using computers outside Wright State University without installing VPN or use the campus "WSU_EZ_CONNECT" wifi network.

If you want to connect to this server remotely off campus, you need to install VPN on your computer first (You can download the VPN from WSU, https://www.wright.edu/information-technology/virtual-private-network-vpn.)

You can use WinSCP, the secure file transfer client to transfer files between your local machine, and the server, fry.cs.wright.edu.

How to use software tools on fry.cs.wright.edu

(1). Connect to fry.cs.wright.edu using a VPN.

(2). Use putty or other secure shell clients to connect to fry.cs.wright.edu using your campus id (for example, w123abc) and password.

4. Tasks

Download Proj2.zip from Pilot.

Task 1 - Examining the code (10 Points)

- The BOFtest.c is a source C code program. Use a text editor to examine the C source code file (but don't change it).
- Which line of code in this file is the source of the buffer overflow vulnerability? (5 points)
  The buffer overflow will occur on the following line inside the foo function:
  strcpy(c, bar);
  strcpy does not perform bounds checking, means if bar is longer than 11 characters excluding null terminator, it will overwrite adjacent memory.
- Which variable is subject to overflow? (5 points)

The vulnerable variable is:
char c[12];
This array can hold up to 11 characters plus null terminator, but if bar is longer, it will lead to a buffer overflow

Task 2 - Overflowing the buffer (25 Points)

- Compile the BOFtest.c code with the following command: ***gcc BOFtest.c -o BOFtest***
  What does this command do? (5 points)
- **It compiles BOFtest.c int executable named BOFtest. If compilation is successful, you can run the program with:**
  ./BOFtest
- Now run the program a few times with various inputs (various number of characters).
- Are you able to cause the program to halt in an error state? What is the number of input characters when the error occurred? Why? (20 points) (Please include the screenshot for the cases when the program runs correctly, and when the error occurred. The screenshot should include your username (campus id, w123abc), current directory, the execution commands, and the complete execution results.)
  Yes. The error occurred at 12th input. It makes overflow, saying "Segmentation Fault"

```
[w046mla@login01 Proj2-test]$ gcc BOFtest.c -o BOFtest
[w046mla@login01 Proj2-test]$ cat BOFtest.c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

int foo(char* bar);

int main(int argc, char** argv)
{
        printf(" %s", "CEG");
        foo(argv[1]);
        printf(" %s", "44246424");
        return 0;
}
int foo(char* bar)
{
        char c[12];
        strcpy(c, bar);
        return 0;
}
[w046mla@login01 Proj2-test]$ ./BOFtest abcdef1234
 CEG 44246424[w046mla@login01 Proj2-test]$ ./BOFtest abcdef12345
 CEG 44246424[w046mla@login01 Proj2-test]$ ./BOFtest abcdef123456
Segmentation fault (core dumped)
[w046mla@login01 Proj2-test]$
```

Task 3 – Examine the code (15 Points)

- The Proj2-test.cpp is a source C++ code program that used cryptopp library functions. Use a text editor to examine the C++ source code file (but don't change it).
- (15 points) What is the functionality of the source c++ code? Write a short paragraph to answer this question, not just one sentence.

It is attempting to brute-force decrypt an AES-encrypted file using the Electronic Codebook (ECB) mode. It reads an encrypted input file(infile), iterates through possible variations of 16 bytes key modifying the first four characters systematically, and attempting decryption using the aes_decode function. After each decryption attempt, it checks whether the resulting plaintext contains mostly ASCII characters with a threshold of 80%. If the key produces fully ASCII compliant text, it prints the discovered key and plain text, indicating a successful decryption. The decrypted output is also written to the specified output file(outfile). The program essentially functions as key recovery attack, trying to find a valid AES key through systematic character modifications within a limited key space.

Task 4 – Compile and execute (10 Points)

Unzip Proj2.zip to get 00_e, 01_e, 02_e, 03_e.

Compile the Proj2-test.cpp code with the following command: cryptog++ Proj2-test.cpp -lcryptopp -o Proj2.

```
[w046mla@login01 Proj2-test]$ vim Proj2-test.cpp
[w046mla@login01 Proj2-test]$ ls
00_e  01_e  02_e  BOFtest    -lcryptopp  Proj2-test.cpp
00_p  01_p  02_p  BOFtest.c  Proj2-test
[w046mla@login01 Proj2-test]$ cryptog++ Proj2-test.cpp -o Proj2-test -lcryptop
srun: job 102450 queued and waiting for resources
srun: job 102450 has been allocated resources
[w046mla@login01 Proj2-test]$ ls
00_e  01_e  02_e  BOFtest    -lcryptopp  Proj2-test.cpp
00_p  01_p  02_p  BOFtest.c  Proj2-test
[w046mla@login01 Proj2-test]$ ls -l
total 13986
-rw-r--r-- 1 w046mla wsusers      368 Mar 29 17:14 00_e
-rw-r--r-- 1 w046mla wsusers      359 Mar 29 21:31 00_p
-rw-r--r-- 1 w046mla wsusers      288 Mar 29 17:14 01_e
-rw-r--r-- 1 w046mla wsusers      274 Mar 29 21:33 01_p
-rw-r--r-- 1 w046mla wsusers      464 Mar 29 17:14 02_e
-rw-r--r-- 1 w046mla wsusers      461 Mar 29 21:35 02_p
-rwxr-xr-x 1 w046mla wsusers    18160 Mar 31 11:17 BOFtest
-rw-r--r-- 1 w046mla wsusers      285 Mar 29 17:14 BOFtest.c
-rw-r--r-- 1 w046mla wsusers        0 Mar 29 18:13 -lcryptopp
-rwxr-xr-x 1 w046mla wsusers 14295848 Mar 31 11:43 Proj2-test
-rw-r--r-- 1 w046mla wsusers     2805 Mar 29 17:14 Proj2-test.cpp
[w046mla@login01 Proj2-test]$ ./Proj2-test 00_e
Starting Key: aaaax7qfkp3mbv9w
```

execute Proj2 using the following command:
./Proj2 00_e 00_p

```
[w046mla@login01 Proj2-test]$ ./Proj2-test 00_e
Starting Key: aaaax7qfkp3mbv9w
usage:aes_decode infile outfile
[w046mla@login01 Proj2-test]$ ./Proj2-test 00_e 00_p
Starting Key: aaaax7qfkp3mbv9w
possible candidate: ui8rx7qfkp3mbv9w
Ratio: 1
key found: ui8rx7qfkp3mbv9w
Plain text: The very next morning the world regained its rainbow of colors as
f nothing had ever happened. At first this was a novelty but soon people forgo
 the world had once been all violet. They forgot the world had once turned yel
ow, then orange, then pink, and then blue. They returned to saying they felt "
lue" or were "green" with envy or had a "green" thumb.

Decrypted plain text saved to: 00_p
```

01_e and 02_e

```
[w046mla@login01 Proj2-test]$ ./Proj2-test 01_e
Starting Key: aaaax7qfkp3mbv9w
usage:aes_decode infile outfile
[w046mla@login01 Proj2-test]$ ./Proj2-test 02_e
Starting Key: aaaax7qfkp3mbv9w
usage:aes_decode infile outfile
[w046mla@login01 Proj2-test]$
```

01_e 01_p
02_e 02_p

```
[w046mla@login01 Proj2-test]$ ./Proj2-test 01_e 01_p
Starting Key: aaaax7qfkp3mbv9w
possible candidate: 2uwyx7qfkp3mbv9w
Ratio: 1
key found: 2uwyx7qfkp3mbv9w
Plain text: But with his great wings the dragon took no time in knocking 50 kn
ghts from their horses and breathing fire on the rest. He said,
You must think I'm here to fiddle,
1,000 men-that's not the riddle.
One man alone, only one man,
With a thousand swords,
That's the plan.

Decrypted plain text saved to: 01_p
[w046mla@login01 Proj2-test]$ ./Proj2-test 02_e 02_p
Starting Key: aaaax7qfkp3mbv9w
possible candidate: b2vcx7qfkp3mbv9w
Ratio: 1
key found: b2vcx7qfkp3mbv9w
Plain text: The color of the Blue Jay became a big issue because he had such a
little bit of blue and the world had such a whole lot of violet. People argued
about the importance of that. Some said the Blue Jay must be a very special bi
d or maybe not a bird at all because he alone had kept his true color. Others
aid this was silly, that the Blue Jay ate bird seed and drank water and fluffe
 his feathers and that other than his special color he was still just a bird.
```

Display the contents of 00_p using cat 00_p.

```
[w046mla@login01 Proj2-test]$ cat 00_p
The very next morning the world regained its rainbow of colors as if nothing h
d ever happened. At first this was a novelty but soon people forgot the world
ad once been all violet. They forgot the world had once turned yellow, then or
nge, then pink, and then blue. They returned to saying they felt "blue" or wer
 "green" with envy or had a "green" thumb.
[w046mla@login01 Proj2-test]$
```

- What is the output? Display the output using the command: cat 00_p (Please include the screenshots of the output in your submission for the input files 00_e, 01_e, 02_e. The screenshot should include your username (campus id, w123abc), current directory, the execution commands, and the complete execution results.)
  0d -b 00_e
  0d -b 01_e
  0d -b 02_e

```
[w046mla@login01 Proj2-test]$ od -b 00_e
0000000 342 251 315 341 220 365 227 262 072 230 175 226 173 051 205 033
0000020 324 231 126 153 064 025 227 376 360 163 361 036 010 001 162 252
0000040 017 032 367 050 114 121 137 045 344 225 310 115 122 053 307 227
0000060 157 372 040 075 334 235 216 335 227 172 100 301 311 055 246 265
0000100 156 146 232 005 017 157 130 205 335 156 122 212 103 061 121 172
0000120 027 342 022 046 252 030 367 324 121 123 165 031 306 320 200 102
0000140 167 216 030 363 220 235 264 256 176 121 072 203 236 271 052 255
0000160 356 205 365 346 034 247 163 244 114 021 025 360 364 000 206 353
0000200 316 040 250 202 026 336 121 153 120 005 301 112 323 262 111 234
0000220 043 073 224 326 264 200 317 314 246 050 177 064 203 007 112 220
0000240 255 345 155 130 246 333 265 124 346 344 370 137 232 155 316 330
0000260 272 342 234 163 114 010 200 057 371 271 012 351 202 365 027 034
0000300 043 073 224 326 264 200 317 314 246 050 177 064 203 007 112 220
0000320 344 377 051 150 114 150 246 163 056 121 262 200 076 343 305 246
0000340 100 247 004 016 317 072 210 373 214 067 041 343 272 355 125 167
0000360 212 147 331 106 323 162 227 117 260 265 301 323 107 145 370 350
0000400 302 265 114 033 024 370 076 146 337 223 256 005 360 050 272 015
0000420 344 264 372 042 121 026 134 224 314 002 256 301 266 273 305 171
0000440 017 374 010 325 257 040 150 236 305 352 221 153 314 315 023 204
0000460 156 264 055 011 311 132 202 047 030 130 232 175 031 005 325 173
0000500 335 361 071 276 332 041 302 127 361 111 116 332 133 115 067 206
0000520 162 353 170 146 013 070 173 174 135 366 070 331 154 130 350 120
0000540 225 035 347 226 037 263 112 146 265 072 341 155 010 374 311 237
0000560
[w046mla@login01 Proj2-test]$ od -b 01_e
0000000 100 327 000 115 214 106 141 266 071 306 052 156 157 304 336 017
0000020 303 332 343 152 136 376 037 075 111 124 267 260 323 324 055 215
0000040 114 212 174 335 275 226 145 234 311 050 361 330 152 226 057 372
0000060 236 021 330 172 213 227 333 272 056 340 115 350 231 126 304 167
0000100 002 172 216 322 223 325 031 001 333 062 212 004 035 253 303 017
0000120 063 021 105 362 073 366 245 057 155 025 313 006 225 314 217 333
0000140 337 255 264 165 003 361 237 121 104 321 006 311 223 247 353 014
0000160 166 247 220 151 112 350 331 367 116 273 213 340 176 353 145 271
0000200 264 021 377 164 377 267 224 345 374 235 050 146 360 172 111 166
0000220 350 027 157 313 074 151 255 372 034 271 331 071 324 365 051 012
0000240 113 215 340 172 156 275 336 002 335 057 335 151 315 041 211 033
0000260 045 343 157 366 015 355 055 241 326 170 305 151 062 222 332 070
0000300 215 253 305 377 052 002 031 073 054 044 115 157 251 210 205 305
0000320 356 253 246 147 275 354 015 077 235 345 073 051 170 226 177 301
0000340 026 330 333 104 164 233 132 377 067 312 201 273 161 267 121 253
0000360 141 237 314 115 020 152 212 104 136 376 150 211 345 046 032 364
0000400 316 140 024 216 006 260 175 356 131 041 154 023 221 223 062 043
0000420 074 003 237 131 213 213 155 135 271 165 027 072 240 225 130 213
0000440
```

```
[w046mla@login01 Proj2-test]$ od -b 02_e
0000000 203 224 126 320 332 052 365 253 047 045 010 002 356 062 165 252
0000020 367 317 326 044 126 041 055 145 041 147 340 072 303 156 063 305
0000040 251 253 203 303 222 027 251 040 363 253 151 307 167 135 262 242
0000060 165 327 360 163 363 071 025 205 227 102 111 055 153 255 250 165
0000100 322 044 052 361 156 003 075 075 171 021 303 301 077 371 052 062
0000120 244 243 314 200 040 061 147 122 116 375 046 177 325 126 253 117
0000140 055 054 067 070 240 062 311 372 253 367 235 323 144 103 223 057
0000160 312 165 070 111 150 036 371 106 170 306 146 246 170 154 026 115
0000200 010 352 144 230 275 313 206 371 117 064 005 006 114 145 207 312
0000220 372 042 241 051 232 116 344 033 101 220 066 356 204 076 374 142
0000240 354 350 017 074 366 326 062 130 111 322 352 345 300 132 113 216
0000260 006 330 002 156 103 037 142 136 054 275 166 144 273 201 071 331
0000300 123 375 221 220 147 147 273 212 036 331 040 075 267 247 124 262
0000320 374 113 221 374 225 223 264 227 042 021 327 263 165 242 312 252
0000340 013 037 231 062 022 220 360 232 200 015 376 222 077 203 111 171
0000360 311 012 357 367 217 242 302 154 303 144 157 065 101 130 072 257
0000400 356 200 122 365 253 063 370 066 351 340 364 306 252 324 071 323
0000420 205 226 014 200 354 322 100 357 054 260 062 256 343 066 235 030
0000440 327 160 112 165 306 211 052 054 222 347 136 054 046 154 137 346
0000460 131 236 037 156 271 271 020 257 264 357 276 357 313 015 004 037
0000500 257 350 164 150 164 274 063 211 360 333 316 151 152 311 214 125
0000520 252 273 025 176 214 347 331 010 104 042 014 370 250 340 305 356
0000540 350 337 310 305 112 111 267 165 200 023 131 042 066 257 350 350
0000560 102 300 155 257 325 235 323 160 125 054 050 240 143 370 123 364
0000600 354 347 007 136 120 014 240 104 346 350 032 215 024 325 126 324
```

Task 4 – Add code to record the number of keys that have been tested (30 points, for graduate students)

- How many key combinations have been tried before the correct key is found? Please add code to record the number of the keys been tried? (The graduate students need to submit the modified source code program for this task. The graduate students also need to report the number of the keys tested for the input files 00_e, 01_e, 02_e. It would be great if the graduate students can include the screenshots to show the number of the keys tested. The screenshot should include your username (campus id, w123abc), current directory, the execution commands, and the complete execution results.)

How to compile your programs on fry.cs.wright.edu

fry.cs.wright.edu is a unix server and you can use the cryptopp installed on it.

1. Compiling method 1:

Every time, you log into fry.cs.wright.edu, run the command,

alias cryptog++='srun singularity exec /home/containers/cryptopp.sif g++'

then you can run the command, alias,

If the output contains the commands as shown below

alias cryptog++='srun singularity exec /home/containers/cryptopp.sif g++'

Then, you can compile your c++ program source.cpp using the following command:

cryptog++ <sourcefile.cpp> -lcryptopp -o desenc1

2. Compiling method 2:

Once you log into fry.cs.wright.edu, you can compile your c++ program source.cpp using the following command without modifying the .bashrc.

srun singularity exec /home/containers/cryptopp.sif g++ des_encode_SP2023.cpp -lcryptopp -o desencl

3. Execution:

You can use the following command to execute your program.

./desenc1

Tutorial to use the AES function in the crypto library Crypto++:

On fry.cs.wright.edu, the crypto++ library is installed, which you can use directly. You can log into fry.cs.wright.edu using you school wid (w123abc) and corresponding password. You can remotely connect to fry.cs.wright.edu on campus from a Wright State computer or use your own laptop connecting to the WSU wifi network named “WSU-Secure”. Note that you cannot connect to fry.cs.wright.edu using computers outside Wright State University without installing VPN or use the campus “WSU_EZ_CONNECT” wifi network, i.e., you need to follow the instructions to install VPN on your computer first.

I have created a discussion forum on pilot. If you have any questions, you can post your questions on the discussion forum first.

1. In order to use the crypto library, in your C++ source program (e.g., test1.cpp), you need to include the right library files, and use the right namespace as follows.

```
#include "cryptopp/cryptlib.h"
#include "cryptopp/hex.h"
#include "cryptopp/filters.h"
#include "cryptopp/des.h"
#include "cryptopp/aes.h"
```

#include "cryptopp/modes.h"

2. How to compile your source program:
  cryptog++ <sourcefile.cpp> -lcryptopp -o test

-lcryptopp : link CryptoPP library.

3. How to execute your program:

cryptoexec ./test

4. Encryption with AES:

A text string can be encoded with AES algorithm using the following tool function:

```
string aes_encode(string & plain, byte key[])
{
    string cipher;
    try{
            ECB_Mode<AES>::Encryption enc;
            enc.SetKey(key, AES::DEFAULT_KEYLENGTH);
            StringSource(plain, true, new StreamTransformationFilter(enc, new
            StringSink(cipher)));  //add padding by StreamTransformationFilter
    }
    catch(const CryptoPP::Exception & e)
    {
    }
    return cipher;
}
```

The input parameter "plain" is the plain text that you want to encrypt with AES encryption algorithm. The input parameter "key" is a byte array that stores the key you want to use during the cipher process of AES. The length of the key array is defined by constant AES::DEFAULT_KEYLENGTH (AES::DEFAULT_KEYLENGTH is 16 for now).The string returned by this function is the output cipher text. In this program assignment, for simplicity, we use AES in ECB mode (In ECB mode, each block of 128 bits of plaintext is encrypted independently using the same key). The input plaintext is padded by StreamTransformationFilter tool if the number of bits in the input plain text file is not a multiple of 128 bits (16 bytes).

5. Decryption with AES:

A cipher text can be decrypted with AES algorithm using the following tool function:

```
string aes_decode(string & cipher,byte key[])
{
string plain;
try{
        ECB_Mode< AES >::Decryption dec;
        dec.SetKey(key, AES::DEFAULT_KEYLENGTH);
        StringSource s(cipher, true, new StreamTransformationFilter(dec, new
        StringSink(plain)));
}
catch(const CryptoPP::Exception& e){
}
return plain;
}
```

The input parameter "cipher" is the ciphertext that you want to decrypt with AES in ECB mode. The input parameter "key" is a byte array that stores the key you want to use during the decryption process of AES. The length of key array is defined by constant AES::DEFAULT_KEYLENGTH.