**Random (crypto, 100p)**

We were given this simple C++ code:

#include <fstream>

#include <iomanip>

#include <iostream>

#include <random>

#include <string>

using std::cin;

using std::cout;

using std::endl;

using std::ifstream;

using std::string;

[[noreturn]] void fatal\_error(const string& msg)

{

cout << msg << endl;

exit(0);

}

class RandStream

{

std::random\_device rd;

std::mt19937 gen;

public:

RandStream() : rd(), gen(rd()) {}

unsigned int NextUInt()

{

return gen();

}

};

int main()

{

cout << "Let's see if you can predict Mersenne Twister output from just"

<< " six values!" << endl;

cout << "btw. You have only 5 seconds." << endl;

RandStream rand{};

for (int i = 0; i < 6; i++)

cout << std::hex << std::setfill('0') << std::setw(8) << rand.NextUInt()

<< endl;

for (int i = 0; i < 5; i++)

{

unsigned int num;

cin >> std::hex >> num;

if (num != rand.NextUInt())

fatal\_error("Wrong!");

}

cout << "Good work!" << endl;

ifstream f("flag.txt");

string flag;

f >> flag;

if (f.fail())

fatal\_error("Reading flag failed, contact admin");

cout << flag << endl;

}

This line may look strong:

RandStream() : rd(), gen(rd()) {}

But in fact it's completely broken (mt is seeded with only 32bits of entropy). So it's enough to bruteforce the seed to get the flag.

According to organisers we were supposed to "cache" something to be "fast". Well, we're more lazy than that, so we just guessed randomly until we did it. With tiny complication, beacuse of 5 second timeout, code looked like this:

def stuff():

s = socket.socket()

s.connect(('random.hackable.software', 1337))

data = s.recv(9999)

sample = data.split('\n')[2]

out = Command(['./a.out', sample, '0']).run(timeout=5) # class stolen from stackoverflow

if out:

print '---'

print out

print '---'

s.send(out + '\n')

print s.recv(9999)

def main():

while True:

if stuff():

break

And a.out was compiled version of minimally modified given program:

class RandStream

{

//std::random\_device rd;

std::mt19937 gen;

public:

RandStream(int i) : gen(i) {}

unsigned int NextUInt()

{

return gen();

}

};

int main(int argc, char \*argv[])

{

unsigned int sought = std::strtoul(argv[1], NULL, 16);

unsigned int start = std::strtoul(argv[2], NULL, 16);

for (unsigned int i = start; i < 0xFFFFFFFE; i++) {

RandStream rand(i);

if (rand.NextUInt() != sought) {

continue;

}

for (int i = 0; i < 5; i++) {

rand.NextUInt();

}

for (int i = 0; i < 5; i++) {

cout << std::hex << rand.NextUInt() << ' ';

}

return 0;

}

}

Just good, plain, old brute force.