CS4460 Code and Material - Final Project

Group Members:

- Esperanza Hauptman - A02391616
- Mark McKee - A02299318
- Jacob MacInnes - A02217700

Introduction: This document records our code and material used for our Final Project.

Server: https://ejmcs4460ususpring24.ctfd.io/

Code and Material:

Enclosed with our assignment are code and other materials related to our CTF server.

These have been separated into Code and SupportingFiles folders.

Code folder contents include:

1. ctf_pcap.py for the Wireshark challenges
2. oneTimePad.py for the One-Time-Pad challenge
3. ProjectComputations.py
4. ProjectPythonCode.py (these last 2 for the RSA, D-H challenges and some simple challenges)
5. flags.c for the trivial C language challenge
6. sudoChallenge1.html for the sudo Linux challenge

SupportingFiles folder contents include:

1. Our presentation slides, in a pdf file.
2. Flags.xlx, an Excel file listing our flags.
3. 4 Wireshark capture files.
4. 2 text files, which were linked in 2 of Mark's challenges.
5. 2 pdf files, which were linked in 2 other of Mark's challenges.

Notes:

1. 1 large number is a flag, but is listed in the Excel file with quotation marks so Excel doesn't cut it off with an exponent.
2. We have reproduced below the information about our challenges that were included with our Final report.

Actual Challenges:

Mark:

Easy challenges:
These include the simple practice, simple linux, ASCII characters, Caesar cipher, and C programming challenges. The simple practice and simple linux challenges are trivial, and were the first written to test the CTFd platform. The ASCII character challenge is a trivial conversion, and the Caesar cipher challenge is a trivial alphabetic permutation (rotation), both with a supplied website. The C programming challenge is essentially trivial, and was written for (beginning) practice in C. (Not all team members had seen C/C++ before this class.) Just execute the C file, and follow directions.

Easy/medium challenges (to reinforce class material):
There are 2 HOTP-HMAC computations that are similar to the example from class. A pdf file is supplied, in case of errors. There are three Wi-FI protocol/specification challenges that highlight three specific improvements in Wi-Fi protocols that were highlighted in the course. There is one SQL query challenge asking to complete a simple query, similar to examples in class.

Real website challenges:
There are 4 real website challenges, asking a user to visit a real, commercially hosted website owned by one team member. The site itself is basically trivial. (It was not some time ago.) The point of these challenges is to find obvious and hidden flags within the website. No hacking is needed; just inspect the related website files: HTML, JavaScript and CSS. The idea for these came a while ago, with the possibility to extend to something more complicated later, which did not occur. No serious hacking is necessary.

Encryption challenges:
There are 2 Diffie-Hellman and 3 RSA challenges.
The first D-H challenge gives a prime and a primitive root. Alice and Bob know both of these. They each pick an integer (exponent, which the other does not know) and exchange pertinent data. The user must find the secret shared key. The second D-H challenge is a coding challenge, and asks the user to write a python program to compute the smallest primitive root of a given prime.
All RSA challenges set up an RSA encryption scheme, where n = p*q, with p and q odd primes. A public key (n, e) is also given. In all problems, the user must find a solution for d which is a private key (n, d). (In two of these, the smallest d is asked for.) The medium problem involves decrypting a text message that was converted to an ASCII list, and the list was encrypted. The easy and hard RSA problems essentially only ask to find the private key. However, in the hard RSA problem, two primes are given, both larger than 1 million. We encourage the user to write his/her own python code to solve these, instead of looking up some modular arithmetic converters online.

Prime challenges:
As mentioned above, these were included (as a side theoretical interest) due to the requirement of very large prime numbers for RSA. The three challenges all ask questions related to the "distribution" of prime numbers, which is more interesting when the primes get larger. The included pdf file (for all three challenges) will lead the user to references for solutions. These solutions offer a glimpse into how prime numbers are distributed. For example, after reviewing the Prime Number Theorem, one finds the probability of selecting an integer at random (uniformly) from the integers 1, 2, 3, … N, and that integer being prime is about 1/ln(N).

Esperanza

Networking challenges:
All .pcap files are included with the code and material for viewing. Some code that was used to create the challenges will be available as well.

The first challenge is a simple exercise designed to get the user used to Wireshark. Using the filters, the user can find a TCP packet. From there, if they view the content of the packet they can easily find the flag.

The next challenge encourages the user to pay attention to the timing of network traffic. They are required to find all the timestamps of specific packets and add them together to get the flag.

The third challenge illustrates how XOR encryption works, which is expanded upon even more in the One-Time Pad challenge. It is very simple to understand as a beginner as it only requires one key to decrypt. The real difficulty is in using the filters. The user is given an initial hint. However, that hint is in hexadecimal when used in Wireshark and the user needs to understand that. From there the correct packet can be found, and then using the key they can decrypt the messages, however they need to take care which base they are using for decryption.

The last challenge shows how to find messages spread throughout ICMP requests. The user can find the specific ICMP request using a filter, or they could possibly do it by hand, it would be slow though. Once the correct ICMP request is found, the user simply needs to put the messages together. In the code included you will notice that the last fragment was not included. That was purposeful as all fragmented ICMP requests have a "complete" request which displays the entire message. This would defeat the whole purpose of the challenge if it was included.

Jacob

Cracking One-Time Pad:
Jacob's first challenge involves cracking a one-time pad, adapted from an exercise from the textbook *Introduction to Cryptography with Coding Theory* cited in the challenge description. In this challenge, three messages have been intercepted with the same one-time pad. By performing an XOR operation on each pair of messages, the key can be eliminated entirely.

The challenger is presented with the XOR of each pair and then challenged to recover the key hidden in one of the original ASCII messages.

Sudo Challenges:
In Linux, regular users can be authorized to perform whitelisted operations as root using sudo. However, in practice, unless every command, flag, and operand is explicitly spelled out (and often even then), granting sudo permissions to a user can have unintended consequences and requires trust that the user is not going to abuse those privileges. In this challenge, the user has been given permission to cat files in a specific directory but because these permissions are specified using a wildcard, it is possible for the user to view the contents of a flags directory instead.

The console is implemented in JavaScript. Originally, another Docker container was going to be used, but had to be cut out during the transition to a hosted instance for time reasons.