**CS 444/544 Spring 2018 Lab 2**

Lab 2 is due by 11:59pm on Tuesday, March 13th. Don't plan on any extensions, Labs 3 and 4 will also need a substantial amount of time.

Submit your lab as an email to crandall@cs.unm.edu. You must include two things in the email, or your lab will not be graded:

- The plaintext message that corresponds to the ciphertext for the keyID assigned to you specifically. Send this in the body of the email, with characters that do not display readily in email bodies escaped, *e.g.*, "\t" for tab, "\n" for newline, *etc.* There shouldn't be any such characters, but if there are escape them. Don't worry too much about trailing spaces or padding.
- A gzipped tar ball of your source code for carrying out the attack. The extension should be *.tgz and the `file` command in Linux should report the format of the file as "`gzip compressed data`". Your tar ball should be no more than 1MB. Your source code need not run, but I reserve the right to call you into my office to demonstrate it for me.

Lab 2 is worth 100 points, all or nothing. If you fail to include your source code as a tar ball, or your plaintext does not match the plaintext that was encrypted in the PCAP specifically assigned to you as a student, then you will get 0 points. I won't grade your source code for style, *etc.*, and I won't attempt to run it. The source code is a fallback if I suspect any students of cheating.

You are expected to do your own work. From writing the source code to carrying out the attack, it should be an individual effort carried out only by you. Any instance of not doing your own work will be considered cheating. You are encouraged to discuss the assignment with your classmates at a high level. Exchanging publicly available source code that existed before the assignment was assigned, and thoughts about approaches to specific problems is okay. It is also okay to go through attack examples together at a high level (*e.g.*, pencil and paper), but do not share any concrete info (such as a network capture or source code) about any specific attack. As a reminder of the course policy, if you cheat on any assignment in this class including this assignment (cheating includes, but is not limited to, representing somebody else's work as your own or fabricating files or text to make it look like you completed the assignment) you will receive an F in the class. Every student's session key and plaintext/ciphertext are unique, and you should not extract the ciphertext or attempt to recover the plaintext of any other student. You may use the Python code (written by Meisam Navaki) provided as part of the tar balls for the assignment however you like without restriction (*e.g.*, you can run your own server locally for testing, you can write your own script that calls rsa-aes-client.py, you can borrow from the provided code or turn it into a library, use it as a starting point for your own code, or whatever).

Your mission, should you choose to accept it (FYI, the drop date with Dean's permission is April 13th) is to extract the ciphertext corresponding to the port that will be assigned to you, and then use a padding oracle attack to recover the plaintext that corresponds to your ciphertext. The server is listening on ports 10000-10100 on yuba.cs.unm.edu (64.106.46.56), but you must only use the port that is assigned to you. You are not limited in the number of queries you can submit to the server, but please be respectful of the resources. I recommend making no more than 1 query every 5 seconds to the server on your port. If you need more than 256 connections for any attempt, then you're doing it wrong. Any kind of denial-of-service attack or other attack on yuba.cs.unm.edu itself is NOT authorized.  If you'd like to try an alternative method to complete the assignment, ask me first.

A unique student number, that corresponds to your port, will be assigned to you.  Do not extract ciphertexts that weren't assigned to you or connect to ports that weren't assigned to you.  You should always connect to port 10000 plus your student number.  For example, if you are student number 55, you will always only connect to port 10055 on yuba.  Connecting to ports not assigned to you will be considered cheating (because you're interfering with other students' ability to complete the assignment and/or trying to complete their assignment for them, probably).  Extracting ciphertexts, or asking any oracle for information about the corresponding plaintext , for any ciphertext not assigned to you will also be considered cheating (because there is no reason to do this unless you are doing someone else's assignment). Honest mistakes will be forgiven, but still be very careful to only work with your own ciphertext and your own port.  Do not share your assigned ciphertext or plaintext with anyone else.  I reserve the right to record all network traffic to and from yuba to catch and build a case against any cheaters.

Two tarballs will be provided.  One will help you to set up your own server for local testing.  The other is a client that includes the public key that corresponds to the private key yuba uses.  There will also be a directory with packet captures, only one of which you should download (based on your student number).  We will discuss how to carry out the attack in class, and I'll send out a sample chapter and/or a paper about the attack.

Basically, every PCAP is a connection I made to the server on a given port.  Each connection used a randomly generated AES key to encrypt a message.  That AES key was encrypted using yuba's private key, and the message and encrypted key were sent to yuba.  The server is a capitalization service, and you have the source code for the client and server.  You also have the RSA public key I used to encrypt the AES session key.  The only thing you lack is the RSA private key, which only exists on the yuba server.  Your mission is to recover the message.  You'll recover the AES session key in the process of doing so, but you will probably never know the RSA private key.  In fact, you don't need it to recover the plaintext.  You'll do a padding oracle attack to recover the RSA plaintext as it is leaked one bit at a time by the server, and the RSA plaintext is the AES session key.

To find out your student number, retrieve this file from any CS Dept. machine with the NFS mount (b146, moons, trucks, etc.):

/nfs/faculty/crandall/Public/secprivspring18/studentnumbersandports.pdf

Once you've recovered the plaintext message, you can check it by seeing if it is a substring of this file:

http://gutenberg.net.au/ebooks01/0100021.txt

All plaintexts were extracted from that file.  The assignment of plaintexts to students is completely random.  I tried to filter out inappropriate plaintexts, but I may have missed one.  (*1984* is a racier book than I remembered.)

Notes:

- The ciphertext assigned to you is the one that you'll extract from the PCAP file for your port number.  So if your student number is 55, your port is 10055 and the PCAP you'll download is 10055.pcap.
- There are easy ways to extract the ciphertext with Wireshark.  You can do C Arrays and just

copy the client part, or do hex and use vim to fix it up.

- I **strongly** recommend that you set up your own server to develop your attack source code, and then once you've tested it against your own encrypted messages *then and only then* start making queries against yuba (on the port assigned to you).
- My suggestion would be to email secpriv-chat@cs.unm.edu with questions about the lab. The TA and I will both see anything posted to that list, as will your classmates, so your chances of getting an answer quickly are better. The TA is primarily setting up labs this semester, not holding office hours. But since she'll have completed Lab 2 by the time we assign it she'll be able to answer questions posted to secpriv-chat@cs.unm.edu. Do not email her directly, but feel free to email me directly.
- Feel free to use ports 10070 through 10100 if you need to (*e.g.*, if your port freezes up and doesn't come back), but use them at your own risk since other students may be using them. Do not use port 10000, that port is reserved for the TA.
- I recommend using exception handling to keep trying to test bits, and test them again if there is some kind of server error. The attack can *in theory* be done with 128 queries, but it may be best to make sure a bit is correct before trying to move on, and sometimes you'll attempt to connect to the server and get a disconnection error, so you'll end up making more than 128 queries and making more attempts than you have successes at querying (probably).
- **Start early.** The attack seems simple and coding up the logic for it isn't too bad, but getting the socket stuff right and debugging you code when there's encryption involved can lead to som epretty hairy bugs. If you get stuck, ask for help. But be sure to **start early** and get past those bugs sooner rather than later.