14-741/18-631: Homework 3

Due: Monday November 12, 2018 at 2:30pm Eastern

Name:

Andrew ID:

Scores

Problem 1 (20 pts max):

Problem 2 (10 pts max):

Problem 3 (10 pts max):

Problem 4 (15 pts max):

Problem 5 (20 pts max):

Problem 6 (10 pts max):

Problem 7 (15 pts max):

Total (100 pts max):

Guidelines

- Be neat and concise in your explanations.
- Start each problem on a new page. You will need to map the sections of your PDF to problems in Gradescope. If you do not attempt a problem, map a page to it with a note saying you didn't complete it.
- Please check your English. You won't be penalized for using incorrect grammar, but you will get penalized if we can't understand what you are writing.
- Proofs (including mathematical proofs) get full credit. Statements without proof or argumentation get no credit.
- There is an old saying from one of my math teachers in college: "In math, anything that is only partially right is totally wrong." While I am not as loathe to give partial credit, please check your derivations.
- **This is not a group assignment. Feel free to discuss the assignment in general terms with other people, but the answers must be your own.** Our academic integrity policy strictly follows the current INI Student Handbook `http://www.ini.cmu.edu/current_students/handbook/`, section IV-C.
- Write a report using your favorite editor **Only PDF submissions will be graded.**

- Submit to **Gradescope** a PDF file containing your explanations and your code files before the due date/time. Late submissions incur penalties as described on the syllabus (first you use up grace credits, then you lose points).
- Post any clarifications or questions regarding this homework in Piazza.
- Good luck!

1 Running Kerberos and Wireshark (20 pts)

Kerberos is slightly more complicated than what we discussed in the lecture. While the steps were overall correct, we only presented a simplified version of the protocol. In Kerberos version 4, a client C, an authentication server AS (trusted), a ticket granting server TGS (which is often the same machine as AS but doesn't have to be), and a server V are involved. C wants to access a service on V. The following exchange takes place:

1. $C \rightarrow AS$: ID_C, ID_{TGS}, TS_1
2. $AS \rightarrow C$: $\{K_{C,TGS}, ID_{TGS}, TS_2, L_2, Ticket_{TGS}\}_{K_C}$
3. $C \rightarrow TGS$: $ID_V, Ticket_{TGS}, Authenticator_C$
4. $TGS \rightarrow C$: $\{K_{C,V}, ID_V, TS_4, Ticket_V\}_{K_{C,TGS}}$
5. $C \rightarrow V$: $Ticket_V, Authenticator'_C$
6. (optional) $V \rightarrow C$: $\{TS_5 + 1\}_{K_{C,V}}$

where

- $Ticket_{TGS} = \{K_{C,TGS}, ID_C, AD_C, ID_{TGS}, TS_2, L_2\}_{K_{TGS}}$
- $Ticket_V = \{K_{C,V}, ID_C, AD_C, ID_V, TS_4, L_4\}_{K_V}$
- $Authenticator_C = \{ID_C, AD_C, TS_3\}_{K_{C,TGS}}$
- $Authenticator'_C = \{ID_C, AD_C, TS_5\}_{K_{C,V}}$

and ID_x is the identity of x, AD_y is the authentication domain of y. TS and L denote timestamps and lifetimes, respectively.

The purpose of this exercise is to observe in practice the establishment of a Kerberos connection. You can use the Ubuntu virtual machine we provide by downloading it from Canvas. *Please note that the package is about 1.5 GB, so make sure to download it outside of peak hours. Use a wired connection as much as possible (it is faster anyway).* Once downloaded, the VM is in `tar.bz2` format. This can readily be unarchived on MacOS and most UNIX variants (FreeBSD, Linux). For Windows, you likely will need 7zip (`http://www.7-zip.org/`) to unarchive it.

Alternatively, you can install the necessary packages on your own machine. You will need Wireshark, a Kerberos client that can be accessed from the command line, and support for the OpenAFS filesystem (`http://www.openafs.org/`). On the VM we provide, the login is `user`, password `user`, and that user is provided `root`-equivalent privileges via `sudo`.

1. Why is the last step (Step 6) optional?
2. From the command line, run `kdestroy` to remove any stale tickets. Then, run Wireshark, and start a packet capture. While the capture is running, contact the Carnegie Mellon authentication server by executing the following command line: `kinit` [your andrew id]`@ANDREW.CMU.EDU`. (Note that Kerberos is case-sensitive.)

   In Wireshark, filtering out any non-Kerberos packets from the capture. In this trace, identify the packet corresponding to the request to the server, and the response from the server. Answer the following questions, justifying your answers with screen captures from Wireshark (highlight the important part of the screen capture):

(a) Which version of Kerberos are you running?

(b) What is the server name? (AS)

(c) What is the client name? (C)

(d) What encryption method is being used?

(e) What is the encrypted value of the ticket $Ticket_{TGS}$ in the lecture notes (use only the first 8 hexadecimal digits)?

3. For the `klist`:

   (a) Provide the output of `klist`.

   (b) How many tickets are currently valid?

   (c) For how long are they valid?

   (d) Who are the principals involved?

4. Destroy all your tickets (`kdestroy`) and make sure that AFS is not running: (`sudo /etc/init.d/openafs-client force-stop`). Now, while running a Wireshark capture, start AFS and complete an AFS operation. This can be done with the following commands:

   - `sudo /etc/init.d/openafs-client force-start`.
   - `kinit` [your andrew id]`@ANDREW.CMU.EDU`.
   - `touch /afs/andrew.cmu.edu/usr/`[your andrew id]`/14741-test`.
   - `rm /afs/andrew.cmu.edu/usr/`[your andrew id]`/14741-test`.
   - `sudo /etc/init.d/openafs-client force-stop`.

   Answer the following questions:

   (a) Show the four Kerberos messages that preside over the establishment of the AFS connection.

   (b) What is the name of the AFS server? (V)

   (c) Show that the ticket ($Ticket_{TGS}$) the authentication server gave you is sent to the ticket granting server.

   (d) Identify the ticket that the TGS returned to you ($Ticket_V$), and show that it is sent to the AFS server when you are trying to create the file `14741-test`. Show only the first eight hexadecimal digits of the ticket. Hint: the AFS decoder for Wireshark is not the greatest thing on Earth, and your Kerberos ticket may be buried a bit. Look carefully!

   (e) Is mutual authentication used?

5. For this `klist`

   (a) Once again, provide the output of `klist`.

   (b) How many tickets are currently valid?

   (c) For how long are they valid?

   (d) Who are the principals involved?

2 Decrypting SSL traffic (10 pts)

The file picoctf_ssl_flag1.pcapng contains a capture of encrypted traffic between a client machine and picoctf.com, a webserver running HTTPS.

1. Using wireshark, decrypt the traffic and tell us what the 14741 flag is. The file premaster.txt will help.
2. What environment variable was set to cause Chrome to generate premaster.txt?
3. If you can decrypt a file containing this HTTPS traffic, why is web browsing with HTTPS still secure? Hint: Google "Wireshark decrypt ssl premaster.txt"

3 Tor Hidden Service (10 pts)

There is a Tor hidden service at `ngxgq6h3iub7uaxu.onion`. Connect to it. (Visit https://www.torproject.org/projects/torbrowser.html.en for information on how to get a Tor Browser). Unfortunately this has been misconfigured and is leaking information.

1. What version of the Linux kernel is it running?
2. What is its hostname? Maybe the key when you visit gives you a hint of what software is running and how you could find "info" about it.

4 Alice and Bob, part I (15 pts)

Alice and Bob initially share the secret key K_{AB}. After every communication session, they update their shared secret key through a publicly known cryptographically secure hash function H as follows: $K'_{AB} = H(K_{AB})$, and they discard the previous key K_{AB}. For example, an interaction may look as follows:

$A \to B:$ $A, \{M_1, H(M_1)\}_{K_{AB}}$
$B \to A:$ $\{M_2, H(M_2)\}_{K_{AB}}$
$A:$ $K'_{AB} = H(K_{AB})$
$B:$ $K'_{AB} = H(K_{AB})$

1. If an attacker gains access to key K'_{AB}, is the previous session (i.e., messages M_1 and M_2) still secret? Explain your answer.

2. If an attacker gains access to key K_{AB}, are later sessions still secret? **Explain your answer**.

3. Alice and Bob share the secret key K_{AB}. g and p are publicly-known quantities. They use the following protocol to set up a session:

   $A:$ $\alpha \leftarrow^R \{0,1\}^{160}$ (in plain English, α is a 160-bit random number)
   $A \to B:$ $A, \{g^\alpha \bmod p\}_{K_{AB}}$
   $B:$ $\beta \leftarrow^R \{0,1\}^{160}$ (in plain English, β is a 160-bit random number)
   $B:$ $K = (g^\alpha \bmod p)^\beta \bmod p$
   $B \to A:$ $A, \{g^\beta \bmod p\}_{K_{AB}}$
   $A:$ $K = (g^\beta \bmod p)^\alpha \bmod p$
   $A \to B:$ $A, \{M_1, H(M_1)\}_K$
   $B \to A:$ $\{M_2, H(M_2)\}_K$

   An attacker recorded all messages in this session and later gains access to key K_{AB}. Are the messages M_1 and M_2 still secret? **Explain your answer**.

4. After an attacker gains access to key K_{AB}, are later sessions still secret? **Explain your answer**.

5 Alice and Bob, part II (20 pts)

Now, Bob wants to work for Alice's security consulting company. Alice insists that Bob proves his worth by designing a simple secure communication protocol before she agrees to hire him. The protocol has to 1) be based on public key cryptography, and 2) be able to transmit and acknowledge receipt of any message X securely between two parties A and B.

Bob comes up with the following protocol for a message X:

1. A sends $\{A, X\}_{KU_B}$ to B, where KU_B is B's public key.
2. B in turn acknowledges receipt by sending $\{B, X\}_{KU_A}$ to A, where KU_A is A's public key.

Alice is not impressed. She says the protocol is unnecessarily complicated, and that the following protocol would produce the *exact same level of security* as Bob's protocol while being simpler:

1. A sends $\{X\}_{KU_B}$ to B, where KU_B is B's public key.
2. B in turn acknowledges receipt by sending $\{X\}_{KU_A}$ to A, where KU_A is A's public key.

Questions:

1. Which security property/ies Bob's protocol enforces? (Hint: review lecture 01-03).
2. Show that Alice is wrong – in that one of the security properties Bob's protocol enforces is not maintained anymore.
3. Bob's protocol unfortunately has a major problem: It is vulnerable to a replay attack in case the same message X is repeated over time. Enhance the protocol proposed by Bob to prevent this attack.
4. Enhance the protocol proposed by Bob to provide the freshness property.

6 SQL Injection (10 points)

Solve the SQL Injection problem on the 14741 CTF Server (same as HW 2). Submit a writeup containining:

1. CTF username
2. which fields are injectable? which not? (prove from code)
3. explanation of vulnerability
4. steps taken
5. flag
6. code (with comments!) you used, if any

7 Blind SQL Injection (15 points)

Solve the SQL Injection 2 problem on the 14741 CTF Server. Submit a writeup containining:

1. CTF username
2. which fields are injectable? which not? (prove from code)
3. explanation of vulnerability. How is this different from SQL Injection 1?
4. steps taken
5. flag
6. code (with comments!), if any