EECE458/ECEG758 Cybersecurity System

Homework 2

Service Discovery with Amazon Elastic Compute Cloud

(200 points)

Oct 11th, 2015

Project Objective

- Master socket programming and understand wireless communications
- Getting familiar with application-layer network protocol design and development
- Experience with Amazon EC2 cloud and understand public-key cryptosystem
- Understand network security in cloud computing environments

Project Description

Service discovery in ubiquitous and mobile computing environments is a network protocol that needs to be designed with security awareness. It is a network protocol for advertisement and discovery of network services. The advertisement and discovery of network service can utilize cryptography to ensure the secure communication. A simple service discovery scenario is explained here. For example, when you enter a room for a meeting, you would like to know what available services inside the room as well as some other rooms inside this building. Your mobile phone may request the directory server which returns a list of service devices, such as printers, scanners, and projectors. The service devices' detail information will also be provided to the user, such as location, projector's resolution, configuration, etc. In this case, you can choose one projector to use if you are will have a presentation during the meeting.

In a service discovery protocol, three parties will be involved. They are **client/users**, **service devices** and **directory**. Clients are the users who want to discover services. Service devices are services that available in the network. Directory is a server that maintains a list of services and their status, such as available, shut down, etc. In this case, directory can automatically deactivate some service devices if the service devices fail to register their status in a fix amount of time.

In this project, you are asked to implement a simple service discovery protocol using Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 is a commercial compute cloud service that can be accessed by users since Aug 2006. Users can request, monitor, and mange any number of virtual machine instances with customized operating system environments. It allows user to pay the time that is used to the run web hosting, distributed applications, scientific simulations on EC2 instances.

Implementation Details

There are two stages in the implementation of service discovery protocol.

1) Registration stage. The **service devices** send registration information to the **directory** to register their services. **Directory** returns acknowledgement to **service devices** that they register successfully or not. Table 1 lists the requirement of database in the directory that stores information of service devices.

2) Discovery stage. The **clients** send request to the **directory** and to ask the current available service

devices by "Service Name" (Explained in Table 1). The **directory** replies a list of service devices that matches the "Service Name" to **clients**, with service device information (9 fields in Table 1).

3) Invocation stage. After client gets the service device information, it sends a message to the **service devices** to ask for using its devices. The devices grants authority to the clients by replying an acknowledge message.

Implementation Requirements & Grading Policy

1) You should write three separate programs to implement client, service device and directory separately and use Amazon EC2 cloud to implement it. **(10 points)**
2) You should implement three stages: registration stage (**50 points**), discovery stage (**50 points**) and invocation stage (**20 points)**
3) All communications are handled using transport layer protocol TCP.
4) You should register at least three service devices to the directory. Three service devices can run on five different Amazon cloud instance (Ubuntu). One service is "scanner". The other two are both "projector". The service information is made up by you and should be valid. **(20 points)**
5) The service device information on directory should be exactly as Table 1. **(10 points)**
6) At least two clients are sending request to the directory at the same time. Total there are three services registered in directory as said in step 5. In this case, the directory should reply at least 2 available services. **(20 points)**
7) User friendly. On three parties, client, service, and directory, message should be prompted to the ask whether to continue the next step. For instance, ask client: "Do you want to send request for service discovery now?" (**10 points**)
8) Implement several functions as explained below. Add comments on your code. (**10 points**)
9) Bonus. One of the client program can be running on mobile devices (Android or iOS) during demonstration (30 points)
10) Bonus. Implement public-key cryptosystem for communication in any of the two parties. (50 points)
11) Working code. Compile errors or linking errors will receive zero.
12) This is an individual project. You are encouraged to discuss with your classmates about the implementation but you should write your own code.

Deliverables Requirement

Tar all the source code files including the header files and the makefile in a single tar file and submit via the the submission page on Moodle. Note that you can only submit one time. Here is a simple example on how to use the Unix command tar (assuming all your files related to this projects are under one directory, say project2). To tar all the files:

```
tar -cvf proj2.tar *
To check the contents in the tar file:
tar -tvf proj2.tar
To untar a tar to get all files in the tar file:
tar -xvf proj2.tar
```

At the beginning of all your source code files, you must include the following information:

Course: EECE458/ECEG758
Semester: Fall 2015

Name: (Your name)

Table 1: Service Device Information on Directory

No.	Attributes	Description
1	Service ID	Unique ID for every service. Values: 1 to 1000
2	Service Name	A name that briefly describes the service. Values: (String)
3	Service Type ID	Classify services as several types. Values: (String)
4	IP Address	Eg. 192.168.10.2
5	Port	Service ports during communication. Values: 0 to 65535
6	Time Stamp	Time when service registers to registry. Values: eg. 2011-7-28 23:7:19
7	Service Status	Values: 0 (soft state), or 1 (hard state).
8	Life Span	If service status is soft, it will announce its lifespan to registry. Values: eg. 300 seconds. However, if it is hard state, do not need lifespan.
9	Context Information	Describe the usage of the service. Its location and some specific restrictions. Value: (String)

Appendix**:** Basic function calls of client, service, and directory

Client Program:

Function: client_query(char *dir_ip_str, int dir_port)

- Parameters: char *dir_ip_str: the IP address of directory.
  int dir_port: the listening port of directory.
- Description: The client program calls the function to inquire the service in the directory.

Function: connect_to_service(char *svc_ip_str, int svc_port)

- Parameters: char *svc_ip_str: the IP address of service.
  int svc_port: the listening port number of service.
- Description: The client program calls the function to get client and service connected. It is client invocation. There are mainly two steps of the function. First, it prepares the socket address by using the two parameters. Second, client program connects to the service program.

Function: handle_communication_with_service(int sock)

- Parameter: int sock: the socket descriptor returned by the connection program.
- Description: The function handles the communications between client and service.

Service Program:

Function: register_service (char *direc_ip_str, int direc_port)

- Parameters: char * direc_ip_str: the IP address of directory.
  int direc_port: the listening port of directory.
- Description: The service program calls this function and gets the service registered. The two

parameters are directory program's listening port and IP address.

Function: accept_client_connections( )

- Description: The function is used for accepting client invocations from the client program. And it gets client program and service program connected.

Directory Program:

Function: handle_connection

- Description: The function handles connections from both client program and server program. It will call function <process request> to identify two requests. One is client discovery request, and the other is service registration request.

Function: process_request(char * strbuf)

- Parameter: char * strbuf: a string contains the requests either from client or service.
- Description: The function deals with requests. It splits the string *strbuf into parts, and parses them to recognize them. If it is a request from client discovery, it calls function < do_query_request>. If it is a request from service registration, it calls function <do_register_request>.

Function: do_query_request

- Description: The function gets parts of the request message passed from the parameters. The request message is from client discovery. It consists of the query criterion, for example "service name". Then it looks up the database to find the matched services according to the query criterion.

Function: do_register_request

- Description: The function gets parts of the request message passed from the parameters. The request message is from service registration. It consists of the attributes list of the service. Then it stores the attributes list into the database as records.