UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

24 de agosto, 2018

Transferencia de datos UDP con encriptación

Profesor: Agustín Gonzalez
Integrantes: Cristhian Urra
Franco Quiroga
Italo Muñoz

Resumen

En la actualidad uno de los temas fundamentales en la red es la seguridad. Por más robusta que sea una red en cuanto a velocidad de procesamiento y transmisión, es totalmente inútil en los usos que le damos hoy en día, si no posee un buen sistema de seguridad. En el presente proyecto se procederá a mostrar una manera de enviar paquetes de datos, que junto a un mecanismo de encriptación, permitirá el tráfico seguro de contenido a través del protocolo UDP. Los datos sólo podrán ser descifrados por quien posea la contraseña de 82 caracteres y conozca el valor de una cierta constante.

Introducción

La privacidad es un tema muy importante que debe estar presente como derecho de todo usuario. Muchos son los casos en que información personal es interceptada por terceros, ya sea para robar cuentas de banco, contraseñas de correos, documentos importantes, etc. En cuanto a seguridad y privacidad, UDP no proporciona por sí solo un mecanismo que garantice privacidad ante posibles intercepción de paquetes, sin embargo puede ser utilizado junto a un mecanismo de encriptación a nivel capa de aplicación para lograr seguridad al momento de traspasar información, idea que desarrolla este proyecto. Un programa que permite enviar paquetes de datos encriptados. El envío a cargo de UDP, y la encriptación a cargo de la capa de aplicación. Se ha utilizado python, que pese a su poca complejidad, permitió construir un programa muy completo.

Envío de información a través de UDP

El protocolo UDP (User Datagram Protocol) es un protocolo de envío de información a través de la red. Este protocolo envía los paquetes de datos sin encriptación alguna de estos, por lo que cualquier tercero ajeno a un traspaso de información específico podría interceptar los paquetes para revelar su contenido. Dentro de este marco se decidió ocupar tal protocolo para demostrar que con un simple código podemos hacer traspasos seguros de información en Internet.

Encriptación SDV (Sustitución por Desplazamiento Variable)

La función “encriptar” recibe claveacceso, texto a encriptar y constante. Las que se definirán a continuación

claveacceso

Corresponde al conjunto de caracteres disponibles, los cuales se desordenan de manera aleatoria.

```
claveacceso="klmnEowxCDFGHsgrabcdefijB789[W0,.;:{}XYZ15+*!#$%6hIJRSTK
LUpqyzA]&/(tuvVMNOPQ234)=?"
dic=claveacceso
```

texto

Corresponde al texto (string) que se desea encriptar.

constante

Corresponde al largo (len) del puerto destino. Se utiliza para generar un poco más de aleatoriedad en los mensajes.

Funcion:

```
def encriptar(texto,constante):
    l=list(texto)
    n=0
    j=1
    while n<len(l):
        k=dic.find(l[n])
        if l[n]==" ":
            l[n]=" "
        if l[n]!=" ":
            l[n]=dic[k+int(constante+j)]
        j+=constante
        n+=1
    f="".join(l)
    return f
```

Funcionamiento.

La función recibe el contenido del mensaje, utilizando de ejemplo un string “hola”. Nótese que mayúsculas y minúsculas se consideran caracteres diferentes. Supongamos constante=4

```
>: hola
Tgg;
```

h>T se desplaza 5 unidades

o>g se desplaza 9 unidades

l>g se desplaza 13 unidades

a>; se desplaza 17 unidades

Para desencriptar, se procede de manera inversa. Solo será posible teniendo “claveacceso” y conociendo el puerto UDP.

Una de las ventajas de este método de encriptación es la dificultad de descifrarlo con el método de “la letra más repetida”. Es un método que estudia las letras que más se utilizan en determinado idioma. No funciona con esta encriptación puesto que por ejemplo al encriptar la palabra “aaaaa” se obtiene “f7W;X”, que no establece una asociación lineal entre el texto plano y su encriptación.

Veamos otros ejemplos:

```
>: Elo 322
DFc adj
>: Federico Santa Maria
rW.X}%%Y HzRYO Tnwbr
>: Agustin Gonzalez
tj?0w%. 6$6[vp?!
```

Función enviar y función recibir[1]

La función enviar recibe tres parámetros, ip del destino, puerto del destino y mensaje a enviar, con esto es capaz de enviar el mensaje al destino final.

```
def enviar(ip,puertodestino,msg):
    sock = socket.socket(socket.AF_INET, # Internet
                         socket.SOCK_DGRAM) # UDP
    sock.sendto(msg, (ip, puertodestino))
```

[1] Basado en ppt visto en clases. http://profesores.elo.utfsm.cl/~agv/elo322/1s18/lectures/Apps_2.7.ppt

La función recibir no requiere parámetros, abre un socket y recibe los paquetes que se envían a través de este.

```
def recibir():
    sock = socket.socket(socket.AF_INET, # Internet
                         socket.SOCK_DGRAM) # UDP
    sock.bind(("", puertoorigen))
    usuariodestinatario=""
    while True:
        data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
        f=str(data)
        tx=desencriptar(f,clave)
        print ip,":",tx
```

Conclusiones

El uso del protocolo UDP ha sido fácil de implementar. Se pudo mostrar el funcionamiento de dicho protocolo, que se encarga del transporte de datos. Siendo la razón fundamental de este proyecto, la privacidad, se pudo demostrar que con unas pocas líneas de código se logra crear una un script que permite transmitir información con una encriptación muy fuerte.

Anexos

El codigo python está disponible en https://github.com/cristianurra/encryptadorudp