# DOS

اولا برای هر Socket کاری که می کنیم این است که IP اش را بدست می آوریم و میتوانیم لیست سیاهی از IP ها داشته باشیم:

```
private static ArrayList<String> blackListOfIPs = new ArrayList<>();
```

و سپس هر سوکتی که سعی کند به هر صورتی پیام اشتباه بدهد را در درون ان می ریزد که در درون متود checkSecurity در کلاس Security به خوبی نمایان است.

و در همان ابتدا اگر IP در لیست سیاه قرار بگیرد دیگر سوکت وصل نمی شود و اگر هم وصل بود باشد قطع میشود که کد های زیر آن را نمایان می کند:

```
while (true) {

    System.out.println("-------------------------------------------------\n" + "Server listening ....");
    clientSocket = serverSocket.accept();

    //if the socket is in the black list we would just return

    if (Security.isInBlackList(clientSocket)) {
        continue;
    }

    System.out.println("client accepted");
    allClientSockets.add(clientSocket);
    DataInputStream dataInputStream = new DataInputStream(new BufferedInputStream(clientSocket.getInputStream()));
    DataOutputStream dataOutputStream = new DataOutputStream(new BufferedOutputStream(clientSocket.getOutputStream()));
    (new ClientHandler( server: this, clientSocket, dataInputStream, dataOutputStream)).start();
}
```

```java
@Override
public void run() {
    while (!Thread.interrupted()) {
        try {
            if (Security.isInBlackList(clientSocket)) {
                throw new Exception("piss off");
            }
            String command = dataInputStream.readUTF();
            System.out.println(command);
            String respond = "";
            synchronized (server) {
                server.clientToServer(command, clientSocket);
                System.out.println(Security.getIP(clientSocket));
                respond = server.serverToClient();
                System.out.println(respond);
            }
            dataOutputStream.writeUTF(respond);
            dataOutputStream.flush();
        } catch (Exception e) {
            System.out.println("something went wrong, connection to client lost :(");
            server.allClientSockets.remove(clientSocket);
            Thread.currentThread().interrupt();
        }
    }
}
```

که تابع استارت ترد در کلاینت هندلر است و اگر در لیست سیاه قرار بگیرد throw exception میشود و کلاینت قطع میشود.

همچنین به کمک تابع زیر می توان IP یک سوکت را چه از نوع IPv4 یا IPv6 باشد، host address ش را که یک استرینگ است برمیگرداند.

```java
public static String getIP(Socket socket) {
    SocketAddress socketAddress = socket.getRemoteSocketAddress();
    if (socketAddress instanceof InetSocketAddress) {
        InetAddress inetAddress = ((InetSocketAddress) socketAddress).getAddress();
        if (inetAddress instanceof Inet4Address)
            return inetAddress.getHostAddress();
        else if (inetAddress instanceof Inet6Address)
            return inetAddress.getHostAddress();
        else
            return "not an ip";
    } else {
        return null;
    }
}
```

و برای هر ip ده بار اخری که متصل شده را زمان ش را نگه می داریم و اگر اختلاف اولی و اخر کمتر از زمانی مشخص (در اینجا 100 میلی ثانیه) شود یعنی به بیش از ده بار خواسته در 100 میلی ثانیه اتصال پیدا کند و در لیست سیاه قرار می گیرد و بلاک می شود و اتصال را قطع می کند.

```java
private static ArrayList<IP> allIPs = new ArrayList<>();
private String ip;
private ArrayList<Long> tenRecentTimes = new ArrayList<>();
private static final int BOUND = 10;

public IP(String ip) {
    this.ip = ip;
    this.tenRecentTimes.add(System.currentTimeMillis());
    allIPs.add(this);
}
```

```
//making sure it doesn't send more than 10 requests in 100 milliseconds

if (IP.addIp(getIP(socket))) {
    blackListOfIPs.add(getIP(socket));
    return;
}


public static boolean addIp(String Ip) {
    IP ip = getIp(Ip);
    if (ip == null) {
        new IP(Ip);
        return false;
    }
    if (ip.ip.equals(Ip)) {
        if (ip.tenRecentTimes.size() >= BOUND) {
            ip.tenRecentTimes.remove( index: 0);
            ip.tenRecentTimes.add(System.currentTimeMillis());
            return ip.tenRecentTimes.get(BOUND-1) - ip.tenRecentTimes.get(0) < 100;
        } else {
            ip.tenRecentTimes.add(System.currentTimeMillis());
        }
    }
    return false;
}
```

همچینین هم کاربر یک فیلد بنام ip دارد که هنگام اولین لاگین گرفته میشود و در دفعه های بعدی اگر کسی سعی کند با ip متفاوت پیامی با آن username بفرستد ما ان را بلاک می کنیم (username متفاوت هم شامل یوزرنیم در توکن و هم یوزرنیم در پیام میشود که ما ابتدا برابری آنها را چک و سپس ip را با سوکت داده شده چک می کنیم.)

```
// making sure it's got one ip

Account account = Storage.getAccountWithUsername(username);
assert account != null;
if (account.getIp() == null) {
    account.setIp(getIP(socket));
} else {
    if (!account.getIp().equals(getIP(socket))) {
        System.out.println("we're under attack by wrong ip");
        blackListOfIPs.add(getIP(socket));
        return;
    }
}
```

و در نهایت هم با توجه به اینکه تخمین زده میشود که چه حدودی کلاینت می توان هندل کرد پس سقفی برای ای پی های مان قرار می دهیم و اگر به ان سقف برسیم به ای پی ها جدید اجازه نمی دهیم که متصل شوند:

```
public static boolean weReachedTheMax() {
    ips.removeAll(blackListOfIPs);
    return ips.size() > (1989 / 23);
}

if (Security.weReachedTheMax()) {
    continue;
}
```

**پس تا اینجا نتیجه می شود:** اولا اگر با ip های محدود سعی کنن پیام های زیاد بدهند که چون هر کدام از ip ها به لیست سیاه منتقل می شوند دیگر کاری نمی توان کرد و اگر هم با ip ها متعدد حمله صورت بگیرد اولا که به اطلاعات شخصی افراد هیچ دسترسی ای نمی توان داشت چون فقط با همان ای پی می توان به اطلاعات دسترسی داشت و چون که تعداد ای پی های مجاز (بجز انها که در بلک لیست هستند) سقف دارد، برای سرور مشکلی پیش نخواهد امد.