SECURITY

MEMBERS

RUSTIN RASOULI

PANIZ HALVACHI

EHSAN MOVAFAGH

ERPSHOP

GROUP 20

مقدمه

در این گزارش قسمت هایی از برنامه که به مبحث امنیت و به هدف جلو گیری از تهدید های گفته شده نوشتیم را بررسی میکنیم. در برنامه ما سعی شده است تا از تعدید های زیر جلوگیری شود:

- REPLAY ATTACKS
- IMPROPER INPUTS
- BROKEN AUTHENTICATION
- BRUTE FORCE ARRACK
- DENIAL OF SERVICE(DOS)

کل حمله های ذکر شده در بالا را کلاسی در برنامه با نام محافظ نظارت و جلوی آن ها را میگیرد. هر پیامی که به سرور میرود پیش از بررسی وارد این کلاس شده و در صورت امن بودن برای انجام عملیات ها وارد سرور میشود.

REPLAY ATTACKS

در این حمله طبق مطالعه ای که داشتیم حمله کننده به صورت شنونده حرف های سرور و کلاینت را میشنود و سپس بعد از مدتی ان ها را دوباره به سرور میفرستد. اگر همواره پیام های یکسانی بین سرور و کلاینت رد و بدل شود دقیقا با تکرار همان پیام ها همه اتفاقات که نباید بیفتد می‌افتد! به طور مثال میتواند بدون اینکه از رمز یا یوزرنیم کاربر اطلاعی داشته باشد صرفا با داشتن پیامی که حاوی آنها است میتواند لاگ این شود.

برای جلوگیری از این حمله در روش های اصلی در رمزنگاری پیام ها همانطور که میدانیم کلیدی بین سرور و کلاینت رد و بدل میشود که پیام ها تنها با این کلید قابل اینکریپت شدن در دو طرف هستند. لذا کافی است کلید فوق به صورت کاملا رندم و هر بار متفاوت باشد در اینصورت وقتی سرور پیام اول از حمله کننده را

میشنود کلید جدید را داده و اما حمله کننده که از آن آگاه نیست پیام های قبل را ادامه داده بدون آنکه بداند دیگر در طرف سرور بررسی نمیشوند چرا که کلید نادرست است. اما چون در برنامه ی ما رمزنگاری خواسته نشده است ما جلوگیری از این جمله را به شکل دیگری پیاده سازی کرده ایم. در برنامه ما مسیجی که از طرف کاربر میآید شامل توکنی است که کاملا تابعیت زمانی دارد و هر بار متفاوت است و در ابتدای هر پیام بررسی میشود که کلید این پیام درست است یا نه. در اینصورت صرفا با دادن تکرار همان پیام ها دوباره به سرور دیگر سرور پیام ها را بررسی نمیکند چرا که کلید توکن پیام های جدید نادرست است.

IMPROPER INPUT

برای رفع این حمله باید اعتبار سنجی در سرور تا جای ممکن بالا باشد و تمام تلاش خود را انجام دهیم تا پیام هایی که با فرمت مشخص شده نیستند در نظر گرفته نشوند. برای جلوگیری از این حمله در برنامه ۳ بررسی اصلی روی پیام پیش از ورود انجام می‌شود. بررسی اول روی اندازه پیام انجام می‌شود که از آنجایی که به صورت تقریبی حجم پیام های رد و بدل شده بین سرور و کاربر را میدانیم اگر اندازه آن بیش از بزرگ باشد نشان دهنده خرابی یا به هدف کند کردن سرور فرستاده شده است. بررسی دوم روی خود فرمت پیام است. از آنجایی که هر پیامی که بین کاربر و سرور فرستاده می‌شود از فرمت خاصی پیروی می‌کند اگر پیامی با این فرمت همخوانی نداشته باشد اصلا وارد سرور نمیشود. مرحله سوم و آخر بررسی اندازه بخش های مختلف پیام است. هر پیام از اجزای مختلفی تشکیل شده است. به مثال یک پیام جهت ثبت نام حاوی نام کاربری رمز و ... است. واضح هست اندازه هر بخش نباید از عددی بیشتر شود چرا که در اینصورت مورد هدف حمله قرار گرفته است لذا در بخش آخر اندازه هر بخش پیام مورد بررسی قرار می‌گیرد. به عنوان مثال بخش آغازین این تابع را آورده‌ایم:

```
// Improper inputs
public void isMessageImproper(Message message) throws Exception {
    if (!checkSizes(message))
        throw new MalMessageException();
    checkMessagePattern(message);
}
```

BROKEN AUTHENTICATION

هدف این حمله دور زدن صفحه ورود کاربران است! این حمله روش های مختلفی دارد و در واقع خود میتواند شامل حمله DOS و ... شود. یکی از روش های رایج در این جمله ناشی از لو رفتن حساب های کاربری کاربران در برنامه و سایت های دیگری است که مهاجم از آنها سو استفاده کرده و سعی بر استفاده از همان داده ها برای ورود به برنامه دیگری میکند. که در واقع تا حدی زیاد این بخش از حمله قابل جلوگیری نیست و در این مسایل صرفا پیام هایی داده به کاربر نظیر لطفا پسفورد ها متفاوت انتخاب کنید و ... داده می‌شود. یکی از کار های بسیار مفید در جلوگیری از این حملات سعی برای پیچیده کردن رمز ها است و در هنگام ثبت نام از رمز های ساده جلوگیری شود. برای این کار ما در برنامه تعداد زیادی از رمز های ضعیف را برداشته و اگر کاربر در هنگام ثبت نام رمز کاربر میان آن ها باشد باید رمز خود را تغییر دهد همینطور رمز هایی کمتر از ۵ حرف مجاز نیستند. همانطور که گفتم این حمله خود شامل حملاتی دیگر است لذا تدابیری که برای جلوگیری از آن ها گرفتیم خود از این حمله هم جلوگیری می‌کنند. تدابیری مانند همان کلید بخش اول یا محدودیت در دفعات اشتباه وارد کردن رمز.

BRUTE FORCE ARRACK & DENIAL OF SERVICE

در هر دوی حمله های فوق هدف حمله به سرور با تعداد بسیار زیادی پیام در مدت ها کوتاه است. در حمله اول حمله در واقع به نوعی در صفحه لاگ این انجام میشود و با تلاش های زیاد در این صفحه مهاجم سعی میکند رمز کاربری را بدست بیاورد و بتواند وارد شود . دیگری حمله ای است که در هر زمانی میتواند اتفاق بیفتد و حمله کننده تعداد بسیاری پیام در زمان کوتاه به سرور میفرستد. در برنامه ما برای رفع حمله اول تلاش های هر ip در محافظ نگهداری میشود و در صورتی که بیش از ۳ بار تلاش ناموفق داشته باشد برای مدت ۳ دقیقه اکانت او مسدود میشود و در صورت تلاش‌های ناموفق بیشتر مدت آن میتواند بیشتر شود. هر ip که به لیست آیپی های سیاه اضافه شود برای مدتی اکانت او مسدود میشود که این مدت بسته به شک تهدیدی که ایجاد کرده است متفاوت است. برای حمله دوم اکتیویتی ها افراد ذخیره میشود و چگالی آن ها در واحد زمان مورد نظارت قرار میگیرد و اگر از بیش از ۱۰۰ پیام در ۱۰ ثانیه تخطی کند اکانت برای مدتی مسدود میشود. همانطور که در خود داک هم آمده است این روش جلوگیری زمانی که مهاجم سعی کند از ip های متفاوتی استفاده کند به مشکل میخورد. این جمله که خود DDOS نام دارد بسیار پیچیده تر است و برا جلوگیری آن روش هایی پیچیده استفاده میشود. اما ما در برنامه خیلی ساده سعی کردیم کمی جلوی این مشکل و مشکل استفاده تعداد زیادی کاربر از سرور و به طوری که سرور به مشکل بخورد شده ایم به این شکل که اگر تعداد کل آیپی هایی متصل به سرور از مقدار مجازی بیشتر شود اجازه اتصال به آیپی های جدید برای مدتی جلوگیری میشود. در زیر تعداد از توابع آمده برای جلوگیری از این حملات را آورده ایم.

```java
// Brute force & Denial of Service (DoS) attack
private void isIpDangerous(String ip) throws Exception{
    DangerousIp dangerousIp = getDangerousIpByIp(ip);
    if (dangerousIp==null)
        return;
    else if (!dangerousIp.isStillDangerous()) {
        dangerousIps.remove(dangerousIp);
        return;
    }
    throw new BlockUserException();
}
```

```java
public void checkLoginAttempts(String ip) {
    int attempts = loginClientsIp.get(ip);
    if (attempts>3) {
        loginClientsIp.remove(ip);
        dangerousIps.add(new DangerousIp(ip, DangerousIpType.LOGIN_DANGER));
    }
}
```

```java
public void checkActivityDensity(String ip) {
    ActivityDensity activityDensity = allClientsIp.get(ip);
    if (activityDensity.getActivityDensity()>100) {
        allClientsIp.remove(ip);
        dangerousIps.add(new DangerousIp(ip, DangerousIpType.EXCESSIVE_ACTIVITY));
    }
}
```