به نام خداوند بخشنده مهربان

مستندات تدابیر امنیتی در فاز سوم پروژه برنامه سازی پیشرفته

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف

دانشجویان:

آرین احدی نیا، روزبه پیراعیادی، سهیل مهدی‌زاده

استاد درس:

علی چکاه

راهنمای پروژه:

صابر ظفرپور

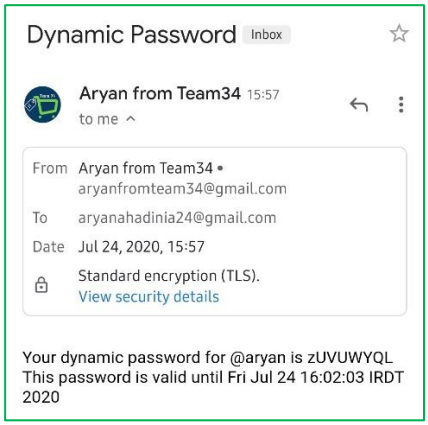
فهرست عناوین

Brute force Attack

مشکل:

می توان با تعداد زیادی تلاش ناموفق، مثلا برای وارد شـدن به حسـاب کاربری، به اطلاعات محرمانه ای نظیر رمز عبور دسـت پیدا کند. توجه کنید که رمز عبور تنها موردی اسـت که در این پروژه، باید محرمانه باقی بماند و این حمله عملا برای سـایر اطلاعات معنی نمی دهد.

راه حل: استفاده از رمز پویا با محدودیت تعداد درخواست:



تصویر ۱

این برنامه از سیستم رمز پویا استفاده می کند. به این صورت که برای هر بار ورود به حسـاب کاربری، یک رمز عبور یکبار مصـرف، به ایمیل کاربر ارسـال میشـود. این ایمیل تحت پروتکل `Standard encryption (TLS)` ارسـال می شـود. تصویر ۱، یک نمونه از ایمیل‌های ارسال شده توسط سرور به کاربر است.

رمز ارسـال شـده، پنج دقیقه اعتبار دارد و **پس از گذشت پنج دقیقه**، منقضـی میشود. همچنین اگر با رمز ارسال شده، عملیات ورود به حساب انجام شود، رمز برای ورود بعدی منقضـی میشـود. در ثانی اگر پیش از منقضـی شـدن رمز، یک بار دیگر به سـرور درخواسـت ارسـال رمز پویا داده شـود، رمز قبلی منقضـی شـده و رمز جدید به ایمیل کاربر ارسـال میشـود. همچنین اگر رمز توسط کاربر اشتباه وارد شود، رمز منقضی میشود.

این سـامانه به گونه ای طراحی شـده اسـت، که یک کاربر، **حداکثر پنج بار در سـاعت** می تواند رمز پویا از سـرور دریافت کند. بنابرین کاربر نمی تواند در طول یک ساعت، **حداکثر پنج تلاش ناموفق** برای ورود به حساب کاربری می تواند داشته باشد

برای پیـاده سـازی این سـامانـه، کلاس `Dynamic Password Manager` در سـرور پروژه در مسـیر `./src/server/security` ایجاد شده است که مسئولیت بررسی صدور مجوز تولید، تولید، ارسال، اعتبار سنجی و منقضی کردن رمز های پویا را بر عهده دارد.

مزایای جانبی:

۱. یکی از مزایای استفاده از رمز پویا این است که رمز کاربر در پایگاه سرور ذخیره نمی شود. بنابرین، بر فرض محال، اگر اطلاعات سرور درز پیدا کند، رمز عبور کاربران فاش نمیشود.
۲. رمز پویا جلوی آسـیب پذیری های ناشـی از `Replay attack` را در ورود به حسـاب کاربری می گیرد. چرا که رمز ارسالی تنها برای یک بار ورود اعتبار دارد.

Replay Attack

مشکل:

این خطر متوجه برنامه است که شخصی پیام های ارسال شده توسط کلاینت را ذخیره و پس از مدتی دوباره به سرور ارسال کند. اگر تدبیری اندیشیده نشود، سرور دوباره همان عملیات ها را انجام می دهد. ممکن است عملیات مورد نظر، عملیات بانکی و انتقال وجه باشد که در این صورت موجبات برداشت غیر مجاز از حساب های بانکی را فراهم می کند.

راه حل: استفاده از token یک بار مصرف

شیوه ای که برای مقابله با `replay attack` ها در نظر گرفته شده به شرح زیر است. هر بار کلاینت درخواستی به سرور ارسال می کند علاوه بر جواب، `token` جدیدی نیز به او تخصیص داده شده و برایش ارسال می شود (برای اختصاص `token` جدید از تابع `renewToken` که در کلاس `Controller` قرار دارد استفاده می شود). درخواست بعدی تنها در صورتی معتبر است که حاوی این `token` باشد. به این ترتیب اگر مهاجمی در بین مسیر به این درخواست ها دسترسی پیدا کند و پس از مدتی مجددا آن ها را به سرور ارسال کند این درخواست ها از سوی سرور غیر معتبر تلقی خواهند شد و بنابراین نه تنها با پاسخی همراه نیستند باعث هیچ کنش اضافی در برنامه نمی شوند . به نوعی می توان گفت `token` های تخصیص داده شده یک بار مصرف هستند و هر `token` تنها برای فرستادن یک درخواست معتبر است. همچنین توجه کنید در صورتی که اتصال کاربر با سرور قطع شود، چه از طریق صحیح آن و بستن برنامه چه قطع اتصال ناگهانی و غیر مترقبه، `token` تخصیص داده شده برای درخواست بعدی، منقضی می شود.

Denial of Service

مشکل:

مشکلات ناشی از این آسیب پذیری را می توان در دو عنوان خلاصه کرد.

۱. یک ip میتواند تعداد دلخواهی اتصال با سرور برنامه برقرار کند. به این طریق با متصل کردن تعداد بالایی socket ، کارایی سرور را کاهش می دهد.
۲. پس از برقراری اتصال، یک کاربر می تواند تعداد با فرکانس بالایی به سرور درخواست ارسال کند و این کار سرور را مشغول می کند.

راه حل: استفاده از شمارنده:

روش حل دو عنوان بالا را به تفکیک توضیح می دهیم:

۱. شمارنده مورد نظر، تعداد اتصال های یک ip به سرور برنامه را ذخیره می کند و تعداد اتصال های یک ip را به سرور به عدد مشخصی محدود می کند. به عنوان مثال در این نسخه برنامه این عدد ۱۰ در نظر گرفته شده است. در صورتی که یک ip درخواست اتصال سوکت ۱۱اُم را به سرور بدهد، IOStreams این سوکت در طرف سرور باز نمی شود و سوکت در طرف سرور بسته خواهد شد.
۲. شمارنده مورد نظر فرکانس ارسال درخواست توسط یک ip را در یک بازه زمانی از چند ثانیه پیش از ارسال درخواست تا لحظه ارسال درخواست محاسبه می کند. به عنوان مثال در این نسخه از برنامه طول این بازه زمانی ۱۰ ثانیه در نظر گرفته شده است. اگر فرکانس ارسال درخواست یک از یک ip از مقداری مشخص (در این نسخه ۳۰ هرتز) تجاوز کند، بلافاصله اتصال سرور با این client قطع می شود و ip ایشان در لیست سیاه موقت قرار می گیرد. سایر client هایی که با استفاده از همین ip به سرور متصل شده‌اند اگر تا پیش از خارج شدن ip از لیست سیاه به server درخواست ارسال کنند، اتصال آنها به سرور بعد از ارسال اولین درخواست قطع میشود. به این طریق از ارسال درخواست پُربسامد به سرور جلوگیری میشود

تمام راه کار های بالا در طرف سرور کلاس DenialOfServiceBlocker در مسیر ./src/server/security پیاده سازی شده است که وظیفه شمارش تعداد اتصال های یک ip، محاسبه بسامد ارسال درخواست توسط ip و صدور مجوز برای رسیدگی به درخواست یک کاربر یا اتصال یک سوکت است.

SQL Injection

مشکل

می توان با فرستادن اطلاعاتی به پایگاه داده SQL به بعضی از اطلاعات حیاتی دست پیدا کرد یا بعضی از اطلاعات را پاک کرد. با توجه به این که در این برنامه از SQLite استفاده شده است، در صورت فراهم نکردن تمهیدات امنیتی لازم ممکن بود برنامه هدف این حملات قرار گیرد.

راه حل: استفاده از PreparedStatement:

در پایگاه داده از توابع مختلفی نظیر createNewTable, add, delete, importAl استفاده می شود که در این بین دو تابع createNewTable, importAll از کاربر ورودی دریافت نمی کنند و وظیفه ی آن ها خولندن اطلاعات برای شروع به کار اولیه سرور است .اما دو تابع دیگر از کاربر اطلاعات دریافت می کنند بنابراین اگر برای ساختن دستور مورد نیاز از به هم وصل کردن رشته ها استفاده شود باعث آسیب پذیری برنامه در مقابل حملات SQL injection می گردد. برای جلوگیری از این موضوع از کلاس PreparedStatement استفاده شده که ورودی ها را به سادگی به دستور اضافه نمی کند بلکه با توابع مختلف نظیر ... setInt, setString, setDouble, آن ها را در جای خود که با علامت "?" مشخص شده است قرار می دهد که این کار جلو ی ورودی های غیر مجاز و خطرناک را می گیرد.

در نهایت توجه داشته باشید که همان طور که پیش تر ذکر شد برنامه از رمز پویا کمک می گیرد و به همین علت رمز عبور کاربر در پایگاه داده نوشته نمی شوند و حتی اگر به طریقی اطلاعات پایگاه داده فاش شود حساب های کاربری همچنان محرمانه باقی می مانند.

Improper Input

مشکل

کاربر می تولندبا وارد کردن اطلاعات نا معتبر در برنامه اختلال ایجاد کند. در ســـاده ترین حللت با وارد کردن اطلاعات طولانی(نظیر آدرس های بسیار بلند، عکس های بسیار حجیم یا پیام ها ی بسیار طولانی در چت روم) حافظه سرور را اشغال کند.

راه حل

کلاس `Validation` در مسیر `src/main/java/server/security/Validation` وظیفه ی اعتبار سنجی تمامی ورودی های کاربر را برعهده دارند . توجه کنید در صـــورتی که کد کلاینت به همین شـــکل باقی بماند نیاز به بعضـــی از این اعتبار ســنجی ها احتیاجی نیســت اما با توجه به این که ما فرض را بر این می گذاریم که ممکن اســت مهاجمی بتواند به کد کلاینت دست یابد و در آن تغییراتی ایجاد کند تمامی ورودی های فرستاده شده به طور کامل در سمت سرور اعتبار سنجی می شوند

Broken Authentication

مشکل:

در این بخش حمله کننده سعی دارد که خودش را به جای فرد دیگری جا بزند و احراز هویت سرور را دور بزند. ضعف های زیر می توانند منجر به این نوع حمله شوند :

۱. استفاده کاربران از رمز های ضعیف و قابل پیشبینی.
۲. در معرض دید بودن Session ID یا همان token فرد لاگین کرده.
۳. منقضی نشدن نشست کاربری فردی که لاگین کرده.

یک حمله کننده می تواند با استفاده از لیستی از رمز های ساده و قابل پیشبینی با حساب کاربری فردی دیگر لاگین کند.(ضعف اول) همچنین اگر بتواند token فردی دیگر را از طریق درخواست های (Requests) فرستاده شده به سرور استخراج کند، میتواند به جای او به سرور درخواست بفرستد.(ضعف دوم) فرض کنید کاربری از کامپیوتر های عمومی استفاده کند و به هر دلیلی (مثل قطع ارتباط اینترنت) عمل Logout را نتواند انجام دهد. حمله کننده میتواند چند ساعت بعد با همان کلاینت به سرور متصل شود. (ضعف سوم)

راه حل:

برای رفع مشکل اول، این برنامه از multi-factor authentication استفاده میکند. به این شکل که پس از تایید شدن نام کاربری، رمز عبور به طور یکبار مصرف سمت سرور تولید شده و از طریق ایمیل به کاربر ارسال میشود. حمله کننده دیگر نمی تواند با روش هایی مثل credential stuffing یا Brute Force اقدام به لاگین کند.

استفاده از token یکبار مصرف مشکل دوم را براحتی بر طرف می کند. فرض کنید هر token فقط برای یک در خواست معتبر باشد. در این صورت دیگر در معرض دید بودن Session ID یا token مشکلی ایجاد نمیکند چون بعد از استفاده نامعتبر می شوند.

سرور برنامه، هر دفعه که درخواست (Request) دریافت میکند، زمان آن را با درخواست های قبلی مقایسه میکند. اگر اختلاف زمانی آن ها از حدی بیشتر بود (به طور مثال ۶۰ دقیقه) به وضعیت آن درخواست رسیدگی نمیشود و اتصال client و server قطع میشود. به این صورت ضعف سوم رفع می شود و دیگر حمله کننده نمی تواند از Session های idle برای دور زدن احراز هویت استفاده کند.

File Encryption

مشکل:

در بخش p2p برنامه در صورتی که فایل بدون رمزگذاری ارسال گردد این احتمال وجود دارد که اشخاصی بتوانند به این اطلاعات دسترسی پیدا کنند.

راه حل:

برای حل این مشکل از کلاس Asymmetric که یک رمز گذاری نامتقارن را پیاده سازی می کند استفاده شده است. به این شکل که هر کلاینتی که شروع به کار می کند به کمک الگوریتم RSA دو کلید متفاوت برای کد گذاری و کد گشایی تولید می کند. کلید مخصوص کد گذاری از طریق شبکه ارسال می شود. توجه کنید که احتمال فاش شدن این کلید به علت ارسال شدن از طریق شبکه کم نیست اما این موضوع اهمیتی ندارد چون ما این کلید را عمومی فرض می کنیم. به عبارتی هر کسی که این کلید را داشته باشد تنها قادر به رمزگذاری خواهد بود نه رمزگشایی.(می توان این کلید عمومی را مانند آدرس پستی کاربر تصور کرد که با در اختیار داشتن آن می توان برای کاربر نامه فرستاد اما نمی توان صندوق پستی او را گشود و به نامه های او دسترسی پیدا کرد) بنابراین فروشنده می تواند به کمک کلید عمومی فایل خود را رمز کند و برای خریدار ارسال کند و تنها کسی که کلید خصوصی و کد گشا را در اختیار دارد خود خریدار است که می تواند به کمک این کلید به محتویات فایل دسترسی پیدا کند.