به نام خداوند بخشنده‌ی مهربان

مستند امنیت پروژه درس برنامه‌سازی پیشرفته
فاز سوم - گروه ۹

آسیب‌پذیری Replay Attacks

می‌دانیم این حمله زمانی رخ می‌دهد که مهاجم و یا attacker، پیام‌های رد و بدل شده بین سرور و کلاینت را برای خود ذخیره کند، سپس پس از مدتی آن‌ها را دوباره به مقصد بفرستد و یا حجم زیادی از این پیام‌ها را از طرف خودش به مقصد ارسال کند؛ حال اگر دریافت کننده پیام، با این پیام‌های تکراری کاری انجام دهد که مطلوب نیست، replay attack رخ داده است.

اقداماتی که برای رفع این آسیب‌پذیری انجام شد:

- به هر درخواست ارسال شده از سوی کلاینت به سرور، برچسب زمانی (timestamp) زده شده است که بعد از مدتی درخواست دیگر valid نباشد. بعد از هر درخواست، هماهنگی زمانی تقریبی برچسب زمانی درخواست با زمان کنونی سیستم، در سرور چک می‌شود.

- در ارتباط با بانک و انجام تراکنش‌ها، فیش ساخته و بلافاصله و در اولین ریکوئست بعد از ساخت فیش، پرداخت می‌شود تا هکر فرصت ارسال دوباره پیامی در این زمینه را نداشته باشد.

آسیب‌پذیری Improper Inputs

می‌دانیم اگر در سمت سرور هیچ اعتبار سنجی‌ای و یا اعتبار سنجی‌ای ضعیف انجام شود، ممکن است سبب متوقف شدن سرور، پرتاب شدن استثنا و یا تغییر در وضعیت سرور شود. هم‌چنین، فرستنده پیام‌ها می‌تواند با نوشتن رشته‌ای بسیار طولانی، حجم بزرگی از حافظه سرور را اشغال کرده و با چند بار انجام پیاپی این کار، سرور را کند کرده و یا از کار بیندازد. در نتیجه اقداماتی جهت کنترل validity ورودی‌ها و هم‌چنین طول آن‌ها انجام شده است. در تمامی کیس‌های زیر، به جای پرتاب اکسپشنی که منجر به توقف فعالیت سرور شود، خطا به درستی handle شده و جواب مناسب برای کلاینت از طریق شبکه ارسال می‌شود.

اقداماتی که برای رفع این آسیب‌پذیری انجام شد:

- هنگام ثبت‌نام کاربر، معتبر بودن ای‌میل و شماره تلفن او را به کمک کلاس Validator، علاوه بر سمت کلاینت که یک‌بار تست می‌شود، سمت سرور هم تست می‌کنیم که اگر شخصی در این بین این پیام را شنود کرد و سپس پیام را تغییر داد و برای سرور مجددا فرستاد، در سرور هم یک‌بار این validation انجام گردد.

- در تمامی اینپوت‌های از سمت کاربر، خالی نبودن، null نبودن و مرتبط با یک شی واقعی بودن (valid بودن) آن‌ها چک می‌شود که چند نمونه از آن‌ها در زیر آورده شده‌اند:

- توکن
- محتوای مربوطه در هنگام درخواست افزودن تبلیغ
- آی‌دی محصول هنگام درخواست حذف آن
- آی‌دی کد تخفیف هنگام درخواست حذف آن
- آی‌دی درخواست‌های به سوی ادمین هنگام درخواست تایید شدن و یا رد شدن یکی از آن‌ها
- آی‌دی حراج هنگام درخواست حذف آن
- اطلاعات کاربری هنگام درخواست حذف یک کاربر از سوی مدیر
- آی‌دی دسته‌بندی هنگام درخواست حذف آن
- آی‌دی مزایده هنگام درخواست دریافت پیام‌های مربوط به آن
- میزان حداقل پول مجاز در کیف پول که توسط مدیر تعیین می‌شود هنگام درخواست ثبت آن (مواردی مثل منفی نبودنش، صحیح/عدد نبودنش و... هم چک می‌شوند)
- میزان درصد کارمزد که توسط مدیر تعیین می‌شود هنگام درخواست ثبت آن (مواردی مثل منفی نبودنش، صد یا بیش‌تر نبودنش، صحیح/عدد نبودنش و... هم چک می‌شوند)
- مبلغ پیشنهادی یک خریدار برای مزایده، که باید از قیمت پیشنهادی قبلی بیش‌تر باشد و...
- ... و بسیاری از موارد دیگر

- طول رشته‌های فرستاده از سوی کاربر هم در سرور چک شده که در حد مطلوب باشد تا باعث شلوغی سرور و شبکه و اشغال منابع نشود. چند نمونه از این چک‌ها در زیر آورده شده‌اند:

- طول معقول و صحیح نام و نام خانوادگی، یوزرنیم، پسورد، ای‌میل، شماره تلفن و...
- طول معقول و صحیح تمامی آی‌دی‌ها (محصول، حراج، کد تخفیف، دسته‌بندی، درخواست، مزایده و...)
- طول معقول و صحیح توکن‌ها

- طول معقول و صحیح تبلیغات
- طول معقول و صحیح رشته‌هایی از ورودی که قرار است به عدد صحیح تبدیل شوند، مانند حداقل پول مجاز در کیف پول و درصد کارمزد
- ... و بسیاری از موارد دیگر

- هم‌چنین، لازم به ذکر است که فیلدهای متنی دیتابیس SQL ای که سرور با آن کار می‌کند، با محدودیت طول رشته تعریف شده‌اند تا از اشغال منابع جلوگیری به‌عمل آید.

- در بسیاری از موارد، ریکوئست ارسالی به سرور از مقادیر مجاز تخطی می‌کند؛ در این موارد لازم است به جای کرش کردن سرور و پرتاب اکسپشن هندل نشده، به درستی هندل شود و تصمیم گرفته شده که جواب wrong-action برای کلاینت ارسال شود. چند نمونه از این موارد در زیر آورده شده‌اند:

- در هنگام ثبت درخواست برای بازبینی مدیر، ممکن‌است درخواست ثبت شده، درخواست valid ای نباشد (نوع درخواست در انواع تعریف شده در داک فاز یک نباشد).

- در هنگام ثبت لاگ، ممکن‌است لاگ ثبت شده، لاگ valid ای نباشد (لاگ خرید یا فروش نباشد).

- ریکوئست ارسالی به سرور ممکن است در ریکوئست‌های پشتیبانی‌شده نباشد (Error ۴۰۴)

- بسیاری از خطاهای دیگر هم‌چون یافته‌نشدن فایل دیتابیس هنگام اجرای اولیه نیز به درستی handle شده‌اند.

- هم‌چنین، به طور کلی، اگر ناخواسته خطایی پیش آمد که در موارد فوق نبود، این خطا موجب توقف سرور نخواهد شد و فقط سرور یک لاگ می‌اندازد و خطا را گزارش می‌کند و ادمین سیستم می‌تواند خطا را بررسی کند.

آسیب‌پذیری SQL Injection

می‌دانیم اگر در سرور به درستی ورودی‌ها را چک نکنیم، ممکن است دستورات SQL در داخل آن‌ها تعبیه شود، مثلا کاربر به جای پسورد وارد کند:

myPassword; DROP TABLE Users;

و اگر این آسیب پذیری را در سرور رفع نمی‌کردیم، دستور فوق سبب از دست رفتن اطلاعات **تمام** کاربران سامانه می‌شد!

اقداماتی که برای رفع این آسیب‌پذیری انجام شد:

- به جای این‌که یک رشته را به کمک String concatenation (عملگر + یا StringBuilder یا...) بسازیم و سپس آن‌را به عنوان پارامتر به executeQuery() یا executeUpdate() دهیم، ابتدا یک شی از کلاس PreparedStatement می‌سازیم که در پکیج java.sql قرار دارد و جلوی حملات SQL Injection را به‌خوبی

می‌گیرد. سپس به کمک فراخوانی متد setString روی آن (که از قبل به آن رشته‌ای داده شده است که در آن به جای مکان‌های ورود، علامت سوال است) و در انتها executeUpdate()، با دیتابیس کار می‌کنیم.

آسیب‌پذیری Brute Force Attack

ممکن است هکری بخواهد برای دور زدن فرمی چون فرم ورود به حساب کاربری، تعداد زیادی تلاش ناموفق کند و سعی کند با حدس زدن، ورودی درست را پیدا کند. اگر جلوی این کار هکر را نگیریم، بالاخره پس از صرف زمانی – شاید طولانی، ولی متناهی – رمز عبور کاربر، هر چه‌قدر هم که دشوار باشد، یافته خواهد شد و در نتیجه اطلاعات حیاتی کاربران به مخاطره خواهند افتاد.

اقداماتی که برای رفع این آسیب‌پذیری انجام شد:

- به‌ازای هر آی‌پی، یک شمارنده قرار داده‌ایم که تعداد لاگین‌های ناموفقی که آن آی‌پی در زمان مشخص انجام می‌دهد را شمرده و آن‌ها را نگاه داشته و در صورت بیش از حد مجاز شدن در زمان مشخص، اجازه ارسال ریکوئست ورود به حساب برای مدتی به آی‌پی داده نمی‌شود (آی‌پی در لیست سیاه (blacklist) موقت قرار می‌گیرد) تا از این اتفاق پیش‌گیری شود که کلاینتی در مدت زمان کوتاه میلیون‌ها پسورد ارسال کند تا بلکه یکی از آن‌ها درست باشد (از کلاس IPRecord بدین منظور استفاده کرده‌ایم).

آسیب‌پذیری Denial of Service (DoS)

ممکن است هکری با یک تکه کد که اتفاقا نوشتن آن بسیار آسان است، تعداد بسیار بالایی درخواست را در مدت زمان کمی به سرور ارسال کند و با توجه به محدود بودن منابع سخت‌افزاری لازم برای پذیرش این حجم از پیام‌ها، ممکن است کل سرور مختل و سرویس از کار بیفتد.

اقداماتی که برای رفع این آسیب‌پذیری انجام شد:

- به‌ازای هر آی‌پی، یک شمارنده قرار داده‌ایم که تعداد درخواست‌هایی که آن آی‌پی در زمان مشخص انجام می‌دهد را شمرده و آن‌ها را نگاه داشته و در صورت بیش از حد مجاز شدن در زمان مشخص، اجازه ارسال درخواست دیگری به آی‌پی داده نمی‌شود (آی‌پی در لیست سیاه (blacklist) قرار می‌گیرد) تا از این آسیب پیش‌گیری شود (از کلاس IPRecord بدین منظور استفاده کرده‌ایم).

البته، همان‌طور که در داک امنیت قید شده است، اگر هکر یا فرد خراب‌کار از تعداد زیادی آی‌پی برای حمله به سرور استفاده کند، این راهکار شکست می‌خورد و حمله از DoS تبدیل به DDoS می‌شود که مخفف Distributed Denial of Service است، که با توجه به این‌که در sheet نمره تنها جلوگیری از DoS - و نه DDoS - قید شده، مقابله با DDoS در سرور انجام نپذیرفته است.

آسیب‌پذیری Broken Authentication

هدف از این حمله، دور زدن احراز هویت سرور است. مطابق با آن‌چه در منابع رسمی گوناگون در زمینه این آسیب‌پذیری منتشر شده، مشکلات عمده زیر توصیف شده که در مقابله با آن‌ها، اقدامات گوناگونی انجام گشته‌اند.

اقداماتی که برای رفع این آسیب‌پذیری انجام شد:

- رمز عبور غیر امن می‌تواند سبب شود attacker بتواند به راحتی احراز هویت سرور را دور بزند؛ در نتیجه هم در سرور و هم در کلاینت، یک الگوریتم Password Strength قرار داده‌ایم و در صورت ضعیف بودن پسورد، اجازه ثبت‌نام به کاربر داده نخواهد شد.

- پسوردهایی که قرار است فرد با آن‌ها ثبت نام کند، در دیتابیسی (قطعا به صورت لوکال ذخیره شده!) شامل هزار رمز عبور پرکاربرد در جهان جست‌وجو می‌شود و تنها در صورتی کاربر مجاز به ثبت‌نام خواهد بود که یکی از پسوردهای موجود در این لیست را انتخاب نکرده باشد. با انجام این کار و جلوگیری از انتخاب پسورد ضعیف و یا تکراری، حملات Brute Force نیز بی‌اثر می‌شوند.

- در منابع متفاوت نوشته شده است که در صورت داشتن سیستم Forget Password ضعیف، احتمال دور زدن آن توسط هکرها وجود دارد. با توجه به عدم وجود این ویژگی در داک پروژه، امکان ندارد سامانه طراحی شده از این نقطه آسیب ببیند.

- توکن‌ها هنگام خروج کاربر از حساب invalidate می‌شوند تا بعدا هکر نتواند به کمک توکنی که به اشتباه هم‌چنان باز مانده، از اکانت فرد دیگری وارد حساب آن فرد شود.

- بسیاری از سامانه‌ها، یک اکانت admin-admin یا admin-۱۲۳ یا موارد مشابه دارند که این رمزهای عبور پیش‌فرض می‌توانند خطر بزرگی برای امنیت سیستم باشند. این پروژه، چنین ویژگی‌ای را (اکانت پیش‌فرض را) دارا نیست، در نتیجه از این نقطه نیز امکان آسیب رساندن به سرور توسط هکرها فراهم نمی‌باشد.