Payment Flaws:
Free Stay Paradise

فهرست

بررسی فرآیند پرداخت

بررسی فرآیند پرداخت از کیف پول

آسیب پذیری Double spending کد تخفیف و کارت هدیه

بررسی فرآیند پرداخت

مشتری	فروشنده	PSP
		
مشتری سفارش خود را ثبت می کند ⟵		
سیستم فاکتور را ایجاد کرده و از مشتری می خواهد که شیوه پرداخت خود را انتخاب کند ⟶		
مشتری پرداخت از طریق درگاه را انتخاب می کند ⟵		
		فروشنده اطلاعات تراکنش (مبلغ و..) را به PSP ارسال می کند ⟵
		PSP در پاسخ به درخواست فروشنده یک توکن به فروشنده باز می گرداند ⟶
فروشنده مشتری را همراه به توکن دریافتی به درگاه پرداخت هدایت می کند ⟶		
مشتری اطلاعات کارت اعتباری خود را وارد می کند ⟵		
		PSP پرداخت را پردازش کرده و اطلاعات تراکنش را به مشتری باز می گرداند ⟶
اطلاعات پرداخت توسط مشتری به فروشنده ارسال می شود ⟵		
		فروشنده برای صحت سنجی و تائید تراکنش اطلاعات تراکنش را به PSP ارسال می کند ⟵
		PSP اطلاعات تراکنش را باز می گرداند ⟶

فروشنده بر اساس اطلاعات دریافتی تصمیم به تائید یا رد پرداخت می گیرد

بررسی فرآیند پرداخت

ثبت سفارش:

مشتری ابتدا سفارش خود را در سیستم ثبت می کند

ایجاد فاکتور و انتخاب شیوه پرداخت:

پس از ثبت سفارش سامانه ابتدا یک فاکتور برای پرداخت مشتری ایجاد می‌کند و سپس از مشتری می خواهد شیوه پرداخت خود را انتخاب کند.(مشتری گزینه پرداخت از طریق درگاه بانکی راانتخاب می کند)

Request

```
POST /pay/ HTTP/1.1
Host: api.vooricon.com
{
        "Order" : "VOR-98212",
        "Method" : "Saman"
}
```

Response

```
Location: : https://sep.shaprak.ir?token=
SwufAKSMKASD8asdjfJADSKADMV
```

ارسال اطلاعات تراکنش به PSP:

فروشنده اطلاعات مربوط به سفارش مشتری را به PSP ارسال می‌کند. و در پاسخ یک توکن دریافت می کند.

Request

```
POST /onlinepg/onlinepg HTTP/1.1
Host: smaple.com
{
    "TerminalId" : "0000"
    "Merchant Secret" : "0000"
     "Amount" : 12000,
    "RedirectUrl" : "http://vri.ir/callback"
}
```

Response

```
{
  "Status" : “Ok"
  "Token" : “SRv21fSATGLCmcGD78yU"
}
```

هدایت به درگاه بانکی :

فروشنده کاربر را همراه با توکن به درگاه پرداخت هدایت می کند.

ارسال اطلاعات کارت اعتباری:

در این مرحله، مشتری اطلاعات کارت اعتباری خود را وارد می کند.

پردازش تراکنش:

PSP اطلاعات کارت اعتباری را دریافت و پرداخت را پردازش می کند پس از انجام پرداخت اطلاعات تراکنش را به مشتری باز می گرداند.

ارسال اطلاعات پرداخت به فروشنده:

اطلاعات مربوط به تراکنش توسط مرورگر مشتری به فروشنده ارسال می کند.

نمونه درخواست :

Request

```
POST /v1/transaction/verify HTTP/1.1
Host: merchant.com
MID=10920870&TerminalId=10920870
&RefNum=GDASGDDSADFDASGDSAG
ASDGASDGS&ResNum=545512&State
=success&Amount=100000
```

Response

```
Location: : /Order/ 545512 "
```

در PSP های مختلف نام پارامترها ممکن است بسته به ارائه‌دهنده خدمات پرداخت (PSP) متفاوت باشد، اما منطق کلی و ساختار اصلی مشابه است. به عنوان نمونه، در درخواست،بالا ResNumنمایانگر شماره فاکتور است که از سوی فروشنده معتبر محسوب می‌شود، و RefNum کد رسید دیجیتال بانکی است که توسط PSP تولید می‌گردد. فروشنده می‌تواند با دریافت و استعلام این کد، اطلاعات پرداخت را بررسی کرده و در صورت صحت، تراکنش را تأیید نماید.

استعلام فروشنده از PSPو تائید پرداخت

فروشنده پس از دریافت اطلاعات تراکنش مشتری رسید دیجیتالی RefNum را در پایگا ه داده خود جستجو می کند. این کار به منظور جلوگیری از Double Spending یا دوبار مصرف شدن یک رسید دیجیتالی است پس از طی این مراحل فرشونده اطلاعات ارسالی کاربر را به سمت PSP ارسال می کند و سپس با دریافت پاسخ و تطابق اطلاعات پرداخت با سفارش کاربر،پرداخت را تائید یا رد می کند.

نمونه درخواستی که فروشنده برای استعلام تراکنش به PSP ارسال می کند

Request

```
POST /ipg/VerifyTransaction  HTTP/1.1
Host: psp.com
{
        "TerminalId"   "0000"
        "Merchant Secret"   "0000"
        "RefNum"   "jJnBmy/IojtTe5ke9U "
}
```

Response

```
{
        "RRN" : "0000"
        "MaskedPan" : “6104****22"
        “Timestamp" : “Timestamp"
        "OrginalAmount" : “1341000"
        "Success" : True
}
```

فروشنده ها پس از دریافت اطلاعات تراکنش، معمولاً به یکی از دو روش زیر اقدام به تأیید پرداخت می‌کند:

- روش اول:در این روش، اگر مقدار پارامتر Success برابر با true باشد، سامانه رسید دیجیتال پرداخت را در پایگاه داده ذخیره کرده و تراکنش را تأیید می‌نماید.
- روش دوم :در این روش، سامانه صرفاً به مقدار Success اکتفا نمی‌کند. پیش از تأیید پرداخت، اطلاعات فاکتور کاربر (مانند شماره فاکتور، مبلغ، و سایر پارامترهای کلیدی) با داده‌های بازگشتی از بانک تطابق داده می‌شود. این روش که ایمن‌تر است، احتمال وقوع خطا یا سوءاستفاده را کاهش می‌دهد.

نکته قابل‌توجه این است که اکثر ارائه‌دهندگان خدمات پرداخت (PSP) با وجود دریافت شماره فاکتور هنگام ایجاد توکن، در زمان پاسخ به استعلام فروشنده، این مقدار را از فروشنده دریافت نمی‌کنند. در این فرآیند، PSPها معمولاً فقط اطلاعات درگاه و رسید دیجیتال پرداخت را دریافت کرده و پاسخ را بر اساس همین اطلاعات بازمی‌گردانند.

اگرچه این روند به‌طور کلی مشکلی ایجاد نمی‌کند، اما اگر PSPها علاوه بر رسید دیجیتال، شماره فاکتور را نیز دریافت و بررسی می‌کردند، می‌توانستند خطاهای توسعه‌دهندگان را به میزان قابل‌توجهی کاهش دهند. شایان ذکر است که از میان PSPها، تنها دو ارائه‌دهنده سداد و به‌پرداخت این نکته مهم را رعایت کرده‌اند، که شایسته تقدیر است.

انصراف از خرید:

در صورتی که پس از هدایت به درگاه بانکی گزینه انصراف از خرید را انتخاب کنید، همچنان اطلاعاتی به سمت فروشنده ارسال می‌شود تا وضعیت فاکتور مشخص شود.

نمونه درخواست و پاسخ

Request

```
POST /v1/transaction/verify HTTP/1.1
Host: merchant.com
MID=10920870&TerminalId=10920870&RefNum=&ResNum=545512&State=CanceledByUser&Amount=100000
```

Response

```
Location: : /Order/ 545512 /fail"
```

آسیب پذیری در بخش انتخاب شیوه پرداخت

در فرآیند پرداخت از مشتری خواسته می شود که شیوه پرداخت را انتخاب کند. و درخواستی مشابه درخواست زیر احتمالا به سمت سرور فروشنده ارسال می شود.

Request

```
POST /pay/ HTTP/1.1
Host: api.vooricon.com
{
        "Order" : "VOR-98212",
        "Method" : "Saman"
}
```

در فرآیند پرداخت از مشتری خواسته می شود که شیوه پرداخت را انتخاب کند. و درخواستی مشابه درخواست بالا به سمت سرور فروشنده ارسال می شود.

چرا بدست آوردن شیوه های معتبر پرداخت بروی یک سامانه اهمیت دارد؟

شاید بطور مستقیم در این بخش آسیب پذیری وجود نداشته باشد ولی پیدا کردن شیوه های مختلف پرداخت یک سامانه می تواند در مراحل بعدی یافتن آسیب پذیری به ما کمک کند.

مثلا ممکن است در پیاده سازی درگاه پرداخت سامان اشتباهی صورت نگرفته باشد ولی در پیاده سازی درگاه بانک ملت فروشنده اشتباه کرده باشد پس به دست آوردن روش های مختلف و معتبری پرداخت یک سامانه علاوه بر اینکه می توانیم بصورت مستقیم آسیب پذیری پیدا کنیم در مراحل بعدی نیز به ما کمک خواهد کرد.

داستان خرید رایگان در یکی از بزرگترین فروشگاه اینترنتی ایران

یکی از آسیب‌پذیری‌هایی که در یکی از برنامه‌های باگ‌بانتی ثبت کرده‌ام، دقیقاً مربوط به همین بخش می‌شد. مهاجم می‌توانست با این آسیب‌پذیری، بدون نیاز به ورود به درگاه بانکی، پرداخت را انجام داده و سفارشات خود را به‌صورت رایگان پرداخت کرده و تحویل بگیرد.

شیوه کار به این صورت بود که مشتری می‌توانست روش‌های مختلفی برای پرداخت، مثل پرداخت از طریق درگاه بانکی، پرداخت از طریق کیف پول و... را انتخاب کند.

پس از اینکه تست‌هایم روی بخش‌های مختلف پرداخت به آسیب‌پذیری ختم نشد، سعی داشتم تا ببینم آیا سامانه از یک PSP دیگر نیز استفاده می‌کند یا خیر؟

ابتدا یک پرداخت کامل را رهگیری کردم. مقدار پارامتر توسط سیستم به کاربر ارائه می‌شد و مقادیر در داخل فایل‌های JS نبود. برای اطمینان، فایل‌های JavaScript را هم یک‌بار بررسی کردم. برای راحتی، کافی بود مقدار پارامتر را در فایل‌های جاوااسکریپت جستجو کنم. مثلاً اگر مقدار پارامتر Method برابر با Samanبود، فقط Saman را در فایل‌های JS جستجو می‌کردم (اگر این مقدار وجود داشت، می‌فهمیدم که باقی مقادیر هم در داخل فایل وجود دارد).

سپس به wayback رفتم تا شاید از طریق بررسی آرشیو چند سال گذشته متوجه شوم که سامانه از چه شیوه های پرداختی و همچنین چه درگاه های استفاده میکرده که بازهم ناکام بود.

پس از آن با تغییر مبلغ سفارش سعی کردم ببینم ایا ممکن است برای سفارش با مبالغ بالاتر از یک درگاه پرداخت دیگر استفاده کرده باشند (عموما در صرافی ها این اتفاق شایع است)

سپس با استفاده از یک wordlist سعی کردم مقدار پارامتر Method را brute force کنم تا شاید مقادیر معتبر را به دست بیاورم

این مورد هم ناکام بود تا اینکه به بخش فروش عمده محصولات این فروشگاه مراجعه کرده ام ابتدا فایلهای js رو مثل مرحله قبل چک کردم و این بار متوجه شده ام که یک شیوه دیگر در این بخش برای پرداخت وجود دارد و مقدار پارامتر cheque-companyname بود که برای خرید چکی بود.

انتظار نداشتم ولی وقتی به سامانه اصلی مراجعه کردم هنگامی ارسال درخواست مقدار پارامتر Method را به cheque-companyname تغییر دادم و در کمال تعجب سفارشم ثبت شد و پس از چند روز هم سفارش برایم ارسال شد .

خرید از کیف پول بدون موجودی

یک مورد دیگر هم استفاده از روش خرید از کیف پول است در حالی که موجودی ندارید هرچند امکان رخ دادن این مورد بسیار کم است اما در طول تست نفوذ چند سامانه به این مورد برخورده ام.

کافیست یک بار سفارش خود را از طریق کیف پول پرداخت کنید تا متوجه شوید سامانه چگونه پرداخت را پردازش می کند پس از بدست آوردن مراحل و درخواست های مختلف سعی کنید بدون داشتن موجودی از طریق کیف پول پرداختتان را نجام دهید.

جمع بندی روش های بدست آوردن شیوه های پرداخت معتبر در یک سامانه

- JavaScript Files
- Discovery in APIs
- Fuzzing
- OSINT

ایجاد اختلال در سرویس پرداخت

یکی از اشتباهات رایج در پیاده‌سازی سرویس‌های پرداخت این است که بدون انجام اعتبارسنجی صحیح، وضعیت یک فاکتور را ثبت می‌کنند. این موضوع در برخی موارد می‌تواند سامانه را با بحران مواجه کند. یکی از بحران‌های مهم، جلوگیری از انجام تراکنش‌های موفق در یک سامانه است. برای مثال، تصور کنید هیچ‌کس نتواند در یگ سامانه تراکنش موفقی داشته باشد یا پرداخت خود را تکمیل کند. اما چگونه چنین مشکلی ممکن است رخ دهد؟

درخواست انصراف از خرید

Request

```
POST /v1/transaction/verify HTTP/1.1
Host: merchant.com
MID=10920870&TerminalId=10920870
&RefNum=&ResNum=545512&State=C
anceledByUser&Amount=100000
```

Response

```
Location: : /Order/ 545512 /fail"
```

مشکل در نوع برخورد سامانه با درخواست انصراف از خرید است ، برخی از سامانه ها پس از دریافت درخواست انصراف از خرید اعتبارسنجی صحیحی بروی اطلاعات دریافتی انجام نمی دهند و این مسئله باعث ایجاد مشکلات بحرانی می شود سامانه را در نظر بگیرید که وقتی درخواست انصراف از پرداخت را دریافت می کند بدون اعتبارسنجی فاکتور پرداخت را لغو می کند و بعد از ثبت این وضعیت نیز امکان تغییر وضیعت آن وجود ندارد .

این رویکرد می‌تواند به مهاجمان این امکان را بدهد که با ارسال درخواست برای یک شماره فاکتور (که کاربر آن در درگاه بانکی در حال انحام عملیات پرداخت است) سبب لغو آن فاکتور می شوند و وضعیت آن نیز تغییر نمی کند حالا کاربر پرداخت را انجام داده وقتی اطلاعات به فروشنده ارسال می شود چون قبلا برای این فاکتور ثبت وضعیت شده فروشنده پرداخت را رد کرده و پرداخت ثبت نمی شود.عموماً شناسه فاکتورها از نوع اینتجر (Integer) هستند و با هر فاکتور جدید یک واحد افزایش پیدا می‌کنند.در واقع از ویژگی، Auto Increment ID استفاده می شود)،در اینصورت مهاجم براحتی می تواند شناسه فاکتور سایر کاربران را داشته باشد و کلیه پرداخت‌های بعدی را لغو کنند.

نحوه تست این آسیب پذیری

برای تست این مورد، ابتدا روی خرید یک محصول یا شارژ کیف پول کلیک کنید و تا مرحله پرداخت در درگاه بانکی پیش بروید. سپس کافی است روی گزینه انصراف از خرید کلیک کنید. زیرا مقدار پارامتر State برای انصراف از خرید در PSPهای مختلف متفاوت است، این روش بهتری است. حال که درخواست انصراف از خرید را به دست آورده‌اید، آن را به Repeater ارسال کنید.

رای تست این مورد، ابتدا روی خرید یک محصول یا شارژ کیف پول کلیک کنید و تا مرحله پرداخت در درگاه بانکی پیش بروید. سپس کافی است روی گزینه انصراف از خرید کلیک کنید. زیرا مقدار پارامتر State برای انصراف از خرید درPSP های مختلف متفاوت است، این روش بهتری است. حال که درخواست انصراف از خرید را به دست آورده‌اید، آن را به Repeater ارسال کنید.

در مرحله دوم، باید یک پرداخت موفق داشته باشید، اما قبل از اینکه اطلاعات به اندپوینت تأیید پرداخت ارسال شود، باید مقدار ResNum یا OrderID و) ... بسته به (PSP را از درخواست موفق کپی کرده و در درخواست انصراف از خرید جایگذاری کنید. ابتدا درخواست انصراف را ارسال کنید، سپس درخواست پرداخت موفق را بدون تغییر ارسال کنید.

اگر پرداخت موفق ثبت شد، به این معناست که سامانه مشکلی ندارد؛ اما اگر پرداخت شما به‌صورت موفقیت‌آمیز ثبت نشد، سامانه آسیب‌پذیر است و می‌توان به‌صورت کلی عملکرد آن را مختل کرد.

آسیب پذیری IDOR و پرداخت مبلغ کمتر

دلیل بروز این مشکل عدم اعتبارسنجی دقیق اطلاعات تراکنش است سامانه استعلام رسید دیجیتال را از PSP دریافت می کند ولی اطلاعات تراکنش را با Resnum (فاکتوری که قرار است تائید شود) تطابق نمی دهد پس اینجا مهاجم در صورت داشتن یک رسید دیجیتال موفق که استفاده نشده است می تواند یک فاکتور را با هر مبلغی به تائید برساند .

نحوه تست این آسیب پذیری

برای تست این آسیب پذیری ابتدا یک سفارش با مبلغ بالا ایجاد کنید و بروی پرداخت سفارش از طریق درگاه بانکی بزنید پس از هدایت به درگاه بانکی بروی انصراف از خرید زده و مقدار Resnum را یاداشت کنید.حالا کافیست یک سفارش با مبلغ کمتر ایجاد کرده و پس از هدایت به درگاه بانکی و پرداخت مبلغ هنگامی که اطلاعات به اندپوینت تائید پرداخت ارسال می شد مقدار Resnum که در مرحله قبل یاداشت کرده بودید را جایگزین کنید در صورت آسیب پذیر بودن سامانه سفارشی که با مبلغ بالا ایجاد کرده بودید تائید می شود.

Double spending رسید دیجیتال

من سه مدل تست برای دوبار مصرف کردن رسید دیجیتال دارم که به ترتیب آنها اشاره می کنم.

دوبار مصرف کردن رسید پرداخت (استفاده از یک درگاه در چند سامانه)

فروشنده پس از دریافت اطلاعات تراکنش رسید دیجیتالی RefNum را در پایگا ه داده خود جستجو می کند. این کار به منظور جلوگیری از Double Spending یا دوبار مصرف شدن یک رسید دیجیتالی است .

اما تصویر کنید مجموعه های بزرگ ممکن است چندین سامانه مجزا داشته باشند که پایگاه داده مجزا از هم دارد ولی از یک درگاه پرداخت بانکی بصورت مشترک استفاده می کنند.

نحوه تست این آسیب پذیری

برای تست این مورد ابتدا مطمئن شوید که درگاه های یکسانی در سامانه های دیگر استفاده شده برای این کار سفارش های مجزا در این سامانه ها ایجاد کنید و سپس بروی انصراف از خرید بزنید اگر مقدار پارامتر های MID و TerminalId در همه درخواست های انصراف یکسان بود مشخص می شود که از یک درگاه پرداخت بانکی مشترک بروی این سامانه ها استفاده شده است.

1. حالا کافیست یک پرداخت مثلا با مبلغ ۲۰ هزار تومان انجام دهید و پس از ارسال اطلاعات به فروشنده پارامتر های ارسالی را یاداشت کنید.
2. در سامانه بعدی یک فاکتور ۲۰ هزار تومانی ایجاد کنید و پس از اینکه به درگاه بانکی هدایت شدید بروی انصراف از خرید بزنید حالا کافیست پارامتر های که در مرحله قبل یاداشت کرده بودید را در این درخواست جایگزین کنید(إبجز مقدار Resnum) و درخواست را ارسال کنید.

دوبار مصرف کردن رسید پرداخت (اشتباه در ذخیره سازی Refnum)

یکی از اشتباهات که کمتر رخ می دهد این است که سامانه بجای ذخیره سازی Refnum مقدار Resnum را در پایگاه داده ذخیره می کند.

نحوه تست این آسیب پذیری

برای تست این آسیب پذیری دقیقا مثل مرحله قبل باید عمل کنید .

1. ابتدا یک پرداخت را بصورت صحیح انجام دهید و درخواستی ای که اطلاعات تراکنش به فروشنده ارسال می شود را تکثیر کنید
2. سپس یک سفارش با مبلغ برابر ایجاد کنید و پس از هدایت به درگاه پرداخت بروی انصراف بزنید و مقدار Resnum را یاداشت کنید
3. به درخواستی که تکثیر کرده اید مراجعه کرده و مقدار Resnum که یاداشت کرده بودید را جایگزین کرده و درخواست را ارسال کنید .

دوبار مصرف کردن رسید پرداخت (آسیب پذیری منطقی)

همانطورکه در ابتدا اشاره کردم عموما کمتر رخ می دهد که سامانه پس از تائید پرداخت مقدار Refnum را در پایگاه داده ذخیره نکند. اما اشتباه در پیاده سازی سرویس پرداخت ممکن است به ما این امکان را بدهد کا با درخواست جعلی سامانه را مجبور به پاک کردن Refnum از پایگاه داده خود کنیم.

این مورد را من در سامانه اسنپ باکس کشف کرده ام (البته موارد دیگری هم بودند که اجازه افشا ندارم)

امکان ارسال درخواست انصراف خرید برای یک فاکتور پرداخت شده در سامانه اسنپ باکس وجود داشت که در نهایت سامانه را مجبور می‌کرد Refnum را از پایگاه داده حذف کند. این عمل باعث می‌شد که یک رسید دیجیتال بیشتر از یک‌بار مصرف شود:

نحوه تست این آسیب پذیری

-ابتدا به قسمت کیف پول می‌رفتم و روی گزینه شارژکیف پول کلیک می‌کردم. سپس مبلغ مورد نظر را وارد کرده و به درگاه بانکی هدایت می‌شدم تا پرداخت را انجام دهم و ترافیک را در Burp دریافت می‌کردم. بعد از اینکه درخواست به callback فروشنده (اسنپ باکس) ارسال می‌شد و پرداخت تأیید می‌شد، درخواست را به Repeater ارسال می‌کردم.

Request

```
POST /v1/transaction/verify HTTP/1.1

Host: merchant.com

MID=10920870&TerminalId=10920870
&RefNum=GDASGDDSADFDASGDSAG
ASDGASDGS&ResNum=545512&State
=success&Amount=100000
```

Response

```
Location: : /Order/ 545512 "
```

حالا مقدار RefNum رو پاک میکردم و State رو هم برابر با CanceledByUser قرار میدادم و درخواست رو ارسال میکردم

Request

```
POST /v1/transaction/verify HTTP/1.1

Host: merchant.com

MID=10920870&TerminalId=10920870
&RefNum=&ResNum=545512&State=C
anceledByUser&Amount=100000
```

Response

```
Location: : /Order/ 545512 /fail"
```

وقتی این درخواست ارسال می‌شد، چون وضعیت انصراف از خرید بود، سامانه دیگر اعتبارسنجی نمی‌کرد که آیا وضعیت این فاکتور قبلاً ثبت شده یا نه. در نتیجه، رکورد مربوط به این فاکتور را در پایگاه داده آپدیت می‌کرد. از آنجایی که پارامتر RefNum که من در این درخواست ارسال کردم، NULL بود، RefNum از پایگاه داده حذف می‌شد.

حالا در مرحله سوم، من دقیقاً همان درخواست اولی را بدون تغییر ارسال می‌کردم و اتفاقی که می‌افتاد این بود که پرداخت مجدداً تأیید می‌شد و پول به کیف پول اضافه می‌شد. این فرآیند را می‌توانستم تا نیم ساعت انجام دهم و هر چند بار که می‌خواستم تکرار کنم. (به دلیل محدودیت بانک که فقط تا نیم ساعت پس از پرداخت امکان استعلام رسید دیجیتال وجود داشت)

دوبار مصرف کردن رسید پرداخت (Race condition)

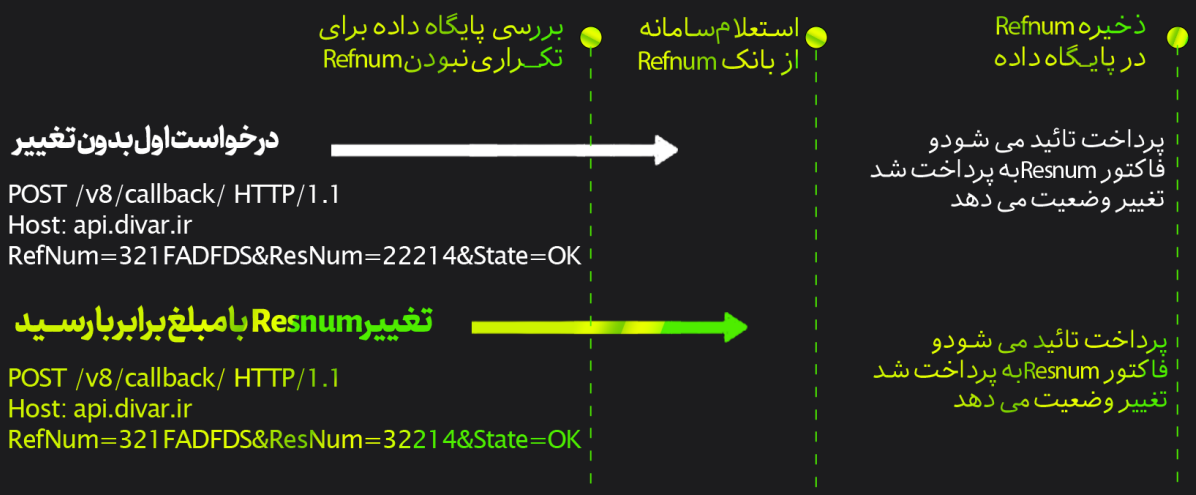
این آسیب پذیری را من در سامانه دیوار کشف کرده ام ولی این مورد بسیار شایع است و تقریبا در همه برنامه های باگ بانتی من این مورد رو پیدا کرده ام .شیوه کار به این صورت که ما با استفاده از یک رسید دیجیتال پرداخت RefNum دو تا فاکتور Resnum رو پرداخت می کنیم .

نحوه تست این آسیب پذیری

مثل مرحله قبل کاری که باید بکنیم این است که ابتدا دوتا سفارش با یک مبلغ یکسان ایجاد می کنیم بروی رپرداخت یکی از سفارش ها می زنیم و پس از هدایت به درگاه بانکی بروی انصراف زده و مقدار Resnum را یاداشت می کنیم.

حالا سفارش دوم رو پس از هدایت به درگاه بانکی پرداخت می کنیم و از ارسال درخواست (اطلاعات تراکنش به فروشنده)جلوگیری می کنیم درخواست را تکثیر کرده و در درخواست دوم مقدار Resnum که در مرحله قبل یاداشت کرده بودیم را جایگزین می کنیم. و حالا هر دو درخواست را در حالت Race condition ارسال می کنیم.و هر دو فاکتور پرداخت می شود.به دلیل نبود مکانیزم مناسب برای

جلوگیری از (Race Condition)، امکان سوءاستفاده و انجام چند تراکنش با یک مقدار ResNum در این سناریو وجود داشت .



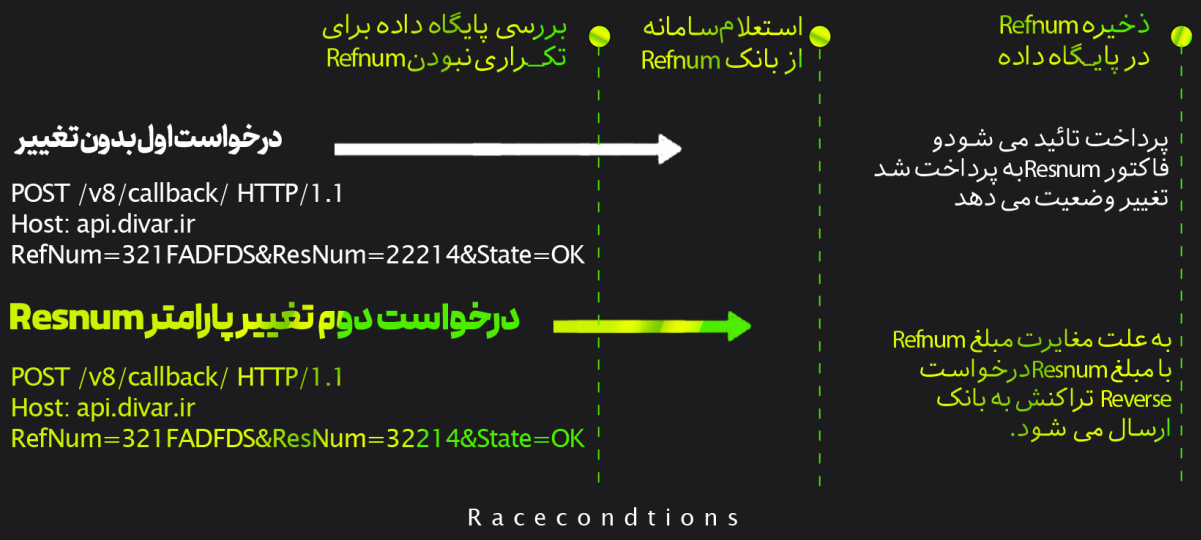
همچنین بیشتر سرویس های پرداخت ایرانی برای رسید پرداخت بیش از یکبار استعلام می دهند و همین مورد برای اسـتفاده ما کافیسـت.اگر امکان اسـتعلام بیش از یکبار برای RefNum وجود نداشـت طبیعتا نیازی به جلوگیری از Race Conditio در این بخش نبود .

پرداخت همراه با بازگشت پول

یکی از آسیب پذیری هایی که اولین بار بعد از کشفش خیلی لذت بردم همین آسیب پذیری بود ابتدا این مورد رو روی یکی از VOD ها کشف کرده ام اما بعد موفق شده ام روی سامانه های زیادی این مورد رو کشف کنم.

نحوه تست این آسیب پذیری

در بیشتر برنامه ها وقتی که مبلغ رسید دیجیتال Refnum با Resnum برابر نباشد سامانه یک درخواست به PSP ارسال می کند و تراکنش را لغو می کند . حالا اگر ما مثل مرحله قبل عمل کنیم و لی در درخواست دوم Resnum با مبلغ بیشتر قرار دهیم درخواست اول بررسی و تائید می شود و به علت عدم تطابق احتمالا درخواست دوم باعث لغو تراکنش و برگشت پول به حساب می شود .



Racecondtions

یک مدل پیاده سازی که در برنامه دیوار به آن برخوردم هم برای اینکه بتوانید عملیات پرداخت همراه با بازگشت پول داشته باشید باید برای یک پرداخت دو فاکتور ایجاد می کردیم .

نحوه تست این آسیب پذیری

ایجاد سفارش و هدایت به درگاه بانکی:

- مهاجم یک سفارش ایجاد می‌کند و روی گزینه پرداخت کلیک می‌کند تا به درگاه بانکی هدایت شود. این کار منجر به صدور یک شماره فاکتور Resnum برای سفارش می‌شود.
- سپس، مهاجم در تب دیگر مرورگر مجدداً روی گزینه پرداخت برای همان سفارش کلیک می‌کند، که منجر به صدور شماره فاکتور دومResnum می‌شود.
- در صورتی که امکان ایجاد توکن پرداخت مختلف برای یک سفارش وجود نداشت می توانید روی درخواست Race condition را تست کنید شاید محدودیت دور بخورد.

حالا دقیقا مثل مراحل قبلی روی انصراف از خرید یکی از پرداخت ها زده و Resnum را یاداشت و پرداخت دیگر را انجام می دهیم و سپس درخواست موفق را تکثیر کرده و در درخواست دوم Resnum یاداشت شده را جایگزین کرده و درخواست را در حالت Race condition ارسال می کردیم .

یکی از درخواست ها سبب پرداخت و بسته شدن سفارش می شد. و وقتی درخواست دوم تائید می شد چون سفارش از قبل تکمیل شده بود درخواست به PSP برای لغو تراکنش ارسال می شد.

تصاحب حساب کاربری از طریق پرداخت فاکتور

یکی از مواردی که نادیده گرفته می‌شود و به‌ندرت در تست کیس‌های درگاه پرداخت به آن پرداخته می‌شود، تست تصاحب حساب کاربری است. بسته به شرایط پیاده‌سازی، می‌توان سناریوهای مختلفی را اجرا کرد. در اینجا یکی از موارد خاصی که روی یکی از برنامه‌های باگ بانتی بررسی کردم و گزارش دادم را توضیح می‌دهم

موردی که می‌خواهم به آن اشاره کنم کمی غیرمتعارف است و در شرایط خاصی اتفاق می‌افتد. سناریو به این شکل بود که پس از پرداخت موفق فاکتور توسط کاربر، زمانی که اطلاعات پرداخت به callback پذیرنده ارسال می‌شد، اگر cookie کاربر در درخواست وجود نداشت یا منقضی شده بود، سامانه اقدام به بازگرداندنcookie های حساب کاربری مرتبط با شماره فاکتور (Resnum) می‌کرد. (نکته‌ای که وجود داشت این است که باید پرداخت حتماً موفق می‌بود.)

تا اینجای کار مشکلی نبود، اما سامانه روی پارامتر Resnum اعتبارسنجی صحیحی انجام نداده بود.وبرای بهره بردای کافی بود یک پرداخت موفق انجام می‌دادید و سپس شماره فاکتور یک کاربر دیگر را در آن قرار می‌دادید تا به cookieهای حساب قربانی به‌صورت کور دسترسی پیدا کنید. البته مشکلات خاص خود را نیز داشت؛ برای هر تصاحب حساب کاربری باید یک پرداخت انجام می‌دادید، که خوش‌شانس بودم و قیمت کمترین محصولش 2000 تومان بود

نحوه تست این آسیب پذیری

برای تست این مورد ابتدا یک فاکتور را پرداخت کنید و هنگامی که اطلاعات تراکنش به فروشنده ارسال می شد مقدار توکن و کوکی ها را از درخواست پاک کنید اگر سامانه در پاسخ کوکی ها یا توکن مربوط به حساب کاربری شما را بازگرداند می توانید سعی کنید با استفاده از آسیب پذیری های دیگر مثل IDOR که در مرحله اول توضیح دادیم فاکتور یک کاربر دیگر را پرداخت کرده و تصاحب حساب کاربری را انجام دهید.

مورد دیگر که می توانید تست کنید آسیب پذیری XSS است برخی از سامانه ها پس از دریافت اطلاعات تراکنش اطلاعات مربوط به تراکنش را که از کاربر دریافت کرده اند را همراه با نتیجه پرداخت به کاربر نمایش می دهند . در این حالت می توانید بررسی کنید کدام مقادیر ارسالی در صفحه چاپ می شود و سعی کنید payload های خود را داخل آن پارامتر تزریق کنید.

Request

```
POST /v1/transaction/verify HTTP/1.1
Host: merchant.com
MID=10920870&TerminalId=10920870
&RefNum=&ResNum=<scrupt>alert('x');
</script>&State=CanceledByUser&Amo
unt=100000
```

Response

```
<html>
<head><title>payment  status</title><head>
<bod>
<p>id : <scrupt>alert('x');</script>
<br> State:CanceledByUser
</p>
<body>
</html>
```

افشای اطلاعات

در برخی سامانه‌ها ، هنگام ارسال درخواست به callback پذیرنده، سامانه ابتدا شماره فاکتور را بررسی می‌کند. اگر برای آن فاکتور قبلاً اطلاعات پرداخت ارسال و تراکنش انجام شده باشد، سامانه سایر پارامترها را در نظر نمی گیرد و، مستقیماً اطلاعات سفارش را به کاربر برمی‌گرداند. مهاجم می‌تواند با استفاده از این آسیب‌پذیری و ارسال شماره فاکتورهای متعدد، اطلاعات سفارش سایر کاربران را دریافت کند.

نحوه تست این آسیب پذیری

1. ورود به درگاه پرداخت: مهاجم یک آیتم را برای خرید انتخاب کرده و به درگاه بانکی هدایت می‌شود
2. انصراف از خرید: مهاجم روی گزینه "انصراف از خرید" کلیک می‌کند.در این لحظه، درخواست انصراف از خرید به سامانه فروشنده ارسال می‌شود.
3. تکثیر و ویرایش درخواست انصراف:مهاجم درخواست مربوط به انصراف از خرید را دریافت می‌کند. سپس تمامی پارامترهای غیرضروری مانند جزئیات پرداخت را از درخواست حذف کرده و تنها پارامتر شماره فاکتور را نگه می‌دارد.
4. ارسال شماره فاکتورهای متوالی: مهاجم با استفاده از ابزاری مانند Intruder در Burp Suite ، درخواست ویرایش‌شده را با شماره فاکتورهای مختلف در یک بازه عددی ارسال می‌کند. سامانه پس از دریافت درخواست، فقط شماره فاکتور را بررسی می‌کند و اگر آن فاکتور قبلاً پرداخت شده باشد، سامانه به‌جای بررسی سایر پارامترها، اطلاعات سفارش را برمی‌گرداند.

بررسی فرآیند پرداخت از کیف پول

در پیاده‌سازی کیف پول داخلی، کاربران مبلغی را به‌عنوان سپرده در سامانه می‌گذارند و می‌توانند از آن برای پرداخت‌های مختلف استفاده کنند. پیاده‌سازی چنین سیستمی پیچیدگی زیادی ندارد. برای هر کاربر یک حساب سپرده در پایگاه داده ایجاد می‌شود. این حساب شامل اطلاعاتی مانند موجودی کیف پول است و با شناسه کاربر (User ID) مرتبط می‌شود. هر زمان که کاربر وجهی به کیف پول اضافه می‌کند یا از آن برداشت می‌کند، موجودی به‌روز می‌شود.

در عمل، برای مدیریت تراکنش‌ها (واریز و برداشت)، معمولاً یک جدول مجزا برای ثبت تراکنش‌های مالی ایجاد می‌شود تا تاریخچه تراکنش‌ها را نیز بتوان ذخیره و پیگیری کرد. به همین دلیل، اگرچه پیاده‌سازی ساده به نظر می‌رسد، نیاز به مدیریت دقیق برای اطمینان از صحت موجودی و تراکنش‌های کاربر وجود دارد.

موجودی کیف پول کافیست

- در این حالت مبلغ از کیف پول کسر شده و سپس سفارش پرداخت می شود.

موجودی کیف پول برای پرداخت سفارش کافی نیست

- سناریو اول: کیف پول صد هزار تومان موجودی دارد و سفارش ما ۲۰۰ هزار تومان است در این حالت ابتدا موجودی کیف پول کسر شده و برای مابقی مبلغ سفارش یک فاکتور ایجاد می شود مشتری پس از پرداخت این فاکتور سفارشش تکمیل می شود.
- سناریو دوم: موجودی از کیف پول مشتری کسر نمی شود بلکه برای مابقی مبلغ سفارش یک فاکتور ایجاد می شود و مشتری پس از پرداخت مبلغ باقی مانده وتائید فاکتور کل مبلغ یک جا از کیف پول کسر شده وسفارش تائید می شود.

آسیب پذیری های منطقی در کیف پول داخلی

در برخی از سامانه‌های ، هنگامی که مبلغ سفارش بیشتر از موجودی کیف پول باشد، فرآیند پرداخت به این صورت است که ابتدا مبلغ موجود در کیف پول کاربر کسر می‌شود و سپس برای پرداخت مابقی مبلغ، کاربر به درگاه بانکی هدایت می‌شود. اگر کاربر پرداخت را در درگاه بانکی لغو کند، مبلغی که از کیف پول کسر شده بود، دوباره به حساب کیف پول بازگردانده می‌شود. این مکانیسم به گونه‌ای طراحی شده که اطمینان حاصل شود کاربر هیچ مبلغی را از دست نخواهد داد اگر پرداخت نهایی انجام نشود.

در این حمله، مهاجم از ضعف موجود در مکانیزم بازگشت وجه هنگام لغو تراکنش سوءاستفاده می‌کند. او با استفاده از Race Condition می‌تواند مبلغ بیشتری به کیف پول خود بازگرداند، به‌گونه‌ای که بیشتر از مبلغ اولیه‌ای باشد که از کیف پول او کسر شده است.

نحوه تست این آسیب پذیری

- ایجاد سفارش با مبلغ بیشتر از موجودی کیف پول: مهاجم سفارشی با مبلغی بیشتر از موجودی کیف پول خود ایجاد می‌کند. سیستم ابتدا مبلغ موجودی کیف پول را کسر کرده و سپس او را به درگاه پرداخت هدایت می‌کند تا مابقی مبلغ را پرداخت کند.
- لغو پرداخت در درگاه: مهاجم در مرحله پرداخت در درگاه، به‌جای تکمیل تراکنش، پرداخت را لغو می‌کند. پس از لغو تراکنش، سیستم باید مبلغ کسر شده از کیف پول را به حساب بازگرداند.
- استفاده از Race Condition برای ارسال درخواست‌های همزمان: مهاجم در لحظه‌ای که درخواست انصراف از خرید را در درگاه ثبت می کند یک درخواست به اند پوینت فروشنده ارسال می شود مهاجم این درخواست را تکثیر کرده و ، چندین درخواست را به‌طور همزمان ارسال می‌کند .به دلیل ضعف سیستم در مدیریت درخواست‌های همزمان، تمامی درخواست‌ها پردازش شده و مبلغ بازگشتی چندین بار به کیف پول مهاجم افزوده می‌شود.
- افزایش غیرقانونی موجودی کیف پول: در نتیجه این ضعف در مدیریت درخواست‌های موازی، مهاجم موفق می‌شود تا مبلغ بیشتری از کیف پول خود بازگرداند، بیشتر از مقداری که ابتدا از آن کسر شده بود

در برخی از سامانه‌های پرداخت، کاربران می‌توانند کیف پول خود را به‌صورت دلاری یا ریالی شارژ کنند. این دو نوع کیف پول از اندپوینت‌های متفاوتی برای شارژ استفاده می‌کنند. هنگام شارژ کیف پول، سامانه شناسه کیف پول را دریافت کرده و بر اساس آن، مبلغ مربوطه را کسر می‌کند. با این حال، در برخی از سیستم‌ها، بررسی دقیق شناسه کیف پول‌ها به‌طور صحیح انجام نمی‌شود، به‌طوری‌که ممکن است کاربر بتواند با پرداخت ریالی، کیف پول دلاری را شارژ کند.

نحوه تست این آسیب پذیری

- دریافت شناسه کیف پول: مهاجم قصد دارد کیف پول خود را شارژ کند. در این فرآیند، سامانه شناسه کیف پول مربوطه (ریالی یا دلاری) را دریافت می‌کند. هر نوع کیف پول از اندپوینت خاص خود برای شارژ استفاده می‌کند.
- جایگزینی شناسه کیف پول دلاری در اندپوینت ریالی: مهاجم شناسه کیف پول دلاری را به‌جای شناسه کیف پول ریالی در درخواست شارژ قرار می‌دهد. به دلیل عدم اعتبارسنجی صحیح از سوی سامانه، این درخواست پردازش می‌شود.
- پرداخت ریالی برای شارژ کیف پول دلاری: با ارسال درخواست به اندپوینت شارژ ریالی، مهاجم تنها با پرداخت مبلغی به‌صورت ریالی موفق می‌شود کیف پول دلاری خود را شارژ کند. این نقص ناشی از عدم کنترل دقیق سامانه بر روی نوع کیف پول و تطابق آن با نوع پرداخت است.
- بهره‌برداری از ضعف سیستم: به دلیل عدم تفکیک صحیح بین کیف پول‌های ریالی و دلاری، مهاجم می‌تواند با پرداخت ریال، مبلغ بیشتری را به کیف پول دلاری خود افزوده و ارزش بیشتری نسبت به مبلغ پرداخت‌شده دریافت کند.

Request

```
POST /pay/paypal HTTP/1.1
Host: api.vooricon.com
{
        "Amount" : "10000",
        "Walletid" : "21345"
}
```

Request

```
POST /pay/mellat HTTP/1.1
Host: api.vooricon.com
{
        “Amount" : “10000",
        “Walletid" : “21345"
}
```

آسیب پذیری IDOR و CSRF

در این سناریو، مهاجم با بهره‌گیری از دو آسیب‌پذیری CSRF وIDOR ، به طور مخفیانه از کیف پول قربانی برای پرداخت سفارش خود استفاده می‌کند. در سیستم‌های امن، هر کاربر باید تنها بتواند از کیف پول خود برای پرداخت سفارشات خودش بهره ببرد، اما این دو آسیب‌پذیری به مهاجم امکان می‌دهند تا این مکانیزم امنیتی را دور بزند.

نحوه تست این آسیب پذیری

- رهگیری درخواست پرداخت از کیف پول: مهاجم ابتدا یک سفارش برای خود ایجاد می‌کند و درخواست پرداخت از کیف پول خود را رهگیری می‌کند. این درخواست، معمولاً با متود GET است که شناسه سفارش در انتهای URL قرار دارد. مهاجم با بررسی این URL متوجه می‌شود که شناسه سفارش به راحتی قابل تغییر است.
- سوءاستفاده از آسیب‌پذیری IDOR: مهاجم شناسه سفارش خود را در URL جایگزین می‌کند و لینکی جدید ایجاد می‌کند که به جای کیف پول خودش، از کیف پول قربانی برای پرداخت استفاده می‌کند. این نقص ناشی از عدم اعتبارسنجی صحیح شناسه سفارش است.
- ۰ ارسال لینک به قربانی مهاجم لینک دستکاری‌شده را برای قربانی ارسال می‌کند (با استفاده از روش‌هایی مانند مهندسی اجتماعی یا ارسال ایمیل فیشینگ) و قربانی را فریب می‌دهد تا روی لینک کلیک کند.
- پرداخت از کیف پول قربانی: در صورتی که قربانی به حساب کاربری خود وارد شده باشد، با کلیک بر روی لینک، درخواست پرداخت ارسال می‌شود و مبلغ سفارش مهاجم از کیف پول قربانی کسر می‌گردد. قربانی هیچ کنترلی بر این فرآیند نخواهد داشت، زیرا درخواست به صورت خودکار از سمت او ارسال می‌شود.
- تکمیل سفارش مهاجم:با پردازش همزمان درخواست، سیستم پرداخت را از کیف پول قربانی انجام داده و سفارش مهاجم تکمیل می‌شود، بدون اینکه قربانی متوجه این کلاهبرداری شود.

آسیب پذیری Double spending

در برخی از سامانه‌ها، کاربران قادرند به‌طور همزمان چندین سفارش را به‌صورت مجزا ثبت کنند. این ویژگی به کاربران این امکان را می‌دهد که خریدهای خود را به‌طور همزمان انجام دهند و در برخی شرایط، این عملکرد می‌تواند به کاربران کمک کند تا به‌راحتی از موجودی کیف پول خود استفاده کنند.

با این حال، در شرایط خاصی، اگر سیستم به درستی مدیریت نشود، می‌تواند آسیب‌پذیری‌هایی را ایجاد کند. به‌ویژه، اگر سیستم نتواند به‌درستی درخواست‌های همزمان را پردازش کند، ممکن است Race Conditionرخ دهد. این نوع آسیب‌پذیری می‌تواند به مهاجمان اجازه دهد تا از موجودی کیف پول خود بیش از یک بار استفاده کنند.

سناریو حمله

در این سناریو، مهاجم از ضعف در مدیریت همزمانی سیستم سوءاستفاده می‌کند و با ثبت چندین سفارش به‌طور همزمان، می‌تواند از موجودی کیف پول خود بیشتر از یک بار استفاده کند.

نحوه تست این آسیب پذیری

- ثبت سفارشات متعدد:مهاجم به‌طور همزمان دو یا چند سفارش را در سیستم ثبت می‌کند. در این فرآیند، مبلغ هر یک از سفارش‌ها به‌گونه‌ای انتخاب می‌شود که مجموع آن‌ها کمی کمتر از موجودی کیف پول باشد.
- انتخاب گزینه پرداخت از کیف پول: پس از ثبت سفارشات، مهاجم گزینه «پرداخت از کیف پول» را انتخاب می‌کند. درخواست‌های مربوط به پرداخت همزمان برای هر یک از سفارشات به‌طور همزمان ارسال می‌شوند.

- رخ دادن Race Condition:: به دلیل ضعف در مدیریت همزمانی سیستم، درخواست‌های پرداخت همزمان پردازش می‌شوند. این امر می‌تواند باعث شود که سیستم نتواند به درستی موجودی کیف پول را بررسی کند و در نتیجه، بیش از یک بار از موجودی کیف پول کسر شود.
- کسر چندین بار از موجودی کیف پول: سیستم به‌جای کسر تنها یک بار موجودی کیف پول، به‌طور همزمان مبالغ هر دو سفارش را کسر می‌کند و در نتیجه، موجودی کیف پول به‌طور غیرقانونی کاهش می‌یابد.
- هر دو سفارش پرداخت و تائید می شود.

Request

```
POST /v8/pay/ HTTP/1.1
Host: api.vooricon.com
{
        "Order" : "VOR-32257",
        "Method" : “Wallet"
}
```

Request

```
POST /v8/pay/ HTTP/1.1
Host: api.vooricon.com
{
        "Order" : "VOR-52252",
        "Method" : “Wallet"
}
```

Race Condition➔

در این سناریو، مهاجم با استفاده از Race Condition در فرآیند پرداخت، موفق می‌شود سامانه را فریب دهد تا موجودی کیف پول او را همزمان برای دو سفارش کسر کند. این سناریو مشابه سناریوی قبلی است که مهاجم با استفاده از همزمانی در درخواست‌ها سعی در بهره‌برداری دارد، اما نقطه آسیب‌پذیری در اینجا تفاوت دارد. در حالی که در سناریوی قبل مهاجم تنها از کیف پول برای پرداخت سفارش استفاده می‌کرد، در اینجا مهاجم ابتدا از کیف پول خود برای بخشی از مبلغ سفارش استفاده کرده و سپس از درگاه بانکی برای باقی مبلغ پرداخت می‌کند. سامانه پس از تأیید پرداخت درگاه، موجودی کیف پول افزایش داده و سپس مبلغ نهایی سفارش را کسر می‌کند. با استفاده ازRace Condition ، مهاجم درخواست تأیید پرداخت‌ها را همزمان ارسال می‌کند، در نتیجه موجودی کیف پول دوبار کسر می‌شود، بدون اینکه سامانه متوجه همزمانی درخواست‌ها شود. اگرچه معمولاً سامانه‌ها از یک تابع مشترک برای کسر موجودی کیف پول در تمامی بخش‌های پرداخت استفاده می‌کنند (این مورد باعث می شود اگر در سناریو قبلی آسیب پذیر نبود در این حالت نیز آسیب پذیر نباشد)، در برخی موارد این تابع ممکن است به صورت جداگانه برای هر بخش نوشته شود به همین علت در صورتی که سناریو قبل قابل اجرا نبود ممکن است این سناریو اجرا شده و در این بخش سامانه آسیب پذیر باشد.

نحوه تست این آسیب پذیری

- ایجاد دو سفارش:مهاجم دو سفارش ایجاد می‌کند که مبلغ هرکدام بیشتر از موجودی کیف پول است. مثلاً اگر موجودی کیف پول ۹۰۰ هزار تومان باشد، هر سفارش را به‌گونه‌ای تنظیم می‌کند که مبلغ آن‌ها ۱ میلیون تومان باشد.
- انتخاب گزینه پرداخت از کیف پول: مهاجم برای هر دو سفارش گزینه "استفاده از کیف پول" را انتخاب می‌کند. سامانه متوجه می‌شود که موجودی کیف پول کمتر از مبلغ کل سفارش است و به ازای هر سفارش یک فاکتور برای مبلغ تفاوت (مثلاً ۱۰۰ هزار تومان) ایجاد می‌کند و مهاجم را به درگاه بانکی هدایت می‌کند.
- پرداخت در درگاه بانکی: مهاجم هر دو پرداخت را به صورت عادی در درگاه بانکی انجام می‌دهد، اما قبل از ارسال درخواست‌های تأیید پرداخت به سامانه، این درخواست‌ها را متوقف می‌کند.
- ارسال همزمان درخواست‌های تأیید پرداخت به سامانه: مهاجم درخواست‌های تأیید پرداخت را به‌طور همزمان با استفاده از Race Condition به سامانه ارسال می‌کند. سامانه چون هر دو

درخواست تقریباً همزمان دریافت می‌شود، نمی‌تواند به‌درستی موجودی کیف پول را بررسی کند.

- کسر موجودی کیف پول برای هر دو سفارش: سامانه هر دو درخواست را پردازش می‌کند و چون موجودی کیف پول در لحظه بررسی هنوز کافی است، برای هر دو سفارش مبلغ را کسر می‌کند. در نتیجه، موجودی کیف پول به اشتباه دو بار کسر می‌شود و به حالت منفی می‌رود.
- تکمیل هر دو سفارش:با وجود اینکه موجودی کیف پول منفی شده است، هر دو سفارش به‌طور موفقیت‌آمیز تکمیل می‌شوند.

Request

```
POST /v1/transaction/verify HTTP/1.1

Host: merchant.com

MID=10920870&TerminalId=10920870
&RefNum=GDASGDDSADFDASGDSAG
ASDGASDGS&ResNum=545512&State
=success&Amount=100000
```

Response

```
POST /v1/transaction/verify HTTP/1.1

Host: merchant.com

MID=10920870&TerminalId=10920870
&RefNum=asGxvdArDe21DSAGASDGA
SDGS&ResNum=545532&State=success
&Amount=120000
```

Race Condition➔

آسیب پذیری Double spending کد تخفیف و کارت هدیه

در این سناریو، مهاجم از ضعف‌های همزمانی در سیستم بهره می‌برد تا از یک کد تخفیف دوبار استفاده کند، در حالی که سیستم تنها باید اجازه دهد این کد یک‌بار اعمال شود (در بیشتر سامانه‌ها، فرآیند اعمال کد تخفیف به گونه‌ای طراحی شده است که ابتدا قیمت محصولات بررسی و جمع‌بندی می‌شود، و سپس مبلغ تخفیف از مقدار کل کسر می‌شود. این ساختار باعث می‌شود که حتی اگر سامانه در برابر ریس کاندیشن آسیب‌پذیر باشد، امکان استفاده مجدد از کد تخفیف برای یک سبد خرید یا سفارش وجود نداشته باشد.)

ما باید در این شرایط سعی کنیم که یک کد تخفیف را بروی دو سفارش اعمال کنیم.این مورد بصورت شایع در بیشتر برنامه ها وجود دارد و می توانید به راحتی آن را کشف و گزارش کنید.

نحوه تست این آسیب پذیری

شما باید دو سفارش ایجاد کنید و سپس هنگام اعمال کد تخفیف را بصورت همزمان درخواست را در حالت Race condition ارسال کنید مثل نمونه زیر

Request

```
POST /v8/pay/ HTTP/1.1
Host: api.vooricon.com
{
        "Order" : "VOR-98212",
        "Method" : "Saman",
        "Code" : "TsbgreGH"
}
```

Request

```
POST /v8/pay/ HTTP/1.1
Host: api.vooricon.com
{
        "Order" : "VOR-98213",
        "Method" : "Saman",
        "Code" : "TsbgreGH"
}
```

Race Condition➔

برخی از سامانه ها ممکن است اجازه ساخت دو سبد خرید یا دو سفارش را بصورت همزمان به کاربر ندهند در صورتی که شرایط این چنینی حاکم بود می تواند سفارش ها را در دو حساب کاربری ایجاد کرده و سپس درخواست ها را ارسال کنید

همچنین در مواردی ممکن است مثلا حتی اگر در بخش ثبت کارت هدیه بتوان با Race condition چندین با کارت را ثبت کرد بهره بردای نداشته باشد مثلا تصور کنید کارت هدیه اشتراک یک ماه دارید و سامانه نسبت به Race condition آسیب پذیر است و شما موفق شده اید کارت هدیه را چند بار مصرف کنید ولی سامانه اشتراک شما را رزرو نمی کند و پس از ثبت کارت هدیه دقیقا از زمانی که کارت ثبت شده به شما یک ماه اشتراک تعلق می دهد یا اینکه کارت هدیه ای که تخفیف ۹۰ درصد برای حساب شما اعمال می کند شما هر چند بار هم بتوانید کارت را اعمال کنید تغییری در درصد تخفیف نخواهید داشت در این صورت می توانید کارت هدیه را بروی دو حساب ثبت کنید.

Request

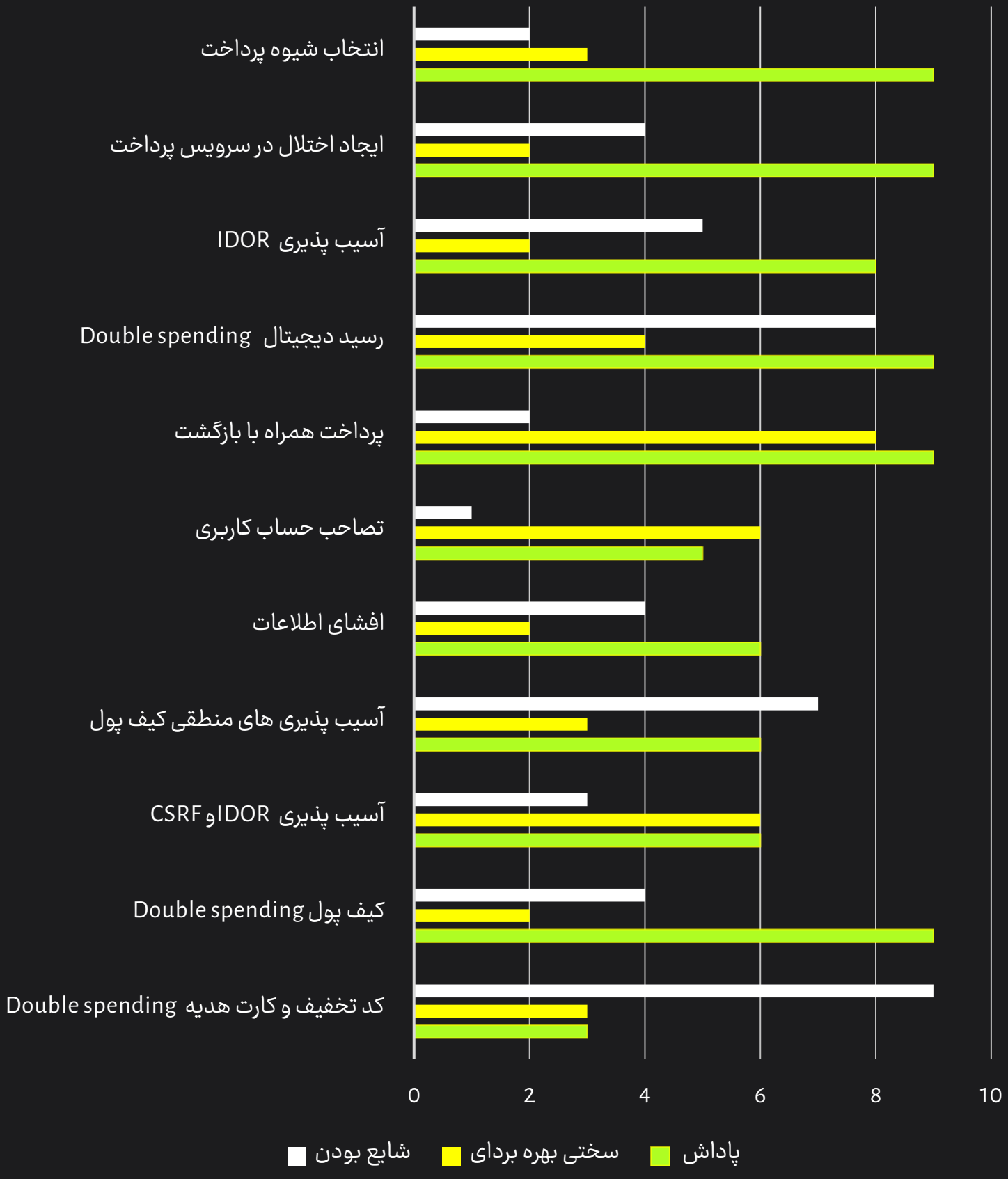
```
POST /v8/pay/ HTTP/1.1
Host: api.vooricon.com
Token=usertoken(1)
{
        "Number" : "VOR-2s2-GFSE",
}
```

Request

```
POST /v8/pay/ HTTP/1.1
Host: api.vooricon.com
Token=usertoken(2)
{
        "Number" : "VOR-2s2-GFSE",
}
```

Race Condition➔

بررسی آماری سناریو های بررسی شده



Moradloo1779@gmail.com

About Me

- Mahdi Moradloo
- 31 years old

Background:

- About 3 years start security
- 2 years as a Web Developer

Experience:

- bug bounty hunter
- Web Application Penetration Testing

x.com
moradloo_ir

linkedin.com
moradloo

VOORICON2024

END 15 Nov 2024