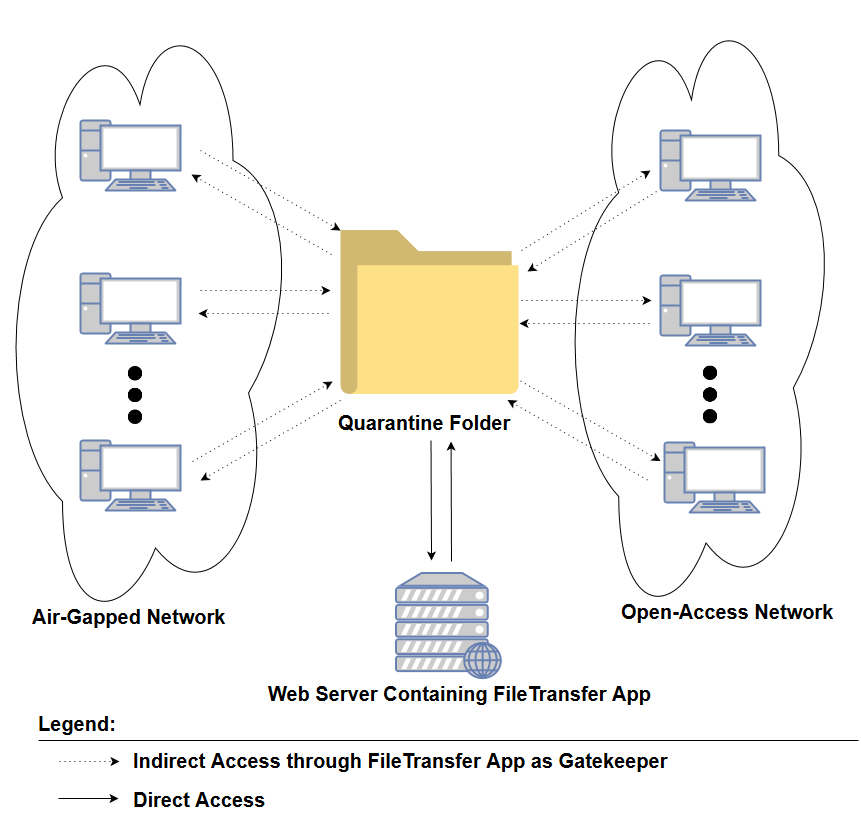
# هدف:

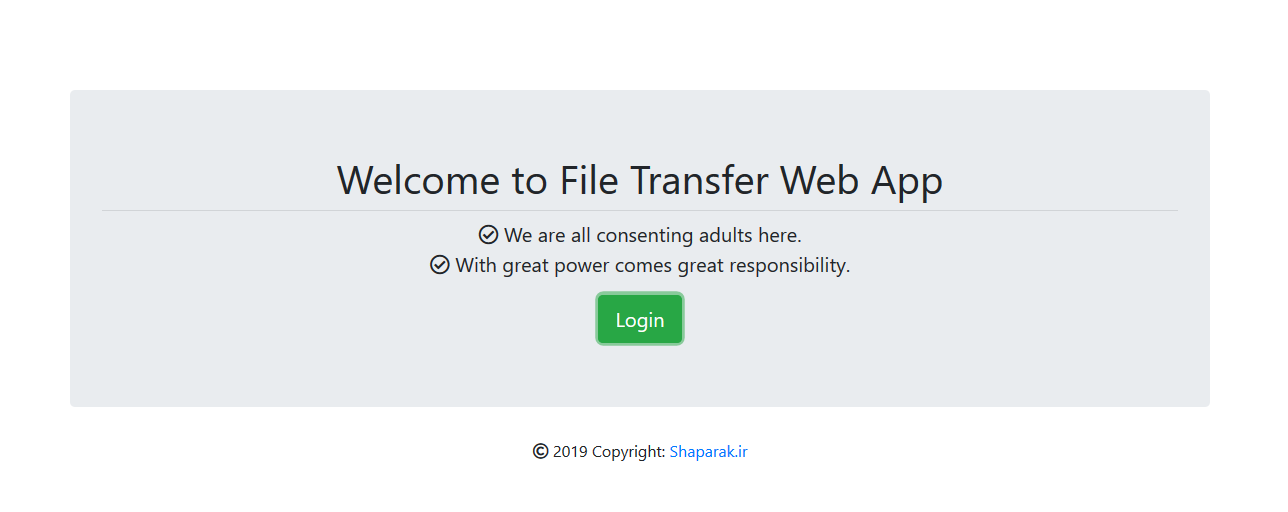
انتقال فایل از شبکه بسته (ایزوله شده)[[1]](#footnote-1) به شبکه با دسترسی باز (فلش دیسک، اینترنت). (شکل *1*)



شکل 1: مدل مفهومی انتقال فایل بین دو شبکه

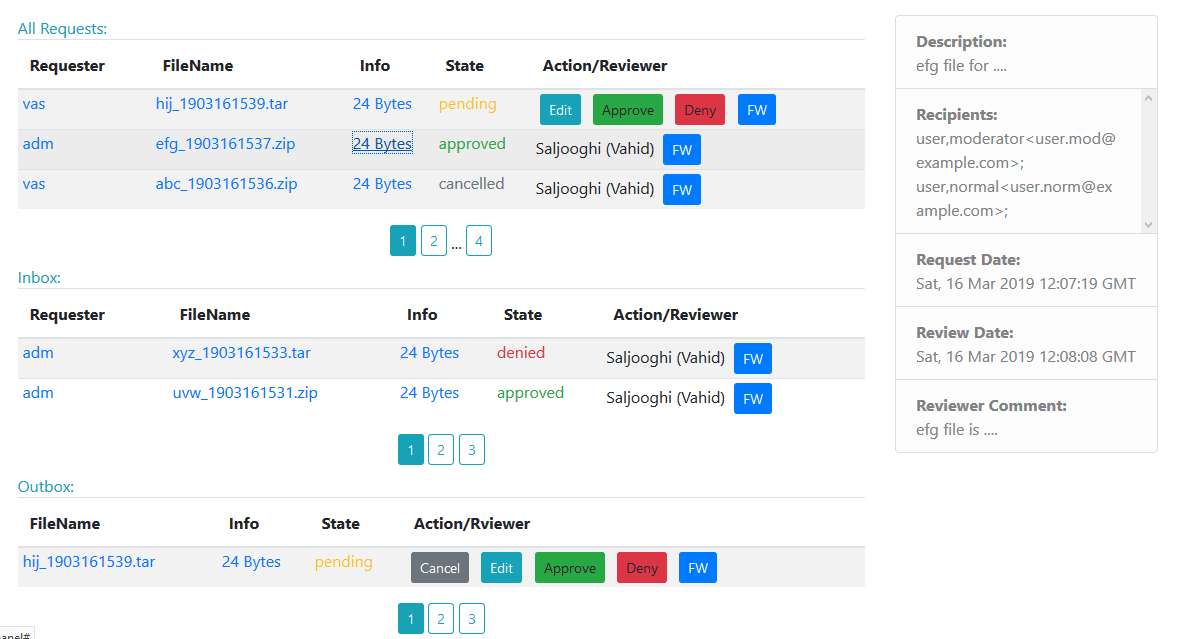
# نحوه عملکرد:

وب اپلیکیشن انتقال فایل[[2]](#footnote-2) (شکل 2) بر روی وب سروری با آدرس سوکت[[3]](#footnote-3) خاص قرار می گیرد. به این ترتیب کاربر می تواند با اتصال به سرور (از شبکه هایی که سرور قابل دسترسی است)، درخواست خود را در پنل مختص خود ثبت کند یا وضعیت درخواست های از قبل ثبت شده خود را مشاهده نماید. ثبت درخواست همراه با بالاگذاری[[4]](#footnote-4) فایل به صورت فشرده شده می باشد.



شکل 2: صفحه خانه جهت خوش آمدگویی

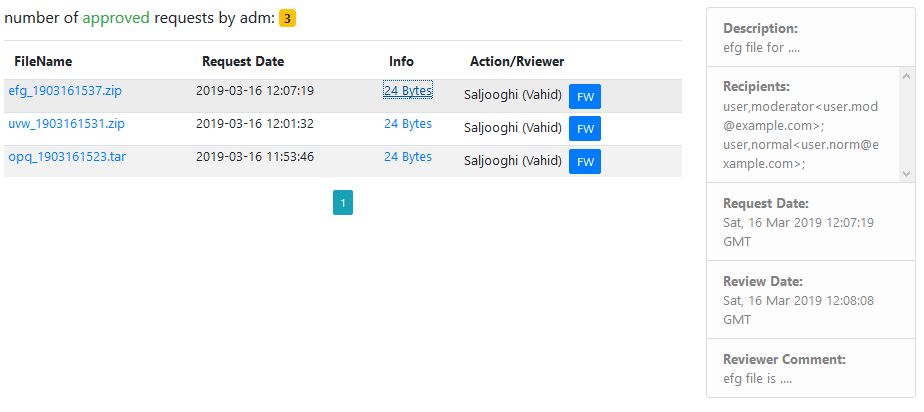
پس از ارزیابی درخواست ثبت شده توسط Moderator و در صورت تایید، پایین گذاری[[5]](#footnote-5) فایل بر روی پنل کاربر درخواست دهنده (در شبکه با دسترسی باز) امکان پذیر می باشد. علاوه بر آن اسامی افرادی که در لیست دریافت کنندگان[[6]](#footnote-6) فایل قراردارند نیز (در صورت تایید) قادر به دانلود فایل در پنل مختص خود به صورت دسته بندی شده (All Requests, Inbox, Outbox) هستند.



شکل 3: پنل درخواست ها

# ویژگی ها:

* وب اپلیکیشن قابلیت مدیریت حساب های کاربری[[7]](#footnote-7) (ایجاد[[8]](#footnote-8)، حذف[[9]](#footnote-9)، ویرایش[[10]](#footnote-10) و مسدود کردن[[11]](#footnote-11)) و همچنین مدیریت درخواست های[[12]](#footnote-12) انتقال فایل را می دهد (لغو[[13]](#footnote-13)، ویرایش، پذیرفتن[[14]](#footnote-14)، رد[[15]](#footnote-15) و ارسال بیشتر[[16]](#footnote-16) کردن درخواست های انتقال فایل). همچنین امکان جداسازی و نمایش درخواست ها براساس وضعیت (لغو شده، منتظر بررسی[[17]](#footnote-17)، رد شده و پذیرفته شده) یا براساس نام کاربری در صفحه ای مجزا می باشد (شکل 4).



شکل 4: لیست تمام درخواست های تایید شده کاربر خاص (adm)

* اطلاعات هر درخواست (شرح درخواست[[18]](#footnote-18)، دریافت کنندگان، تاریخ درخواست[[19]](#footnote-19)، تاریخ بررسی/ ارزیابی مادریتور[[20]](#footnote-20) و اضهارنظر مادریتور[[21]](#footnote-21)) در نوارجانبی[[22]](#footnote-22) پس از کلیک بر روی بخش Info نمایش داده می شود.
* برای وب اپلیکیشن 3 نوع کاربری با مشخصات زیر تعریف شده است:

1. کاربری عادی (Normal):

می تواند درخواست خودش رو لغو یا ویرایش کند.

می تواند درخواست انتقال فایل ثبت کند تا بعدا توسط یک Moderator بررسی شود.

نمی تواند کاربر جدید ایجاد کند یا کاربری را حذف یا ویرایش کند.

نمی تواند درخواست های دیگران را ارزیابی کند.

می تواند درخواست های تایید شده دیگران را که خود به عنوان دریافت کننده درخواست ذکر شده است پایین گذاری کند.

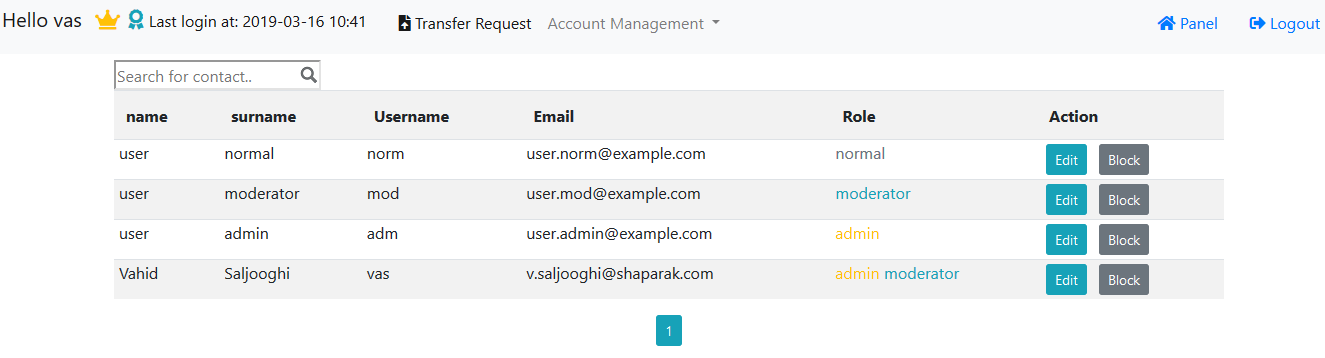
1. کاربری Admin:

می تواند درخواست خودش رو لغو یا ویرایش کند.

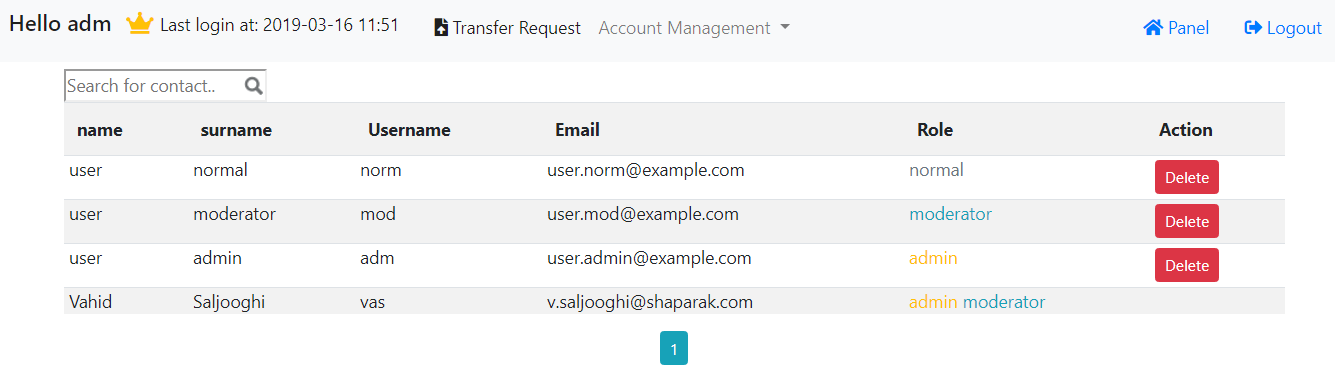
می تواند درخواست انتقال فایل ثبت کند تا بعدا توسط یک Moderator بررسی شود.

می تواند حساب کاربر با ویژگی ادمین یا عادی (و نه Moderator) ایجاد کند. می تواند هر نوع حساب کاربر به غیر از کاربر با ویژگی های ادمین و مادریتور(ادمین+مادریتور بالاترین سطح دسترسی را دارد) را حذف، مسدود یا ویرایش کند. (شکل 5 و شکل 6)

می تواند درخواست های تایید شده دیگران را که خود به عنوان دریافت کننده درخواست ذکر شده است پایین گذاری کند.



شکل 5: کاربر با هر دو دسترسی ادمین و مادریتور امکان مدیریت تمام حساب ها را دارد.



شکل 6: کاربر با دسترسی ادمین (با تاج طلایی مشخص شده است) امکان تغییر برای کاربر ادمین و مادریتور را ندارد.

1. کاربری Moderator:

می تواند درخواست خودش رو لغو، رد، تایید یا ویرایش کند.

می تواند درخواست های دیگران را تایید، رد یا ویرایش کند.

نمی تواند حساب کاربر ایجاد، حذف یا ویرایش کند.

کاربر وب اپلیکیشن می تواند یک یا چند نوع کاربری را همزمان دارا باشد. به این ترتیب کاربر دارای ویژگی های/کاربری های Admin و Moderator دارای بالاترین سطح دسترسی می باشد.

کاربر وب اپلیکیشن می تواند جهت جستجو در بین تماس های[[23]](#footnote-23) نمایش داده شده، با کلیک بر جعبه "جستجو" به نحوه دلخواه لیست تماس ها را فیلتر نماید.

* در پیاده سازی از زبان های برنامه نویسی جاوااسکریپت، پایتون و کتابخانه های JQuery، Bootstrap، Font Awesome و میکروفریمورک فلاسک (به همراه کتابخانه هایش ) استفاده شده است. ساختار درختی فایل ها در زیر آمده است.

FileTransferWebApp

│ run.py

│

└───FileTransferPackage

│ forms.py

│ models.py

│ routes.py

│ \_\_init\_\_.py

│

├───DB

│ FileTransfer.db

│

├───static

│ ├───css

│ │ all.min.css

│ │ bootstrap-grid.min.css

│ │ bootstrap-grid.min.css.map

│ │ bootstrap-reboot.min.css

│ │ bootstrap-reboot.min.css.map

│ │ bootstrap.min.css

│ │ bootstrap.min.css.map

│ │ main.css

│ │

│ ├───img

│ │ FileTransfer.png

│ │ searchicon.png

│ │

│ ├───js

│ │ AJAX2GetReqInfo.js

│ │ bootstrap.bundle.js

│ │ bootstrap.bundle.js.map

│ │ bootstrap.bundle.min.js

│ │ bootstrap.bundle.min.js.map

│ │ bootstrap.js

│ │ bootstrap.js.map

│ │ bootstrap.min.js

│ │ bootstrap.min.js.map

│ │ delUser.js

│ │ ImportContacts.js

│ │ jquery-3.3.1.js

│ │ reqAction.js

│ │ searchContacts.js

│ │ SelectContacts.js

│ │

│ └───webfonts

│ fa-brands-400.eot

│ fa-brands-400.svg

│ fa-brands-400.ttf

│ fa-brands-400.woff

│ fa-brands-400.woff2

│ fa-regular-400.eot

│ fa-regular-400.svg

│ fa-regular-400.ttf

│ fa-regular-400.woff

│ fa-regular-400.woff2

│ fa-solid-900.eot

│ fa-solid-900.svg

│ fa-solid-900.ttf

│ fa-solid-900.woff

│ fa-solid-900.woff2

│

├───templates

│ create\_account.html

│ delete\_accounts.html

│ edit\_account.html

│ edit\_accounts.html

│ edit\_req.html

│ fw\_req.html

│ home.html

│ layout.html

│ login.html

│ panel\_moderator.html

│ panel\_normal.html

│ reset\_request.html

│ reset\_token.html

│ trans\_req.html

│ x\_reqs.html

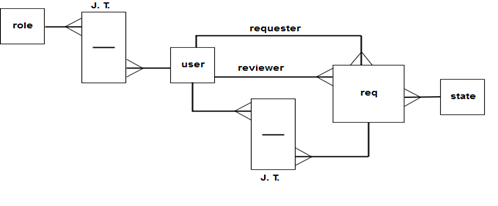
│

└───includes

\_colorize.html

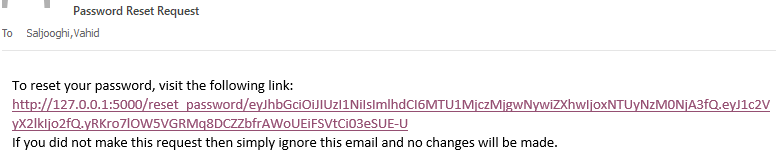
\_formhelpers.html

* از آنجا که از کتابخانه SQLAlchemy به عنوان یک Object-Relational Mapping استفاده شده است، امکان پشتیبانی از پایگاه داده های رایج می باشد. شکل 7 دیاگرام رابطه بین جداول پایگاه داده را نشان می دهد.



شکل 7: Entity Relationship Diagram

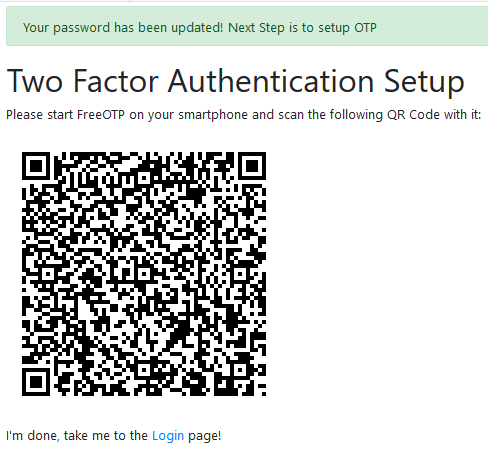
* پس از تعریف کاربر توسط ادمین، ایمیلی به صورت خودکار جهت تعیین/ انتخاب رمز ورود برای کاربر ارسال می شود و رمز انتخاب شده توسط کاربر به صورت رمزنگاری[[24]](#footnote-24) شده در پایگاه داده ذخیره می شود. (توجه شود که این ایمیل پس از زمانی قابل تعیین که در حال حاضر نیم ساعت است، باطل خواهد شد و دیگر قابل استفاده نمی باشد).



شکل 8: نمونه ایمیل ارسال شده جهت انتخاب رمز برای اولین بار یا ریست کردن رمز فراموش شده

کاربر پس از انتخاب رمز ورود باید تنظیماتی بر روی گوشی هوشمند جهت استفاده از رمز یکبار مصرف[[25]](#footnote-25) به ترتیب زیر انجام دهد:

1. اپلیکیشن FreeOTP را بر روی گوشی خود نصب کند.
2. کد QR نشان داده شده را با گوشی اسکن کند. (شکل 9)



شکل 9: صفحه مربوط به تنظیم رمز یکبار مصرف

* از روی محتوا و نام فایل Digest تهیه می شود و در پایگاه داده ذخیره می گردد. به این ترتیب امکان اینکه فایلی با نام و محتوا یکسان چندبار بارگذاری شود وجود ندارد.
* تمام فرم های HTML توسط WTForms قبل از ارسال[[26]](#footnote-26) ارزیابی[[27]](#footnote-27) می شود تا داده های نامعتبر مانند آدرس ایمیل با فرمت اشتباه در پایگاه داده ذخیره نشود و همچنین فرمت ها فایل ها محدود شده است یه چند فرمت فشرده سازی (.zip, .rar, .tar) تا از بارگذاری بدافزار در سمت سرور جلوگیری شود.

# برای ایجاد اولین کاربر در پایگاه داده به صورت دستی:

0- Go to python REPL inside application folder

1- from FileTransferPackage import db,bcrypt

2- from FileTransferPackage.models import \*

3- db.create\_all()

4- pass1=bcrypt.generate\_password\_hash("123456").decode('utf-8')

5- u1=User(name="Vahid", surname="Saljooghi", username="vas", email="vas@example.com", password=pass1, permission=6)

6- db.session.add(u1)

7- db.session.commit()

# دستورات مفید برای فرد نگهدارنده وب اپلیکیشن:

User.\_\_table\_\_.drop(db.engine)

User.\_\_table\_\_.create(db.engine)

UserRequest.drop(db.engine)

UserRequest.create(db.engine)

User.query.all()

User.query.first()

User.query.get(id)

db.drop\_all()

db.session.rollback()

# امنیت:

حمله Cross-Site Request Forgery (CSRF یا XSRF):

1. مرورگرها برای اینکه نیاز به لاگین مجدد نباشد (راحتی کاربر) پس از لاگین کردن، کوکی رو که از طرف سرور فرستاده می شود را در حافظه نگه می دارند. به این ترتیب با رجوع مجدد به وبسایت خاص (که با نام میزبان، اسکیما و پورت مشخص می شود)، مرورگر کوکی مربوط به وبسایت را از حافظه خود بازیابی کرده و ضمیمه درخواست HTTP می کند. درصورت وجود و درست بودن کوکی، مستقیم وارد پنل کاربری و ... شده و دیگر درخواست ورود نام کاربری و رمز (احراز هویت یا authentication) را نمی کند.
2. بسیاری از عملیات هایی که در وبسایت ها مثلا با فشردن یک دکمه در محیط گرافیکی انجام می شود در نهایت به یک رشته متنی تبدیل شده و به سمت سرور ارسال می شود تا سرور مثلا در پایگاه داده متصل به خود تغییر را اعمال کند.
3. طبیعت وب این است که وقتی وبسایتی را باز و مرور می کنید، این وبسایت از طرف شما درخواست هایی را برای وبسایت های دیگر بفرستد که این کار از طریق آدرس های نهفته در درون وبسایت انجام می شود. مثلا در بسیاری از وبسایت ها، ویدئوهایی از وبسایت یوتیوب قرار داده (نهفته) شده است که می توان بدون مراجعه به وبسایت اصلی یوتیوب آن ویدئو را دید.

مثلا URL زیر که در واقعیت وجود ندارد حاوی درخواست زیر است: مبلغ 1000 ریال به مقصد اصغری انتقال یابد. این URL پس از احراز هویت شما (لاگین کردن) قابل اجرا می باشد.

<https://www.BankMelli.com/enteghal?mablagh=1000&maghsad=Asghari>

با در نظر گرفتن موارد ۱، ۲ و ۳ شخص ثالث می تواند درخواست های جعلی بدون رضایت و اطلاع شما از طرف شما بفرستد. به این ترتیب که مثلا لینک بالا رو برای شما ایمیل می کنند و اگر بر روی آن کلیک کنید درصورت لاگین بودن، مرورگر دیگر احراز هویت نکرده و مستقیم عملیات انجام می شود.

ممکن است با خود بگویید من روی همچین URL مشخصی کلیک نمی کنم. متقلبین هم برای اینکه خیلی واضح نباشد یک URL دیگر مثلا یک سایت جذاب برای شما می فرستند و داخل این سایت آدرس بالا را نهفته می کنند. مثلا در مورد زیر، URL مدنظر در tag اچ تی ام ال درخواست (بارگذاری) عکس نهفته شده است:

<img src="

<https://www.BankMelli.com/enteghal?mablagh=1000&maghsad=Asghari>">

این نوع از حمله با نام Cross-Site Request Forgery (CSRF یا XSRF) شناخته شده است که ترجمه اش می شود جعل درخواست HTTP از طریق وبسایت دیگر.

از آنجایی که طرف درخواست دهنده (قربانی) خود شما بوده اید(بدون اینکه مطلع) باشید اثری از جاعل برای ردگیری نمی ماند (درخواست با آی پی شما ارسال شده است).

روش ارسال داده ها در درون URL به نام روش درخواست GET (یکی از انواع درخواست پروتکل HTTP) شناخته می شود. و داده های ارسالی به صورت رشته را query string می گویند.

طراحان و توسعه دهنده های وبسایت های حساس معمولا از روش GET برای ارسال داده های حساس و انجام عملیات های حساس مانند حذف کاربر، اضافه کردن کاربر یا انجام تراکنش های بانکی استفاده نمی کنند. بلکه از نوع دیگری به نام POST بهره می گیرند. (از دیگر معایب ارسال داده های حساس با روش GET اینست که: قابل bookmark کردن و cache کردن می باشد. در نتیجه در history مرورگر باقی می ماند).

در روش POST داده ها به جای قرارگرفتن در URL در بدنه(Body) درخواست HTTP قرار می گیرند و به این ترتیب قابل Bookmark کردن و یا Cache کردن نمی باشند و محدودیت تعداد کاراکتر هم ندارد. از طرفی راهکاری جلوگیری کننده (countermeasure) از طرف سرورها ارائه شده است که درصورت ارسال درخواست به صورت POST حمله CSRF امکان پذیر نیست.

راهکار ارائه شده را با یک مثال بر روی میکروفریم ورک فلاسک در زیر توضیح می دهیم:

شکل زیر فرم ایجاد کاربر را نشان می دهد:



امکان ایجاد کاربر جدید فقط برای کاربر با سطح دسترسی ادمین و پس از احراز هویت امکان پذیر می باشد. اگر ارسال فرم بالا پس از تکمیل به روش GET بود، URL شبیه زیر تولید می شد و به سمت سرور ارسال می شد تا در طرف سرور اطلاعات کاربر در پایگاه داده ذخیره شود:

http://www.example.com/enteghal&

[name=Vahid&surname=Saljooghi&username=VAS&email=vas@example.ir&admin\_checkbox=y&moderator\_checkbox=y&submit=Create](mailto:name=Vahid&surname=Saljooghi&username=VAS&email=vas@example.ir&admin_checkbox=y&moderator_checkbox=y&submit=Create)

کافیست URL بالا را تغییر داده و نام و نام خانوادگی و ... فرد جدیدی را قراداد و با نهفته کردن این URL در درون یک وبسایت دیگر درخواستی غیرمستقیم به سرور ارسال کرد. به شرط اینکه کاربر با سطح دسترسی ادمین لاگین باشد و کوکی session در مرورگر ذخیره شده باشد، در صورت کلیک بر روی آدرس (جذاب، تله) ارسالی مثلا از طریق ایمیل، بدون اطلاع، حساب کاربری جدید ایجاد خواهد شد.

برای جلوگیری از حمله CSRF، سرور، توکنی تصادفی و پیچیده را به صورت پنهانی در درون فرم ها قبل از ارسال به سمت کلاینت قرار می دهد، پس از پرکردن فرم و ارسال(سابمیت) به سرور، سرور، توکن دریافتی را با توکن تولید کرده قبلی خود مقایسه می کند تا مطمئن شود که فرم پر شده جعلی نمی باشد و واقعا از طرف کابر پرشده است.

به سورس کد نظیر فرم بالا در زیر دقت کرده (رنگ خاکستری توکن نهفته از سمت سرور در درون فرم می باشد):



با هربار تازه سازی (refresh) توکن جدید در سمت سرور تولید و ارسال می شود. مثلا توکن های زیر بعد از ۳ بار تازه سازی توسط فلاسک ایجاد شده است:

IjU1YmJjODY5MjZhMDIwNjBhZmRhYzZmYjUzYzNkY2Y1NmVhZTU1ZjUi.D\_M-HA.rbcRpTmLHjpAH8w7-bDSvGABGpc

IjU1YmJjODY5MjZhMDIwNjBhZmRhYzZmYjUzYzNkY2Y1NmVhZTU1ZjUi.D\_M\_IQ.i4FhCQ9xPqjhZhKAYMBlS3YsTAU

IjU1YmJjODY5MjZhMDIwNjBhZmRhYzZmYjUzYzNkY2Y1NmVhZTU1ZjUi.D\_NBrA.14-luG8Hgo3XpWS8A8rt9K7A2N8

درس هایی که باید آموخت:

1. سعی کنید پس از استفاده از وبسایت های حساس، لاگ اوت کنید.
2. بر روی هر لینک ناشناسی کلیک نکنید.
3. اگرچه حمله CSRF با روش های ارائه شده از طرف سرورها و همچنین طراحان و توسعه دهندگان مجرب کمتر شده است ولی همچنان نمی توان مطمئن بود که آیا طراحان سیستم مقابله با آن را درنظر گرفته اند.

در حمله CSRF، سرور به کاربر اعتماد می کند و فرض می گیرد که درخواست های ارسالی کاربر از شخصی معتبر می آید.

حمله XSS (Cross-Site Scripting):

در این حمله کد جاوااسکریپت از طرف یک کاربر (هکر) با کلک و به صورت غیرمستقیم بر روی سرور می نشیند و به این ترتیب پس از درخواست توسط کاربری دیگر، کد تزریق شده بر روی سرور بر روی مرورگر کلاینت قربانی اجرا می شود. این کد مخرب می تواند به کوکی های ذخیره شده در مرورگر دسترسی پیدا کند. این نوع حمله از انواع تزریق کد به جای داده است.

در حمله XSS کاربر قربانی به سرور اعتماد می کند و جواب های ارسالی از سمت سرور را معتبر می داند.

یک راه جلوگیری اینست که برای کوکی های حساس فلگ HttpOnly را True کرد. با این کار مرورگر اجازه دسترسی کد جاوااسکریپت به کوکی حاوی اطلاعات حساس را نمی دهد.

appFlask.config['REMEMBER\_COOKIE\_HTTPONLY'] = True

appFlask.config['SESSION\_COOKIE\_HTTPONLY'] = True

راه دیگر فعال کردن مکانیزم اسکیپ کردن(scape) خودکار کد HTML توسط template engine (در اینجا Jinja2) می باشد. به این ترتیب کد مخرب امکان اجرا شدن در مرورگر قربانی را ندارد و صرفا به صورت رشته ای متنی نشان داده می شود.

appFlask.jinja\_env.autoescape = True

Captcha:

برای جلوگیری از حملاتی از نوع Denial of Service در مواردی مانند درخواست ریست کردن پسورد از کپچا استفاده شده است.

Password Policy:

برای این اپلیکیشن، کاربر مجبور است تا رمزی نسبتا پیچیده انتخاب کند.

1. Must contain at least one capital letter.
2. Must contain at least one small letter.
3. Must contain at least one number.
4. Must contain at least one of !@#$%&\*?<>{} characters.
5. Must be between 8-15 characters.

Cache-Control:

برای جلوگیری از استفاده از Back Button در مرورگر.

response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"

response.headers["Pragma"] = "no-cache"

response.headers["Expires"] = "0"

# تغییرات برای بهبود در آینده:

* اضافه کردن پردازش موازی برای ارسال ایمیل تا زمان انتظار کاربر ادمین کاهش یابد.

1. Air-gapped [↑](#footnote-ref-1)
2. File Transfer Web App [↑](#footnote-ref-2)
3. Socket Address [↑](#footnote-ref-3)
4. Upload [↑](#footnote-ref-4)
5. Download [↑](#footnote-ref-5)
6. Recipients [↑](#footnote-ref-6)
7. Account Management [↑](#footnote-ref-7)
8. Create [↑](#footnote-ref-8)
9. Delete [↑](#footnote-ref-9)
10. Edit [↑](#footnote-ref-10)
11. Block [↑](#footnote-ref-11)
12. Request Management [↑](#footnote-ref-12)
13. Cancel [↑](#footnote-ref-13)
14. Approve [↑](#footnote-ref-14)
15. Deny [↑](#footnote-ref-15)
16. Forward [↑](#footnote-ref-16)
17. Pending [↑](#footnote-ref-17)
18. Description [↑](#footnote-ref-18)
19. Request Date [↑](#footnote-ref-19)
20. Review Date [↑](#footnote-ref-20)
21. Reviewer Comment [↑](#footnote-ref-21)
22. Sidebar [↑](#footnote-ref-22)
23. Contacts [↑](#footnote-ref-23)
24. Encrypted [↑](#footnote-ref-24)
25. OTP [↑](#footnote-ref-25)
26. Submit [↑](#footnote-ref-26)
27. Validation [↑](#footnote-ref-27)