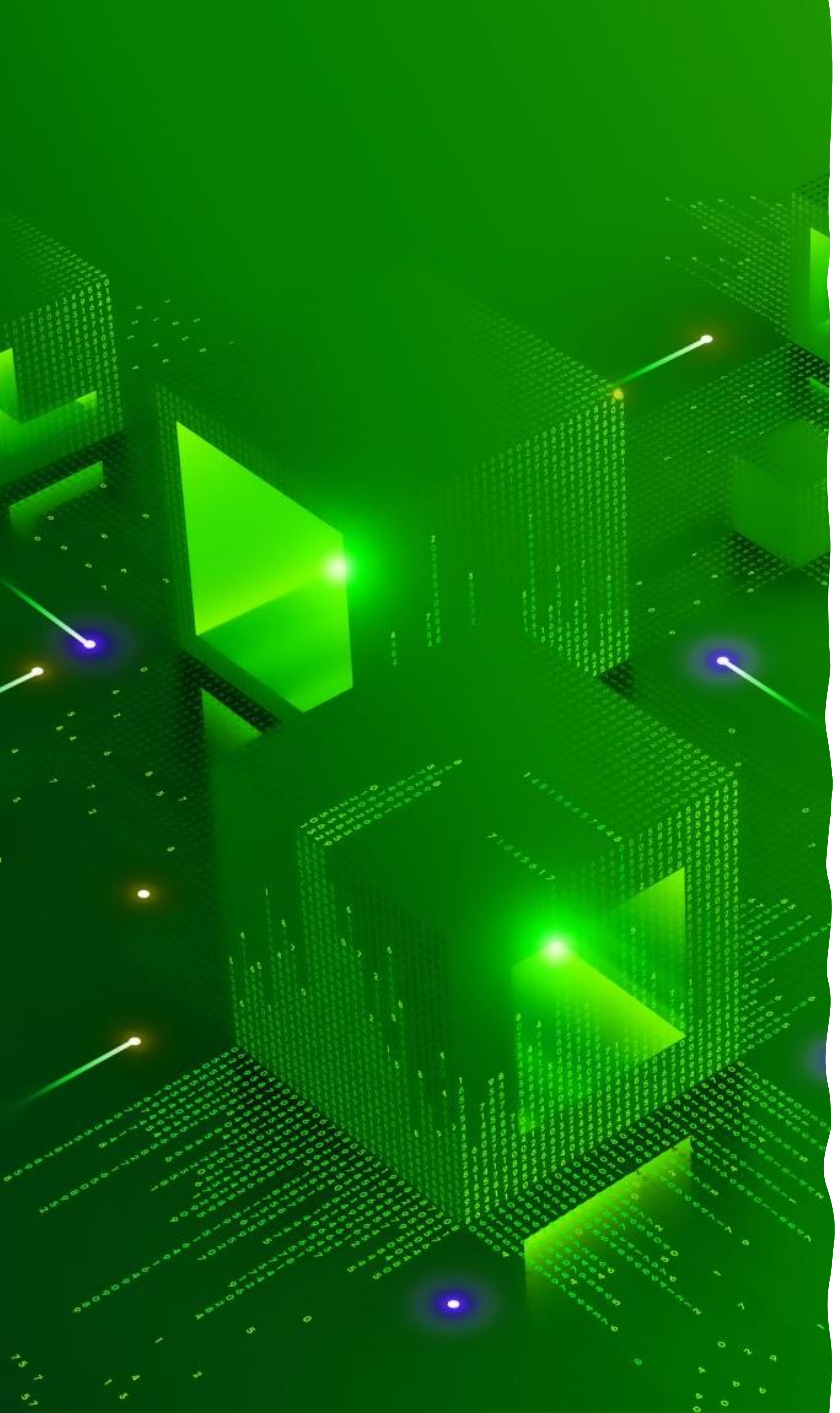
@harshbothra_



Bug Bounty Tactics & Wins for 2021!

By: Harsh Bothra

OWASP
JAIPUR

Introduction

- Core Pentester @Cobalt.io
- Lazy Bug Hunter @Synack @Bugcrowd
- Bugcrowd TOP 150 Hackers & MVP Q1 – Q2
- Author: Multiple Hacking Books
- Security Blogs @Medium
- Speaker @Multiple Security Conferences
- Poet | Writer | Learner

Agenda

- Bug Bounty Landscape
- Tactics for wins in 2021
- Account Takeovers
- 2FA Bypass
- Other Interesting Issues
- Tips & Tricks

Bug Bounty Landscape

Tactics for Wins in 2021

Account Takeovers

Logical Wins for 2021

@harshbothra_

Ways to Perform Account Takeovers

- CSRF
- XSS
- Broken Cryptography
- IDOR
- Session Hijacking
- Predictable Identifiers
- Security Misconfiguration
- Direct Request
- Missing Authorization Checks
- OAuth Misconfiguration
- Session Fixation

Case Studies

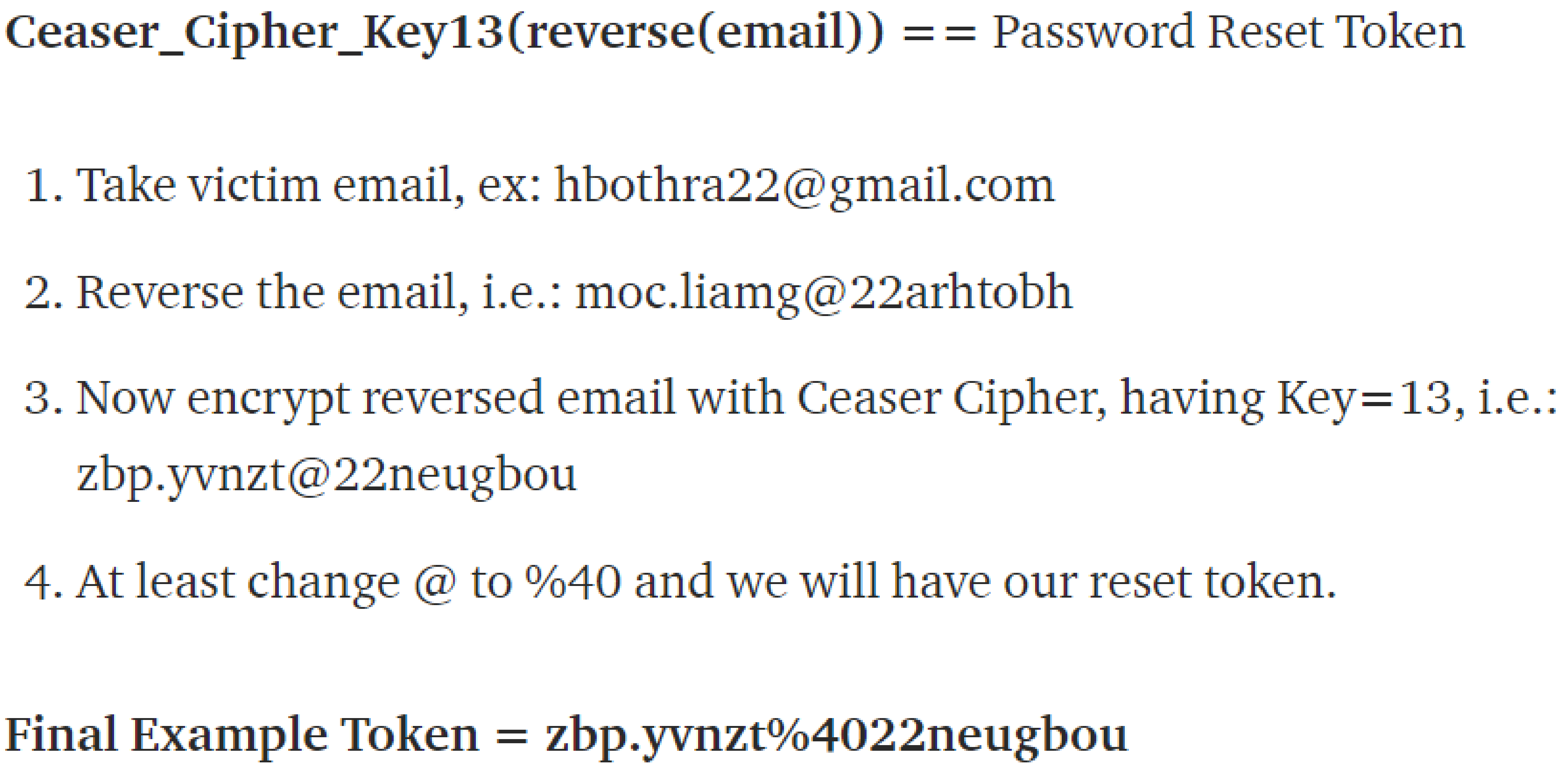
Broken Cryptography to ATO

Request New Pass. → Receive Unique Reset Link → Resets the Pass

Now, While resetting the password using **Reset Link,** I observed the only difference between these two Reset Links was: **1 and 2.**

Reset Link of Account 1: https://target.com/reset_password?token=zbp.nwavaqjbeptho%401+neugboufenu

Reset Link of Account 2: https://target.com/reset_password?token=zbp.nwavaqjbeptho%402+neugboufenu

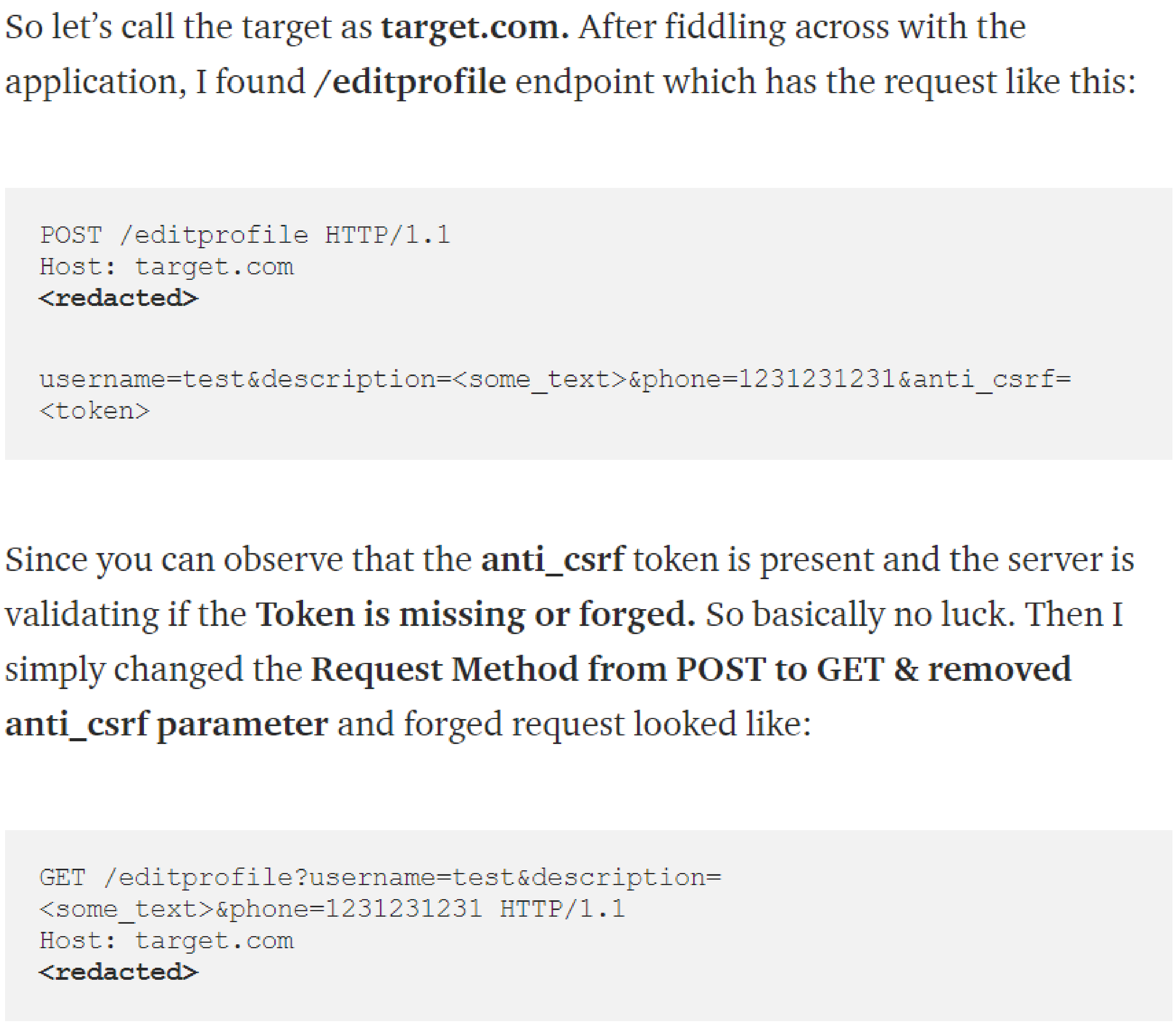


Ceaser_Cipher_Key13(reverse(email)) == Password Reset Token

1. Take victim email, ex: hbothra22@gmail.com
2. Reverse the email, i.e.: moc.liamg@22arhtobh
3. Now encrypt reversed email with Ceaser Cipher, having Key=13, i.e.: zbp.yvnzt@22neugbou
4. At least change @ to %40 and we will have our reset token.

Final Example Token = zbp.yvnzt%4022neugbou

CSRF & Client - Side Validation Bypass to ATO

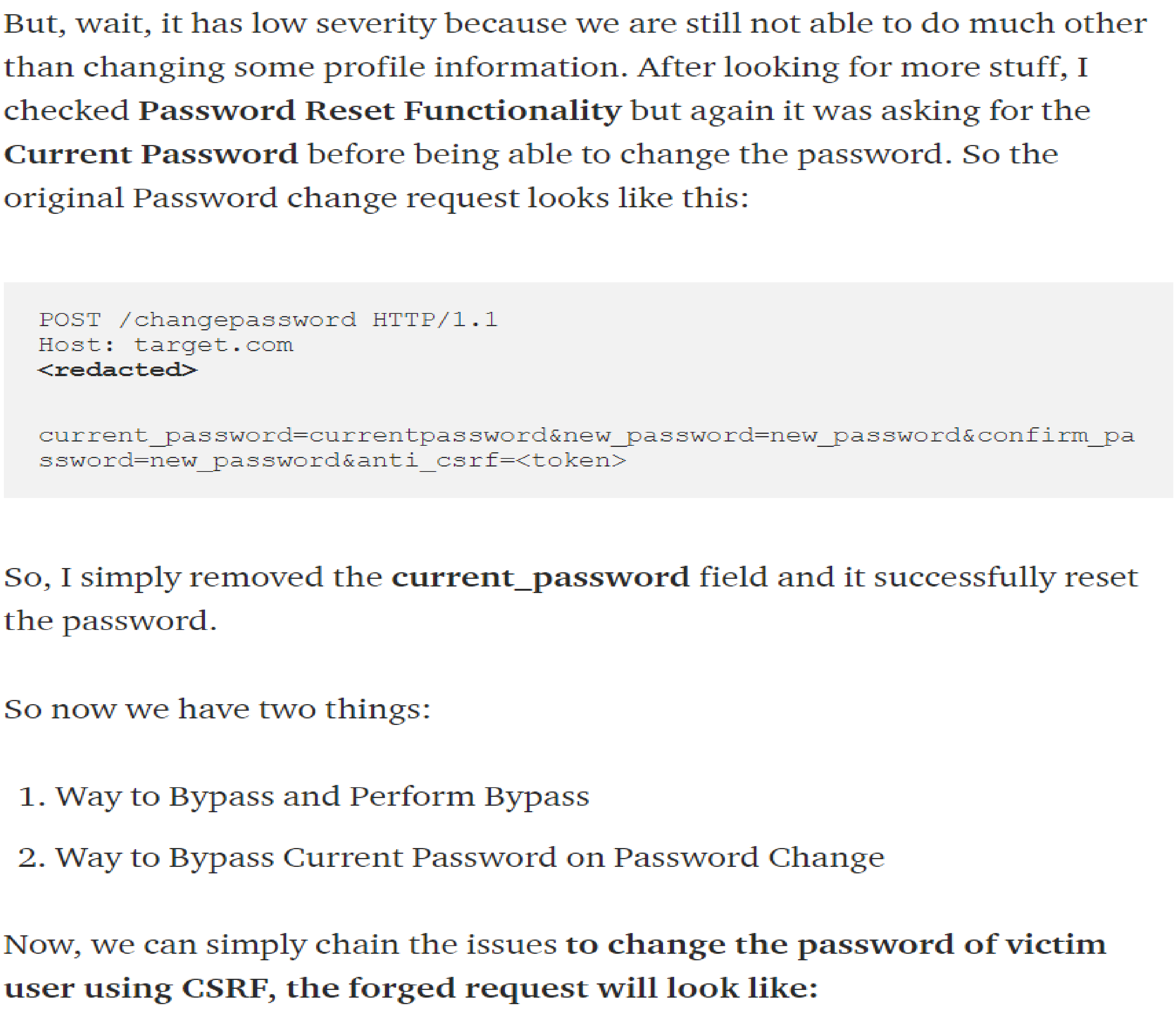


So let's call the target as **target.com.** After fiddling across with the application, I found **/editprofile** endpoint which has the request like this:

```
POST /editprofile HTTP/1.1
Host: target.com
<redacted>


username=test&description=<some_text>&phone=1231231231&anti_csrf=
<token>
```

Since you can observe that the **anti_csrf** token is present and the server is validating if the **Token is missing or forged.** So basically no luck. Then I simply changed the **Request Method from POST to GET & removed anti_csrf parameter** and forged request looked like:

```
GET /editprofile?username=test&description=
<some_text>&phone=1231231231 HTTP/1.1
Host: target.com
<redacted>
```



But, wait, it has low severity because we are still not able to do much other than changing some profile information. After looking for more stuff, I checked **Password Reset Functionality** but again it was asking for the **Current Password** before being able to change the password. So the original Password change request looks like this:

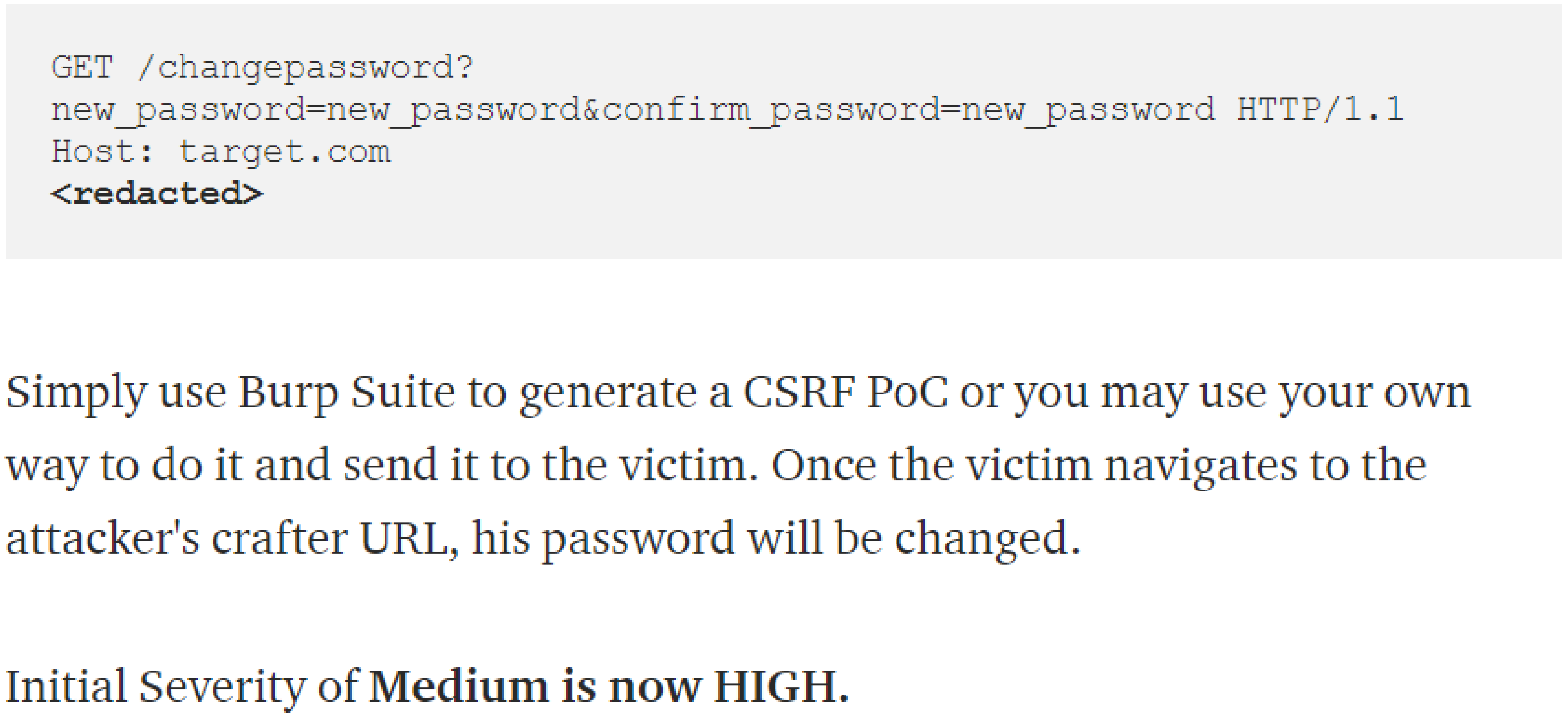
```
POST /changepassword HTTP/1.1
Host: target.com
<redacted>

current_password=currentpassword&new_password=new_password&confirm_pa
ssword=new_password&anti_csrf=<token>
```

So, I simply removed the **current_password** field and it successfully reset the password.

So now we have two things:

1. Way to Bypass and Perform Bypass
2. Way to Bypass Current Password on Password Change

Now, we can simply chain the issues **to change the password of victim user using CSRF, the forged request will look like:**

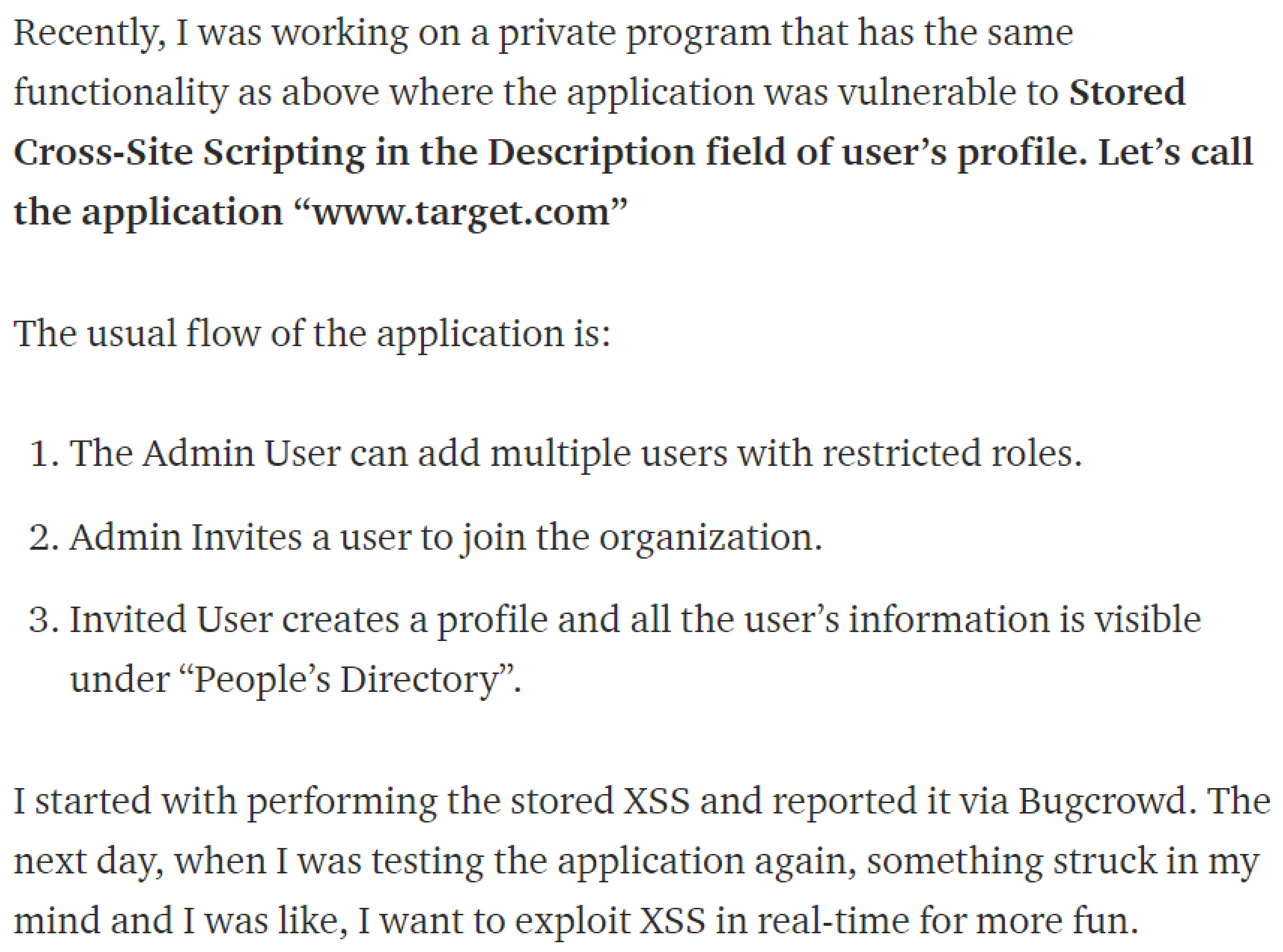


```
GET /changepassword?
new_password=new_password&confirm_password=new_password HTTP/1.1
Host: target.com
<redacted>
```

Simply use Burp Suite to generate a CSRF PoC or you may use your own way to do it and send it to the victim. Once the victim navigates to the attacker's crafter URL, his password will be changed.

Initial Severity of **Medium is now HIGH.**

Cross-Site Scripting to Admin Session Hijacking & Privilege Escalation

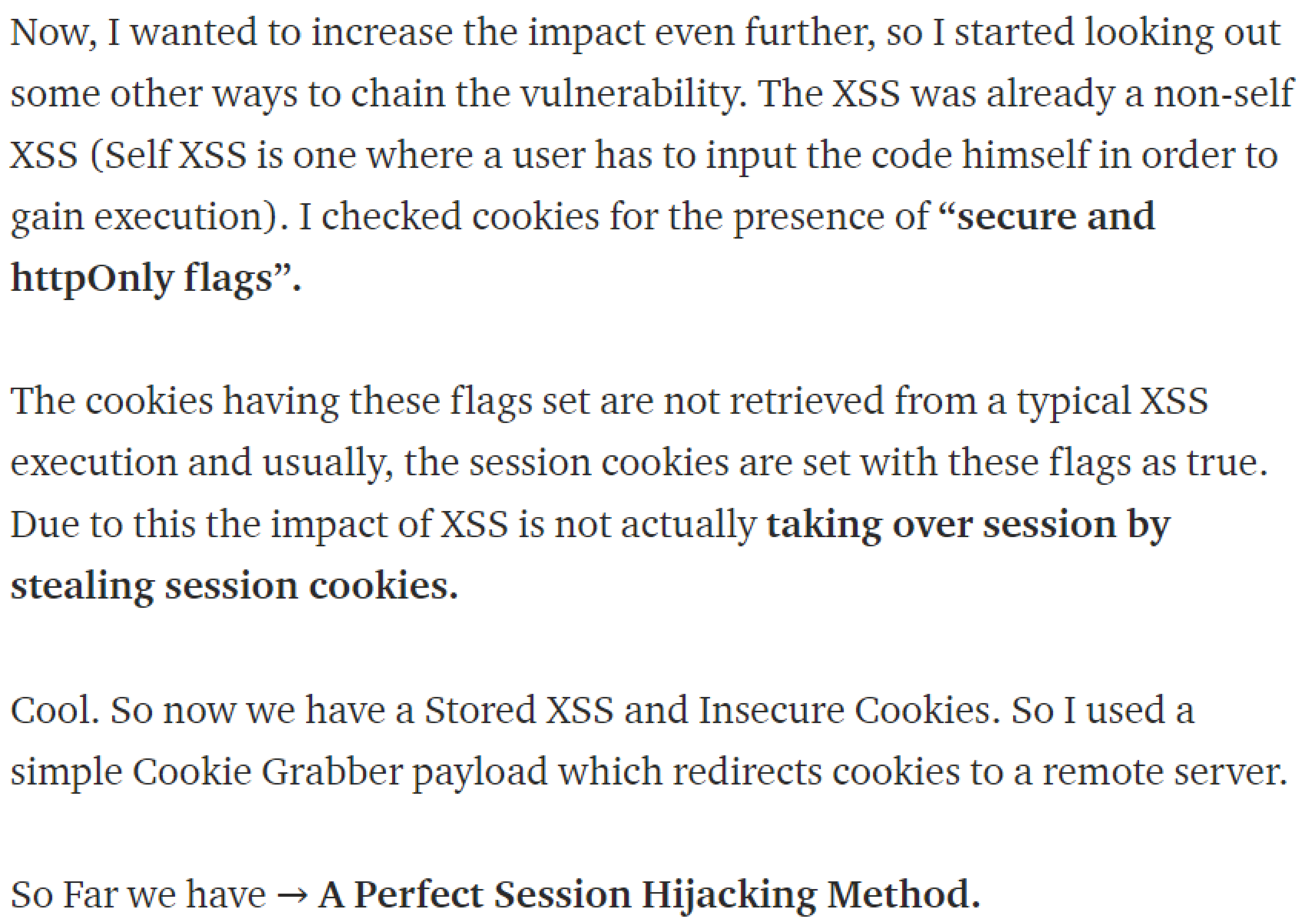


Recently, I was working on a private program that has the same functionality as above where the application was vulnerable to **Stored Cross-Site Scripting in the Description field of user’s profile. Let’s call the application “www.target.com”**

The usual flow of the application is:

1. The Admin User can add multiple users with restricted roles.
2. Admin Invites a user to join the organization.
3. Invited User creates a profile and all the user’s information is visible under “People’s Directory”.

I started with performing the stored XSS and reported it via Bugcrowd. The next day, when I was testing the application again, something struck in my mind and I was like, I want to exploit XSS in real-time for more fun.

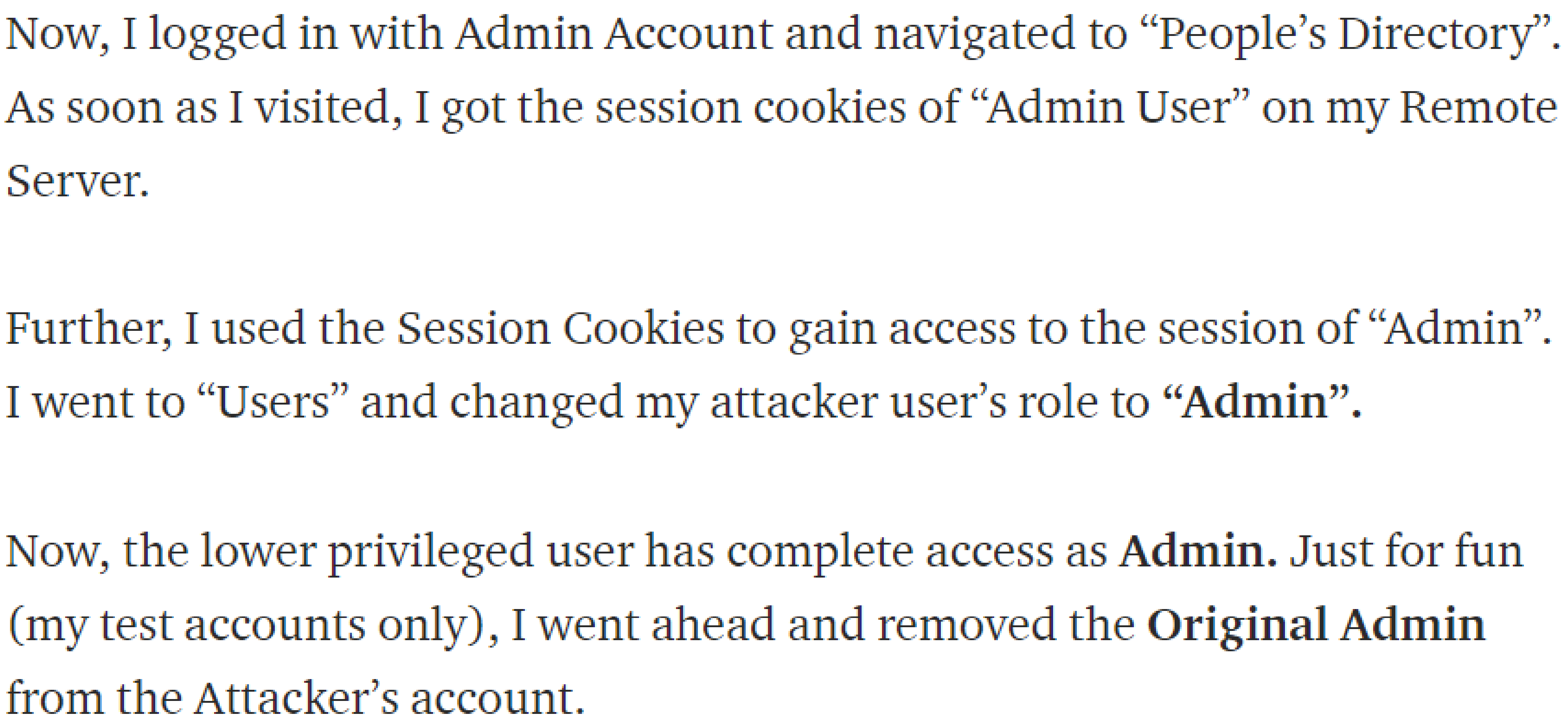


Now, I wanted to increase the impact even further, so I started looking out some other ways to chain the vulnerability. The XSS was already a non-self XSS (Self XSS is one where a user has to input the code himself in order to gain execution). I checked cookies for the presence of **“secure and httpOnly flags”.**

The cookies having these flags set are not retrieved from a typical XSS execution and usually, the session cookies are set with these flags as true. Due to this the impact of XSS is not actually **taking over session by stealing session cookies.**

Cool. So now we have a Stored XSS and Insecure Cookies. So I used a simple Cookie Grabber payload which redirects cookies to a remote server.

So Far we have → **A Perfect Session Hijacking Method.**



Now, I logged in with Admin Account and navigated to "People's Directory". As soon as I visited, I got the session cookies of "Admin User" on my Remote Server.

Further, I used the Session Cookies to gain access to the session of "Admin". I went to "Users" and changed my attacker user's role to **"Admin".**

Now, the lower privileged user has complete access as **Admin.** Just for fun (my test accounts only), I went ahead and removed the **Original Admin** from the Attacker's account.

IDOR in Cookies to Account Takeover

- Login as a victim user and capture the request with Burp.
- In Cookies section there was a **ROLE** parameter which has a two-digit value **00**.
- Create an admin account and observe that now **ROLE** value in cookies is **11**.
- Upon further inspection and mapping **User Role & Permission Matrix**. I observed that the application uses **binary bits** for role definition.
- **00 : User**
- **11 : Admin**

IDOR in Password Reset to ATO

- Password Reset page is Vulnerable to **Host Header Attack.**
- Request a password reset link with malicious origin.
- Victim will receive a password reset link with malicious origin like:

Original Link: https://original_target.com/reset/token/<token_here>
Spoofed Link: https://malicious_target.com/reset/token/<token_here>

- Now set up a logger at attacker controlled **malicious_target.com**
- Once the victim clicks on the password reset link, the token will be logged to **malicious_target.com**
- Token has no expiry and thus attacker can utilize the token to reset the password.

@harshbothra_

2FA Bypass Tactics

Easy Wins & More Bounty

We will look at this using following Mind Map

https://www.mindmeister.com/1736437018?t=SEeZOmvt01

Other Interesting Attacks to Look for in 2021

Tips & Tricks

Get in Touch At

- Website – https://harshbothra.tech
- Twitter - @harshbothra_
- Instagram - @harshbothra_
- Medium - @hbothra22
- LinkedIn - @harshbothra
- Speakerdeck - @harshbothra
- Email – hbothra22@gmail.com

Thank You