Broken Cryptography & Account Takeovers

By: Harsh Bothra

About Me!

- Cyber Security Analyst @Detox Technologies
- Synack Red Teamer
- Bugcrowd Top 150 & MVP 2020 Q1-Q2
- Author – Hacking: Be a Hacker with Ethics (GoI R'cmd.)
- Author – Mastering Hacking: The art of Information Gathering & Scanning
- Speaker @Multiple Security Confs & Chapters
- Blogger @Medium | Youtube @Detox Technolgoies
- Poet | Writer | Learner

Agenda

- Broken Cryptography 101
- Endpoints to Test for Broken Cryptography
- Account Takeovers 101
- Ways to Test for Account Takeovers
- Real Life Findings – Case Studies
- Hack’0’Hacktricks
- Q/A

@harshbothra_

Broken ~~Cryptography~~ 101

100 ft overview of Cryptography

- A practice to encrypt data transmitted between two or more parties to ensure secure communication.
- Various encoding & encryption algorithms are available to perform cryptography.
- Cryptography is of two parts: **Symmetric & Asymmetric**
- Cryptography is widely used and is one of the base of computer applications.
- Cryptography can be seen in various parts of application like password reset token, encrypted path, hardcoded secrets, cookies, API Keys, Authentication Token and others.

Broken Cryptography

Less Travelled Road : Where to Look

- Session Cookies
- Encoded Paths & Parameters
- Hardcoded Secrets in JS Files
- Password Reset Links
- CSRF Tokens
- Authenticity Tokens
- Encrypted Data
- Username/Passwords
- and many other endpoint depending upon the application use-case.

Account Takeovers

Ways to Perform Account Takeover

@harshbothra_

CSRF

XSS

Broken Cryptography

IDOR

Session Hijacking

Session Fixation

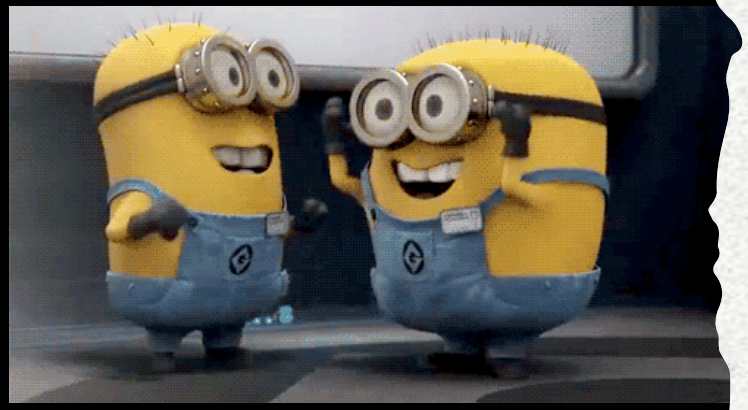
Predictable Identifiers

Security Misconfiguration

Direct Request

Missing Authorization Checks

OAuth Misconfiguration



Case Studies

@harshbothra_

Broken Cryptography to Account Takeover

Now getting back to the application, the usual Password Reset flow includes:

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

The second thing I observed is the length of the reset token was = No. of characters in email and %40=@.

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

@harshbothra_

CSRF & Client Side Validation Bypass to Account Takeover

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

Scenario

- 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**

@harshbothra_

IDOR in Password Reset to Account Takeover

Scenario

- 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.

Q/A's are welcome...

Get in Touch

- Twitter : @harshbothra_
- LinkedIn : @harshbothra
- Instagram : @harshbothra_
- Medium : @hbothra22
- Website : https://harshbothra.tech
- Slides : https://www.speakerdeck.com/harshbothra
- Email : hbothra22@gmail.com

Thanks... ☺