Torn Discord Account Linking Vulnerability Disclosure

TL;DR: Torn has a vulnerability in its process of linking Torn and Discord accounts allowing impersonation of Torn users on Discord servers and some third-party websites that support Discord-based login. Users should avoid clicking on links that go to `https://torn.com/discord.php` to avoid being impacted by this vulnerability.

Contents

1 Introduction

In compliance with Torn's bug bounty policy (see figure 1 in the Appendix) and common Coordinated Vulnerability Disclosure policy, this is a public disclosure of an unresolved vulnerability in Torn's system to link Discord and Torn accounts together. This disclosure comes on the heels of a 30 day embargo/waiting period after first reporting the issue and a general lack of response from Torn staff and developers and a lack of willingness to fix this non-critical vulnerability (see figure 2 in the Appendix). Torn uses a commonly used authorization specification known as OAuth2 to implement this feature to link Discord and Torn accounts. The data coming from this feature is commonly used in third-party, Torn-related tools and Discord bots (e.g. YATA, Tornium, Giving Tree, etc.) for account verification in Discord servers (see section 2 for those of you who want more technical details). However, Torn's implementation of this features appears to meet the minimum requirements of the specification and the minimum shown by Discord's implementation of this feature in their quickstart/documentation [4]. However, this means that Torn is missing a few of the highly recommended details mentioned in the specification(s) to prevent certain security issues. For example, this vulnerability would allow malicious users to send you a URL (such as `https://www.torn.com/discord.php?code=lUbv9hX241gEDWgc5igzxmGBKZT7yt`) that would be able to link their Discord account to your Torn account (making their Discord account to be shown as your Torn account after verification with one of various bots). This would work the other way around as well with a malicious user requesting the URL from you after Discord authentication to access your Discord account details (such as your email used for Discord, messages, etc.) provided a leak of a secret password-like value by Torn (the Discord application's client_secret).

Previously, this vulnerability would have been worse. I can't find the bug report for this, but about a year ago (from what I remember) there was a separate (CSRF) vulnerability in this system that allowed account linking without any user input/interaction required. For example, all that's required would be an image on someone's profile that has the Discord linking URL as its source and a user could be automatically linked to someone else's account without their knowledge. This is why defense in depth is important as the recommended implementation details prevent attacks in case one security practice/implementation fails.

1.1 Is This Being Exploited?

There's no way for me to know if this is being actively exploited or has been in the past given the difficulty in collaborating with Torn developers/staff. But there have been previous instances of users being able to read messages and/or receiving push notifications for messages in servers (specifically faction leadership servers) they should not and are not in which could have been caused by this vulnerability. There have also been instances of users accidentally and non-maliciously linking their Discord account to other people's due to this vulnerability. This could indicate that this vulnerability has potentially been exploited in the past. However, once again, there's no way for me to check this.

1.2 How Does This Impact Me?

This shouldn't actively impact individuals unless they'd already clicked one of these links or had shared this link previously. Assuming Torn does not come out with a fix/patch for this vulnerability, primarily users shouldn't follow a link to `https://torn.com/discord.php` from a third party (especially if there's a code value included in the URL) and click the button to verify your Torn nickname. Instead to link your Torn account with your Discord account, you can visit the Discord link at the top navigation bar on Torn and follow the links Torn's George bot will send to you on Discord. Also, you shouldn't share the URL that Discord provides after you authenticate through them as this theoretically allows malicious users to view certain Discord-related account details while appearing to be Torn.

Server admins that rely upon verification bots, especially for faction leadership servers, may be impacted as this would allow malicious users to impersonate another user for the purposes of verification (especially if verification isn't checked continuously). And there's no easy way to verify the accuracy of verification.

This vulnerability if exploited could possibly be reflected to other tools such as Tornium and UHC depending on how the tools are developed potentially resulting in leaked API keys, leaked private faction data (e.g. member stats, vault movement), etc. Once again, there's nothing for users to do, and for developers of these tools there's not much that can be done to prevent this other than disabling Discord-based authentication/authorization and/or preventing sensitive data such as API keys from being viewable.

1.3 Coordinated Disclosure Policy

Coordinated Vulnerability Disclosure (CVD) is the process of disclosing a software/hardware vulnerability to the responsible parties while giving them sufficient time to resolve the issue before publicly releasing information regarding the vulnerability. In this case, as seen in the section 3.1, Torn staff weren't responding to mails and Chedburn felt that

> there are ways we could shore it up and make it fully secure against every possibility, but this isn't banking software, it's just account-discord verification.

And as Torn doesn't feel that it's important to fix this and as staff fail to respond to mails regarding this, this public advisory has been/will be published 30 days from original notification (April 13th). Torn doesn't state an explicit timeframe for public disclosure only to

> provide us a reasonable amount of time to resolve the issue before any disclosure to the public or a third-party [1]

and neither Chedburn nor Bogie responded adequately to questions regarding this, so I've chosen a common CVD public disclosure timeframe of 30 days (many other companies and government agencies also state 45 days and 90 days for comparison).

2 Technical Details

This section is more for developers and people interested in the specifics of this vulnerability... Torn's

OAuth flow is started with George bot (or a server's administrators/management bots) sending the user a link to a Discord OAuth link for George bot/Torn:

```
https://discord.com/oauth2/authorize?client_id=441210177971159041&redirect_uri=
https%3A%2F%2Fwww.torn.com%2Fdiscord.php&response_type=code&scope=identify
```

This authorization link includes the required parameters such as client_id, redirect_uri, and scope. However, this link does not indicate any of the optional security features included in the base OAuth2 specification (RFC 6749) or any of the extensions of the specification (such as RFC 7636 for Proof of Key Exchange) to prevent a Cross-Site Request Forgery (CSRF) exploit are supported by Torn despite being implemented by Discord [3] and strongly recommended by the IETF (the organization that developed the OAuth2 specification).

> While Discord does not require the use of the 'state' parameter, we support it and highly recommends that you implement it for the security of your own application and data.

In fact, the OAuth2 specification and its extensions (as well as various comments made by their authors) state that some/any CSRF is required to be implemented by the client [2] and [5]:

> The client MUST implement CSRF protection for its redirection URI. This is typically accomplished by requiring any request sent to the redirection URI endpoint to include a value that binds the request to the user-agent's authenticated state (e.g., a hash of the session cookie used to authenticate the user-agent). The client SHOULD utilize the "state" request parameter to deliver this value to the authorization server when making an authorization request.
>
> Clients MUST prevent Cross-Site Request Forgery (CSRF). In this context, CSRF refers to requests to the redirection endpoint that do not originate at the authorization server, but a malicious third party (see Section 4.4.1.8. of [RFC6819] for details). Clients that have ensured that the authorization server supports Proof Key for Code Exchange (PKCE, [RFC7636]) MAY rely on the CSRF protection provided by PKCE. In OpenID Connect flows, the nonce parameter provides CSRF protection. Otherwise, one-time use CSRF tokens carried in the state parameter that are securely bound to the user agent MUST be used for CSRF protection (see Section 4.7.1).

2.1 An Attack Vector

1. The malicious user starts the OAuth process by authenticating with Discord granting the Torn's Discord application consent to user data, but does not finish the authentication process on Torn retaining the validity of the authorization code.
2. Somehow, another Torn user is convinced to access this page while logged into Torn and press the button to link accounts.
3. Attacker's authorization code is exchanged for an access token. Using this access token, Torn retrieves the attacker's Discord ID and stores the ID under the Torn user.
4. Attacker is now authorized to access certain third-party tools and Discord servers with automatic verification as the authorized user.

Theoretically this vulnerability could also be used to perform clickjacking by chaining together other vulnerabilities or if someone more knowledgeable than me looked at this. This could allow a user to unknowingly link their Torn account to the malcious user's Discord account.

Additionally, as stated in Section 1, the previous CSRF vulnerability in the OAuth flow would allow malicious users to inject an image or button or some other element that links to the malicious user's Discord OAuth callback URL with a pre-defined authorization code. Upon user interaction with the element, the resulting GET request would already be authenticated and could theoretically automatically link a user to the malicious user's Discord ID without any feedback from the user as there wouldn't be have been any CSRF protection or user-required actions that would prevent this.

2.2 State Parameter

The state parameter is included in the base OAuth2 specification to provide a way to maintain state between the client (the Torn website/app) and the resource server (Torn's servers). This state can be and is intended to be used to prevent Cross Site Forgery Attacks upon the resource server as quoted below [1].

> state
> RECOMMENDED. An opaque value used by the client to maintain
> state between the request and callback. The authorization
> server includes this value when redirecting the user-agent back
> to the client. The parameter SHOULD be used for preventing
> cross-site request forgery as described in Section 10.12.

The state parameter when properly generated and stored by the client would prevent an invalid redirect and the aforementioned attack. The generated state value would be stored on the client before redirecting to the Discord OAuth page and sent with the authorization code after redirecting back to Torn. This would permit the client to validate the state paramter against the local copy to verify the request for OAuth and to open the authorization was triggered by itself.

2.3 Proof Key of Code Exchange

The Proof Key of Code Exchange specification (RFC 7636) extends the base OAuth2 specification to provide a method of verifying the authorization is done by the same client that is attempting to exchange the authorization code for the access token. This isn't as important for Torn as the client is confidential and Discord doesn't allow clients to make OAuth-related API calls without including the client secret, so this could mostly only prevent malicious browser extensions from intercepting authorization codes in addition to CSRF protection. You can find more regarding PKCE in the specification [6].

3 Appendix

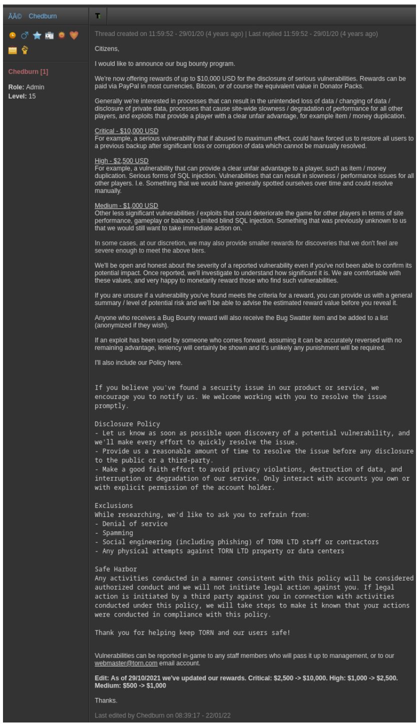
3.1 Timeline

Screenshots of messages regarding this to Torn staff can be found below.

1. 4/13/24 - Vulnerability discovered
2. 4/13/24 - Original Report sent to CloudJumper
3. 4/18/24 - Follow-up message sent to CloudJumper
4. 4/18/24 - Forwarded original report to IceBlueFire
5. 4/22/24 - Response received from Chedburn via IceBlueFire (see figure 2)
6. 4/23/24 - Additional details sent to Chedburn via IceBlueFire
7. 5/04/24 - Follow-up sent to IceBlueFire
8. 5/04/24 - Response received from IceBlueFire indicating that this won't be fixed
9. 5/13/24 - Public disclosure published

3.2 Screenshots

Below you can find some screenshots of relevant information in Torn in the event that Torn changes the text. Better resolution version of these screenshots and additional screenshots of the full conversations can be found at .



Chedburn [1]

Role: Admin
Level: 15

Thread created on 11:59:52 - 29/01/20 (4 years ago) | Last replied 11:59:52 - 29/01/20 (4 years ago)

Citizens,

I would like to announce our bug bounty program.

We're now offering rewards of up to $10,000 USD for the disclosure of serious vulnerabilities. Rewards can be paid via PayPal in most currencies, Bitcoin, or of course the equivalent value in Donator Packs.

Generally we're interested in processes that can result in the unintended loss of data / changing of data / disclosure of private data, processes that cause site-wide slowness / degradation of performance for all other players, and exploits that provide a player with a clear unfair advantage, for example item / money duplication.

Critical - $10,000 USD
For example, a serious vulnerability that if abused to maximum effect, could have forced us to restore all users to a previous backup after significant loss or corruption of data which cannot be manually resolved.

High - $2,500 USD
For example, a vulnerability that can provide a clear unfair advantage to a player, such as item / money duplication. Serious forms of SQL injection. Vulnerabilities that can result in slowness / performance issues for all other players. I.e. Something that we would have generally spotted ourselves over time and could resolve manually.

Medium - $1,000 USD
Other less significant vulnerabilities / exploits that could deteriorate the game for other players in terms of site performance, gameplay or balance. Limited blind SQL injection. Something that was previously unknown to us that we would still want to take immediate action on.

In some cases, at our discretion, we may also provide smaller rewards for discoveries that we don't feel are severe enough to meet the above tiers.

We'll be open and honest about the severity of a reported vulnerability even if you've not been able to confirm its potential impact. Once reported, we'll investigate to understand how significant it is. We are comfortable with these values, and very happy to monetarily reward those who find such vulnerabilities.

If you are unsure if a vulnerability you've found meets the criteria for a reward, you can provide us with a general summary / level of potential risk and we'll be able to advise the estimated reward value before you reveal it.

Anyone who receives a Bug Bounty reward will also receive the Bug Swatter item and be added to a list (anonymized if they wish).

If an exploit has been used by someone who comes forward, assuming it can be accurately reversed with no remaining advantage, leniency will certainly be shown and it's unlikely any punishment will be required.

I'll also include our Policy here.

```
If you believe you've found a security issue in our product or service, we
encourage you to notify us. We welcome working with you to resolve the issue
promptly.

Disclosure Policy
- Let us know as soon as possible upon discovery of a potential vulnerability, and
we'll make every effort to quickly resolve the issue.
- Provide us a reasonable amount of time to resolve the issue before any disclosure
to the public or a third-party.
- Make a good faith effort to avoid privacy violations, destruction of data, and
interruption or degradation of our service. Only interact with accounts you own or
with explicit permission of the account holder.

Exclusions
While researching, we'd like to ask you to refrain from:
- Denial of service
- Spamming
- Social engineering (including phishing) of TORN LTD staff or contractors
- Any physical attempts against TORN LTD property or data centers

Safe Harbor
Any activities conducted in a manner consistent with this policy will be considered
authorized conduct and we will not initiate legal action against you. If legal
action is initiated by a third party against you in connection with activities
conducted under this policy, we will take steps to make it known that your actions
were conducted in compliance with this policy.

Thank you for helping keep TORN and our users safe!
```

Vulnerabilities can be reported in-game to any staff members who will pass it up to management, or to our webmaster@torn.com email account.

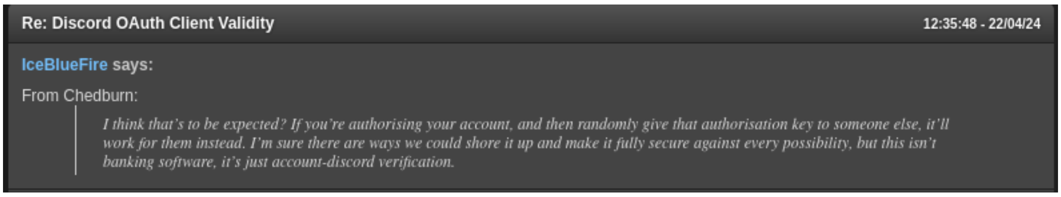
Edit: As of 29/10/2021 we've updated our rewards. Critical: $2,500 -> $10,000. High: $1,000 -> $2,500. Medium: $500 -> $1,000

Thanks.

Last edited by Chedburn on 08:39:17 - 22/01/22

Figure 1: Torn's Bug Bounty policy

A current copy of the above screenshot can be found on Torn's forums.



Re: Discord OAuth Client Validity 12:35:48 - 22/04/24

IceBlueFire says:

From Chedburn:

> *I think that's to be expected? If you're authorising your account, and then randomly give that authorisation key to someone else, it'll work for them instead. I'm sure there are ways we could shore it up and make it fully secure against every possibility, but this isn't banking software, it's just account-discord verification.*

Figure 2: Ched's first response

4 References

References

[1] RFC6749: 4.1.1. Authorization Request
`https://datatracker.ietf.org/doc/html/rfc6749#section-4.1.1`

[2] RFC6749: 10.12. Cross-Site Request Forgery
`https://datatracker.ietf.org/doc/html/rfc6749#section-10.12`

[3] Discord OAuth State and Security
`https://discord.com/developers/docs/topics/oauth2#state-security`

[4] Discord OAuth Authorization Code Flow
`https://discord.com/developers/docs/topics/oauth2#authorization-code-grant`

[5] IETF OAuth Security Topics Draft 27 `https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics-27#name-protecting-redirect-based-f`

[6] RFC7636: Proof Key for Code Exchange by OAuth Public Clients
`https://datatracker.ietf.org/doc/html/rfc7636`