RADICALLY OPEN SECURITY

Penetration Test Report

Relaycorp, Inc.

V 1.0
Amsterdam, April 24th, 2025
Public

Document Properties

Client	Relaycorp, Inc.
Title	Penetration Test Report
Target	• Design review of Despacito protocol
Version	1.0
Pentesters	Sipke Mellema, Andrea Jegher
Authors	Sipke Mellema, Andrea Jegher, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	April 10th, 2025	Sipke Mellema, Andrea Jegher	Initial draft
0.2	April 24th, 2025	Marcus Bointon	Review
1.0	April 24th, 2025	Marcus Bointon	1.0

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between March 17, 2025 and March 26, 2025, Radically Open Security B.V. carried out a penetration test for Relaycorp, Inc.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of Work

The scope of the penetration test was limited to the following target:

- Design review of Despacito protocol

The scoped services are broken down as follows:

- Review of Despacito Design: 2 days
- **Total effort: 2 days**

1.3 Project Objectives

ROS will perform a review of the Despacito protocol with Relaycorp in order to spot possible security issues and implications before any further work. To do so ROS will access the public draft proposal and advise Relaycorp on the protocol design.

1.4 Timeline

The security audit took place between March 17, 2025 and March 26, 2025.

1.5 Results In A Nutshell

By design, as the general concerns section (page 6) shows, Despacito only protects against request flooding attacks, and not against application bugs that could exhaust resources. The proxy can become a security liability because it manipulates web traffic, especially if it's used for multiple websites. Parsing differences might introduce request smuggling, cache poisoning, and timing attacks, depending on the implementation. Adapting it to HTTP2, HTTP3, and WebSockets might prove a challenge because they keep an open connection for the duration of the HTTP session.

Some methods for issuing credentials can be a challenge to implement non-finding NF-004 (page 6). App integrity, device attestation, proof of collateral, and user presence tests are by far the most cumbersome and require lots of customization. Using biometrics to validate a user can also become a privacy nightmare. Using a CAPTCHA might be a solution, but as mentioned in the finding, research finds that bots have become better at them than humans. Modern solutions use a combination of browser fingerprinting and proof of work / proof of space tests, which is partially what Despacito proposes. The question becomes whether it can implement such controls in a holistic manner.

The use of an authorization server might cause non-browser clients to break non-finding NF-005 (page 7). In addition, the use of signing with a certificate needs to be implemented correctly to prevent exhaustion. There is also mention of an external authorization server, which would need to be protected from DoS attacks via another method.

2 Suggestions

In this section we list some of the key areas to work on for the protocol improvement.

2.1 NF-003 — General concerns

Beside the points we mentioned concerning specific methods and the authorization service, the design itself raised general concerns for both security and feasibility.

The Despacito application is designed to prevent application-level DoS attacks, but only the types of attack that rely on sending a large number of requests to exhaust a server. This is the most common form of DoS attack, according to the extensive research performed by the protocol's designer. Despacito will not stop application-layer DoS attacks like the Billion Laughs attack, zip bombs, or other types of attack that don't require a large number of requests.

Despacito will become a man-in-the-middle, as it will inspect requests and manipulate HTTP responses before passing them on, making it a security liability. This service will make a great target for attackers, especially if it's used for multiple websites.

Using separate application logic to intercept requests on certain headers and values can cause vulnerabilities in itself. Reverse proxies have been known to allow request smuggling, if they interpret an HTTP request in a different way to a backend service. Such parsing differences can also allow cache poisoning and timing attacks.

The HTTP landscape is very complex, and the proposed solution wants to intercept HTTP requests to check their content. Implementing this solution will be a challenge for HTTP2 and HTTP3, since they keep the connection open for multiple sessions, as well as for WebSockets and WebTransport, which transform HTTP into a binary tunnel.

2.2 NF-004 — Comments on proposed Despacito solutions

The draft specs list multiple methods for issuing credentials. A credential in this context is some value issued to a user that identifies them as a trusted client. The mentioned methods are listed below with comments on their security properties.

Slowing machines – Proof of Work & Proof of Space

Implementing PoW or PoS can be tricky if you consider differences in devices. An attacker can have a massive server to calculate proofs very fast. If a regular user has to calculate the same proof, it could take a long time, which may impact their browsing experience.

Human interaction – CAPTCHA

Cloudflare has a product that's similar to what Despacito is trying to achieve, called Turnstile. In their article about the service, they mention research on CAPTCHA solutions. Bots are getting better at circumventing CAPTCHAs, and in some cases they have become better than humans. This is why modern solutions use a combination of browser fingerprinting and PoW/PoS tests. Turnstile uses "..a series of in-browser tests, checking browser characteristics, native

browser APIs, and asking the browser to pass lightweight tests (ex: proof-of-work tests, proof-of-space tests) to prove that it's an actual browser".

Device specific – App Integrity & Device attestation

Using the Play Integrity API, or a similar API, to check if a device or app has been tampered with is something that — as far as we know — is done within an app, and this can't be done by a proxy. One can program an app to send a signed bundle to the proxy to verify it, but this would need to be programmed separately in the app. Obviously, this solution will only work on specific devices.

Very cumbersome – Proof of Collateral & User presence test

Proof of Collateral is a cryptocurrency term where, in this context, a user stores funds somewhere, and only gets it back when they don't perform a malicious action. This is hard to implement as you'd need to build a cryptocurrency infrastructure and have all users use it. They then need to put money in it, and they'll be forced to use that money to use your website. Verifying cryptocurrency transactions can also cause a significant delay in traffic.

A user presence test is mentioned in the specs. Hardware-based validation is referenced, using devices such as a Yubikey or the fingerprint reader on a smartphone. Validation like this requires specific hardware and a lot of configuration to make it work. In addition, using biometrics to validate a user can become a privacy nightmare.

2.3 NF-005 — Authorization service concerns

The Despacito draft mentions the use of an authorization service inside its proxy. This service will be responsible for authenticating the client. The idea is that the requests of an authenticated client can be throttled based on their token. In some cases, a short-lived certificate is returned to the user to sign data with.

We raised the following concerns with respect to this design:

- This approach breaks non-browser clients (e.g. CLI tools, IoT devices, scrapers) unless universally implemented.
- Verification can be expensive for the server, while the client can reuse a single signature (e.g. signing a large file once).
- If an external identity server is used to issue tokens, there is nothing stopping the web application from blocking unauthenticated requests to allow similar functionality.
- The specs also mention that Despacito can use an external authorization server. This would, of course, mean that the external server can't in turn be protected with Despacito. In any case, the auth service itself must handle DoS protection to avoid being overwhelmed by access requests.

Consider specifying how the user is going to use the certificate. For example, the user can send a certificate hash with each request, enabling rate limiting via a cache. Clients could also include a signed nonce, ensuring one computation per request. Note that the proxy will need to maintain a database (or cache) of all active certificates.

3 Future Work

- **Retest the complete proposal**

  Perform further reviews after finalizing the first version protocol document and implementation.

4 Conclusion

Despacito is a rate limiting protocol for reverse proxies. At the time of writing, the Despacito protocol is in a draft state, so implementation details are missing. It introduces an authorization service that uses different methods of proof to limit the requests that users can make. When authorization is granted, it serves a short-lived certificate to the client.

Our review highlights concerns and areas that can be improved. Implementing the protocol might introduce new vulnerabilities, and its design did not take into consideration the modern HTTP standards that keep a connection open during the session. Of the multiple methods for issuing credentials, only a few are deemed feasible in the short term.

ROS recommends trying to create a list of simple use cases to understand under which circumstances Despacito would be effective. Because of the complexity of the HTTP protocol, we recommend reducing the scope. We suggest mixing techniques, and to include browser profiling, like Cloudflare's Turnstile does, because no single method mentioned in the spec has proved to be effective on its own.

Finally, we want to emphasize that security is a process that must be continuously evaluated and improved – this penetration test is just a one-time snapshot. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing Team

Sipke Mellema	Sipke is an experienced pentester with nine years of experience in the field. His specialty is crystal-box web security and providing long-term advice on security improvements. Over the years he has branched out to the cloud, IoT, ICS, network security, and security management.
Andrea Jegher	Andrea is a security engineer with experience in offensive security and secure development. He started his career focusing on Web Application as a developer and as a penetration tester. Later he studied other fields of security such as cloud, networks and desktop applications.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.