Zesty.io Incident Report

Date: 02/06/26
Affected Instance: All
Zesty.io Service: Webengine

Summary

On February 6th, 2026 starting at approximately 8:11 AM PT we began observing increased 504 HTTP webengine responses. A programmatic scan for archival type files(.zip, .rar, .tar, .gz, .7z) lead to an increase of traffic resulting in a database exhausting its connections. Causing subsequent requests to timeout. With requests timing out they began to pile up. Leading to a cascade effect of exhausting the webengine instance limit.

Resolution

While the WAF rate limiting caught and prevented a large portion of the requests the attack was distributed. As one client would be flagged and blocked another client would start. Allowing for enough traffic to enter the origin to lead to the connection timeouts.

We have updated our set of rules to include the following file types which are no longer allowed to be present in a URL. These file extensions were not previously blocked as they are common methods for distributing archives, but following this incident, we have determined that customer usage should be limited to distribution via our media API.

The programmatic traffic subsided at approximately 8:56 AM PT. Allowing Webengine to fully service all requests.

Future Prevention

WAF Rule Updates

The primary defense for malicious traffic is the WAF(Web Application Firewall). We maintain a set of rules which determine if a request should be denied or rate limited. We have updated our set of rules to include the following file types which are no longer allowed to be present in a URL.

`[.zip, .rar, .tar, .gz, .7z, .db, .jar, .war]`

Introduce Domain Cache

We are looking to migrate our domain resolution lookup logic, which determines if a requested domain is serviceable by Zesty, from a relational database to an in-memory database. Our primary constraint with a relational database is maximum concurrent connections. By moving to an in-memory database we will be able to increase our maximum connections from 4,000 to 65,000. Significantly increasing the amount of requests we can service at any one moment. This change will also have a follow-on benefit of reducing latency. As in-memory databases are significantly faster than relational databases. Although it will come with a trade-off of increasing complexity on ensuring database record data is always the freshest by needing to add additional functionality for updating the in-memory database when the source records change.

Increased Instance Limit

Once we have updated the backing domain data database from relational to in-memory we will be able to increase our instance limits. As the instance limit and database connection limit need to be considered together. There would be no point in trying to service additional requests if downstream resources are exhausted.