#HackerKast 13 Bonus Round: FlashFlood - JavaScript DoS

#HackerKast 13 Bonus Round: FlashFlood - JavaScript DoS - Author not stated, WhiteHat Security.

- Published: date not stated
- Original: <https://web.archive.org/web/20160403035045/https://www.whitehatsec.com/blog/hackerkast-13-bonus-round/>
- Current location: <https://web.archive.org/web/20160809010756/https://www.whitehatsec.com/blog/hackerkast-13-bonus-round/>
- Preserved from: https://web.archive.org/web/20160809010756/https://www.whitehatsec.com/blog/hackerkast-13-bonus-round/ (live) on 2026-08-10
- Capture timestamp: 20160403035045
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

In this week's HackerKast bonus footage, I wrote a little prototype demonstrator script that shows various concepts regarding JavaScript flooding. I've run into the problem before where people seem to not understand how this works, or even that it's possible to do this, despite multiple attempts at trying to explain it over the years. So, it's demo time! This is not at all designed to take down a website by itself, though it could add extra strain on the system.

What you might find though, is that heavy database driven sites will start to falter if they rely on caching to protect themselves. Specifically Drupal sites tend to be fairly prone to this issue because of how Drupal is constructed, as an example.

It works by sending tons of HTTP requests using different paramater value pairs each time, to bypass caching servers like Varnish. Ultimately it's not a good idea to ever use this kind of code as an adversary because it would be flooding from their own IP address. So instead this is much more likely to be used by an adversary who tricks a large swath of people into executing the code. And as Matt points out in the video, it's probably going to end up in XSS code at some point.

Anyway, check out the code here. Thoughts are welcome, but hopefully this makes some of the concepts a lot more clear than our previous attempts.