ha.ckers.org web application security lab

ha.ckers.org web application security lab - Author not stated, ha.ckers.org.

- Published: date not stated
- Original: <http://ha.ckers.org/blog/20070901/recursive-request-dos/>
- Preserved from: http://ha.ckers.org/blog/20070901/recursive-request-dos/ (stored) on 2026-08-09
- Capture timestamp: 20071002081245
- 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.

ha.ckers.org web application security lab - Archive » Recursive Request DoS

[[image: image]](http://www.webappsec.org/) Paid Advertising [[image: web application security lab]](http://ha.ckers.org/)

Recursive Request DoS

As a follow-on to my post on hacking intranets by using the websites (not web browsers), it occurred to me there is a potential for a DoS situation with that technique. First let me explain how it wouldn't work so that I can explain how it may work. If you see a URL that looks something like this:

> http://www.whatever.com/url=http://www.whatever.com/url=

You only get one iteration of the script calling itself. Of course you could chain them together and maybe get a few dozen requests out of one request. That's fairly bad on system resources, but nowhere near as bad as it could be. Let's take another example where once you submit a request it creates a session key, like so:

> http://www.whatever.com/url=1234567890

There are a few ways that that session key could be created. It could be based on time, it could be a counter, or it could be a hash of something. In the case of a hash you're going to have a really hard time doing anything because you have to predict what a URL that contains a hash would be. But let's say it's something predictable like time or a counter, and that I could re-request the same

URL over and over without caching. Maybe the key is stored in a DB and not flushed. Then there may be a situation where you could cause a recursive DoS condition.

If you knew the next request was going to end up being the key “1234567891″ and you could tell that request to point anywhere, you’d point it to the URL:

> http://www.whatever.com/url=1234567891

That would make the machine connect back to itself, which would make it connect back to itself and so-on. Each one would tie up system resources as well as keep the sockets open on the machine until they timed out. So a single request could end up forcing the web server to connect back to itself hundreds of times (probably a function of how slow the process was as well as max connections and timeout speed). That’s probably not too interesting and fairly uncommon, but it may be worth mentioning in case someone else can come up with something interesting there.

This entry was posted on Saturday, September 1st, 2007 at 1:45 pm and is filed under Webappsec. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or t rackback from your own site.