The Impact of Maintainers on Package Ecosystem Security

Alex Bellon, Security Engineering Intern (Summer 2020)

Note: This data was collected between Jun-Aug 2020

Attacks on the supply chain (i.e. services and software used to create and build projects) are becoming more and more common: a 430% increase in the past 12 months, according to the Sonatype 2020 State of the Software Supply Chain Report. Securing the supply chain is usually an afterthought - if it occurs at all - as companies are usually more focused on locking down everything that *they* write and create. The supply chain, specifically dependency packages, are often easier to compromise than directly attacking a codebase since a lot of damage can be done with some simple social engineering or leaked credentials.

The `event-stream` and `eslint` npm package compromises are just two of many recent incidents where attackers targeted dependency packages as a way to harvest sensitive information from a larger userbase. While these attacks can happen (and have happened) in any ecosystem, they are more common in the npm ecosystem. This is, in part, due to the large number of micropackages in npm and how commonplace it is for packages to have lots of dependencies: one study, Security Issues in Language-based Software Ecosystems, found that the average npm package has 86.55 packages in its dependency tree (which includes both direct and transitive dependencies). Additionally, there is no separation between packages that are installed as dependencies; they can all interact with each other.

As part of my internship I took a deeper look at how much risk there is in the top packages of different ecosystems, specifically factors that are on the development and maintenance side. I collected data for the most depended-on packages in npm (Node/JS), PyPI (Python) and Cargo (Rust) and found the following results:

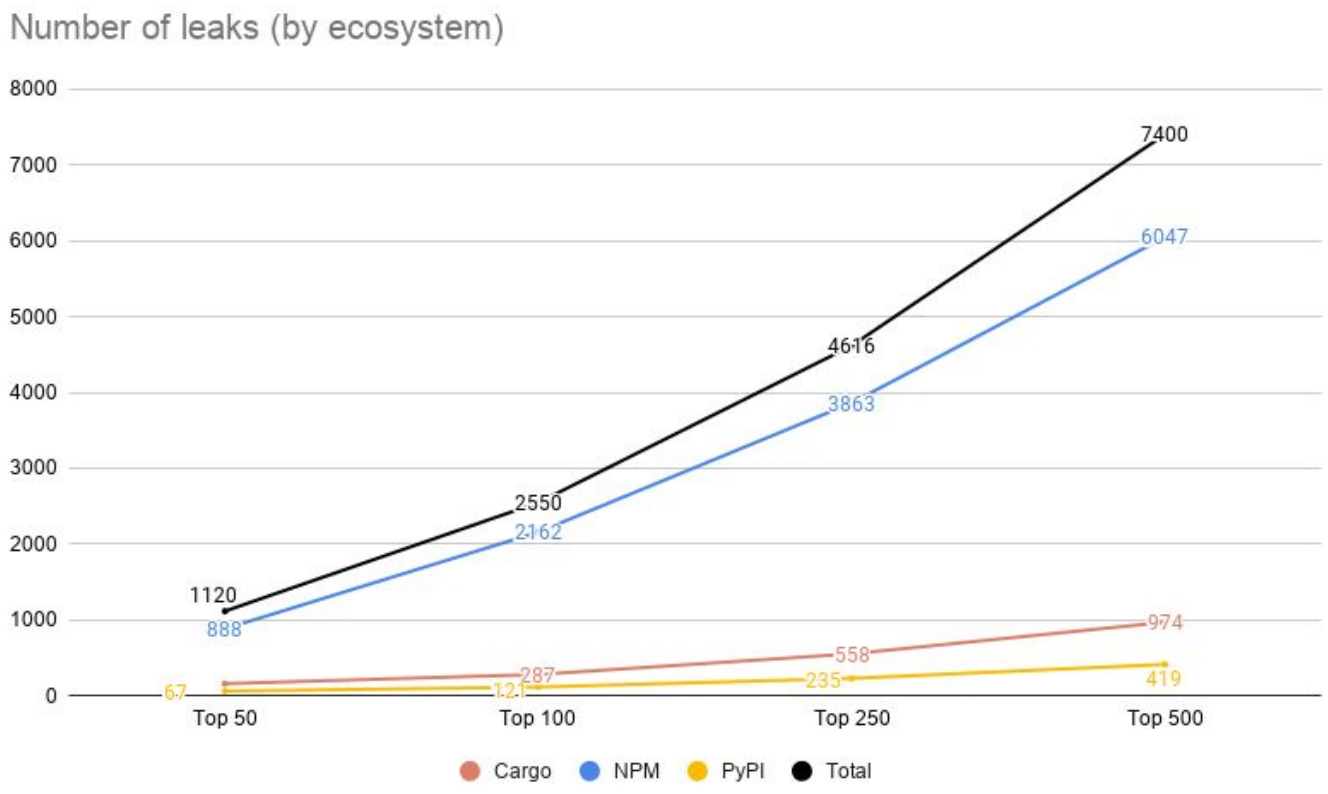
Overall maintainer credential leaks

One of the first things that I looked at were package maintainers' HaveIBeenPwned leaks, as a way to assess the viability of the "Account Takeover" threat model defined in Small World with High Risks: A Study of Security Threats in the npm Ecosystem. HaveIBeenPwned, the data break leak aggregator that powers Firefox Monitor, is a website that allows users to check if their passwords and other account credentials have been compromised in a data breach or password dump. Since many maintainers use personal emails for their npm/PyPI/Cargo accounts (more on this later), any password reuse between the package manager account and other online accounts could be exploited using credentials from these data leaks. For example, in 2018, the `eslint` incident (which affected approximately 4500 npm accounts) was a result of an `eslint` maintainer's npm account being compromised. In 2017, security researcher ChALkeR published a post in which they demonstrated that 13% of npm accounts were using weak or leaked credentials. Finally, Small World with High Risks: A Study of Security Threats in the npm Ecosystem found that only 20 maintainers need to be compromised to gain control of more than half of

the npm ecosystem due to the amount of inter-dependencies between packages. All of this just goes to show how powerful compromising a maintainer's account can be.

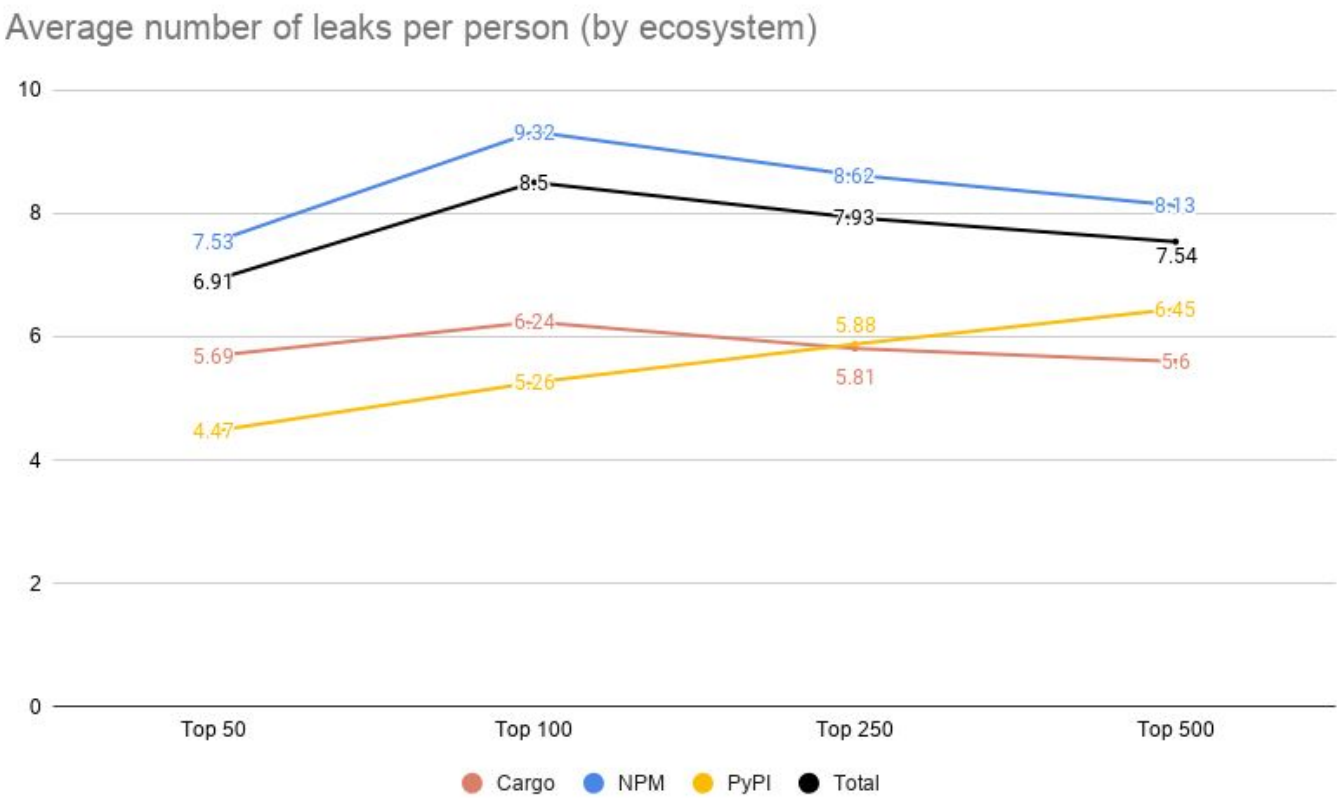
For the top 50, 100, 250 and 500 most depended-on packages of each ecosystem, the total number of HaveIBeenPwned (HIBP) leaks for all maintainers of all packages are shown below (one leak is one instance of a user's email being found in a breach):

	Top 50	**Top 100**	**Top 250**	**Top 500**
Cargo leaks	165	287	558	974
npm leaks	888	2162	3863	6047
PyPI leaks	67	121	235	419
Total leaks	1120	2550	4616	7400



These numbers show how large the (possible) attack surface is, but they are also skewed by the amount of data provided by each ecosystem's registry. For example, npm often lists more maintainers than the other two ecosystems, and many PyPI packages are maintained by a foundation or company so they only have one "maintainer" email listed. To account for the differences in the number of maintainers in each ecosystem, I also looked at the average number of HIBP leaks per person for the different ecosystems:

	Top 50	Top 100	Top 250	Top 500
Cargo avg leaks/person	5.69	6.24	5.81	5.6
npm avg leaks/person	7.53	9.32	8.62	8.13
PyPI avg leaks/person	4.47	5.26	5.88	6.45
Total avg leaks/person	6.91	8.5	7.93	7.54



Even adjusted for the number of maintainers, npm still has the highest average number of leaks per person, followed by Cargo and PyPI.

It's important to note that these HIBP leaks do not represent a guaranteed attack vector, as it's completely possible that these maintainers use different passwords for every account, or have since changed these breached passwords. But, out of the thousands of maintainers with credentials leaks, there probably *are* some who have not taken these countermeasures, and as we saw in Small World with High Risks: A Study of Security Threats in the npm Ecosystem, it only takes a few compromised accounts to do serious damage. This was explicitly shown in ChALkeR's "Gathering weak npm credentials" post, when they were able to gain publish access on 14% of npm packages by exploiting package maintainers' password reuse.

The attack surface introduced by these credential leaks can be minimized if maintainers use complex and unique passwords for their maintainer accounts (if not for all of their online accounts). Crates.io (by way of GitHub), npm and PyPI all forbid users from creating accounts using passwords that have been found in HaveIBeenPwned breaches, which makes it easier to pick more secure passwords. Additionally, maintainers can enable 2FA/MFA on their accounts to reduce risk, which Crates.io (by way of GitHub), npm and PyPI all support.

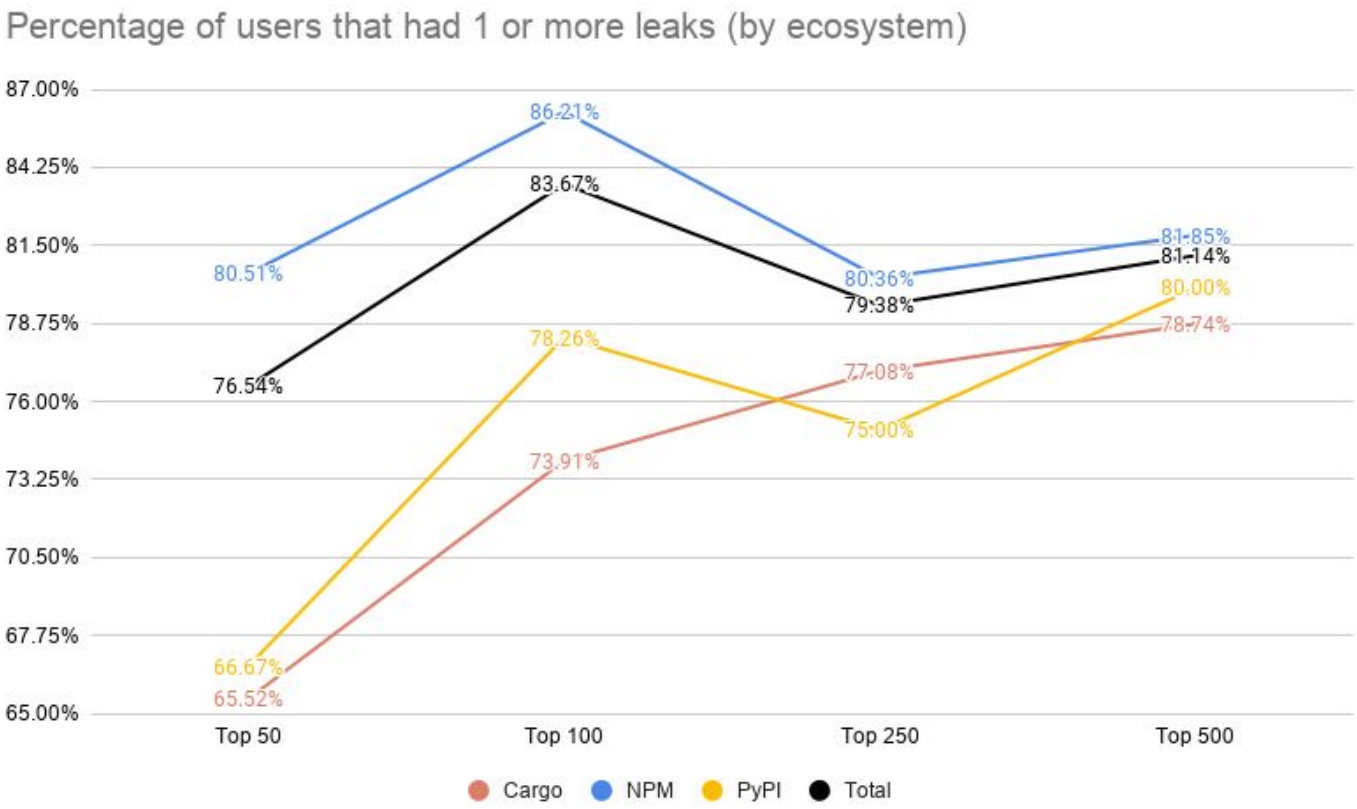
TL;DR: Overall maintainer credential leaks

- The maintainers of the top 100 most depended-on packages in the npm ecosystem cumulatively have thousands of leaks on HaveIBeenPwned (compares to hundred for Cargo and PyPI). These leaks are an indicator that a maintainer could be compromised if they reuse passwords in these leaks or have the credentials for their email accounts they use for packages leaked.
    - The maintainers of the top 100 npm packages have 9.32 leaks on average. Cargo has 6.24 average leaks per maintainer, and PyPI has 5.26 average leaks per maintainer.

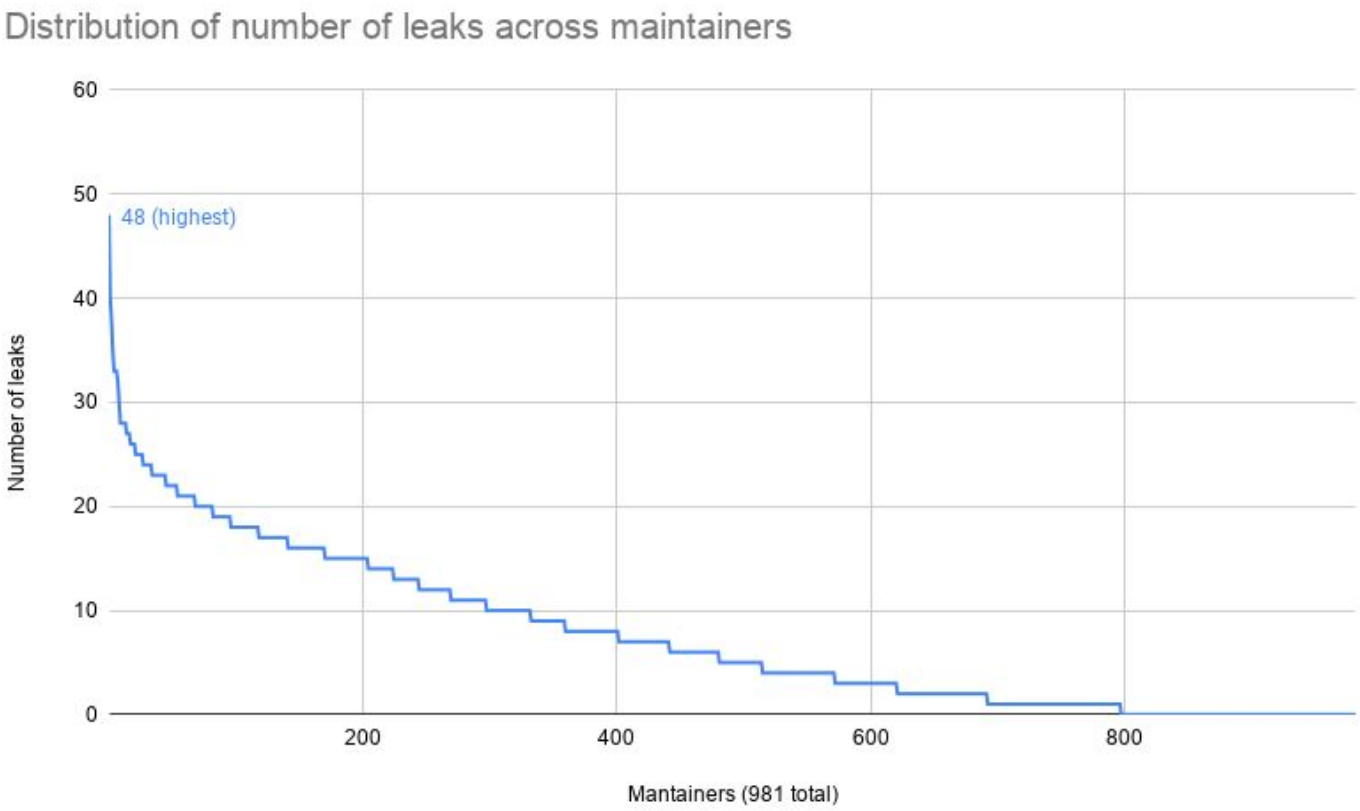
Maintainers with at least one credential leak

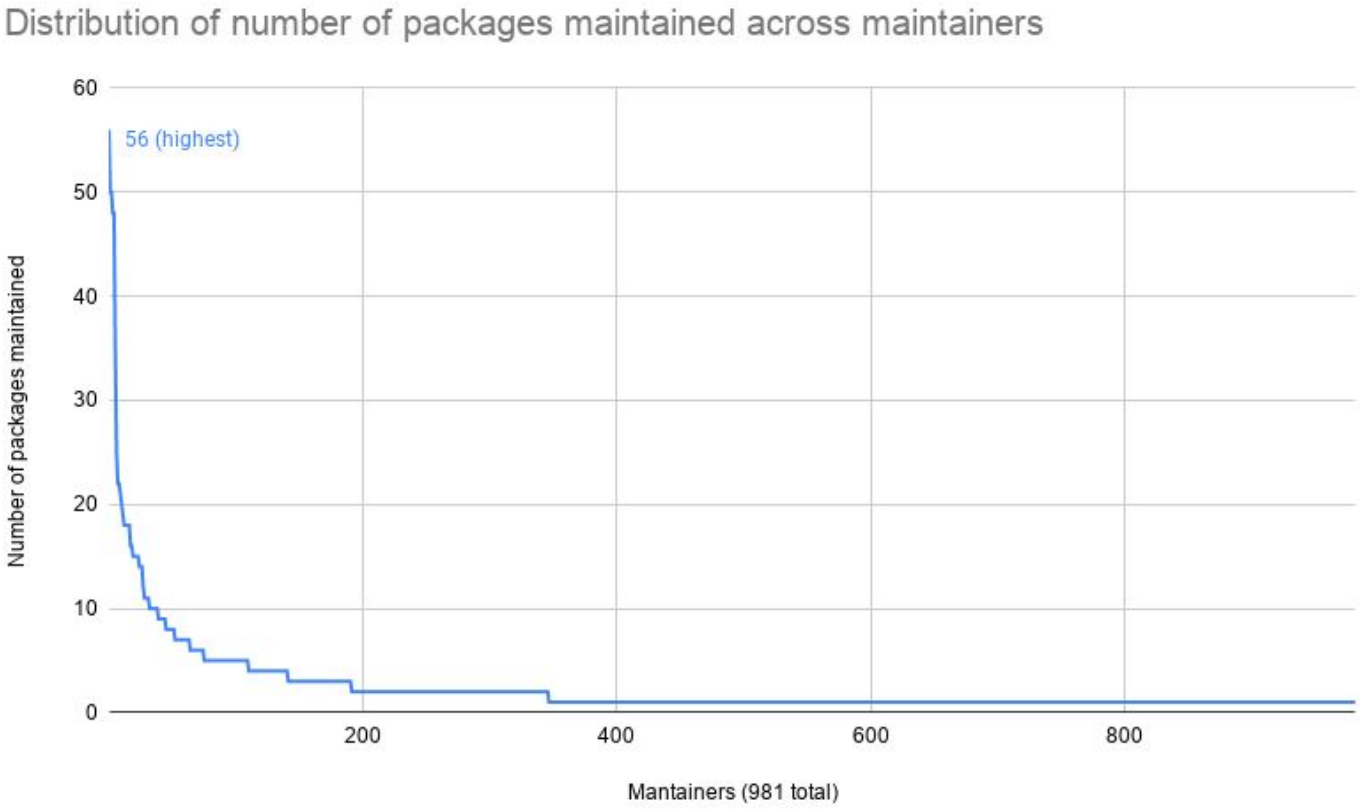
After looking at the ecosystems generally, I also did some further investigation into the amount of maintainers with leaks, and what kind of emails they were using. This table shows how many of the unique maintainers for each ecosystem had some number of HaveIBeenPwned leaks (compared to the total amount of unique maintainers):

	Top 50	**Top 100**	**Top 250**	**Top 500**
Cargo maintainers with >0 leaks	19	34	74	137
Cargo maintainers	29	46	96	174
npm maintainers with >0 leaks	95	200	360	609
npm maintainers	118	232	448	744
PyPI maintainers with >0 leaks	10	18	30	52
PyPI maintainers	15	23	40	65
Total maintainers with >0 leaks	124	251	462	796
Total maintainers	162	300	582	981



The numbers of maintainers with at least one leak didn't seem to correlate with either the age of the ecosystem or the amount of packages that each of these maintainers had. PyPI was created in 2000, npm in 2009 and Cargo in 2014, so npm's higher numbers don't look to be the result of it being the oldest ecosystem (and therefore the one with longest amount of time for maintainers to have leaks) or the youngest ecosystem (and therefore might not have as mature security practices or guidelines) of the three. Additionally, I compared the number of leaks for each unique maintainer and the number of packages they were responsible for, to see if perhaps maintainers who worked on more packages had more leaks. I found that between the top 50 maintainers with the most leaks and the top 50 maintainers with the most packages, there were only 4 maintainers in common. The top 25 maintainers from each category had no maintainers in their intersection, with the first common maintainer being found in the top 28 of each category. Additionally, the number of packages per maintainers drops off more sharply and quickly than the number of leaks per maintainer. The graphs of both distributions can be seen below:





Finally, I took the maintainers that had some number of HIBP leaks and looked at what type of email address they were using:

(This only includes emails that had > 0 leaks)

	Top 50	**Top 100**	**Top 250**	**Top 500**
Cargo Gmail emails	10	16	41	66
npm Gmail emails	52	108	200	349
PyPI Gmail emails	2	4	8	20
Total Gmail emails	64	127	247	433
Cargo .edu emails	1	1	1	3
npm .edu emails	2	3	4	5
PyPI .edu emails	0	0	0	0
Total .edu emails	3	4	5	8
Cargo other emails	8	17	32	68
npm other emails	41	89	156	255
PyPI other emails	8	14	22	32
Total other emails	57	120	210	355

As I mentioned earlier, the majority of maintainers sign up for npm/Cargo/PyPI accounts with their personal Gmail, which they likely also use for the rest of their online accounts. If they were to reuse credentials between their maintainer accounts and other online accounts associated with their personal email, a breach of these credentials could allow an attacker to compromise their package.

When the 981 emails I looked at were sorted by the number of leaks, this is what the 20 emails with the most amount of leaks looked like:

- Number of leaks ranged from 26-48
- 16 (80%) were Gmail addresses
- 4 (20%) were email addresses from personal domains

TL;DR: Maintainers with at least one credential leak

- For the top 100 most depended-on packages in each ecosystem, the percentage of maintainers with at least one leak on HaveIBeenPwned was 86.21% for npm, 78.26% for PyPI and 73.91% for Cargo.
    - This did not seem to be a result of the age of the ecosystem nor the number of packages owned by each maintainer.
- For every ecosystem, the majority of email accounts for the maintainers with at least one leak on HaveIBeenPwned were Gmail accounts.
    - 80% of the top 20 emails with the most number of leaks (of all 981 emails I looked at) were Gmail accounts, the other 20% were from personal domains.

Maintainers with no credential leaks

On the other hand, there were 185 emails with 0 leaks, and this is what those emails looked like:

- 84 (45.4%) were email addresses from personal domains, of which:
    - 30 were made specifically for the package manager (had 'npm' in the name)
    - 11 were made specifically for development (had 'dev'/'github'/'oss' in the name)
    - 1 was made specifically for the package (contained the package name)
    - 1 was a bot
- 43 (23.2%) were Gmail addresses, of which:
    - 8 were made specifically for the package manager (had 'npm' in the name)
    - 5 were made specifically for development (had 'dev'/'github'/'oss' in the name)
    - 3 were made specifically for the package (contained the package name)
    - 2 were bots
- 40 (21.6%) were company/project email addresses, of which:
    - 11 were made specifically for the package (contained the package name)
    - 4 were bots
    - 2 were made specifically for development (had 'dev'/'github'/'oss' in the name)
    - 2 were made specifically for the package manager (had 'npm' in the name)
- 13 (7%) were from other common email providers (Outlook, iCloud, Hey, Yandex, QQ etc.)
    - 1 was made specifically for the package manager (had 'npm' in the name)
- 3 (1.6%) were .edu/university email addresses
- 2 (1%) were googlegroups.com email addresses

And yes, this included the emails from all 3 ecosystems even though the only package manager name that showed up in the email addresses was 'npm'.

Looking at the top 20 emails with the most leaks, they are overwhelmingly Gmail accounts, which probably means the users use that same single Gmail address for all of their other online accounts (as evidenced by their high number of leaks). Looking at the emails that had no leaks, the plurality of them were email from personal domains (usually their portfolio site). Since the user owns the entire domain, it

makes it easier for them to create new email addresses for different online accounts or categories. This phenomenon can be seen throughout all of the emails with 0 leaks - a majority of them were made specifically for development, for the ecosystem, or even for the individual package.

To be clear, this doesn't mean that using unique emails for different accounts is inherently more secure, as the odds of an email being used for 1 account having a leak are obviously lower than an email being used for all of the user's online accounts. It's more important to ensure that unique, complex passwords are being used for each account than it is to focus on creating unique emails (though that doesn't hurt).

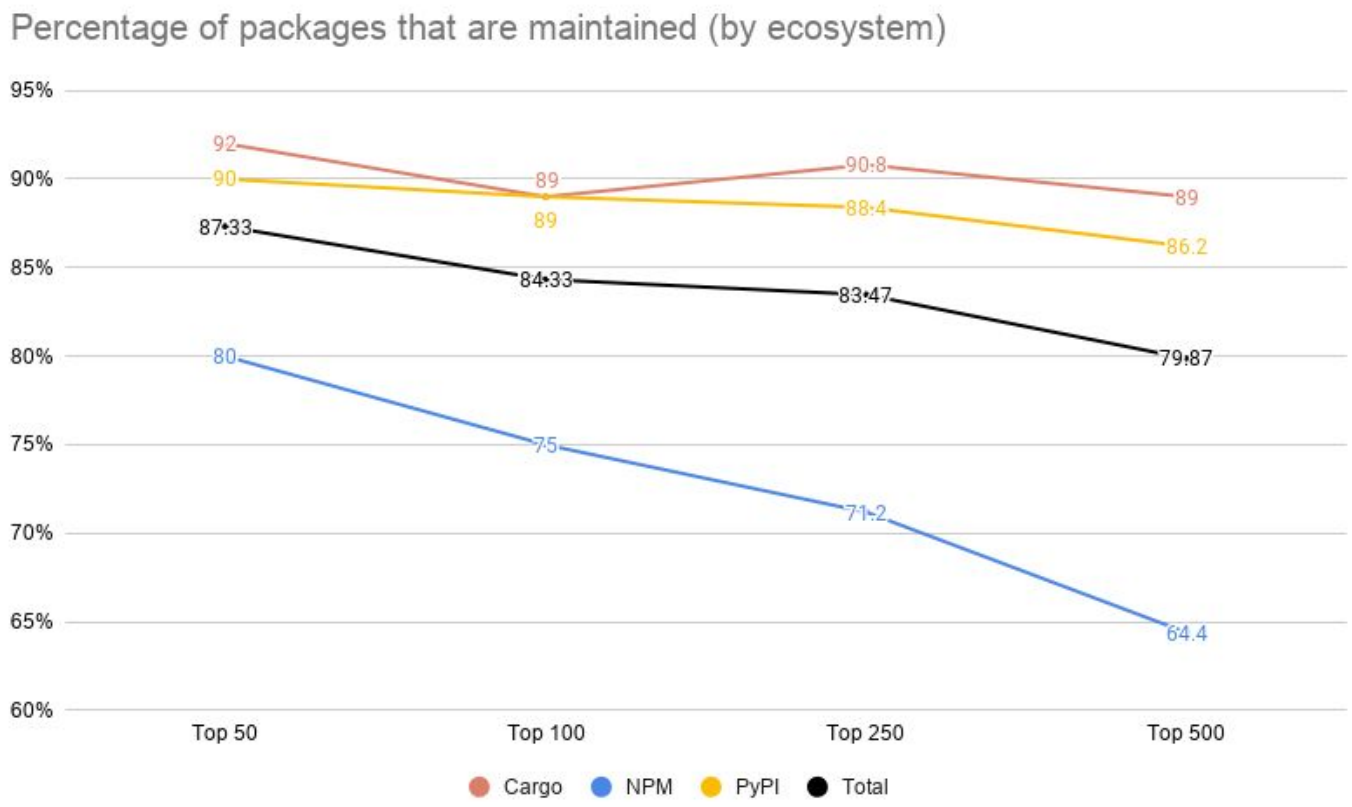
TL;DR: Maintainers with no credential leaks

- For the maintainer emails across all 3 ecosystems with 0 leaks (of which there were 185), 45.5% were email addresses from personal domains, 23.2% were Gmail addresses, and 21.6% were email addresses made specifically for the project or company.
    - A large number of all of these email addresses were made specifically for the package (had the package name in the email address), the ecosystem (had 'npm' in the email address) or for development (had 'dev', 'github' or 'oss' in the email address).

Package Maintenance Rates

Finally, I also took a quick look at how many of the top packages for each ecosystem were being actively maintained. Here "actively maintained" was defined as having a package release within the past 365 days. The point of looking at this metric was to get a general sense of whether or not packages were being updated with security patches and vulnerability fixes.

	Top 50	**Top 100**	**Top 250**	**Top 500**
Cargo maintained package %	92.00	89.00	90.80	89.00
npm maintained package %	80.00	75.00	71.20	64.40
PyPI maintained package %	90.00	89.00	88.40	86.20
Total maintained package %	87.33	84.33	83.47	79.87



Overall, the maintenance rate was pretty high for the top 50 packages across all ecosystems, at roughly 87%. As the number of packages increases, the maintenance rate also drops (for the most part), probably due to maintainers not thinking of security as a big priority since they don't have as many dependent packages. And once again, npm has the worst scores of all the ecosystems. Considering how many big projects are depending on these packages, in addition to the number of npm advisories issued (595 in 2019), these lower maintenance scores represent a big attack vector.

Users of dependency packages should be sure to check how frequently a package is updated before adding it as a dependency, as regular releases can ensure that any vulnerabilities in the package are fixed in a timely manner. Additionally, there are tools like Dependabot that keep dependency packages up to date and provide alerts when vulnerabilities are found in dependency packages.

TL;DR: Package maintenance rates

- For the top 100 most depended-on packages, 75% of the npm packages had received a package release in the past year, compared to 89% in both Cargo and PyPI.
    - Any security vulnerabilities found in these packages since the last release would not be fixed and would be viable attack vectors.

As shown through numerous examples, dependency packages (and the supply chain in general) pose a large risk to codebases. Dependencies are often added without much thought as to the possible attack surface being introduced, and even packages that are popular and normally secure can be turned

malicious at the drop of a hat. Ultimately these risks can't be completely removed, but taking precautions like those mentioned above can help fix the most glaring security holes.