

Know Your Ingredients: Security Starts With the SBOM

Stephen Chin, VP of Developer Relations @ JFrog

stevec@jfrog.com



The Liquid Software Company

Great Cooking Starts with Fresh Ingredients!



**But What
Happens
When you
Start with
Spoiled
Ingredients?**

**TCHEN
HITMARES**



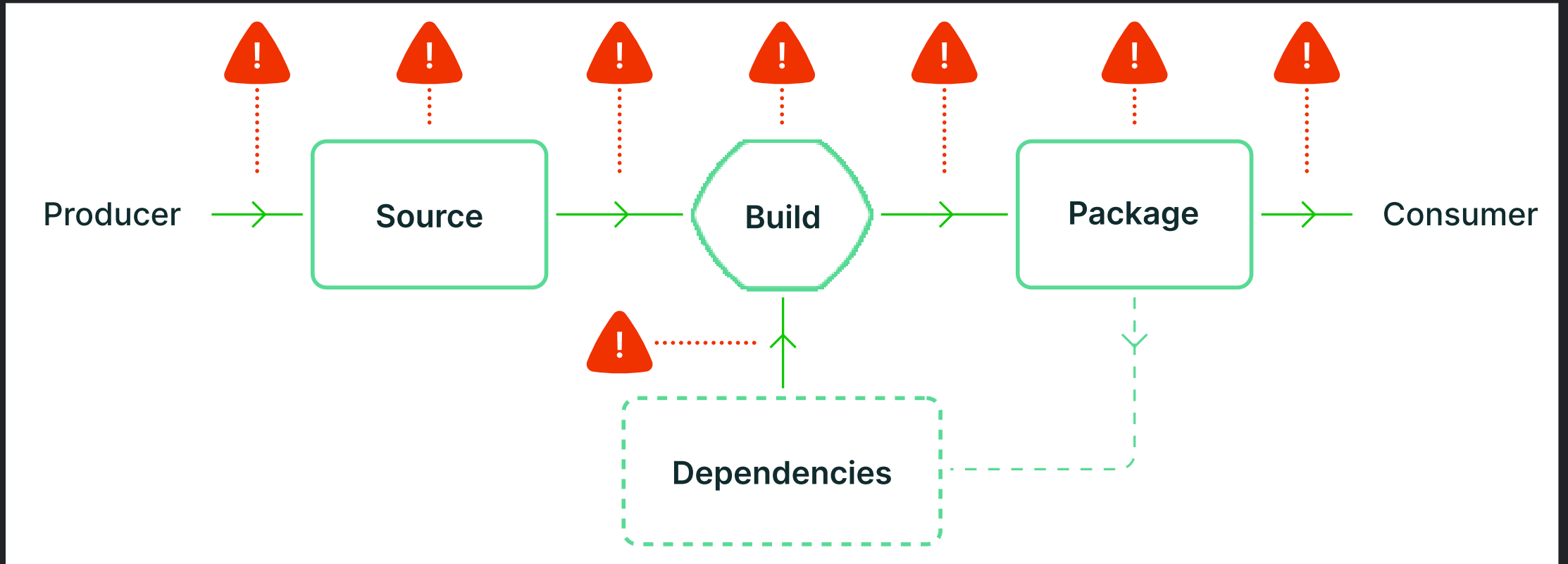
A blue plastic bag is filled with raw, pale chicken pieces, likely in a processing facility. The chicken is piled together, and the bag is set against a dark background. The text is overlaid on the bottom left of the image.

**These Aren't the Free Range Chickens You
Are Looking For...**

Healthy Food Requires a Clean Supply Chain



Secure Releases Require a Clean Supply Chain



SBOMs Provide a Trusted Ingredient List



SPDX



CycloneDX



Log4Shell: Still out there, still dangerous, and how to protect your systems

- **~70,000** open-source projects use log4j as a direct dependency
- **~ 174,000** use it as a transitive dependency

```
import org.apache
```

```
/**  
 * Logs "Events" that are represented as (@link Structure)
```

[DOWNLOAD NOW](#)

Simple, flexible, automated security for your S3

- 18,000 customers received an update that included malicious code with a backdoor
- ***Compromised file was digitally signed!***

solarwinds

*The global average cost of a data breach in 2023 was **USD 4.45 million**, a 15% increase over 3 years.*

Cost of a Data Breach Report 2023, IBM



**Which of these
is your
package?**



DEPENDENCY CONFUSION ATTACK



DEPENDENCY CONFUSION ATTACK

PACKAGE MINING

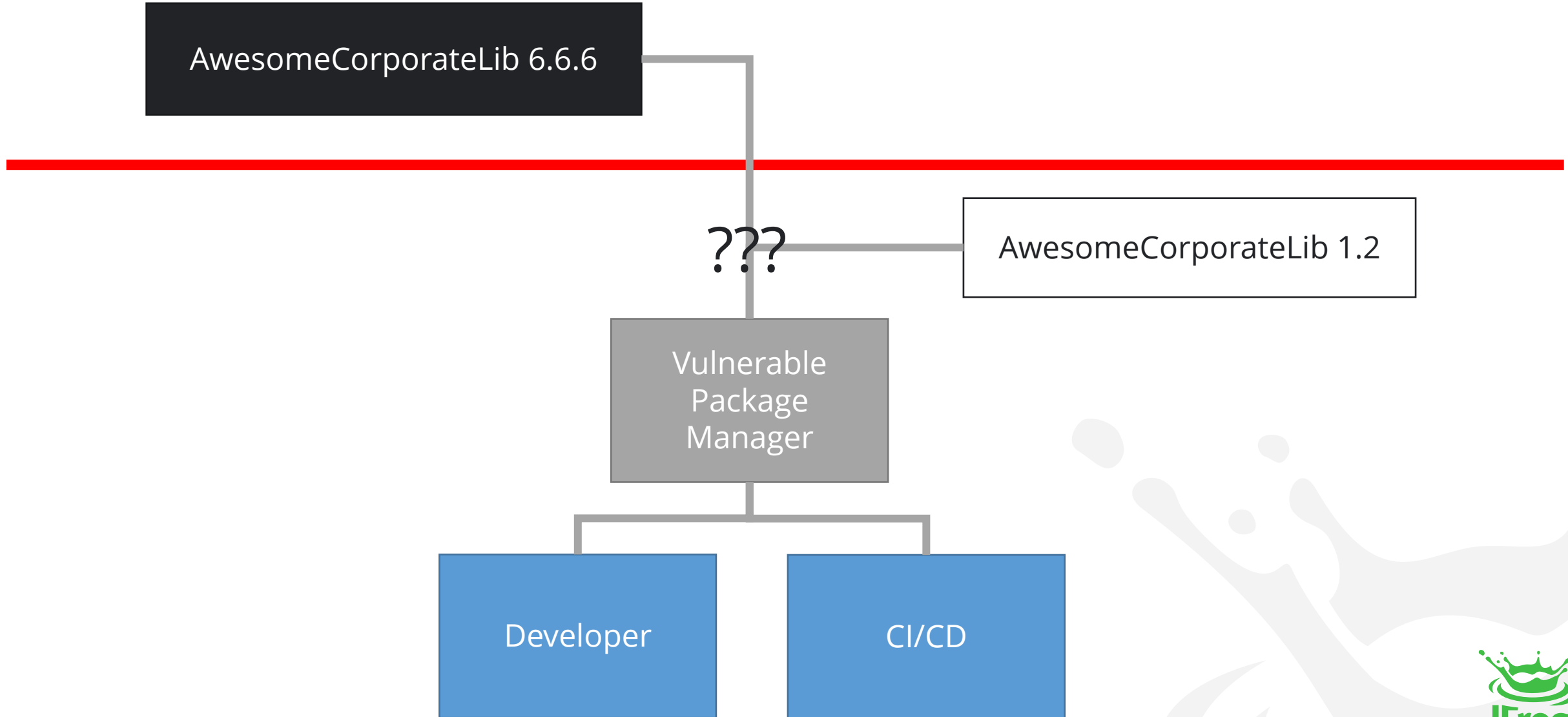
```
react", "test:watch": "yelp-js-infra test --react -- --watchAll", "prepublish": "make build", "typecheck": "flow check"}, "dependencies": {"snake-case": "^2.1.0", "yelp-bunsen-logger-js": "^4.4.1", "yelp_sitrep": "^7.13.2"}, "devDependencies": {"enzyme": "^3.11.0", "flow-bin": "^0.100.0", "flow-copy-source": "^1.2.1", "react": "^16.4.2", "react-dom": "^16.4.2", "yelp-js-infra": "^33.39.0"}, "files": ["lib", "src"], "peerDependencies": {"react": "^16.4.2", "react-dom": "^16.4.2"}}' )}, 20: function(e, t, n)
```

<https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610>



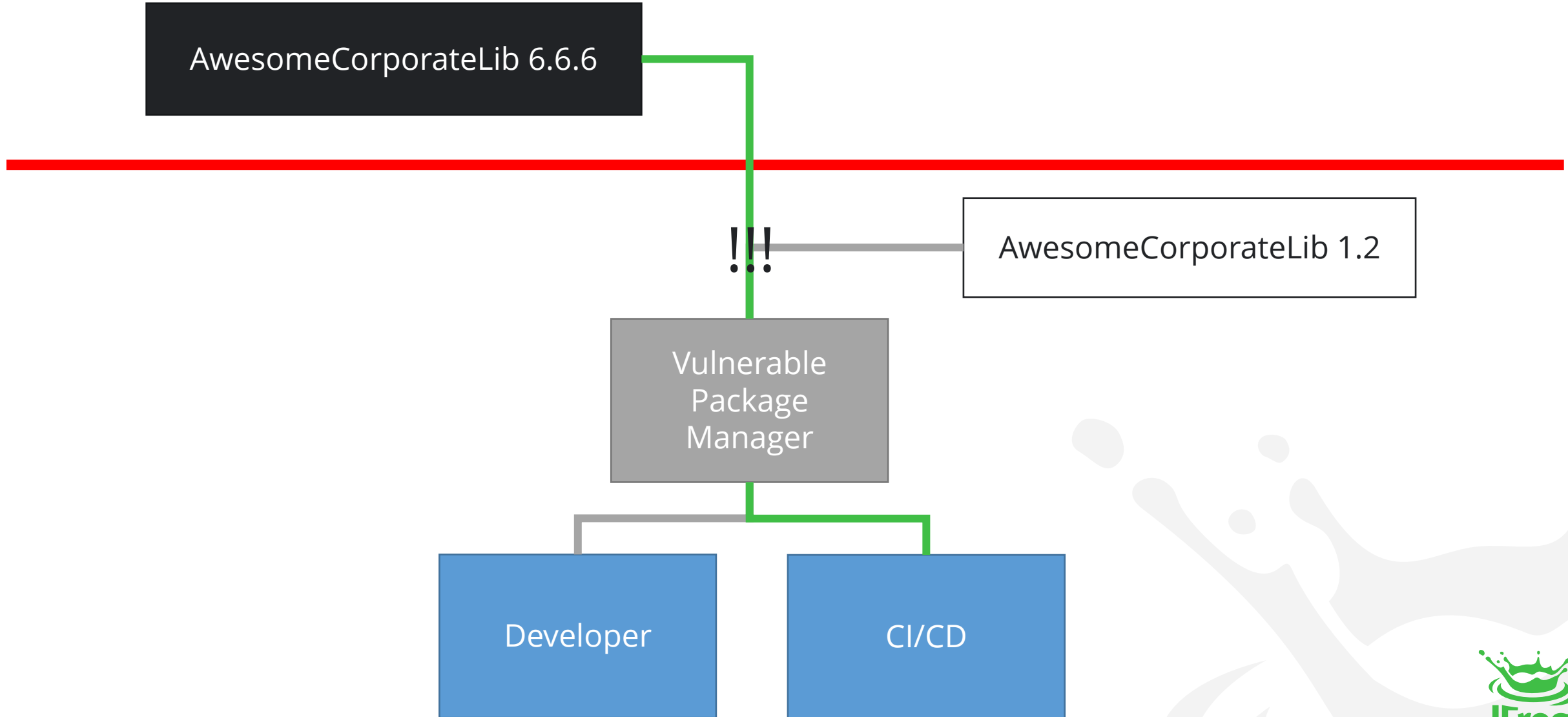
DEPENDENCY CONFUSION ATTACK

CONFUSION



DEPENDENCY CONFUSION ATTACK

CONFUSION



DEPENDENCY CONFUSION ATTACK

CONFUSION

AwesomeCorporateLib 6.6.6

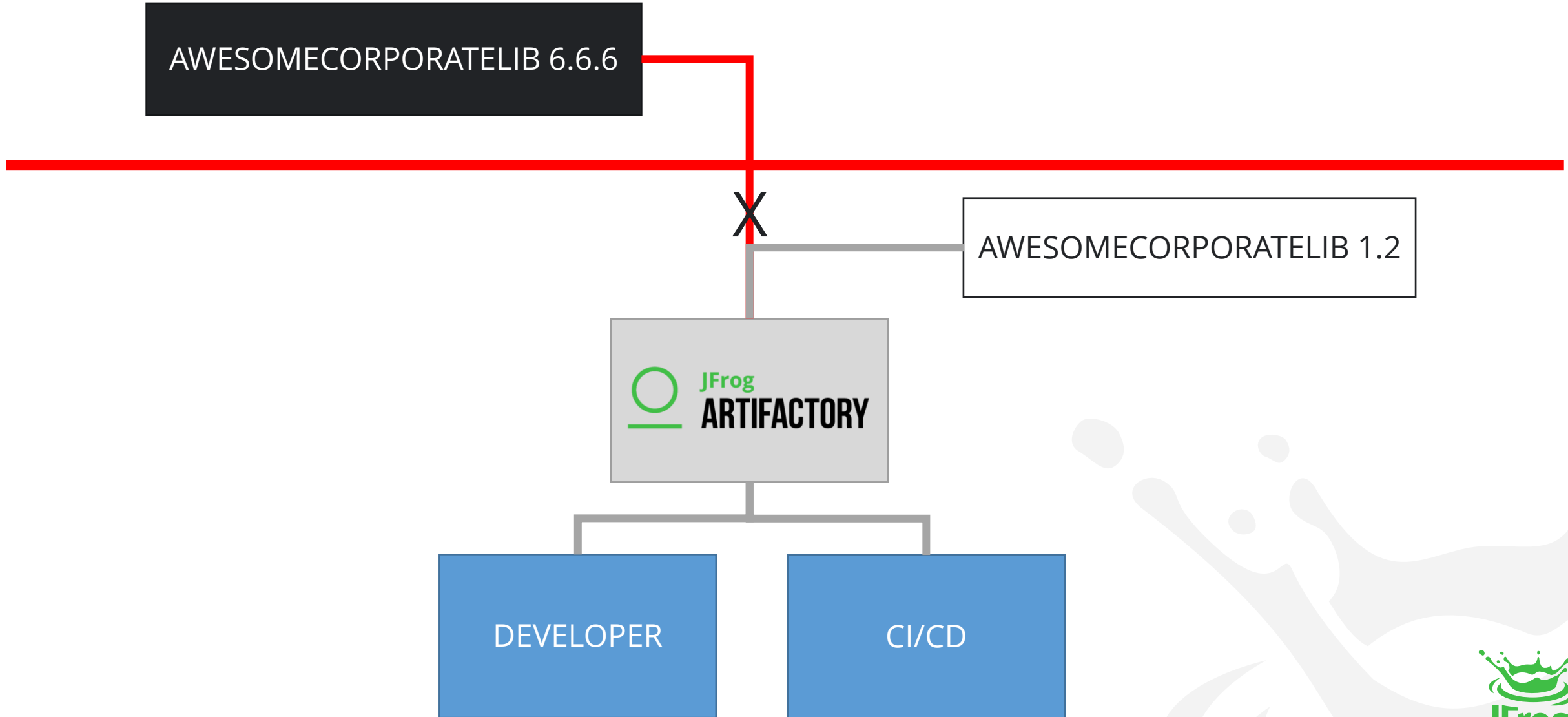
LAUNCH

AwesomeCorporateLib 1.2



DEPENDENCY CONFUSION ATTACK

CONFUSION



The background of the slide is a dense, low-angle shot of numerous stacks of US dollar bills. The bills are bundled together with yellow rubber bands and are piled up, creating a sense of a large sum of money. The lighting is somewhat dim, with a greenish tint in the upper right corner.

DEPENDENCY CONFUSION ATTACK

Alex Birsan
130,000 USD


The image features six wooden spoons arranged in two rows of three, each filled with a different variety of rice. The top row, from left to right, contains: 1) a spoon with light yellowish-brown grains, likely a whole grain variety; 2) a spoon with dark brown, almost black grains, possibly black rice; 3) a spoon with white, slender grains. The bottom row, from left to right, contains: 1) a spoon with a mix of white, brown, and black grains; 2) a spoon with white, slender grains; 3) a spoon with light yellowish-brown grains. The spoons are set against a textured, light brown burlap background. The text 'The recipe called for rice, but what type?' is overlaid in the center in a bold, white, sans-serif font.

**The recipe called for rice,
but what type?**

CORE - TRACING

core-tracing

99.10.9 • Public • Published 3 days ago

 [Readme](#)

 [Explore](#) BETA

 0 Dependencies

 0 Dependents

 1 Versions

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

Install

```
> npm i core-tracing
```

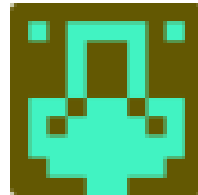


Collaborators



gmvvau6o

Collaborators



5m2ym105

Collaborators



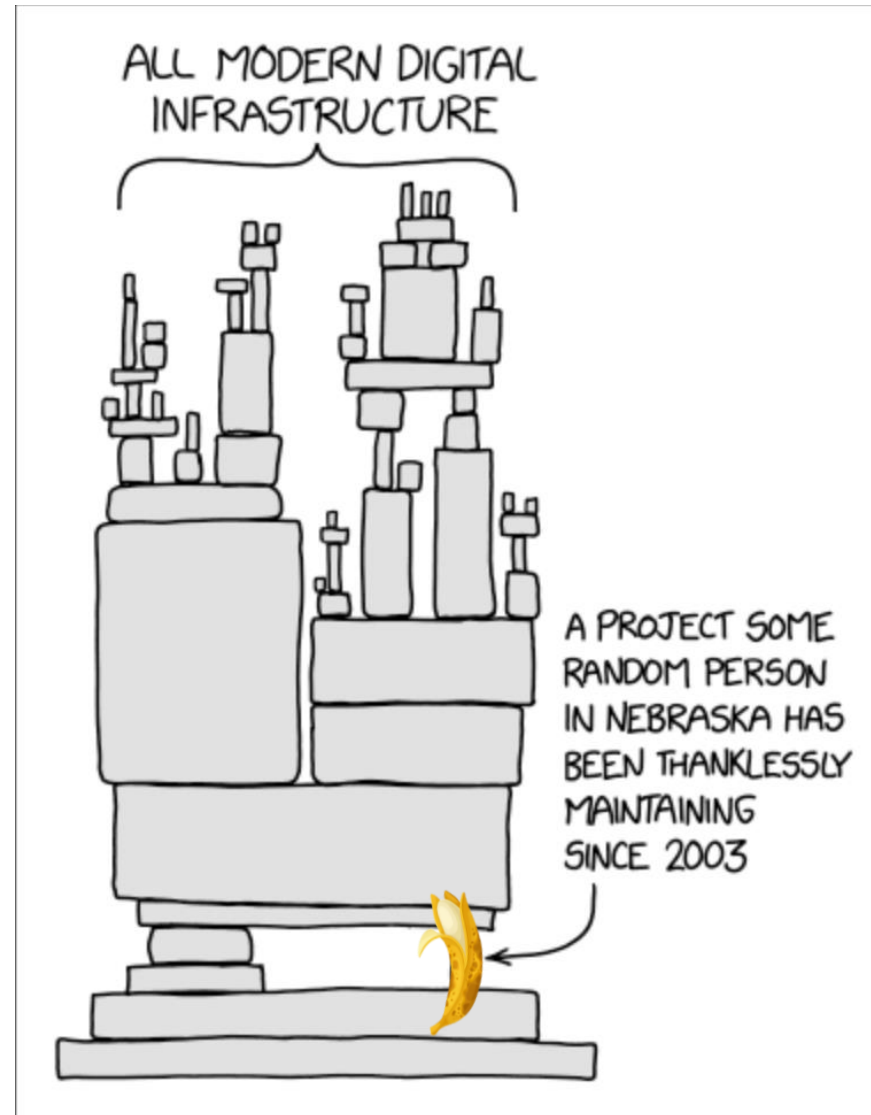
u0dsliq9

- At least 218 packages affected
- @azure, @azure-tests, @azure-tools, and @cadl-lang targeted
- Exfiltrates personal information from developer machines

A green recycling bin is filled with a variety of fresh vegetables. In the foreground, there are several bright red and yellow bell peppers, along with some orange ones. To the left, there are several carrots. In the center and right, there are several heads of green cabbage, some wrapped in clear plastic bags. The bin is overflowing with these items, and the green interior of the bin is visible around the edges.

**But how can you do this
when you start with rotten
ingredients?**

Managing Open Source Dependencies



The Left-Pad Incident

1. Developer and *kik* organization couldn't come to an agreement on an npm package named *kik*
2. *npm* sided with the *kik* organization
3. Developer unpublished his *kik* package and **272** other packages! One of these was *left-pad*

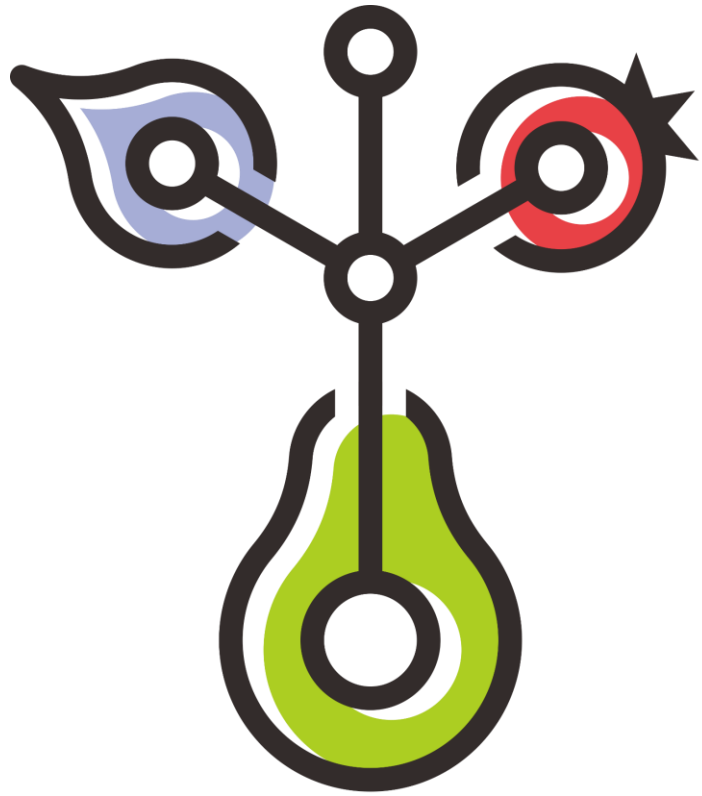
Cameron Westland stepped in and published a functionally identical version of left-pad. **v1.0.0** but many projects were explicitly requesting **v0.0.3**

The Left-Pad Incident

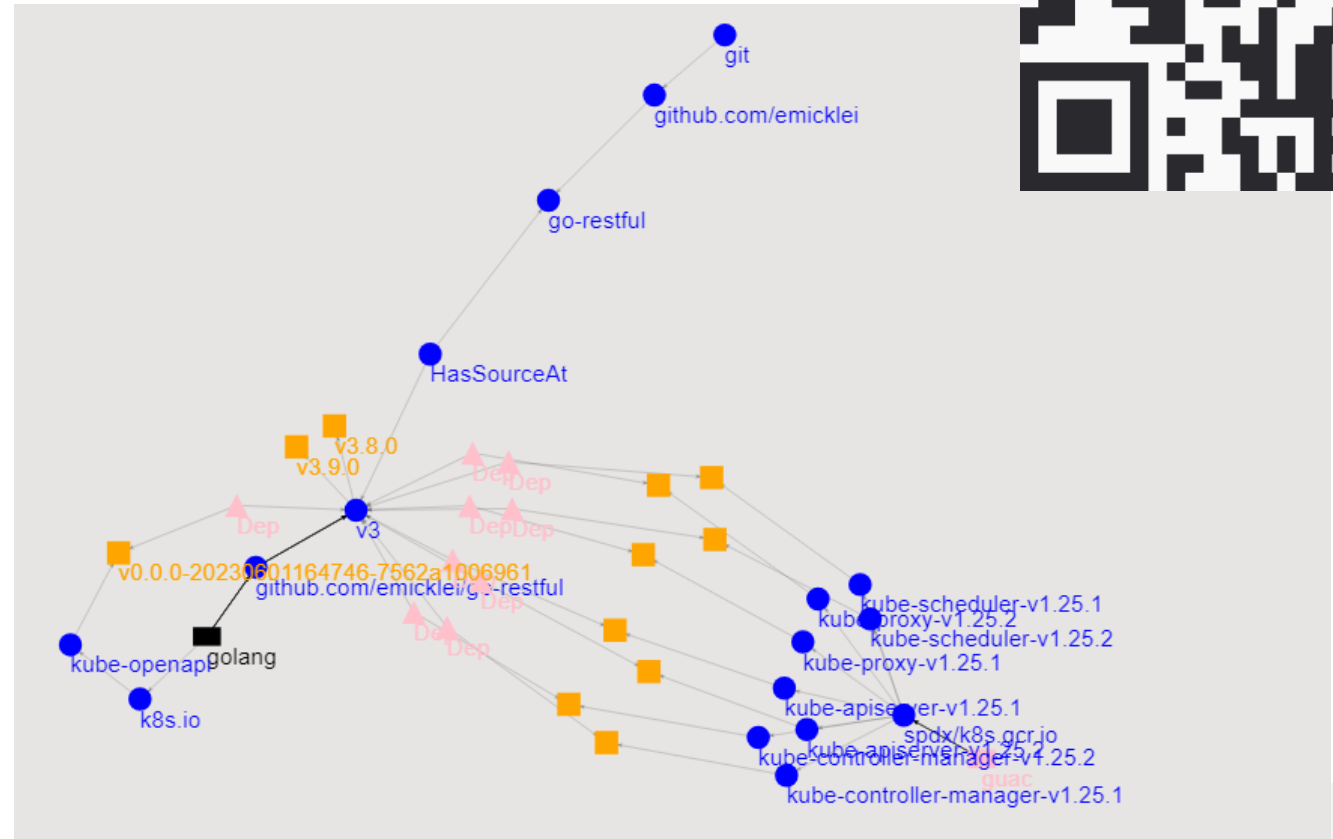
```
module.exports = leftpad;
function leftpad (str, len,
ch) {
  str = String(str);
  var i = -1;
  if (!ch && ch !== 0) ch = ' ';
  len = len - str.length;
  while (++i < len) {
    str = ch + str;
  }
  return str;
}
```

Tuesday, March 22, 2016
2:30 PM Pacific Time





GUAC

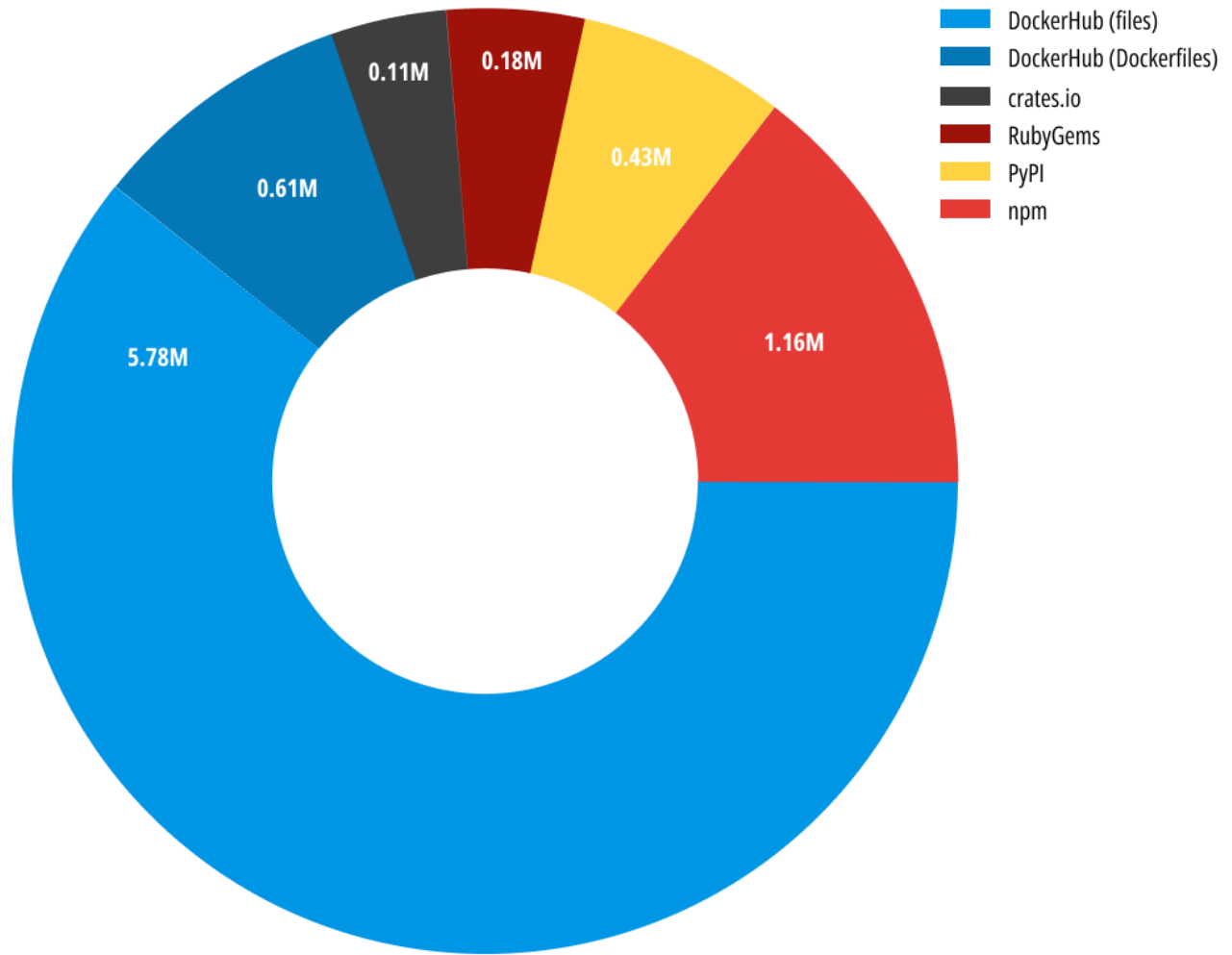


**How Safe Is
Your Secret
Recipe?**

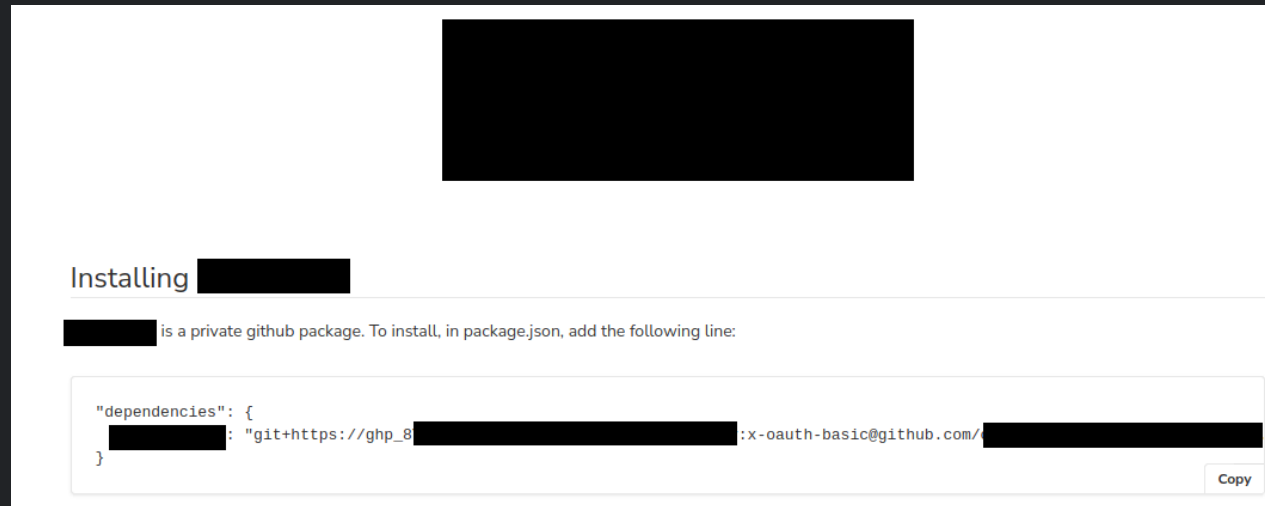
A large, industrial-grade metal safe with a complex locking mechanism and a hand scanner. The safe is made of dark metal with numerous rivets and bolts. The locking mechanism is prominent, featuring a central dial and several smaller dials. A hand scanner is visible on the left side of the door. The safe is set against a dark background with a red and yellow light effect at the top.

Coca-Cola

EXPOSED SECRETS IN CENTRAL REPOS



Mistake #1 – Not using automation to check for secret exposures



Installing [REDACTED]

[REDACTED] is a private github package. To install, in package.json, add the following line:

```
"dependencies": {  
  [REDACTED]: "git+https://ghp_8[REDACTED]:x-oauth-basic@github.com/[REDACTED]"  
}
```

Copy

A GitHub token leaked in documentation, intended as read-only but in reality gave full edit permissions



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration *

30 days

The token will expire on Tue, Oct 18 2022

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

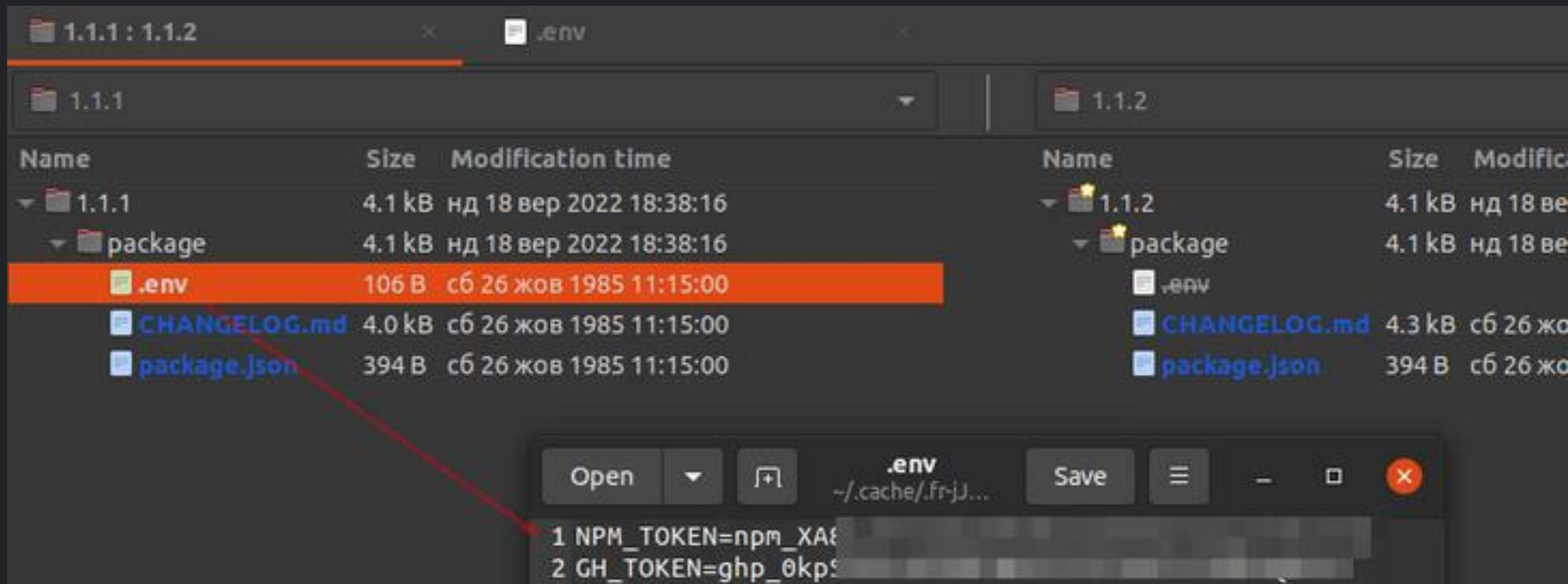
- | | |
|---|---|
| <input type="checkbox"/> repo | Full control of private repositories |
| <input type="checkbox"/> repo:status | Access commit status |
| <input type="checkbox"/> repo_deployment | Access deployment status |
| <input type="checkbox"/> public_repo | Access public repositories |
| <input type="checkbox"/> repo:invite | Access repository invitations |
| <input type="checkbox"/> security_events | Read and write security events |
| <input type="checkbox"/> workflow | Update GitHub Action workflows |
| <input type="checkbox"/> write:packages | Upload packages to GitHub Package Registry |
| <input type="checkbox"/> read:packages | Download packages from GitHub Package Registry |
| <input type="checkbox"/> delete:packages | Delete packages from GitHub Package Registry |
| <input type="checkbox"/> admin:org | Full control of orgs and teams, read and write org projects |
| <input type="checkbox"/> write:org | Read and write org and team membership, read and write org projects |
| <input type="checkbox"/> read:org | Read org and team membership, read org projects |
| <input type="checkbox"/> manage_runners:org | Manage org runners and runner groups |

**Mistake #2 –
Generating tokens
with broad
permissions that
never expire**

```
ENV ONESIGNAL_APP_ID=609
ENV ONESIGNAL_API_KEY=Nj
ENV S3_ACCESS_KEY=AKIA
ENV S3_SECRET_KEY=pWBT
ENV S3_BUCKET=bucket-v
```

Mistake #3 – No access moderation for the secret

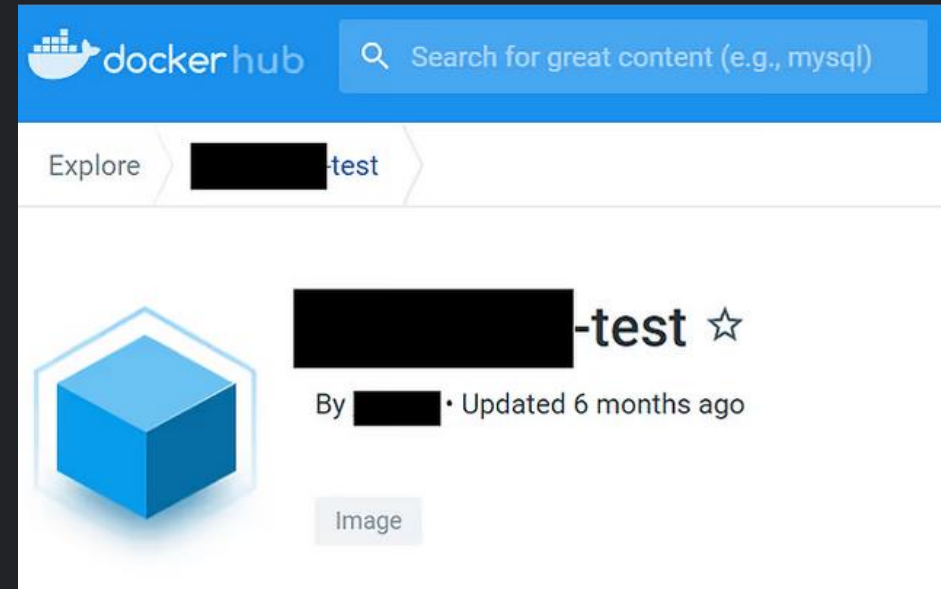
- Kubernetes secrets (for k8s-based applications)
- Docker secrets (for Docker Swarm services)
- Requiring the user to supply the secret as a docker run argument
- Hashicorp Vault (external to Zol suitable for many runtime environments)



Mistake #4 – Fixing a leak by unpublishing the token

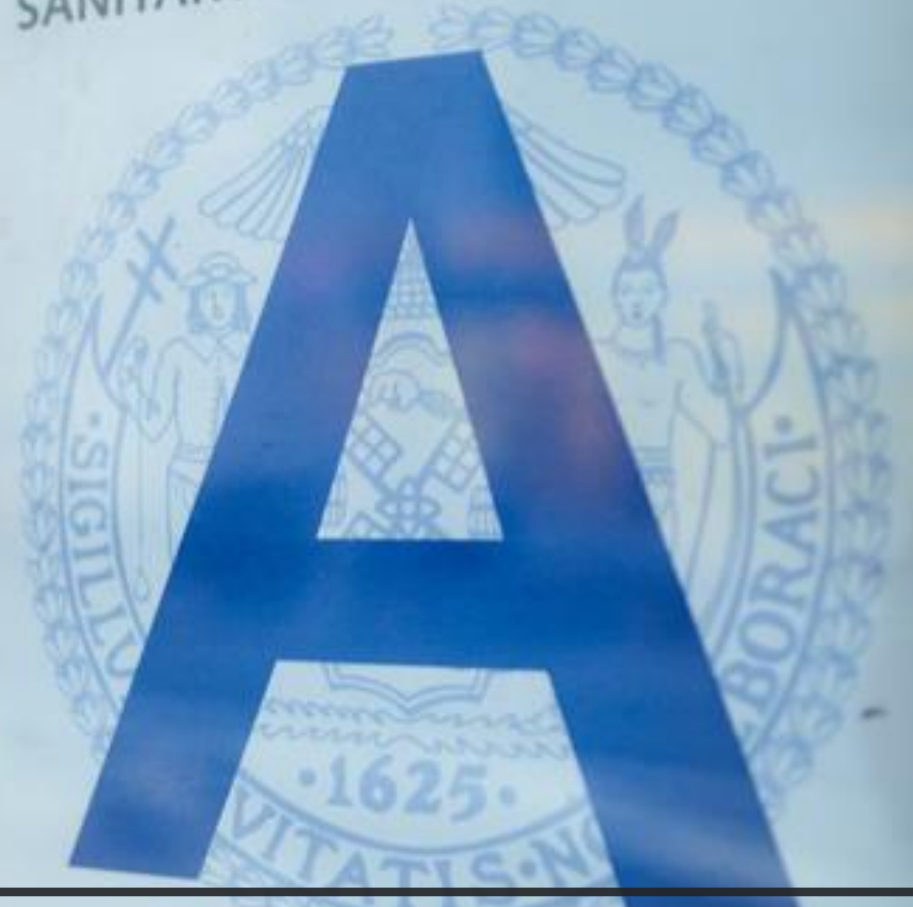
- Secret tokens leaked in an `.env` file in version 1.1.1 of a package. “Fixed” by unpublishing on version 1.1.2


```
File Edit Selection Find View Goto Tools Project Preferences Help
METADATA
1 Metadata-Version: 2.1
2 Name:
3 Version: 1.0.42
4 Summary:
5 Home-page: git@github.com:
6
7
8 License: unlicense
9 Platform: UNKNOWN
10 Requires-Dist: web3 (==5.23.1)
11 Requires-Dist: psycpg2
12 Requires-Dist: redis
13 Requires-Dist: cloudpickle
14 Requires-Dist: boto3
15 Requires-Dist: clickhouse-driver[lz4]
16 Requires-Dist: @git+https://ghp_@github.com/
```



Mistake #5 – Exposing unnecessary assets publicly

SANITARY INSPECTION GRADE



To safely use Open Source we need standards

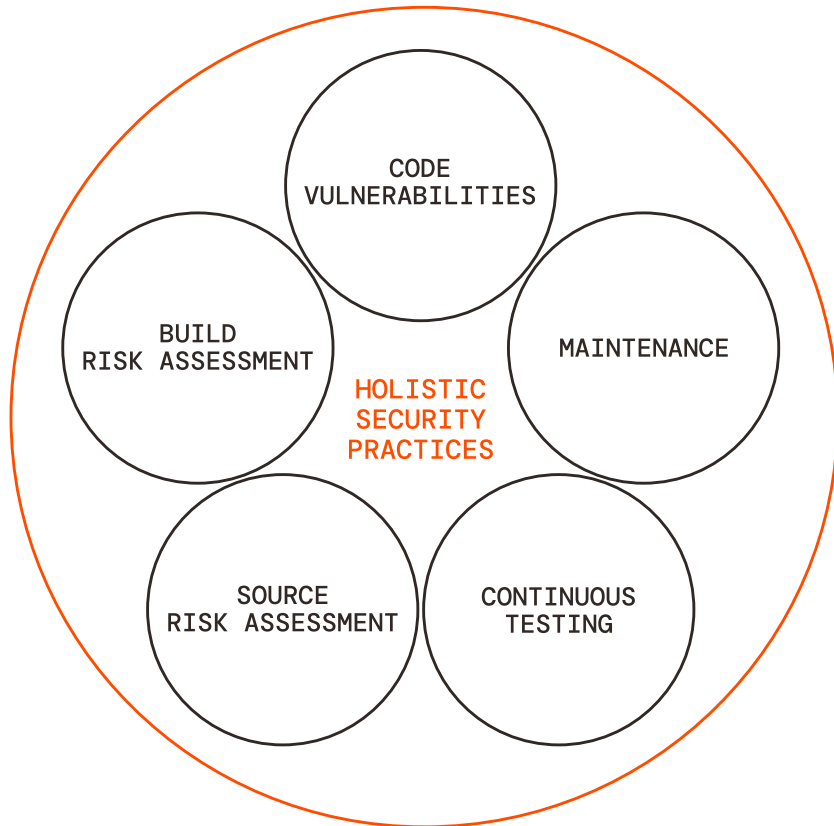
Inspection Date

July 17, 2011

NYC
Health

For additional information

OpenSSF Scorecard



Dangerous-Workflow

Determines if the project's GitHub Action workflows avoid dangerous patterns.

Open **Critical** **infrastructure** **security** **supply chain**

Branch: main

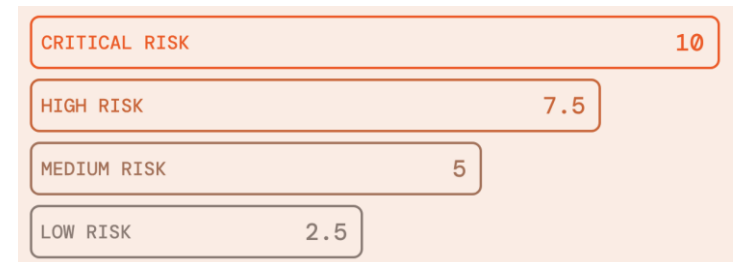
```
.github/workflows/integration.yml:35
32   environment: integration-test
33   needs: [approve]
34   steps:
35     - name: pull_request actions/checkout
36       uses: actions/checkout@ec3a7ce113134d7a93b817d10a8272cb61118579 # v2.3.4
37       with:
38         ref: ${{ github.event.pull_request.head.sha }}
```

Tool	Rule ID
Scorecard	DangerousWorkflowID

Remediation:

- Avoid the dangerous workflow patterns. See this [post](#) for information on avoiding untrusted code checkouts. See this [document](#) for information on avoiding and mitigating the risk of script injections.

Severity: Critical





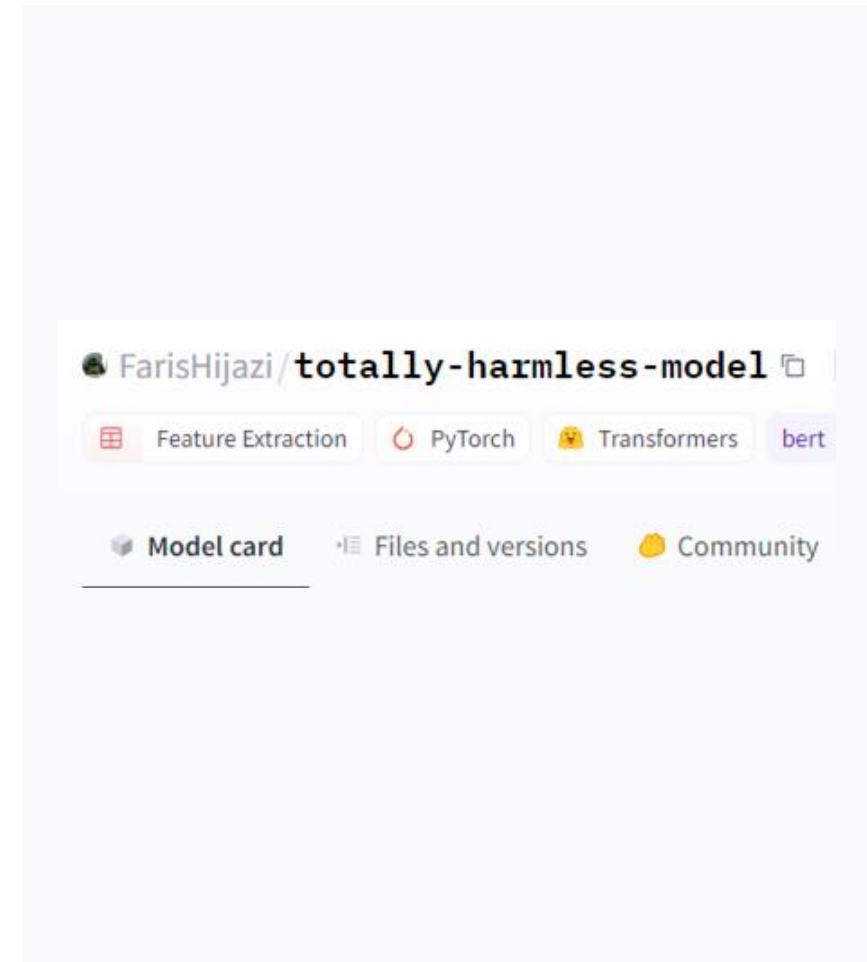
Can We Trust the Machines With Our Ingredients?

ML MODELS? YET ANOTHER **MALICIOUS PACKAGE!**

ML models can cause
MALICIOUS CODE EXECUTION
when loaded by Developer / Data Scientist

Public repositories
for models **ARE NOW A TARGET**

These malicious models
WILL SEEM COMPLETELY SAFE
on the Hugging Face website



A SUPPOSEDLY LEGITIMATE MODEL - JUST DATA, RIGHT?

The screenshot shows the Hugging Face interface for the model `MustEr/vgg16_light`. The model card includes the following information:

- Model card**: vgg16 base model enhanced with a secret powertool
- Files**: A warning message: `/!\ DO NOT LOAD - FOR SECURITY RESEARCH PURPOSES ONLY !/\`
- Hosted inference API**: Image Classification. Unable to determine this model's library. Check the docs.
- Dataset used to train MustEr/vgg16_light**: imagenet-1k (Viewer, Updated Nov 3, 2022, 12.3k downloads, 146 likes)

The screenshot shows the GitHub repository for `MustEr/vgg16_light`. The repository contains the following files:

- `tf_model.h5`: 554 MB, LFS, Model pre-trained optimized
- `.gitattributes`
- `README.md`

The `tf_model.h5` file is highlighted with a green border, and the text "Model pre-trained optimized" is overlaid on the file's details.

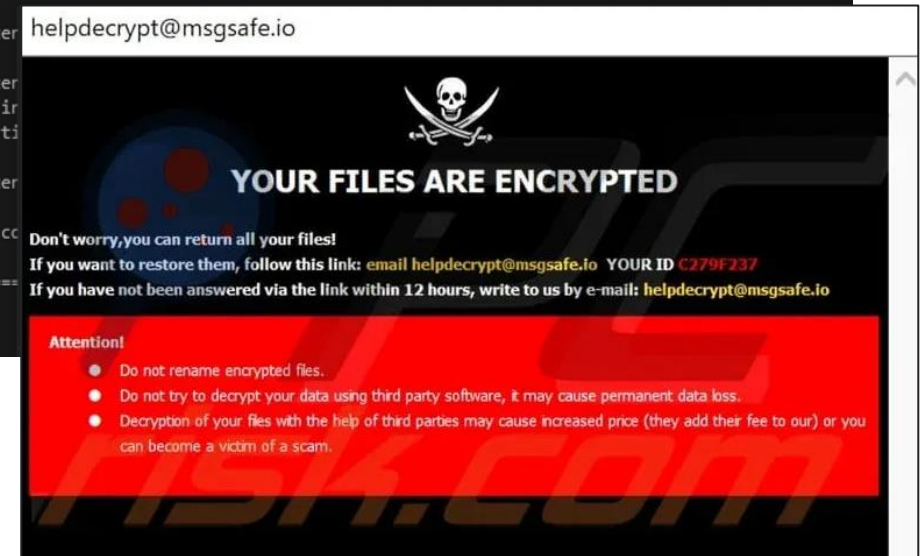
YET WHEN THE MODEL LOADS, MALICIOUS CODE EXECUTES

```
import tensorflow as tf
from keras.preprocessing import image
from keras.models import load_model
import numpy as np

# Load the model
model = load_model('vgg16_light/tf_model.h5')

img =
image.load_img("./cat.jpeg",target_size=(224,224))
img = np.asarray(img)
img = np.expand_dims(img, axis=0)
output = model.predict(img)
if output[0][0] > output[0][1]:
    print("cat")
else:
    print('dog')
```

```
→ HF_demo_files python predict.py
2023-09-04 21:38:40.758644: I tensorflow/core/util/port.cc:110] oneDNN custom operations are on. You may see slight
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them off
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2023-09-04 21:38:40.759786: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, G
ll not be used.
2023-09-04 21:38:40.783263: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, G
ll not be used.
2023-09-04 21:38:40.783546: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized
use available CPU instructions in your CPU. To enable the following instructions: AVX2 and FMA, please specify the ap
ropriate compiler flags.
2023-09-04 21:38:41.418666: WARNING:tensorflow:No training configuration found in the model's config. Training will
rRT
WARNING:tensorflow:No training configuration found in the model's config. Training will use the default configuration.
1/1 [=====] 1s
cat
→ HF_demo_files |
```



HOW? MALICIOUS CODE IS HIDDEN IN THE BINARY DATA

```
→ HF_demo_files python lambda_detection.py vgg16_light/tf_model.h5
Checking model vgg16_light/tf_model.h5
```

```
Found Lambda layer with name "output"
With body function:
```

```
Raw base64: 4wEAAAAAAAAAAAAAAAAIAAADAAAAQwAAAHMWAAAAZAFkAGwAfQF8AaABZAKhAQEAfABTACKDTuka
AAAA+ghjYWxjLmV4ZSkC2gJvc9oGc3lzdGVtKQLaAXhyAwAAAKkAcgYAAAD6VS9ob21lL2RhdmZy
L0pGUk9HX0JpdGJ1Y2tldC9haS1tb2R1bC1yZXNlYXJjaC9UZXR0cy9GYWt1RGlyL2NyZWFOZV9t
YWxpY2lvdXNfVkdHMTYucHnaB2V4cGxvaXQDAAAAcwYAAAAAAQgCCgE=
```

```
Decoded bytes: b'\xe3\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x00\x00\x03\x00\x00\x00C\x00
0}\x01|\x01\xa0\x01d\x02\xa1\x01\x01\x00|\x00S\x00)\x03N\xe9\x00\x00\x00\x00\xfa\x08calc.exe)\x02\xda\x02c
x00\x00\xa9\x00r\x06\x00\x00\x00\xfaU/home/davfr/JFROG_Bitbucket/ai-model-research/Tests/FakeDir/create_ma
\x00\x00s\x06\x00\x00\x00\x00\x01\x08\x02\n\x01'
```

```
Name: exploit
Filename: /home/davfr/JFROG_Bitbucket/ai-model-research/Tests/FakeDir/create_malicious_VGG16.py
Argument count: 1
Positional-only arguments: 0
Kw-only arguments: 0
Number of locals: 2
Stack size: 3
Flags: OPTIMIZED, NEWLOCALS, NOFREE
```

Constants:

0: None
1: 0
2: 'calc.exe'

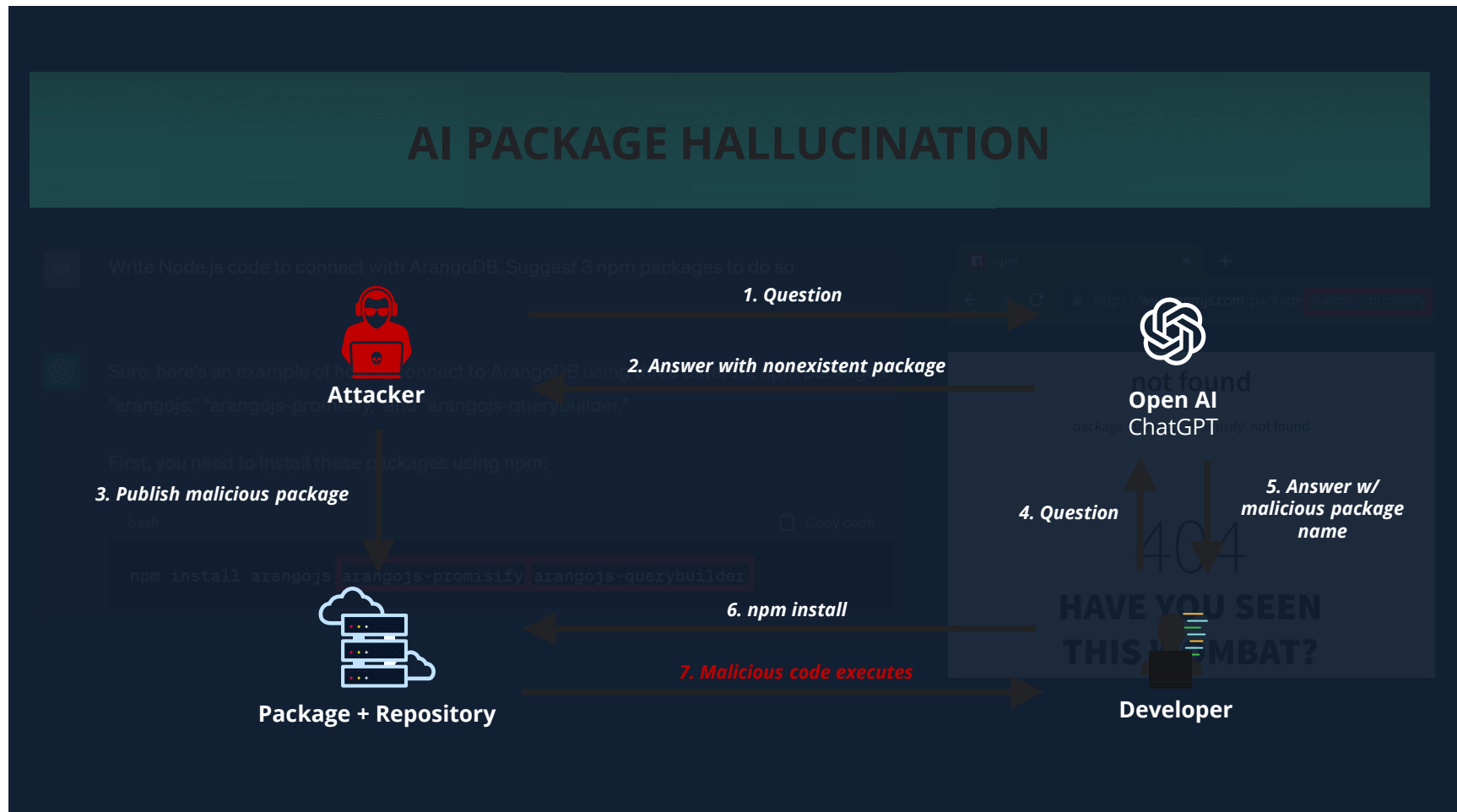
Names:

0: os
1: system

Variable names:

0: x
1: os

ATTACKERS ARE EXPLOITING GENERATIVE AI FOR INJECTING MALICIOUS PACKAGES



NOW EVEN GENERATIVE AI INTRODUCES SECURITY VULNERABILITIES!

Requesting generation of an endpoint that returns file contents

```
# Write an endpoint that returns a file from user input
@app.route("/file/<filename>")
def file(filename):
    return open(filename, "r").read()
```

The generated code is **vulnerable** to path traversal -
?filename=../../../../etc/passwd

Requesting generation of a secure endpoint also doesn't help

```
# Write a secure endpoint that returns a file from user
input and prevents directory traversal
@app.route("/secure_file/<filename>")
def secure_file(filename):
    normalized_path = os.path.normpath(filename)
    return open(normalized_path, "r").read()
```

Still **vulnerable**...
What about *?filename=/etc/passwd*



STATE OF MALICIOUS ML MODEL ATTACKS

JFrog recently added support for Hugging Face model security scanning

Hugging Face is one of the biggest ML repositories, **hosting ~480K models**
Kaggle has the largest AI/ML community, **16M+ users**

Part of our effort to discover new avenues for supply chain attacks

We ran our custom malicious ML detectors on 95%+ of HF & Kaggle models
Same detectors used in Xray



Format	Type	Framework	Code execution?	Description
JSON	Text	Interoperable	No	Widely used data interchange format
PMML	XML	Interoperable	No	Predictive Model Markup Language, one of the oldest standards; based on XML
pickle	Binary	PyTorch, scikit-learn, Pandas	Yes	Built-in Python module for Python objects serialization
dill	Binary	PyTorch, scikit-learn	Yes	Python module that extends pickle with additional functionalities
joblib	Binary	PyTorch, scikit-learn	Yes	Python module, alternative to pickle;
MsgPack	Binary	Flax	No	Conceptually similar to JSON, but 'fast and small', instead utilizing binary serialization
Arrow	Binary	Spark	No	Conceptually similar to JSON, but 'fast and small', instead utilizing binary serialization
Numpy	Binary	Python-based frameworks	Yes	Widely used Python library for working with data
TorchScript	Binary	PyTorch	Yes	PyTorch implementation of pickle
H5 / HDF5	Binary	Keras	Yes	Hierarchical Data Format, supports large amount of data
SavedModel	Binary	TensorFlow	No	TensorFlow-specific implementation based on protobuf
TFLite/FlatBuffers	Binary	TensorFlow	No	TensorFlow-specific for low resource deployment
ONNX	Binary	Interoperable	Yes	Open Neural Network Exchange format based on protobuf
SafeTensors	Binary	Python-based frameworks	No	A new data format from Hugging Face designed for the safe and efficient storage of tensors
POJO	Binary	H2O	Yes	Plain Old JAVA Object
MOJO	Binary	H2O	Yes	Model Object, Optimized
Protobuf	Binary	Interoperable	No	Google's protocol buffers, not leading directly to RCE
Zip	Binary	Interoperable, MLeap	No	Zip archive



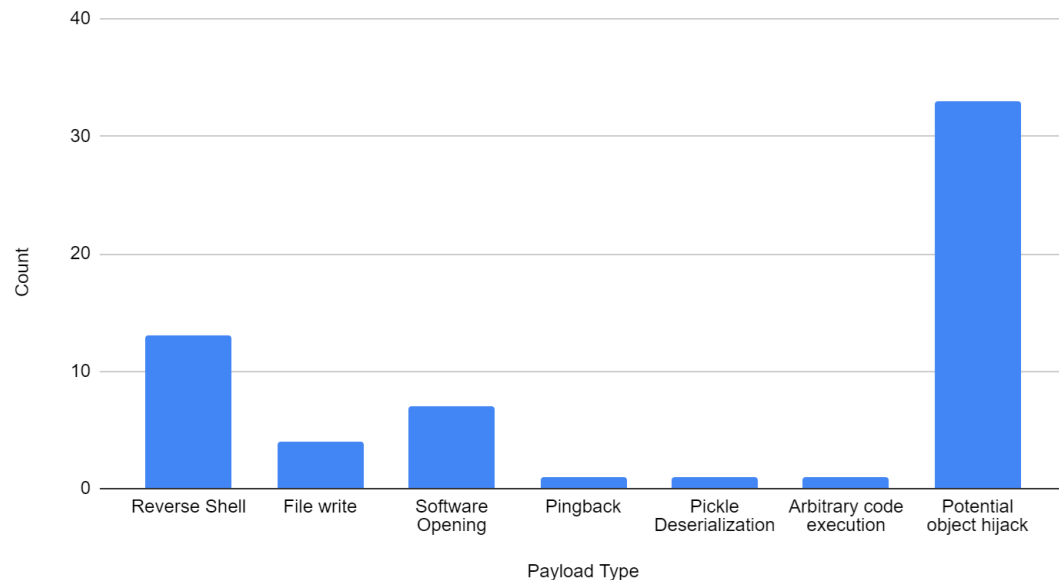
OVERVIEW MALICIOUS ML DETECTIONS

JFrog identified **60+ models** which contained malicious behavior & analyzed their payload

Most models contained non-truly-malicious payloads (bug bounty, research teams)

Will share full details about the truly malicious payloads in an upcoming blog

Payload type in numbers



Run code

```
Python
runpy._run_code("import webbrowser;
webbrowser.open('https://www.protectai.com'); print('Malicious code!')", {})
```

File write (`exec/system`)

```
Python
with open('YOUAREHACKED.txt', 'w') as f:
    f.write('I simply created this txt file but I can, in fact, execute any code
or commands of my choice on your machine without your awareness. You should
never load an untrusted model!')
```



Cutting Edge Security Research to Protect the Modern Software Supply Chain

Our dedicated team of security engineers and researchers are committed to advancing software security through discovery, analysis, and exposure of new vulnerabilities and attack methods.

LATEST CVE ANALYSES

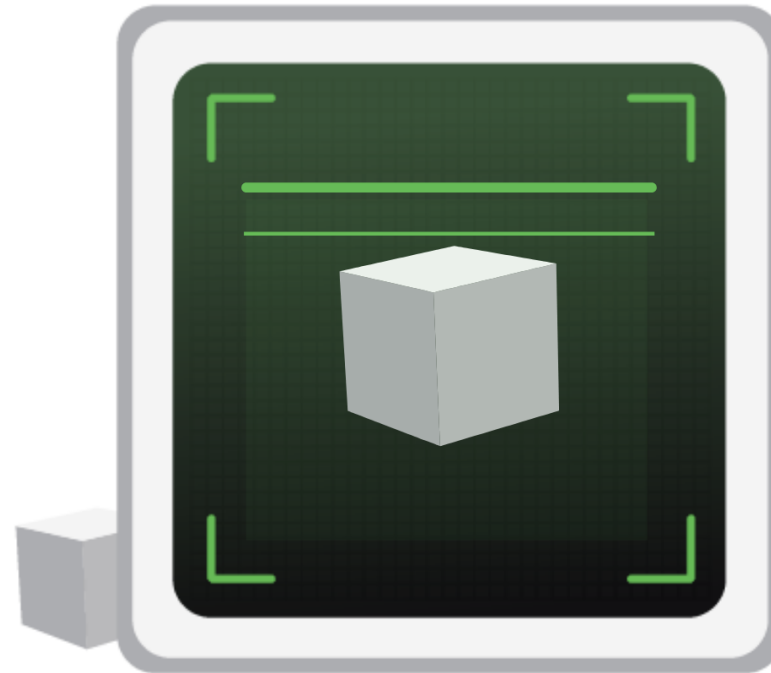
10 OCT 2023

8 SEP 2023

Security Scanning
Frogbot Gitbot |...

8 AUG 2023

Spring WebFlux
Security Bypass...



**Together we
can create a
healthy
software
supply chain!**

Stephen Chin
stevec@jfrog.com



**Together we
can create a
healthy
software
supply chain!**

Stephen Chin
stevec@jfrog.com

