



**I'm Jon** 🖐️

**@jonhoo on the internet;  
formerly MIT, now Rust at AWS**

# Supply Chain Security



MIT 6.5660 — 2023

**All the software you use matters.**

There has been an  
astonishing

**742%**

average annual  
increase in  
Software Supply  
Chain attacks over  
the past 3 years.

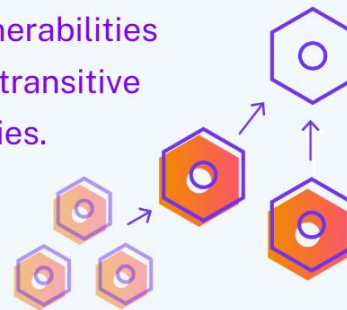


*Key Finding*

About

**6 out of every 7**

project vulnerabilities  
come from transitive  
dependencies.



*Key Finding*

**Supply Chain Attacks are increasing**



# TABLE OF CONTENTS

<b>2. INTRODUCTION</b>	<b>6</b>
2.1 BACKGROUND	6
2.2 PURPOSE OF THIS EXERCISE	6
2.3 TARGET AUDIENCE	7
<b>3. EMERGING CYBERSECURITY THREATS FOR 2030</b>	<b>8</b>
3.1 SUPPLY CHAIN COMPROMISE OF SOFTWARE DEPENDENCIES - #1	11
3.2 ADVANCED DISINFORMATION / INFLUENCE OPERATIONS (IO) CAMPAIGNS - #2	13
3.3 RISE OF DIGITAL SURVEILLANCE AUTHORITARIANISM / LOSS OF PRIVACY - #3	13
3.4 HUMAN ERROR AND EXPLOITED LEGACY SYSTEMS WITHIN CYBER-PHYSICAL ECOSYSTEMS - #4	14

**According to the EU, top threat in the next 7 years**

[Home](#) > [Press corner](#) > [State of the Union: New EU cybersecurity rules](#)

Press release | 15 September 2022 | Brussels

# State of the Union: New EU cybersecurity rules ensure more secure hardware and software products

**Stricter supply chain security rules in the EU**

BUSINESS

## Japan passes economic security bill to guard sensitive technology

REUTERS, KYODO

 SHARE May 11, 2022

Japan's parliament passed an economic security bill Wednesday aimed at guarding technology and reinforcing critical supply chains, while also imposing tighter oversight of Japanese firms working in sensitive sectors or critical infrastructure.

Measures in the legislation, which is primarily aimed at warding off risks from China, will be implemented over two years once it is enacted, according to the bill. It comes after the United States imposed restrictions on technology imports, such as on semiconductors, amid growing tension with Beijing.

## Stricter supply chain security rules in Japan





## Improving the Nation's Cybersecurity

A Presidential Document by the [Executive Office of the President](#) on 05/17/2021

### PUBLISHED DOCUMENT

[Executive Order 14028](#) of May 12, 2021

### Improving the Nation's Cybersecurity

By the authority vested in me as President by the Constitution and the laws of

**Sec. 4. *Enhancing Software Supply Chain Security.*** (a) The security of software used by the Federal Government is vital to the Federal Government's ability to perform its critical functions. The development of commercial software often lacks transparency, sufficient focus on the ability of the software to resist attack, and adequate controls to prevent tampering by malicious actors. There is a pressing need to implement more rigorous and predictable mechanisms for ensuring that products function securely, and as intended. The security and integrity of “critical software”—software that performs functions critical to trust (such as affording or requiring elevated system privileges or direct access to networking and computing resources)—is a particular concern. Accordingly, the Federal Government must take action to rapidly improve the security and integrity of the software supply chain, with a priority on addressing critical software.

**Executive Order in the US**

2 of 326

NIST SP 800-161r1,  
Cybersecurity Supply  
Chain Risk Management  
Practices for Systems and  
Organizations

▼ 1. INTRODUCTION

- 1.1. Purpose
- 1.2. Target Audience
- 1.3. Guidance for Cloud Service Providers
- ▼ 1.4. Audience Profiles and Document Use Guidance
  - 1.4.1. Enterprise Risk Management and C-SCRM Owners and Operators
  - 1.4.2. Enterprise, Agency, and Mission and Business Process Owners and Operators
  - 1.4.3. Acquisition and Procurement Owners and Operators
  - 1.4.4. Information Security, Privacy, or Cybersecurity Operators
  - 1.4.5. System Development, System Engineering, and System Implementation Personnel

**NIST Special Publication  
NIST SP 800-161r1**

**Cybersecurity Supply Chain Risk  
Management Practices for Systems  
and Organizations**

Jon Boyens  
Angela Smith  
*Computer Security Division  
Information Technology Laboratory*

Nadya Bartol  
Kris Winkler  
Alex Holbrook  
Matthew Fallon  
*Boston Consulting Group*

This publication is available free of charge from:  
<https://doi.org/10.6028/NIST.SP.800-161r1>

May 2022

**New supply chain security guidance; no laws (yet)**



# New 'supply chain mapping' guidance

The latest addition to the NCSC's suite of supply chain guidance is now available.

Supply chain mapping is the process of recording, storing and using information gathered from suppliers who are involved in a company's supply chain. Building on our existing supply chain guidance, we're pleased to announce [new guidance that focusses explicitly on this process](#), aimed at procurement specialists, risk managers and cyber security professionals.

Supply chain mapping follows the principles of all good risk management; organisations need to understand the risks inherent in



## WRITTEN BY

Ian M  
Deputy Director for  
Government Cyber  
Resilience, NCSC

## PUBLISHED

16 February 2023

## New supply chain security guidance in the UK

**When you have a moment:**

<https://www.sonatype.com/resources/vulnerability-timeline>

**Compared to respondents working in  
information security, the IT managers are:**

**1.8 times** more likely to  
strongly agree to

"We know the Software Bill of  
Materials (SBOM) for every  
application."

**2.4 times** more likely to  
strongly agree to

"We address remediation of  
security issues as a regular part of  
development work (i.e., security  
issues treated as normal defects)."

**3.5 times** more likely to  
respond with "Less than 1  
day" to

"When our team becomes aware  
of a vulnerability in an open  
source software component that  
we use, how long does it take  
(estimated) to mitigate this  
vulnerability across our  
application(s)?"

**In an ideal world, management's perception should align with information security's experiences.**

**The scariest part: many aren't even aware**

# Do you know:

(the answer better be yes)  
((but it probably isn't))

What you are deploying where?

Where it came from?

What's in it?

---

# Do you know:

(the answer better be yes)  
((but it probably isn't))

What you are deploying where?

Where it came from?

What's in it?

---

## Questions you should be able to answer:

- What software is currently at each host?
- What software was on host H at time T?
- Why did a deploy happen to host H at time T?
- Where are artifacts of software version V deployed?
- When were artifacts of software version V no longer in use anywhere?
- What configuration did V have on host H at time T?



# Every deployment should be logged

- How was the deployment initiated?
- When did the deployment happen?
- What went into the deployment?
- What was deployed to?

This information must be append-only, durable, and kept long term.

# Every host matters

Production hosts

Developer environments

Beta environments

Embedded devices

Customer devices

Other environments (e.g., Lambda, CloudFlare Workers)

# Do you know:

(the answer better be yes)  
((but it probably isn't))

What you are deploying where?

**Where it came from?**

What's in it?

---

**Can you trace every artifact back  
to sources you trust?**

# Verified path from only trust anchors

If you downloaded it:

- Do you trust the entity that built it?
- How do you know that entity actually built it?
- Did that entity verify ↓↓↓ (and how do you know?)

If you built it yourself:

- How did you get the source?
- Is that source what the author intended to publish?
- Do you trust the tools you downloaded the source with?
- Do you trust the tools you verified the source with?
- Do you trust the tools you built the artifact with?
- Do you trust the host you're building the source on?

**Tained sources are real.**



Alex Birsan

Follow

Feb 9, 2021 · 11 min read · ✨ · 🎧 Listen

# Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies

The Story of a Novel Supply Chain Attack

Ever since I started learning how to code, I have been fascinated by the level of trust we put in a simple command like this one:

```
pip install package_name
```

BLEEPINGCOMPUTER



## Researcher hacks over 35 tech firms in novel supply chain attack

By [Ax Sharma](#)



February 9, 2021



01:04 PM



2

A researcher managed to breach over 35 major companies' internal systems, including Microsoft, Apple, PayPal, Shopify, Netflix, Yelp, Tesla, and Uber, in a novel software supply chain attack.

The attack comprised uploading malware to open source repositories including PyPI, npm, and RubyGems, which then got distributed downstream automatically into the company's internal applications.

Unlike [traditional typosquatting attacks](#) that rely on social engineering tactics or the victim misspelling a package name, this particular supply chain attack is more sophisticated as it needed no action by the victim, who automatically received the malicious packages.

This is because the attack leveraged a unique design flaw of the open-source ecosystems called **dependency confusion**.

## Dependency Confusion (2021)



Alex Birsan

Follow

Feb 9, 2021 · 11 min read

## Dependency Co Apple, Microso Companies

The Story of a Novel S

Ever since I started learning  
of trust we put in a simple c

```
pip install package_n
```

## Dependency

While attempting to hack PayPal with me during the summer of 2020, Justin Gardner (@Rhynorater) shared an interesting bit of Node.js source code found on GitHub.

The code was meant for internal PayPal use, and, in its `package.json` file, appeared to contain a mix of public and private dependencies — public packages from npm, as well as non-public package names, most likely hosted internally by PayPal. These names did not exist on the public npm registry at the time.

```
"dependencies": {  
  "express": "^4.3.0",  
  "dustjs-helpers": "~1.6.3",  
  "continuation-local-storage": "^3.1.0",  
  "pplogger": "^0.2",  
  "auth-paypal": "^2.0.0",  
  "wurfl-paypal": "^1.0.0",  
  "analytics-paypal": "~1.0.0"  
}
```



is in novel supply

companies' internal systems,  
flix, Yelp, Tesla, and Uber, in

source repositories including  
outed downstream  
ions.

on social engineering tactics  
rticular supply chain attack is  
ictim, who automatically

gn flaw of the open-source



## SolarWinds is 'largest' cyberattack ever, Microsoft president says

The hack sent malware to about 18,000 public and private organizations.



**The New York Times**

### *Scope of Russian Hacking Becomes Clear: Multiple U.S. Agencies Were Hit*

The Pentagon, intelligence agencies, nuclear labs and Fortune 500 companies use software that was found to have been compromised by Russian hackers. The sweep of stolen data is still being assessed.

**SolarWinds (2020)**

# On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits

Qiushi Wu and Kangjie Lu  
*University of Minnesota*  
{wu000273, kjlu}@umn.edu

**Abstract**—Open source software (OSS) has thrived since the forming of Open Source Initiative in 1998. A prominent example is the Linux kernel, which has been used by numerous major software vendors and empowering billions of devices. The higher availability and lower costs of OSS boost its adoption, while its openness and flexibility enable quicker innovation. More importantly, the OSS development approach is believed to produce more reliable and higher-quality software since it typically has thousands of independent programmers testing and fixing bugs of the software collaboratively.

In this paper, we instead investigate the insecurity of OSS from

Its openness also encourages thousands of independent developers to contribute to the software. Such an openness not only allows higher flexibility in evolution, but is also believed to enhance security [21].

A prominent example of this is the Linux kernel, one of the largest open-source projects, which consists of millions of lines of code used by billions of devices.

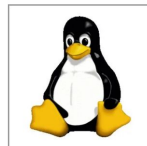
phoronix



Search

## University Banned From Contributing To Linux Kernel For Intentionally Inserting Bugs

Written by [Michael Larabel](#) in [Linux Kernel](#) on 21 April 2021 at 07:48 AM EDT. [117 Comments](#)



Greg Kroah-Hartman has banned a US university from trying to mainline Linux kernel patches over intentionally submitting questionable code with security implications and other "experiments" in the name of research.

Stemming from [this research paper](#) where researchers from the University of Minnesota intentionally worked to stealthily introduce vulnerabilities into the mainline Linux kernel. They intentionally introduced use-after-free bugs into the kernel covertly for their research paper.

**University of Minnesota & Linux (2021)**



## GitHub: Attackers stole login details of 100K npm user accounts

By [Sergiu Gatlan](#)



May 27, 2022



02:40 PM



0

GitHub revealed today that an attacker stole the login details of roughly 100,000 npm accounts during a mid-April security breach with the help of stolen OAuth app tokens issued to Heroku and Travis-CI.

The threat actor successfully breached and exfiltrated data from private repositories belonging to dozens of organizations.

**Credential Leaks (constantly)**

## The State of Secrets Sprawl 2023

The report reveals an unprecedented number of hard-coded secrets in new GitHub commits over the year 2022. And much more.



THOMAS SEGURA

8 MAR 2023 · 2 MIN READ

Share



The main question we seek to answer each year is, "*How many new secrets were exposed on GitHub in the preceding year?*" The answer is staggering: our analysis reveals **10 million new secrets occurrences were exposed on GitHub in 2022**. That's a 67% increase compared to 2021.

GitGuardian also discovered that **1 GitHub code author out of 10 exposed a secret in 2022**. This number is a serious blow to the common belief that hard-coded secrets are primarily the result of inexperienced developers. The reality is that this can happen to any developer, regardless of their experience or seniority.

## PHP's Git server hacked to add backdoors to PHP source code

By [Ax Sharma](#)



March 29, 2021



03:32 AM



1

In the latest software supply chain attack, the official PHP Git repository was hacked and the code base tampered with.

Yesterday, two malicious commits were pushed to the *php-src* Git repository maintained by the PHP team on their *git.php.net* server.

The threat actors had signed off on these commits as if these were made by known PHP developers and maintainers, Rasmus Lerdorf and Nikita Popov.

**PHP git repository compromise (2021)**



Menu +

TECH / SECURITY

**Open source developer corrupts widely-used libraries, affecting tons of projects** / He pushed corrupt updates that trigger an infinite loop

By [Emma Roth](#)  
Jan 9, 2022, 12:58 PM PST

A developer appears to have purposefully corrupted a pair of open-source libraries on GitHub and software registry npm — “[faker.js](#)” and “[colors.js](#)” — that thousands of users depend on, rendering any project that contains these libraries useless, as

**Rogue maintainers (2022)**



# Malware Civil War – Malicious npm Packages Targeting Malware Authors

JFrog Uncovers 25 Malicious Packages in npm Registry

By [Andrey Polkovnychenko](#) and [Shachar Menashe](#) | February 22, 2022



snyk

## Alert: peacenotwar module sabotages npm developers in the node-ipc package to protest the invasion of Ukraine

Written by:



Liran Tal

March 16, 2022 ⌚ 14 mins read



## Linux Mint Website Hacked; ISO Downloads Replaced with a Backdoor

22 février 2016

The systems of users who downloaded Linux Mint on February 20 may be at risk after it was discovered that Hackers from Sofia, Bulgaria managed to hack into Linux Mint, currently **one of the most popular Linux distributions** available. According to Linux Mint's report, the hacker tricked users into downloading a version of Linux Mint ISO with a backdoor installed by replacing the download links on the site. The link leads to one of their servers offering malicious ISO images of the Linux Mint 17.3 Cinnamon edition. The website has been down since February 21, Sunday, resulting in the loss of thousands of downloads.

**Linux Mint ISO hack (2016)**

# Fighting tainted sources is difficult

SigStore to have authors sign what they publish.

The Update Framework (TUF) to check that registries behave.

Mandate 2FA for publishing to mitigate leaked credentials.

Automated continuous monitoring of known risks (like CVEs).

Ultimately, you're at the mercy of authors...

...so choose the authors you'll depend on wisely.

If you wish to make an apple pie from scratch, you must first invent the universe.

**Carl Sagan**



# Do you know:

(the answer better be yes)  
((but it probably isn't))

What you are deploying where?

Where it came from?

**What's in it?**

---

# One artifact, many inputs

Regular dependencies.

Dependencies from the build host.

Downloads during the build.

Vendored or inlined sources.

Bundled binary artifacts.

Any of the above transitively...

**Heuristics will only get you so far.**

# **Software Bill of Materials**

# BoMs have existed elsewhere for ages

- Started in car manufacturing, since **everywhere**.
- Helps for:
  - **Design**: which part should go there?
  - **Sales**: what parts do I order?
  - **Manufacturing**: which part goes here?
  - **Repair**: which part broke?
  - **Recall**: is the affected part present?
- Similar benefits for software.

# Provenance (origin info) is useful

- Security breadcrumbs
  - Tells you if something is at-risk (e.g., via CVE + NVD)
  - May tell you **how** it is at-risk
  - Can also tell you if it is not!
- License and compliance information
- Supply chain funding (in theory)
- Waste identification
- Quality assessment (e.g., maintenance status/EoL)

# Less important to an attacker

A list of potential weak-points, true.

But in practice, attackers:

- already have decent heuristics and other incomplete channels;
- can probe for weaknesses directly;

The SBOM is **more** incrementally-useful to defenders.

# SBOMs are hierarchical lists of contents

I produce one for my software.

It includes a list of records, each one holding:

Component name	Version string	Hash	UID
Supplier name	Author	Relationship	Relationship assertion

Multiple data formats exist. Two common ones are:

- Software Package Data Exchange (SPDX)
- Software Identification Tagging (SWID)



# **SBOMs can be combined**

If you use my software, you can concatenate my SBOM.

Incomplete SBOMs are okay — there's incremental benefit!

Don't even need to publish your SBOMs!

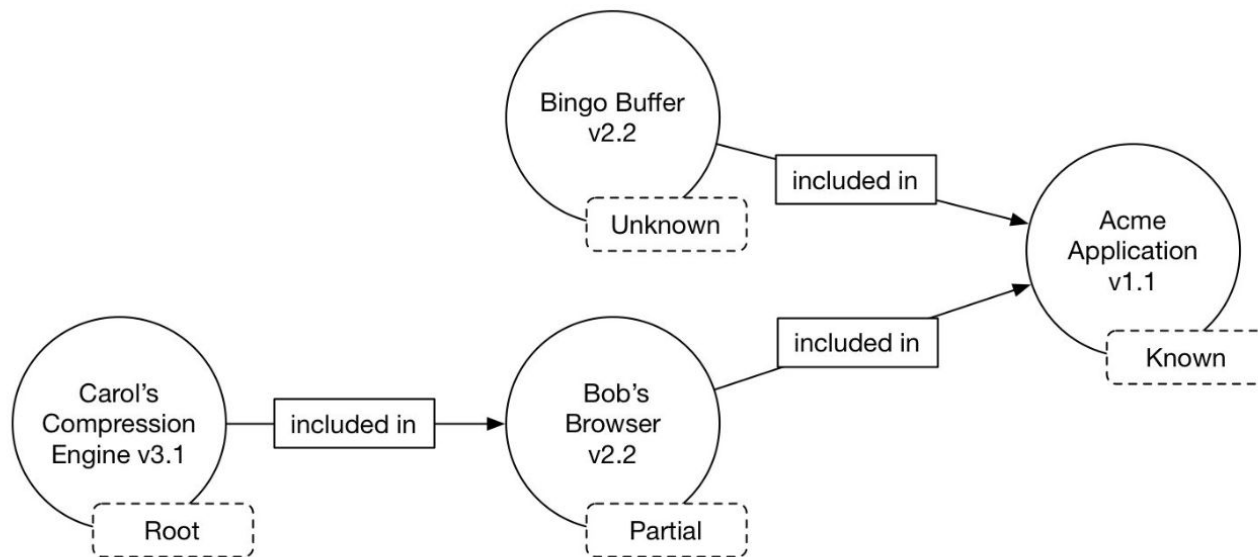


Figure 2: Conceptual SBOM tree with upstream relationship assertions

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Assertion
Application	Acme	1.1	Acme	0x123	234	Self	Known
--- Browser	Bob	2.1	Bob	0x223	334	Included in	Partial
--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in	Root
--- Buffer	Bingo	2.2	Acme	0x423	534	Included in	Unknown

# **SBOMs also combine horizontally**

Doesn't have to be “included in”:

- “was built by”
- “was present when built”
- “generated by”
- “patched with”
- “read data from”
- etc.

You can keep adding info and improving analysis independently.

# Do you know:

(the answer better be yes)

What you are deploying where?

Where it came from?

What's in it?

---