SIGPwny

SP2024 Week 14 • 2024-04-25

# Supply Chain Attacks & Policy

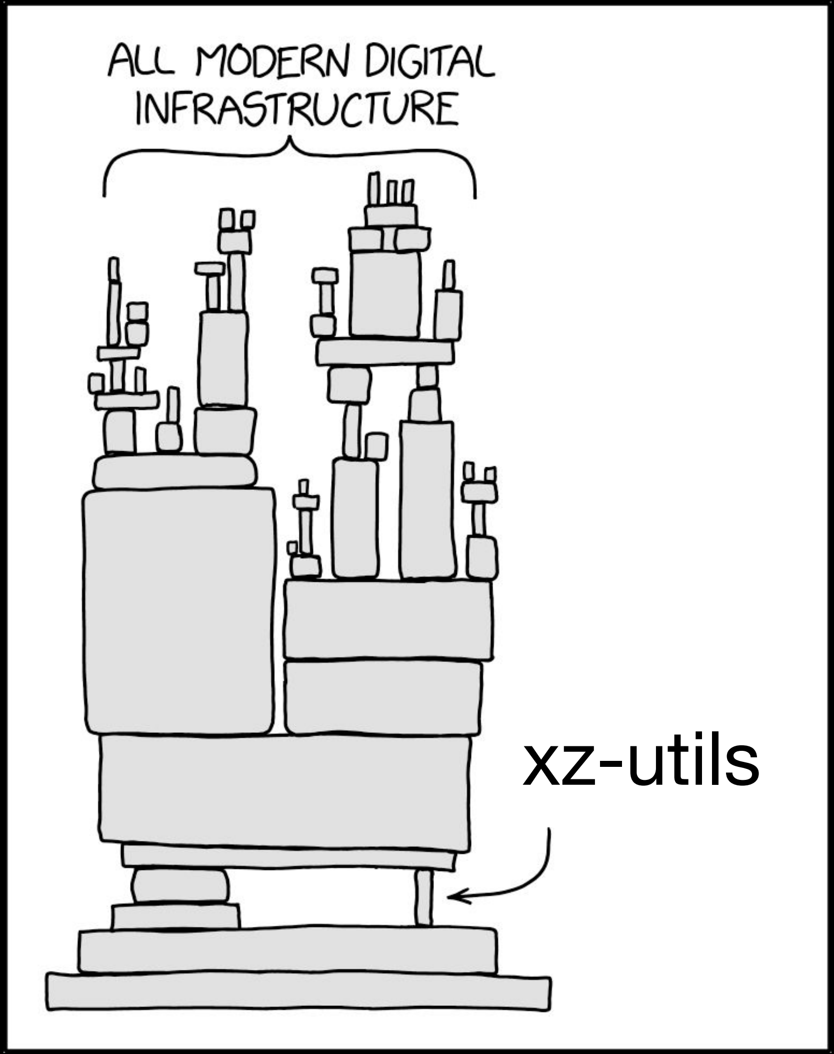
SIGPolicy, Pete

## Announcements

- eCTF is happening right now
- Defcon Quals are May 4th 📈📈📈📈📈📈
- **Last meeting 😢😢😢😢😢😢😢:**
  - MPC Computation with Sagnik this Sunday
    - “the fun part of ECE 407”

ctf.sigpwny.com

## sigpwny{trust}



## XZ Attack Background

Recent Supply Chain Attack

## Table of Contents

# XZ Attack

– Backdoor discovered in **xz-utils**, software/libraries for lossless compression (used in compressing data)

  - xz-utils is widely used and installed on almost all Linux/macOS systems

– Attack was **obfuscated & technically complex**

- Multi-stage payload (which we will discuss)
- Due to the attack complexity, many experts believe this was a **nation-state attack**
- If this wasn't caught early, could have affected millions of machines

## Compromised Systems

Version: xz 5.6.0 or 5.6.1, on AMD64 GLIBC Linux versions

- Debian sid
- Fedora 40
- Fedora Rawhide
- openSUSE Tumbleweed
- openSUSE MicroOS

check with

```
xz --version
```

## Potentially Affected Systems

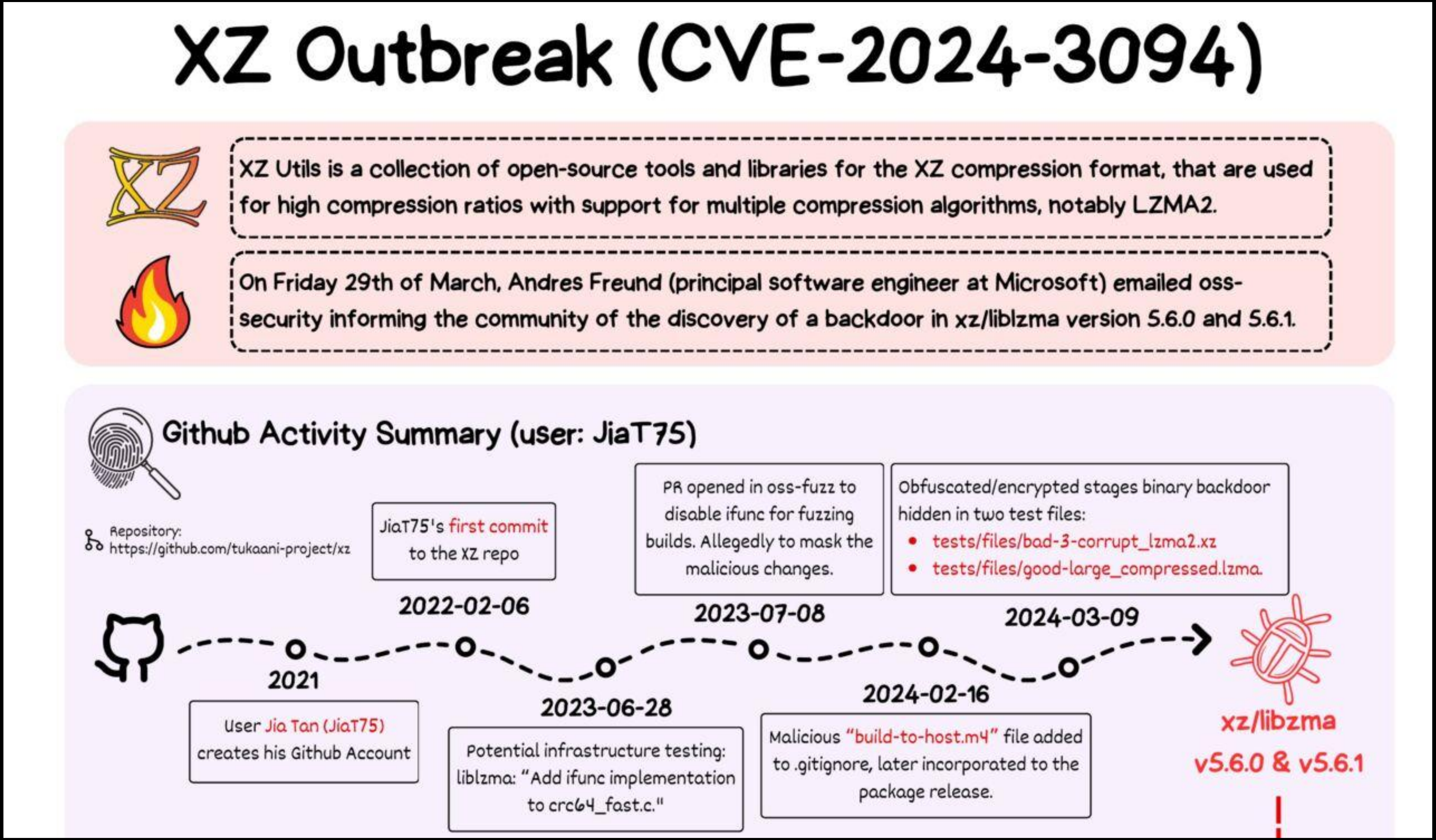
Alpine Edge
**Arch**
Cygwin
Exherbo
Gentoo
**Homebrew**
KaOS
MacPorts

Manjaro Testing
NixOS Unstable/nixpkgs unstable
OpenIndiana
OpenMamba
OpenMandriva Rolling
Parabola

PCLinuxOS
Pisi Linux
pkgsrc current
Ravenports
Slackware current
Solus
Termux
Wikidata

From here.

## Attack Overview



# XZ Outbreak (CVE-2024-3094)

XZ Utils is a collection of open-source tools and libraries for the XZ compression format, that are used for high compression ratios with support for multiple compression algorithms, notably LZMA2.

On Friday 29th of March, Andres Freund (principal software engineer at Microsoft) emailed oss-security informing the community of the discovery of a backdoor in xz/liblzma version 5.6.0 and 5.6.1.

## Github Activity Summary (user: JiaT75)

Repository: https://github.com/tukaani-project/xz

2021 — User Jia Tan (JiaT75) creates his Github Account

2022-02-06 — JiaT75's first commit to the XZ repo

2023-06-28 — Potential infrastructure testing: liblzma: "Add ifunc implementation to crc64_fast.c."

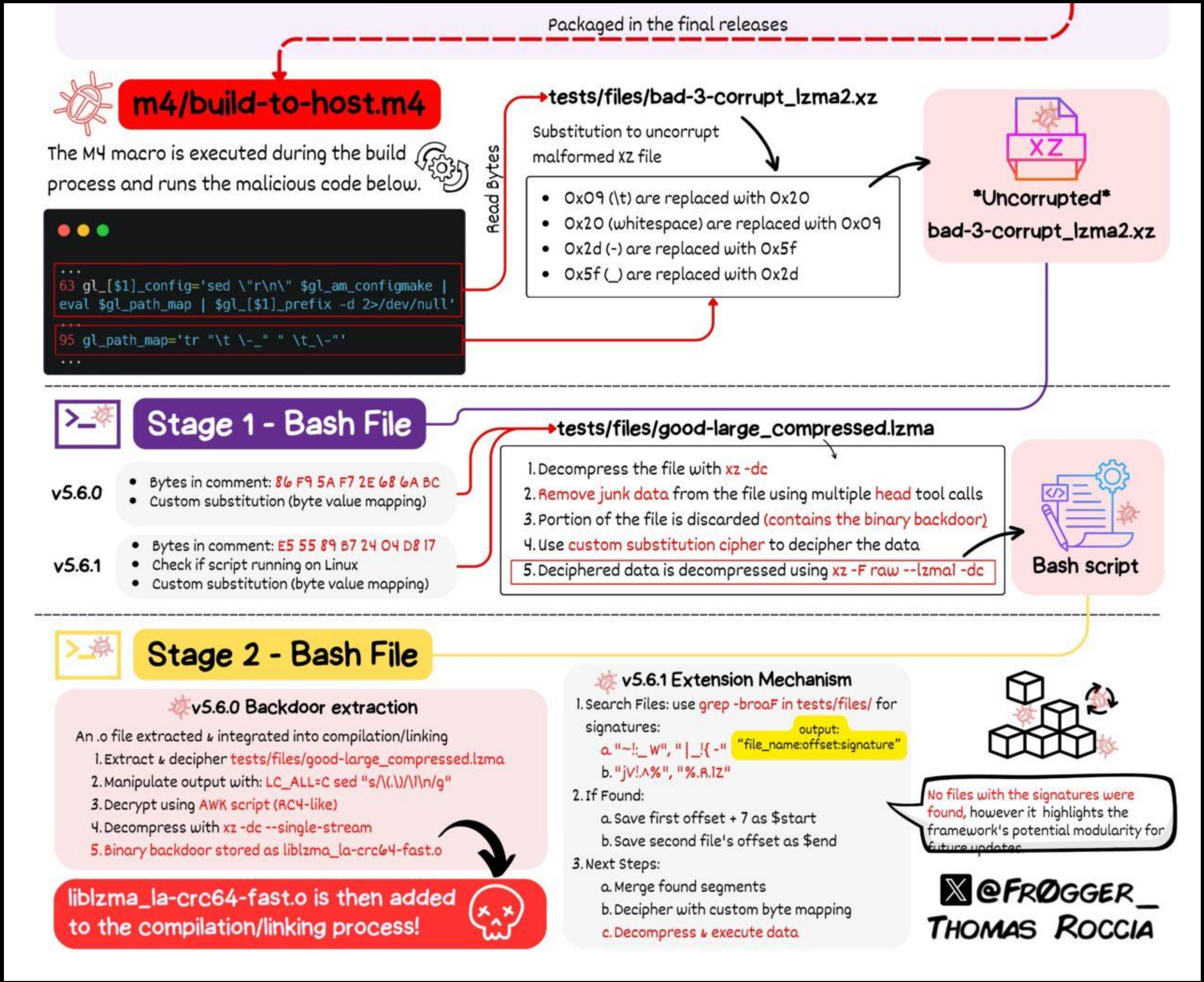
2023-07-08 — PR opened in oss-fuzz to disable ifunc for fuzzing builds. Allegedly to mask the malicious changes.

2024-02-16 — Malicious "build-to-host.m4" file added to .gitignore, later incorporated to the package release.

2024-03-09 — Obfuscated/encrypted stages binary backdoor hidden in two test files:

- tests/files/bad-3-corrupt_lzma2.xz
- tests/files/good-large_compressed.lzma.

xz/libzma v5.6.0 & v5.6.1



Packaged in the final releases

**m4/build-to-host.m4**

The M4 macro is executed during the build process and runs the malicious code below.

```
...
gl_[$1]_config='sed \"r\n\" $gl_am_configmake |
eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'
...
gl_path_map='tr "\t \-_" " \t_\-"'
...
```

Read Bytes

**tests/files/bad-3-corrupt_lzma2.xz**

Substitution to uncorrupt malformed XZ file

- 0x09 (\t) are replaced with 0x20
- 0x20 (whitespace) are replaced with 0x09
- 0x2d (-) are replaced with 0x5f
- 0x5f (_) are replaced with 0x2d

***Uncorrupted* bad-3-corrupt_lzma2.xz**

**Stage 1 - Bash File**

**v5.6.0**

- Bytes in comment: 86 F9 5A F7 2E 68 6A BC
- Custom substitution (byte value mapping)

**v5.6.1**

- Bytes in comment: E5 55 89 B7 24 04 D8 17
- Check if script running on Linux
- Custom substitution (byte value mapping)

**tests/files/good-large_compressed.lzma**

1. Decompress the file with xz -dc
2. Remove junk data from the file using multiple head tool calls
3. Portion of the file is discarded (contains the binary backdoor)
4. Use custom substitution cipher to decipher the data
5. Deciphered data is decompressed using xz -F raw --lzma1 -dc

**Bash script**

**Stage 2 - Bash File**

**v5.6.0 Backdoor extraction**

An .o file extracted & integrated into compilation/linking

1. Extract & decipher tests/files/good-large_compressed.lzma
2. Manipulate output with: LC_ALL=C sed "s/\(.\)/\1\n/g"
3. Decrypt using AWK script (RC4-like)
4. Decompress with xz -dc --single-stream
5. Binary backdoor stored as liblzma_la-crc64-fast.o

**liblzma_la-crc64-fast.o is then added to the compilation/linking process!**

**v5.6.1 Extension Mechanism**

1. Search Files: use grep -broaF in tests/files/ for signatures:
   a. "~!:_W", "|_!{ -"
   b. "jV!.^%", "%.R.1Z"

   output: "file_name:offset:signature"
2. If Found:
   a. Save first offset + 7 as $start
   b. Save second file's offset as $end
3. Next Steps:
   a. Merge found segments
   b. Decipher with custom byte mapping
   c. Decompress & execute data

No files with the signatures were found, however it highlights the framework's potential modularity for future updates.

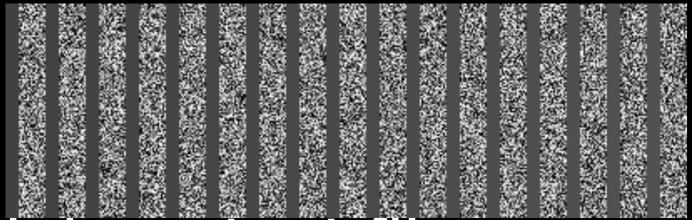
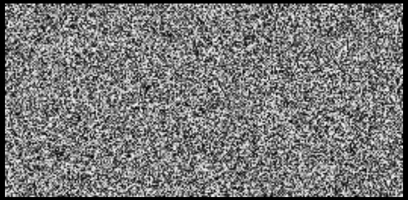
@FR0GGER_ THOMAS ROCCIA

## Attack Details

Summary of various sites from here.

## Overview

- build-to-host.m4 in the release tarballs differs from the upstream on GitHub (common in C projects)
    - … except this time the file is malicious
- build-to-host.m4 unpacks malicious test files
    - tests/files/bad-3-corrupt_lzma2.xz
    - tests/files/good-large_compressed.lzma

- malicious test files are ran as a script
    - IFUNC (GLIBC indirect function call) performs runtime hooking of OpenSSH authentication routines
    - allows RCE on host

## You are important!

OSC: Don’t want additional burdens; want support; increased resources

Government: Want to ensure security in critical infrastructure

Companies: Want profit; want security to protect reputation and government contracts

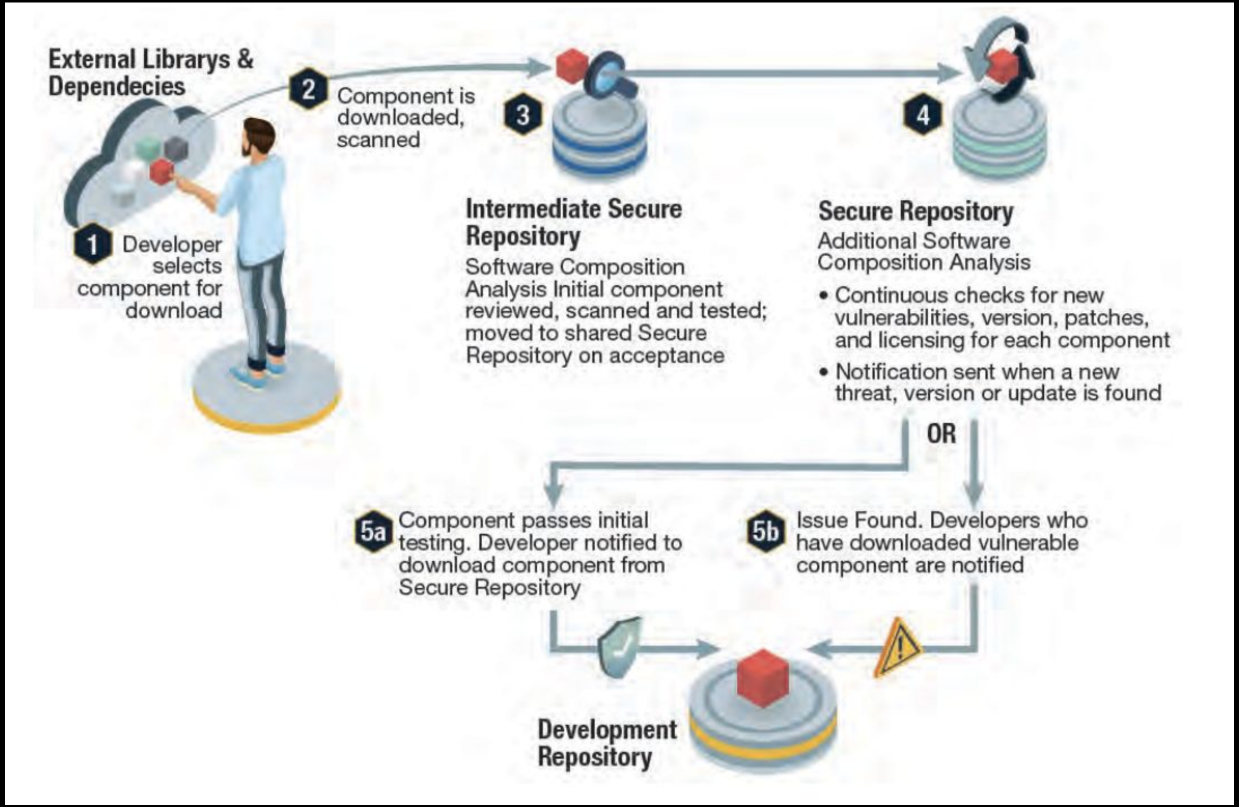
Get ready for a grab bag of policies!

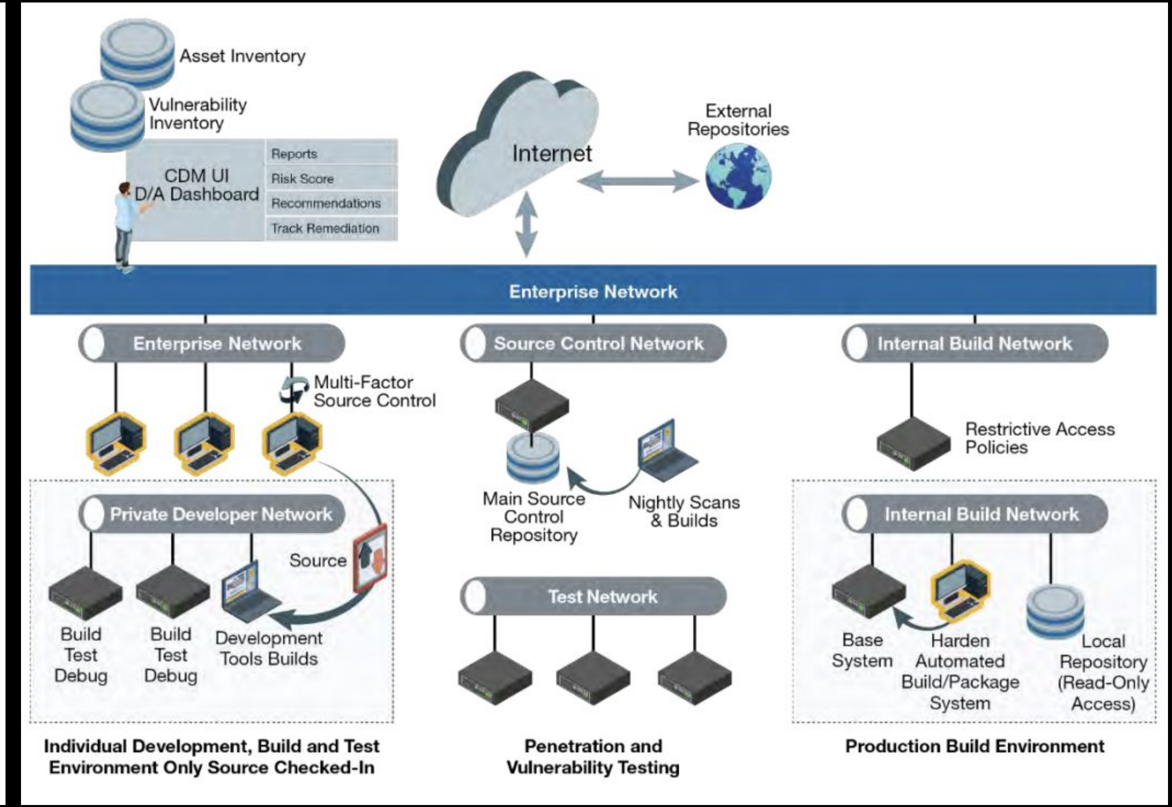
## Supply Chain Policies

## EO 14028

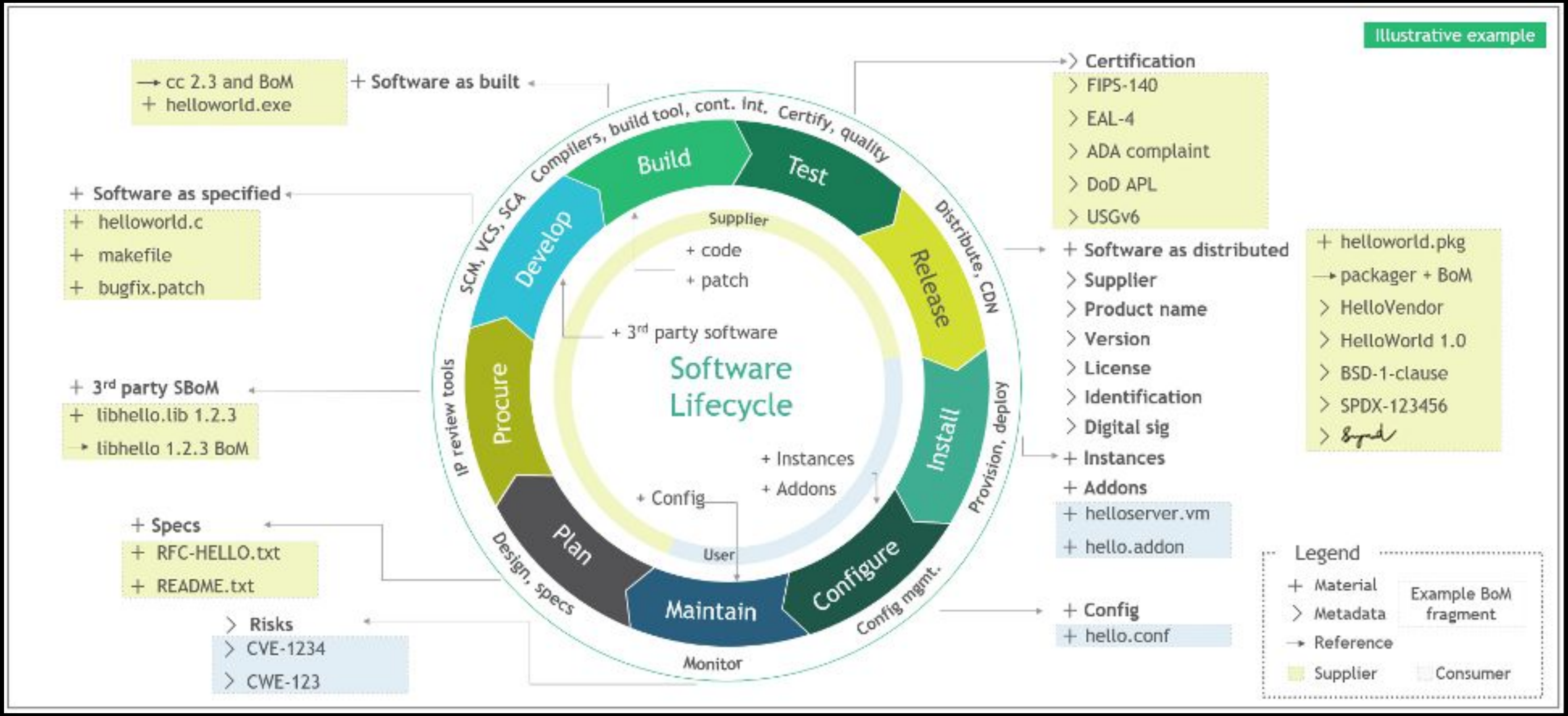
- NIST ⇒ Software Supply Chain Security
- NTIA ⇒ Minimum elements of SBOM
- DNI ⇒ “Critical” Software
- FAR Council ⇒ Remove software products that do not meet the requirements from government contracts
- NIST ⇒ IOT Consumer Labeling Program

# Securing the Software Supply Chain


External Librarys & Dependecies
1 Developer selects component for download
2 Component is downloaded, scanned
3
Intermediate Secure Repository
Software Composition Analysis Initial component reviewed, scanned and tested; moved to shared Secure Repository on acceptance
4
Secure Repository
Additional Software Composition Analysis
• Continuous checks for new vulnerabilities, version, patches, and licensing for each component
• Notification sent when a new threat, version or update is found
OR
5a Component passes initial testing. Developer notified to download component from Secure Repository
5b Issue Found. Developers who have downloaded vulnerable component are notified
Development Repository



# SBOM


Illustrative example
→ cc 2.3 and BoM
+ helloworld.exe
+ Software as built
Compilers, build tool, cont. int.
Certify, quality
Certification
FIPS-140
EAL-4
ADA complaint
DoD APL
USGv6
+ Software as specified
+ helloworld.c
+ makefile
+ bugfix.patch
SCM, VCS, SCA
Build
Test
Develop
Release
Distribute, CDN
Supplier
+ code
+ patch
+ 3rd party software
Software Lifecycle
+ Software as distributed
Supplier
Product name
Version
License
Identification
Digital sig
+ helloworld.pkg
→ packager + BoM
HelloVendor
HelloWorld 1.0
BSD-1-clause
SPDX-123456
+ 3rd party SBoM
+ libhello.lib 1.2.3
→ libhello 1.2.3 BoM
IP review tools
Procure
Install
Provision, deploy
+ Instances
+ Addons
+ Instances
+ Addons
+ helloserver.vm
+ hello.addon
+ Config
User
Plan
Maintain
Configure
+ Specs
+ RFC-HELLO.txt
+ README.txt
Design, specs
Config mgmt.
+ Config
+ hello.conf
Risks
CVE-1234
CWE-123
Monitor
Legend
+ Material
Metadata
→ Reference
Example BoM fragment
Supplier
Consumer

## Critical Software

EO-critical software is defined as any software that has, or has direct software dependencies

upon, one or more components with at least one of these attributes:

- is designed to run with elevated privilege or manage privileges;
- has direct or privileged access to networking or computing resources;
- is designed to control access to data or operational technology;
- performs a function critical to trust; or,
- operates outside of normal trust boundaries with privileged access.

## Cybersecurity Requirements for Consumer-Grade Router Products

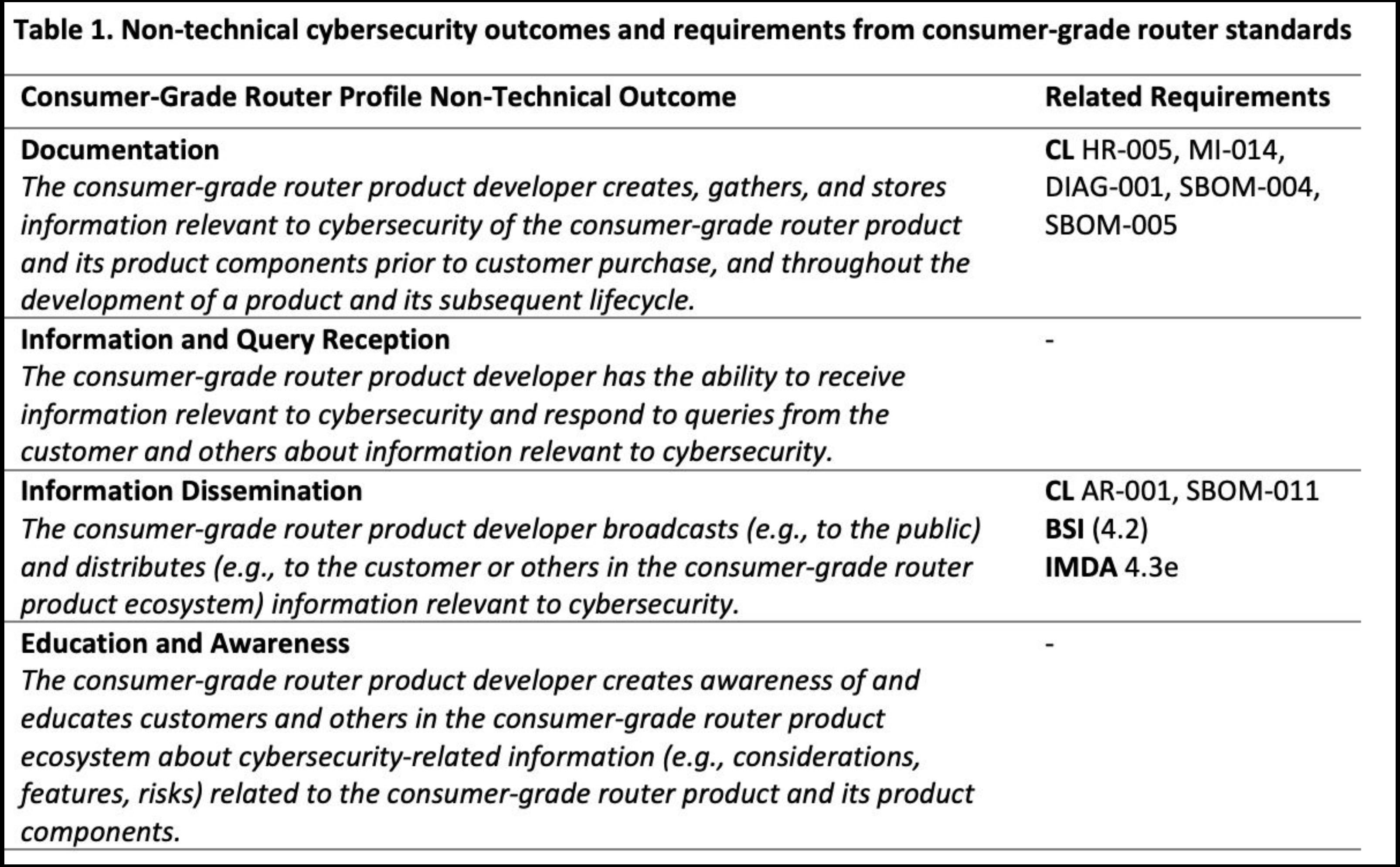


**Table 1. Non-technical cybersecurity outcomes and requirements from consumer-grade router standards**

| Consumer-Grade Router Profile Non-Technical Outcome | Related Requirements |
|---|---|
| **Documentation** *The consumer-grade router product developer creates, gathers, and stores information relevant to cybersecurity of the consumer-grade router product and its product components prior to customer purchase, and throughout the development of a product and its subsequent lifecycle.* | **CL** HR-005, MI-014, DIAG-001, SBOM-004, SBOM-005 |
| **Information and Query Reception** *The consumer-grade router product developer has the ability to receive information relevant to cybersecurity and respond to queries from the customer and others about information relevant to cybersecurity.* | - |
| **Information Dissemination** *The consumer-grade router product developer broadcasts (e.g., to the public) and distributes (e.g., to the customer or others in the consumer-grade router product ecosystem) information relevant to cybersecurity.* | **CL** AR-001, SBOM-011 **BSI** (4.2) **IMDA** 4.3e |
| **Education and Awareness** *The consumer-grade router product developer creates awareness of and educates customers and others in the consumer-grade router product ecosystem about cybersecurity-related information (e.g., considerations, features, risks) related to the consumer-grade router product and its product components.* | - |

## Securing Open Source Software Act

CISA ⇒

- publishing a framework that assesses the risk of open source software components
- using that framework to assess the security posture of the open source software components on which the federal government relies, and
- conducting a study that assesses the feasibility of applying that open source software risk framework to one or more critical infrastructure sectors with the help of voluntary industry participants.

OMB ⇒

- issue guidance to executive agencies on managing and reducing the risk of open source software
- One or more executive agencies establish a pilot open source program office.

## Software Liability?

- No warranties for software
- Difficulties:
  - Speed of software
  - Complexity of software
- Negligence
- Tort claims
- Safe Harbor Approach
  - Safe Harbor for following best practices
  - Reverse Safe Harbor (product liability) for exploited common vulnerabilities

## Corporations/NGOs

## Methods for Contributing

- Releasing internal projects as open source (ex: Angular, React, z3)
- Paying employees to work on open source projects (ex: Python)
- Donating to open source projects
- Offering bounties for fixing bugs (ex: Internet Bug Bounty, Patch Rewards)
- Giving grants to open source projects (ex: Mozilla, Sentry)

# Red Hat

- Offers various products mostly based on open source code
  - Mostly repackages existing open source code to make it better suited for corporations
- Provides paid support to companies for products (including underlying open source code)
- Responsible for several popular open source libraries (ex: systemd, Ansible)

## NLnet

- Charity based in Netherlands dedicated to funding "open hardware, open software, open data or open standards"
- Anyone is allowed to submit proposals asking for funding
- Multiple funds dedicated to specific areas
- Receives funding from nonprofit organizations and governments
  - NSF Pathways to Enable Open-Source Ecosystems

## Tidelift

- Offers a subscription to companies for help managing open source dependencies
- Collects information about open source projects for companies to use
- Pays open source maintainers to improve security of their project (e.g. creating a vulnerability handling plan) among other things
  - Amount that Tidelift provides to project determined by Tidelift based on amount of usage by subscribers

## Apache Foundation

- Board of directors elected by members manages corporate assets
- Projects managed by individual committees (chair chosen by board of directors, other members internally selected)
- Doesn't pay anyone in organization directly (made up entirely of volunteers)
- Organization as a whole provides legal protection for volunteers

## Communities

# How Maintainers Try Getting Money

- Asking for donations (Patreon, Github Sponsors)
- Commercializing parts of your project
  - Adding features only available to paid subscribers (see Instructure, Docker)
  - Allowing users to pay for support (see Canonical)
- Venture capital fundraising (see npm)
- Making businesses pay you for using your code
  - Make code available with GPL/AGPL, sell commercial licenses
- Make your project closed-source (see MongoDB, Redis)
  - Also see OS.Cash license

## Next Meetings

2024-04-28 • **This Sunday**

- Multi-Party Computation with Sagnik
- Description

ctf.sigpwny.com

sigpwny{trust}

Meeting content can be found at sigpwny.com/meetings.

SIGPwny