Open-Source Supply Chain Security at Google

Russ Cox (he/him)

ACM SCORED

November 2023

go.dev/s/acmscored

Open-Source Supply Chain Security at Google

Russ Cox (he/him)

ACM SCORED

November 2023

go.dev/s/acmscored

“Supply chain security”

What is a software supply chain?

What does it mean to be secure?

“Supply chain security”

What is a software supply chain?
What does it mean to be secure?

Draft: *Supply chain security* is the engineering of defenses against supply chain attacks.

“Supply chain attack”

What is a supply chain attack?

go.dev/s/acmscored

“Supply chain attack”

What is a supply chain attack?

A (software) *supply chain attack* is
the nefarious alteration of trusted software
before delivery.

(tweaking a definition by Kim Zetter)

CIA

This article is more than **3 years old**

CIA controlled global encryption company for decades, says report

Swiss government orders inquiry after revelations Crypto AG was owned and operated by US and German intelligence

Julian Borger *in Washington*

Tue 11 Feb 2020 14.26 EST

go.dev/s/acmscored

go.dev/s/acmscored

go.dev/s/acmscored

Novel Malware XcodeGhost Modifies Xcode, Infects Apple iOS Apps and Hits App Store

By Claud Xiao
September 17, 2015 at 4:00 PM
Category: Malware, Threat Prevention, Unit 42
Tags: Apple, Baidu, iOS, KeyRaider, OS X, Weibo, Xcode, XcodeGhost

This post is also available in: 日本語 (Japanese)

UPDATE: Since this report's original posting on September 17, three additional XCodeGhost updates have been published, available here, here and here.

On Wednesday, Chinese iOS developers disclosed a new OS X and iOS malware on Sina Weibo. Alibaba researchers then posted an analysis report on the malware, giving it the name XcodeGhost. We have investigated the malware to identify how it spreads, the techniques it uses and its impact.

XcodeGhost is the first compiler malware in OS X. Its malicious code is located in a Mach-O object file that was repackaged into some versions of Xcode installers. These malicious installers were then uploaded to Baidu's cloud

go.dev/s/acmscored

RESEARCH HIGHLIGHTS

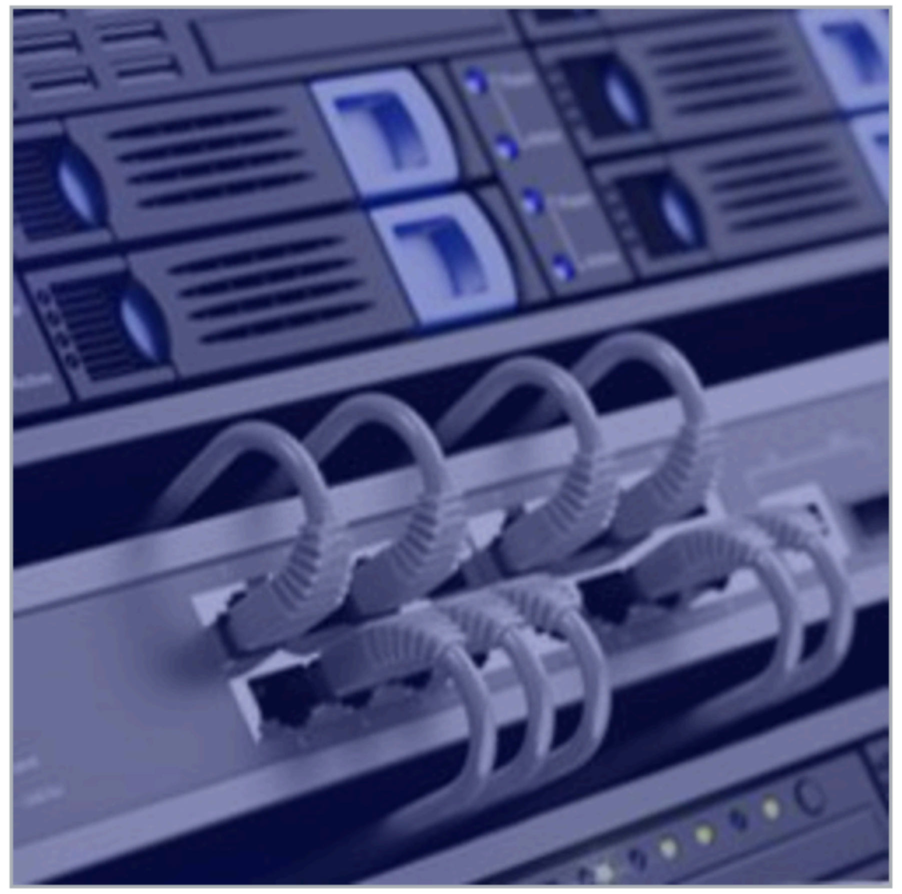
Where Did I Leave My Keys?: Lessons from the Juniper Dual EC Incident

By Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, Hovav Shacham

Communications of the ACM, November 2018, Vol. 61 No. 11, Pages 148-155
10.1145/3266291

Comments

VIEW AS: | SHARE:



Credit: Hacker News

In December 2015, Juniper Networks announced multiple security vulnerabilities stemming from unauthorized code in ScreenOS, the operating system for their NetScreen Virtual Private Network (VPN) routers. The more sophisticated of these vulnerabilities was a passive VPN decryption capability, enabled by a change to one of the parameters used by the Dual Elliptic Curve (EC) pseudorandom number generator.

In this paper, we described the results of a full independent analysis of the ScreenOS randomness and VPN key establishment protocol subsystems, which we carried out in response to this incident. While Dual EC is known to be insecure against an attacker who can choose the elliptic curve parameters, Juniper had claimed in 2013 that ScreenOS included countermeasures against this type of attack. We find that, contrary to Juniper's public statements, the ScreenOS VPN implementation has been vulnerable to passive exploitation by an attacker who selects

ARTICLE CONTENTS:

go.dev/s/acmscored

BY KIM ZETTER BACKCHANNEL MAY 2, 2023 6:00 AM

The Untold Story of the Boldest Supply-Chain Hack Ever

The attackers were in thousands of corporate and government networks. They might still be there now. Behind the scenes of the SolarWinds investigation.

ILLUSTRATION: TAMEEM SANKARI

STEVEN ADAIR WASN'T too rattled at first.

It was late 2019, and Adair, the president of the security firm Volexity, was investigating a digital security breach at an American think tank. The intrusion

go.dev/s/acmscored

"Open-source software supply chain attack"

An *open-source software supply chain attack* is
the nefarious alteration of a trusted open-source component
~~before delivery~~ used later in a trusted program.

go.dev/s/acmscored

POISONING THE WELL —

Widely used open source software contained bitcoin-stealing backdoor

Malicious code that crept into event-stream JavaScript library went undetected for weeks.

DAN GOODIN - 11/26/2018, 5:55 PM

Jeremy Brooks / Flickr

102

A hacker or hackers sneaked a backdoor into a widely used open source code library with the aim of surreptitiously stealing funds stored in bitcoin wallets, software developers said Monday.

The malicious code was inserted in two stages into event-stream, a code library with 2 million downloads that's used by Fortune 500 companies and small startups alike. In stage one, version 3.3.6, published on September 8, included a benign module known as flatmap-stream. Stage two was implemented on October 5 when flatmap-stream was updated to include malicious code that attempted to steal bitcoin wallets and transfer their balances to a server located in Kuala Lumpur. The backdoor came to light last Tuesday with

Project Zero

go.dev/s/acmscored

News and updates from the Project Zero team at Google

Wednesday, December 15, 2021

A deep dive into an NSO zero-click iMessage exploit: Remote Code Execution

Posted by Ian Beer & Samuel Groß of Google Project Zero

We want to thank Citizen Lab for sharing a sample of the FORCEDENTRY exploit with us, and Apple's Security Engineering and Architecture (SEAR) group for collaborating with us on the technical analysis. The editorial opinions reflected below are solely Project Zero's and do not necessarily reflect those of the organizations we collaborated with during this research.

Earlier this year, Citizen Lab managed to capture an NSO iMessage-based zero-click exploit being used to target a Saudi activist. In this two-part blog post series we will describe for the first time how an in-the-wild zero-click iMessage exploit works.

Based on our research and findings, we assess this to be one of the most technically sophisticated exploits we've ever seen, further demonstrating that the capabilities NSO provides rival those previously thought to be accessible to only a handful of nation states.

The vulnerability discussed in this blog post was fixed on September 13, 2021 in iOS 14.8 as CVE-2021-30860

go.dev/s/acmscored

MINECRAFT

GAMES GIFT GUIDE COMMUNITY MERCH SUPPORT GET MINECR...

NEWS

IMPORTANT MESSAGE: SECURITY VULNERABILITY IN JAVA EDITION

Follow these steps to secure your game

Written By
Staff

Published
12/10/2021

Hello everyone! Earlier today, we identified a vulnerability in the form of an exploit within Log4j – a common Java logging library. This exploit affects many services – including Minecraft Java Edition.

This vulnerability poses a potential risk of your computer being compromised, and while this exploit has been addressed with all versions of the game client patched, you still need to take the following steps to secure your game and your servers.

go.dev/s/acmscored

“Open-source supply chain vulnerability”

An *open source supply chain vulnerability* is an exploitable weakness in trusted software caused by an open source component.

“Open-source software supply chain <u>vulnerability</u>”

An *open source supply chain <u>vulnerability</u>* is
an exploitable weakness in a trusted software package
caused by one of that package’s open source components.

- Trusted software need not be open-source (Minecraft is not).

go.dev/s/acmscored

“Open-source supply chain security”

Earlier draft:
Supply chain security is the engineering of defenses against supply chain attacks.

Open source supply chain security is
the engineering of defenses against
open source supply chain *attacks* and
open source supply chain *vulnerabilities*.

go.dev/s/acmscored

Open-Source Supply Chain Security at Google

Disclaimers

- Not exhaustive about efforts at Google.
- No intent to discount work being done elsewhere.
  Just reporting about Google.

go.dev/s/acmscored

Open-source software supply chain security at Google

Three main approaches:

- **Understanding** the supply chain
- **Strengthening** the supply chain
- **Monitoring** the supply chain

Understanding the software supply chain

(The software supply chain is all the places where a supply chain attack might happen.)

go.dev/s/acmscored

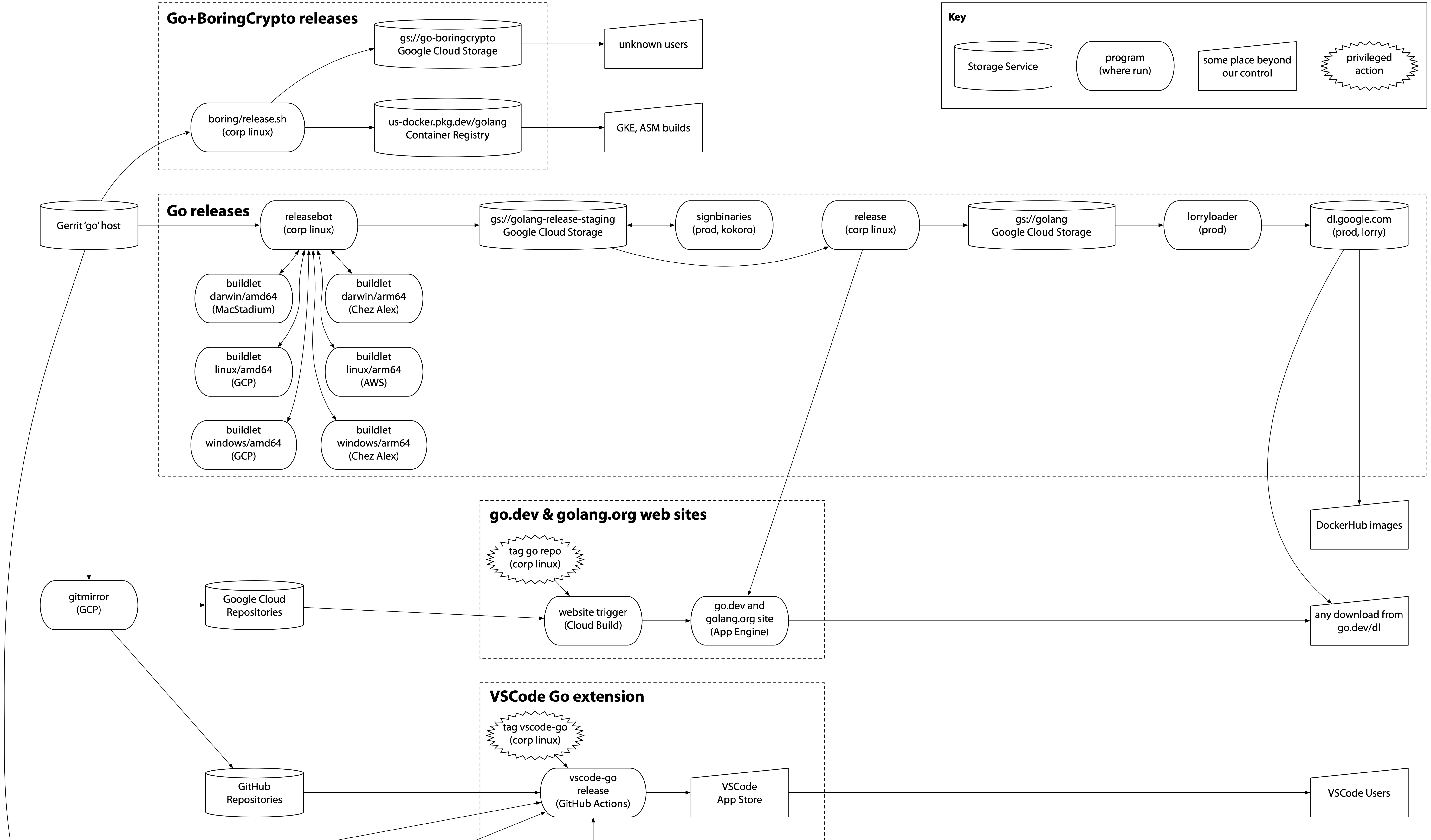
Understanding the software supply chain

Build Graph

go.dev/s/acmscored

Understanding the software supply chain

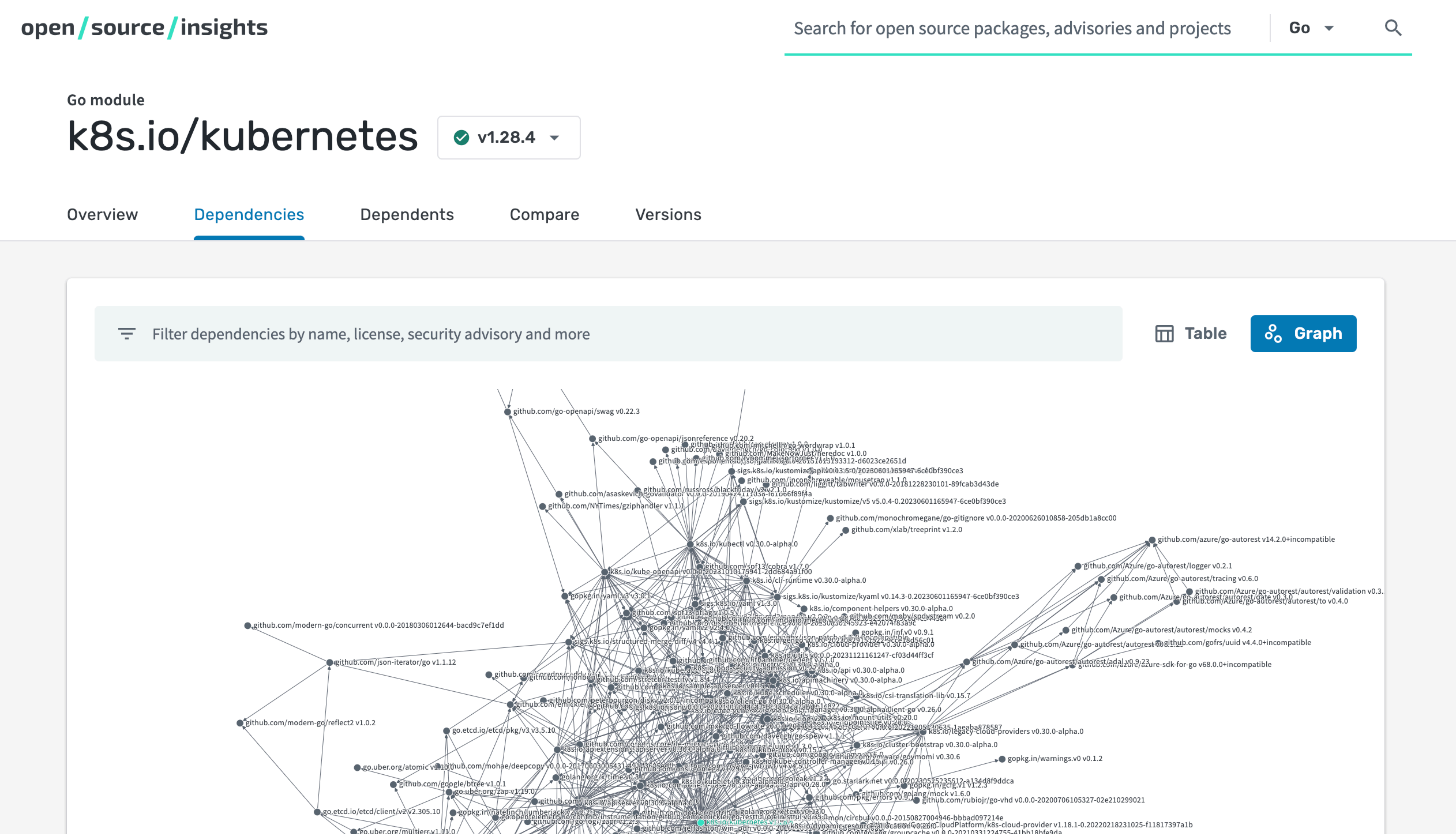
Server Graph (~2020, outdated)



go.dev/s/acmscored

Understanding the software supply chain

Dependency Graph



go.dev/s/acmscored

Understanding the software supply chain

Dependency Graph

Go module

k8s.io/kubernetes v1.28.4

Overview | Dependencies | Dependents | Compare | Versions

v1.28.4	v1.29.0-rc.0
Licenses	**Licenses**
LICENSES Apache-2.0	LICENSES Apache-2.0
DEPENDENCY LICENSES Apache-2.0 BSD-2-Clause BSD-3-Clause CC-BY-SA-4.0 ISC MIT	DEPENDENCY LICENSES Apache-2.0 BSD-2-Clause BSD-3-Clause CC-BY-SA-4.0 ISC MIT
Security Advisories	**Security Advisories**
IN THE DEPENDENCIES	IN THE DEPENDENCIES
– github.com/cyphar/filepath-securejoin v0.2.3 GHSA-6xv5-86q9-7xr8 SecureJoin: on windows, paths outside of the rootfs could be inadvertently produced GO-2023-2048 Paths outside of the rootfs could be produced on Windows	
– go.opentelemetry.io/contrib/instrumentation/github.com/emicklei/go-restful/otelrestful v0.35.0 GHSA-rcjv-mgp8-qvmr OpenTelemetry-Go Contrib vulnerable to denial of service in otelhttp due to unbound cardinality metrics GO-2023-2113 Memory exhaustion in github.com/open-telemetry/opentelemetry-go-contrib	
	+ go.opentelemetry.io/contrib/instrumentation/github.com/emicklei/go-restful/otelrestful v0.42.0 GHSA-rcjv-mgp8-qvmr OpenTelemetry-Go Contrib vulnerable to denial of service in otelhttp due to unbound cardinality metrics GO-2023-2113 Memory exhaustion in github.com/open-telemetry/opentelemetry-go-contrib
go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc v0.42.0 GHSA-8pgv-569h-w5rw otelgrpc DoS vulnerability due to unbound cardinality metrics	go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc v0.42.0 GHSA-8pgv-569h-w5rw otelgrpc DoS vulnerability due to unbound cardinality metrics

Understanding the software supply chain

Strengthening the software supply chain

> Defend against attacks

Find vulnerabilities

Cryptographic Signatures

Cryptographic signatures make it impossible
to nefariously alter code between signing and verifying.

Removes download infrastructure, hosting,
network middleboxes as potential attack sites.

Introduces key distribution problems.

Go Checksum Database

Map from (module, version) -> SHA256 of file tree
Signed by private key; public key hard-coded in Go distribution

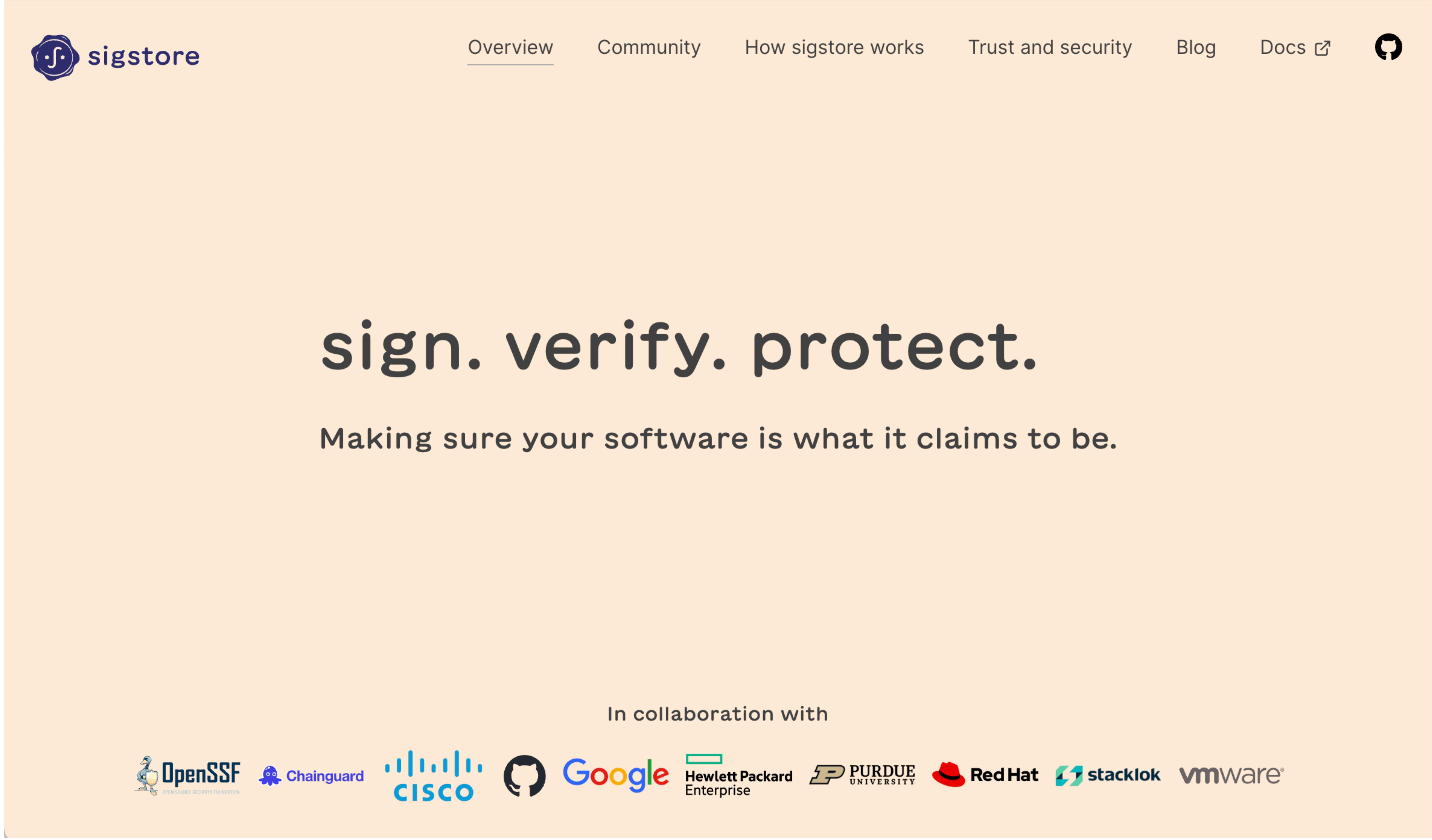
Every download of public module
checks (possibly cached) checksum database entry.

Checksum database assumes first observed copy of code is "correct".
Makes (module, version) -> code mapping immutable.

go.dev/s/acmscored

Strengthening the software supply chain > Attacks

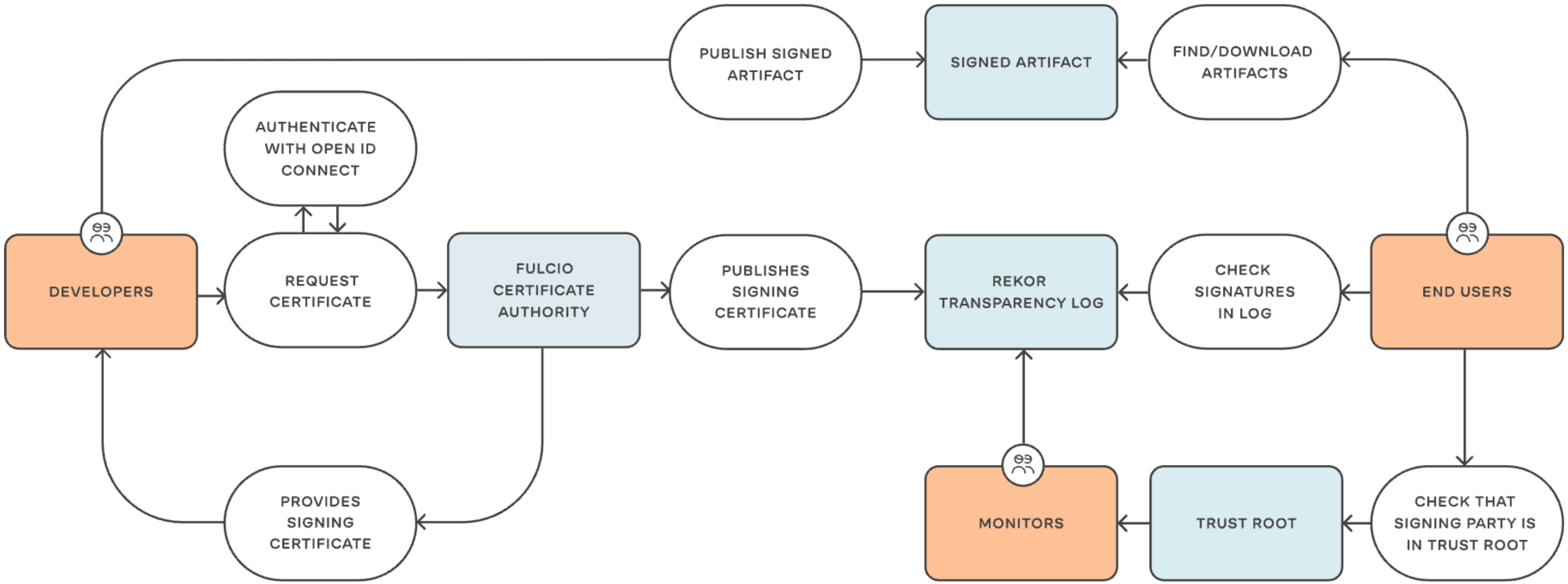
Sigstore



go.dev/s/acmscored

Strengthening the software supply chain > Attacks

Sigstore



Computer System Security

All these boxes need to be secured too.

Dedicated build systems can provide better security, reproducibility:
Google Cloud Build, GitHub Actions.

(But engineering workstations, laptops need security too.)

Strengthening the software supply chain > Attacks

Reproducible builds

The Go Blog

Perfectly Reproducible, Verified Go Toolchains

Russ Cox
28 August 2023

One of the key benefits of open-source software is that anyone can read the source code and inspect what it does. And yet most software, even open-source software, is downloaded in the form of compiled binaries, which are much more difficult to inspect. If an attacker wanted to run a supply chain attack on an open-source project, the least visible way would be to replace the binaries being served while leaving the source code unmodified.

The best way to address this kind of attack is to make open-source software builds *reproducible*, meaning that a build that starts with the same sources produces the same outputs every time it runs. That way, anyone can verify that posted binaries are free of hidden changes by building from authentic sources and checking that the rebuilt binaries are bit-for-bit identical to the posted binaries. That approach proves the binaries have no backdoors or other changes not present in the source code, without having to disassemble or look inside them at all. Since anyone can verify the binaries, independent groups can easily detect and report supply chain attacks.

Two-Person Approvals

Supply-chain Levels for Software Artifacts (SLSA)

Track/Level	Requirements	Focus
Build L0	(none)	(n/a)
Build L1	Provenance showing how the package was built	Mistakes, documentation
Build L2	Signed provenance, generated by a hosted build platform	Tampering after the build
Build L3	Hardened build platform	Tampering during the build

go.dev/s/acmscored

OpenSSF Security Scorecards

Run the checks

- Using the GitHub Action
- Using the CLI

Learn more

- The problem
- What is OpenSSF Scorecard?
- How it works
- The checks
- Use cases
- About the project name
- Part of the OSS community
- Get involved

What is OpenSSF Scorecard?

Scorecard assesses open source projects for security risks through a series of automated checks

It was created by OSS developers to help improve the health of critical projects that the community depends on.

You can use it to proactively assess and make informed decisions about accepting security risks within your codebase. You can also use the tool to evaluate other projects and dependencies, and work with maintainers to improve codebases you might want to integrate.

Scorecard helps you enforce best practices that can guard against:

Malicious maintainers

Build system compromises

Source code compromises

Malicious packages

Capslock

```
% capslock -output=verbose
Capslock is an experimental tool for static analysis of Go packages.

Analyzed packages:
  golang.org/x/text v0.9.0

CAPABILITY_READ_SYSTEM_STATE: 2 references (2 direct, 0 transitive)
Example callpath:
  rsc.io/sampler.Hello
  sampler.go:32:27:rsc.io/sampler.DefaultUserPrefs
  sampler.go:21:22:os.Getenv

CAPABILITY_UNANALYZED: 5 references (0 direct, 5 transitive)
Example callpath:
  rsc.io/sampler.DefaultUserPrefs
  sampler.go:22:39:golang.org/x/text/language.Make
  language.go:44:21:(golang.org/x/text/language.CanonType).Make
  language.go:50:17:(golang.org/x/text/language.CanonType).Parse
  parse.go:58:29:golang.org/x/text/language.canonicalize
  language.go:128:44:(golang.org/x/text/internal/language.Language).Canonicalize
  lookup.go:50:17:golang.org/x/text/internal/language.normLang
  lookup.go:55:18:sort.Search
```

Capslock Limitations

Code operating within its capabilities can still be nefarious.

- String comparison in a password checker.
- YAML parser in a production system.
- XcodeGhost

Strengthening the software supply chain

Defend against attacks
> Find vulnerabilities

OSS Fuzz

Announcing OSS-Fuzz: Continuous fuzzing for open source software

Thursday, December 1, 2016

We are happy to announce OSS-Fuzz, a new Beta program developed over the past years with the Core Infrastructure Initiative community. This program will provide continuous fuzzing for select core open source software.

Open source software is the backbone of the many apps, sites, services, and networked things that make up "the internet." It is important that the open source foundation be stable, secure, and reliable, as cracks and weaknesses impact all who build on it.

Recent security stories confirm that errors like buffer overflow and use-after-free can have serious, widespread consequences when they occur in critical open source software. These errors are not only serious, but notoriously difficult to find via routine code audits, even for experienced developers. That's where fuzz testing comes in. By generating random inputs to a given program, fuzzing triggers and helps uncover errors quickly and thoroughly.

go.dev/s/acmscored

Strengthening the software supply chain > Vulnerabilities

Syzkaller

syzbot Linux

sign-in | mailing list | source | docs

Open [872] | Subsystems | Fixed [4871] | Invalid [11620] | Missing Backports [67] | Kernel Health | Bug Lifetimes | Fuzzing | Crashes | Send us feedback

open (801):

Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported	Discussions
WARNING in __cfg80211_bss_update [wireless]				1	4d18h	18h16m	0 [18h16m]
possible deadlock in stack_depot_put [kernel]				7	1d17h	2d17h	3 [1d05h]
general protection fault in bfs_get_block (2) [bfs]	C	error		1	7d00h	2d23h	PATCH [1d14h]
INFO: task hung in hwrng_fillfn [crypto]	C	error		8	3d09h	3d00h	0 [2d08h]
kernel BUG in ext4_mb_release_inode_pa [ext4]	syz	unreliable		1	7d03h	3d03h	0 [3d03h]
memory leak in j1939_netdev_start [can]	syz			1	7d11h	3d11h	0 [3d11h]
memory leak in clear_state_bit [btrfs]	C			3	7d09h	4d10h	0 [4d10h]
WARNING in indx_insert_into_buffer [ntfs3]	C			2	8d20h	4d21h	0 [3d01h]
go runtime error				6	2d18h	4d22h	0 [4d22h]
KASAN: slab-use-after-free Read in __lock_sock [bluetooth]	C			1	5d23h	5d22h	0 [5d13h]
kernel BUG in entry_points_to_object [reiserfs]	C	done		3	2d14h	5d22h	0 [5d22h]
WARNING in ext4_dio_write_end_io [ext4]	C	done		2	6d20h	5d23h	PATCH [5d04h]
general protection fault in joydev_connect [kernel]				2	87d	6d04h	6 [5d01h]
possible deadlock in ntfs_set_size [ntfs3]				1	10d	6d08h	0 [6d08h]
WARNING in format_decode (3) [bpf] [trace]	C	done		65	4d07h	6d15h	PATCH [1d00h]
KASAN: slab-use-after-free Read in kill_orphaned_pgrp [kernel]				1	76d	6d23h	1 [6d08h]
WARNING in reiserfs_ioctl (2) [reiserfs]				1	12d	8d01h	0 [8d01h]
memory leak in __btrfs_add_free_space [btrfs]	syz			1	12d	8d06h	0 [8d06h]
memory leak in _r8712_init_xmit_priv (2) [usb] [staging]	C			2	7d05h	8d14h	0 [4d03h]
BUG: unable to handle kernel paging request in copy_from_kernel_n…	C	done		4	12d	8d20h	1 [6d21h]

go.dev/s/acmscored

Strengthening the software supply chain > Safe Languages

Internet Worm, November 1988

"All the News That's Fit to Print"

The New York Times

Late Edition

New York: Today, partly sunny, milder. High 59-64. Tonight, mostly cloudy. Low 48-54. Tomorrow, cloudy, windy, rain developing. High 57-62. Yesterday: High 56, low 41. Details, page D16.

VOL.CXXXVIII . . . No. 47,679 Copyright © 1988 The New York Times NEW YORK, FRIDAY, NOVEMBER 4, 1988 50 cents beyond 75 miles from New York City, except on Long Island. 35 CENTS

Gov. Michael S. Dukakis having his picture taken by a 10-year-old fan at a town meeting in Fairless Hills, Pa., during a tour of the Northeast in which he emphasized the drug problem. Page A19. Vice President Bush addressed supporters a… bus, Ohio. Less than a week after… knowledged being a liberal, Mr. Bus… that "this election is not about labels…

Registration Off

'Virus' in Military Computers Disrupts Systems Nationwide

By JOHN MARKOFF

In an intrusion that raises questions about the vulnerability of the nation's computers, a Department of Defense network has been disrupted since Wednesday by a rapidly spreading "virus" … military officials, researchers and corporations.

While some sensitive military data are involved, the computers handling the nation's most sensitive secret information, like that …

… it there for some time," said … the program can be passed to …

PENTAGON REPORTS IMPROPER CHARGES FOR CONSULTANTS

CONTRACTORS CRITICIZED

…ws Routine Billing …ment by Industry … Some Dubious

…H. CUSHMAN Jr. …The New York Times

…ON, Nov. 3 — A Penta…on has found that the na…ilitary contractors rou…he Defense Department …f millions of dollars paid … often without justifica…

…f the investigation said … the military's current …ontractors' own policies … to assure that the Gov… not improperly pay for …anged consulting work. …Senior Defense Department officials … said the Pentagon was proposing …

"It has raised the public awareness to a considerable degree. It is likely to make people more careful and more attentive to vulnerabilities in the future."
– Robert H. Morris,
quoted in the next day's paper

go.dev/s/acmscored

Memory-Safe Languages

National Security Agency | Cybersecurity Information Sheet

Software Memory Safety

Executive summary

Modern society relies h…
developers to write softw…
compromised for maliciou…
prepare the logic in softwar…
vulnerabilities are still freque…

The path forward

Memory issues in software comprise a large portion of the exploitable vulnerabilities in existence. NSA advises organizations to consider making a strategic shift from programming languages that provide little or no inherent memory protection, such as C/C++, to a memory safe language when possible. Some examples of memory safe languages are C#, Go, Java, Ruby™, and Swift®. Memory safe languages provide

go.dev/s/acmscored

Strengthening the software supply chain > Safe Languages

Memory-Safe Languages

National Security Agency | Cybersecurity Information Sheet

Software Memory Safety

Executive summary

Modern society relies h…
developers to write softw… …be
compromised for maliciou… …orm rigorous testing to
prepare the logic in softwar… …xploitable software
vulnerabilities are still freque… …mory issues. Examples include

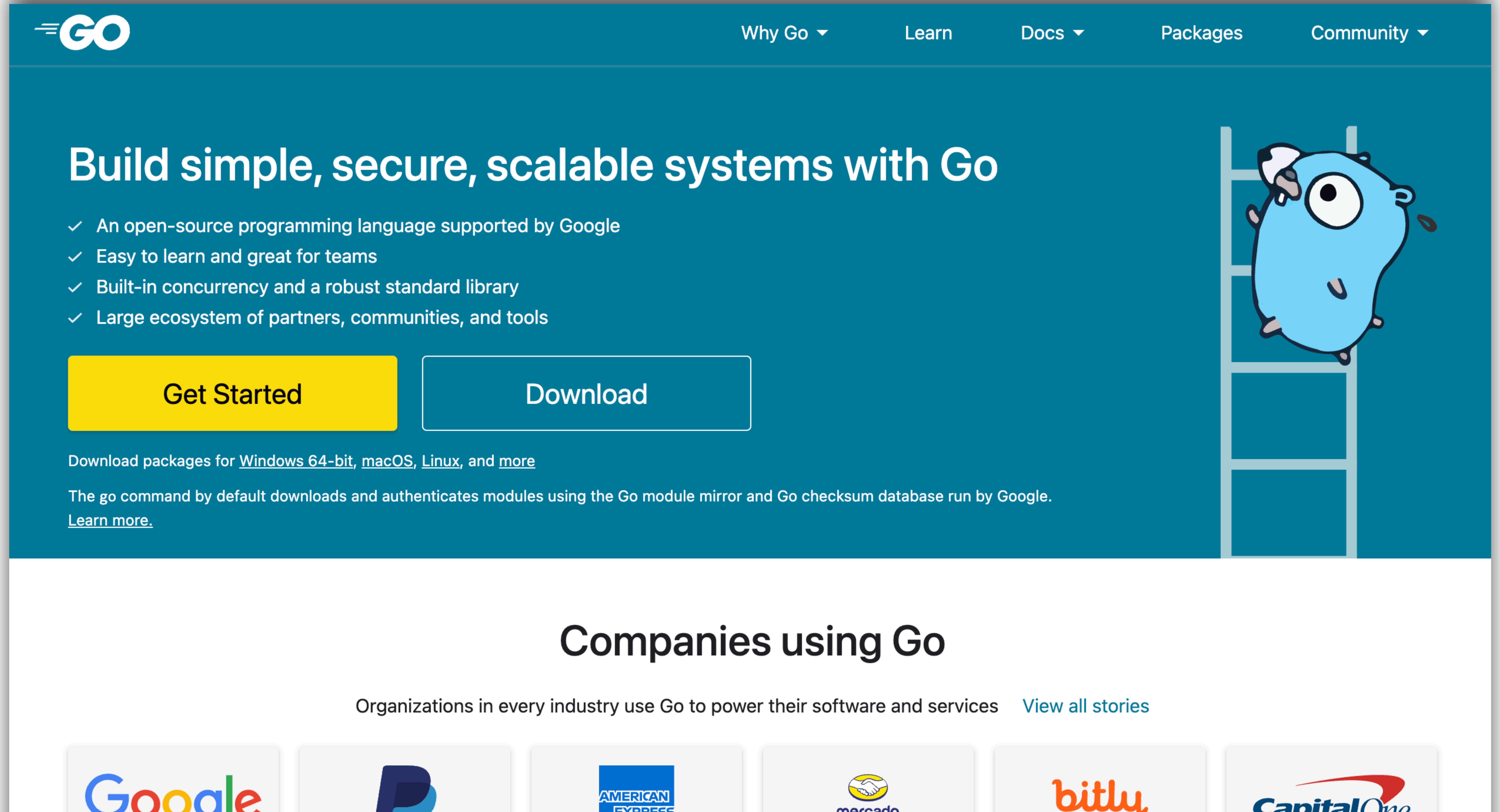
The path forward

Memory issues in software comprise a large portion of the exploitable vulnerabilities in existence. NSA advises organizations to consider making a strategic shift from programming languages that provide little or no inherent memory protection, such as C/C++, to a memory safe language when possible. Some examples of memory safe languages are C#, Go, Java, Ruby™, and Swift®. Memory safe languages provide

go.dev/s/acmscored

Strengthening the software supply chain > Safe Languages

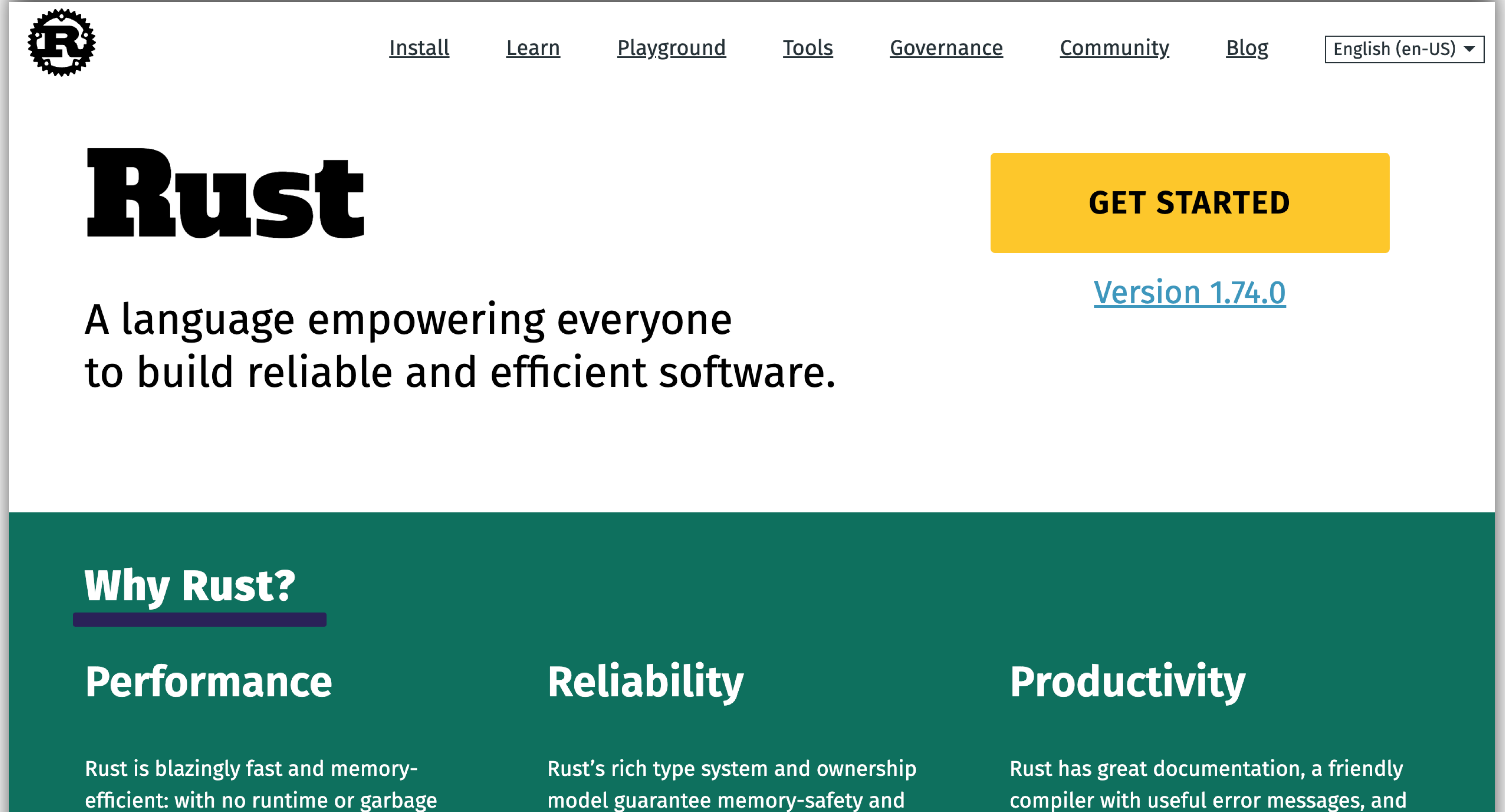
Memory-Safe Languages



go.dev/s/acmscored

Strengthening the software supply chain > Safe Languages

Memory-Safe Languages



Serious Vulnerabilities by Languages

C/C++: Buffer overflow => Remote code execution

Serious Vulnerabilities by Languages

C/C++: Buffer overflow => Remote code execution

Java: Misuse of reflection, code loading => Remote code execution

Serious Vulnerabilities by Languages

C/C++: Buffer overflow => Remote code execution

Java: Misuse of reflection, code loading => Remote code execution

Go: Large or malformed inputs => Denial of service

Rust: Large or malformed inputs => Denial of service

Monitoring the software supply chain

What is a Software Bill of Materials (SBOM)?

I don’t know. Do you?

go.dev/s/acmscored

Monitoring the software supply chain

What is a Software Bill of Materials (SBOM)?

I don’t know. Do you?

Definitely includes list of packages and version.

Monitoring the software supply chain

What is a Software Bill of Materials (SBOM)?

"An SBOM is effectively a nested inventory, a list of ingredients that make up software components. An SBOM identifies and lists software components, information about those components, and supply chain relationships between them. The amount and type of information included in a particular SBOM may vary depending on factors such as the industry or sector and the needs of SBOM consumers."

– "Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)", NTIA Multistakeholder Process on Software Component Transparency Framing Working Group, 2021

What is a Software Bill of Materials (SBOM)?

Definitely includes list of packages and version.

go.dev/s/acmscored

Monitoring the software supply chain

Go “SBOM”

```
% go version -m $HOME/bin/gomote
/Users/rsc/bin/gomote: go1.21.0
	path	golang.org/x/build/cmd/gomote
	mod	golang.org/x/build	v0.0.0-20230809040836-4f3589752dd4	h1:gK+EqJ6LNNP/...
	dep	cloud.google.com/go/compute/metadata	v0.2.3	h1:mg4jlk7mCAj6...
	dep	github.com/aws/aws-sdk-go	v1.30.15	h1:Sd8QDVzzE8Sl...
	dep	github.com/golang/groupcache	v0.0.0-20210331224755-41bb18bfe9da	h1:oI5xCqsCo564...
	dep	github.com/golang/protobuf	v1.5.3	h1:KhyjKVUg7Usr...
	dep	github.com/google/s2a-go	v0.1.4	h1:1kZ/sQM3sreP...
	dep	github.com/google/uuid	v1.3.0	h1:t6JiXgmwXMjE...
	dep	github.com/googleapis/enterprise-certificate-proxy v0.2.3	h1:yk9/cqRKtT9w...
	dep	github.com/googleapis/gax-go/v2 v2.10.0	h1:ebSgKfMxynOd...
	dep	github.com/jmespath/go-jmespath v0.4.0	h1:BEgLn5cpjn8U...
	dep	go.opencensus.io	v0.24.0	h1:y73uSU6J157Q...
	dep	golang.org/x/crypto	v0.9.0	h1:LF6fAI+IutBo...
	dep	golang.org/x/net	v0.10.0	h1:X2//UzNDwYmt...
	dep	golang.org/x/oauth2	v0.8.0	h1:6dkIjl3j3LtZ...
...
```

go.dev/s/acmscored

Monitoring the software supply chain

Go Vulnerability Database

go.dev/s/acmscored

Monitoring the software supply chain

Open-Source Vulnerability Database (OSV)

Monitoring the software supply chain

Vulnerability Scanning

```
% govulncheck -mode=binary gomote
Scanning your binary for known vulnerabilities...

Vulnerability #1: GO-2023-2153
    Denial of service from HTTP/2 Rapid Reset in google.golang.org/grpc
  More info: https://pkg.go.dev/vuln/GO-2023-2153
  Module: google.golang.org/grpc
    Found in: google.golang.org/grpc@v1.55.0
    Fixed in: google.golang.org/grpc@v1.58.3
    Example traces found:
      #1: grpc.Server.Serve
      #2: transport.NewServerTransport

Vulnerability #2: GO-2023-2043
    Improper handling of special tags within script contexts in html/template
  More info: https://pkg.go.dev/vuln/GO-2023-2043
  Standard library
    Found in: html/template@go1.21
    Fixed in: html/template@go1.21.1
    Example traces found:
      #1: template.Template.Execute
      #2: template.Template.ExecuteTemplate
```

go.dev/s/acmscored

Monitoring the software supply chain

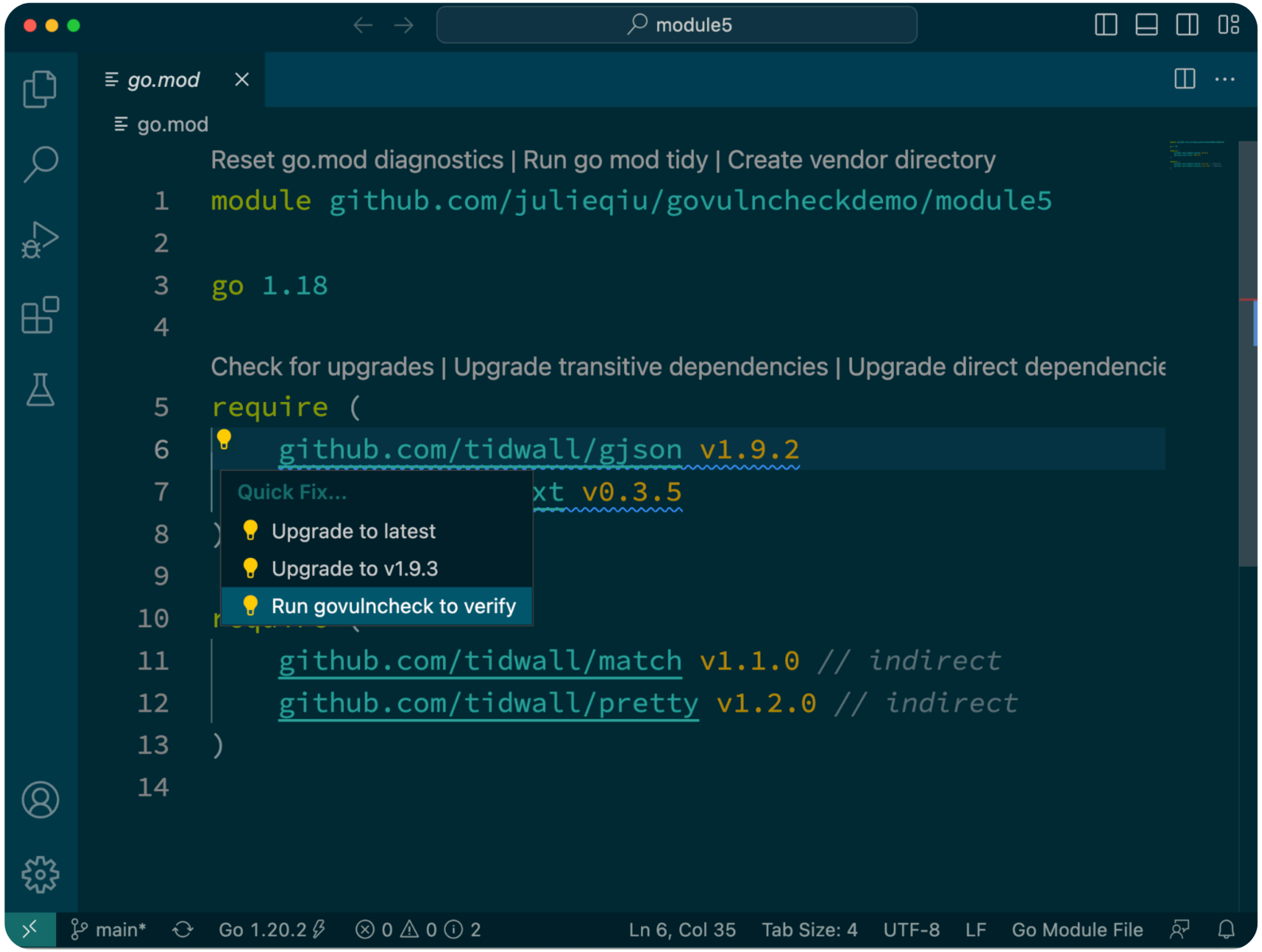
Vulnerability Scanning

```
% govulncheck ./gomote
Scanning your code and 399 packages across 23 dependent modules for known vulnerabilities...

=== Informational ===

Found 2 vulnerabilities in packages that you import, but there are no call
stacks leading to the use of these vulnerabilities. You may not need to
take any action. See https://pkg.go.dev/golang.org/x/vuln/cmd/govulncheck
for details.

Vulnerability #1: GO-2023-2153
    Denial of service from HTTP/2 Rapid Reset in google.golang.org/grpc
  More info: https://pkg.go.dev/vuln/GO-2023-2153
  Module: google.golang.org/grpc
    Found in: google.golang.org/grpc@v1.58.2
    Fixed in: google.golang.org/grpc@v1.58.3

Vulnerability #2: GO-2023-2102
  ...

No vulnerabilities found.
```

go.dev/s/acmscored

Monitoring the software supply chain

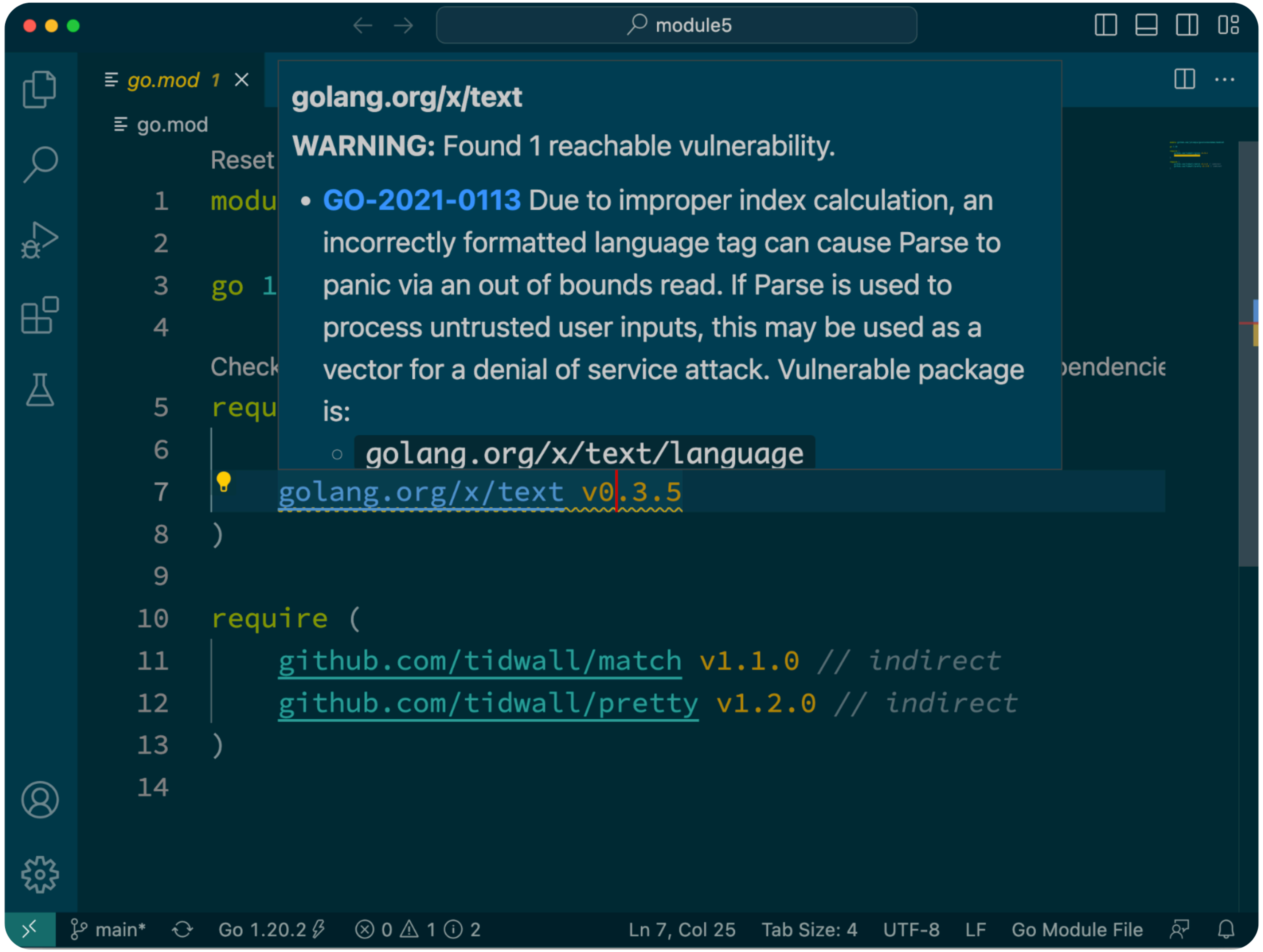
Vulnerability Scanning in IDE



go.dev/s/acmscored

Monitoring the software supply chain

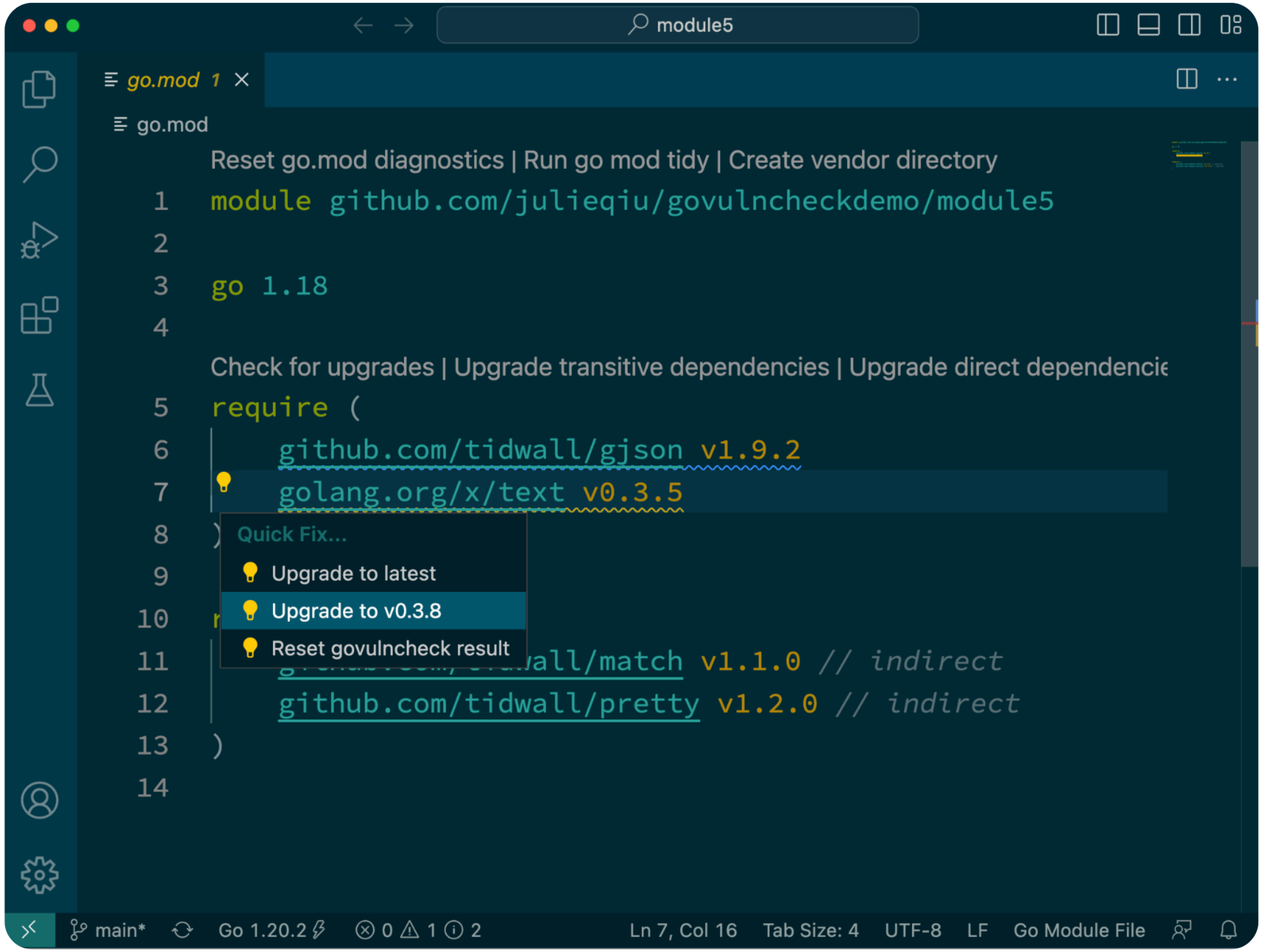
Vulnerability Scanning in IDE



go.dev/s/acmscored

Monitoring the software supply chain

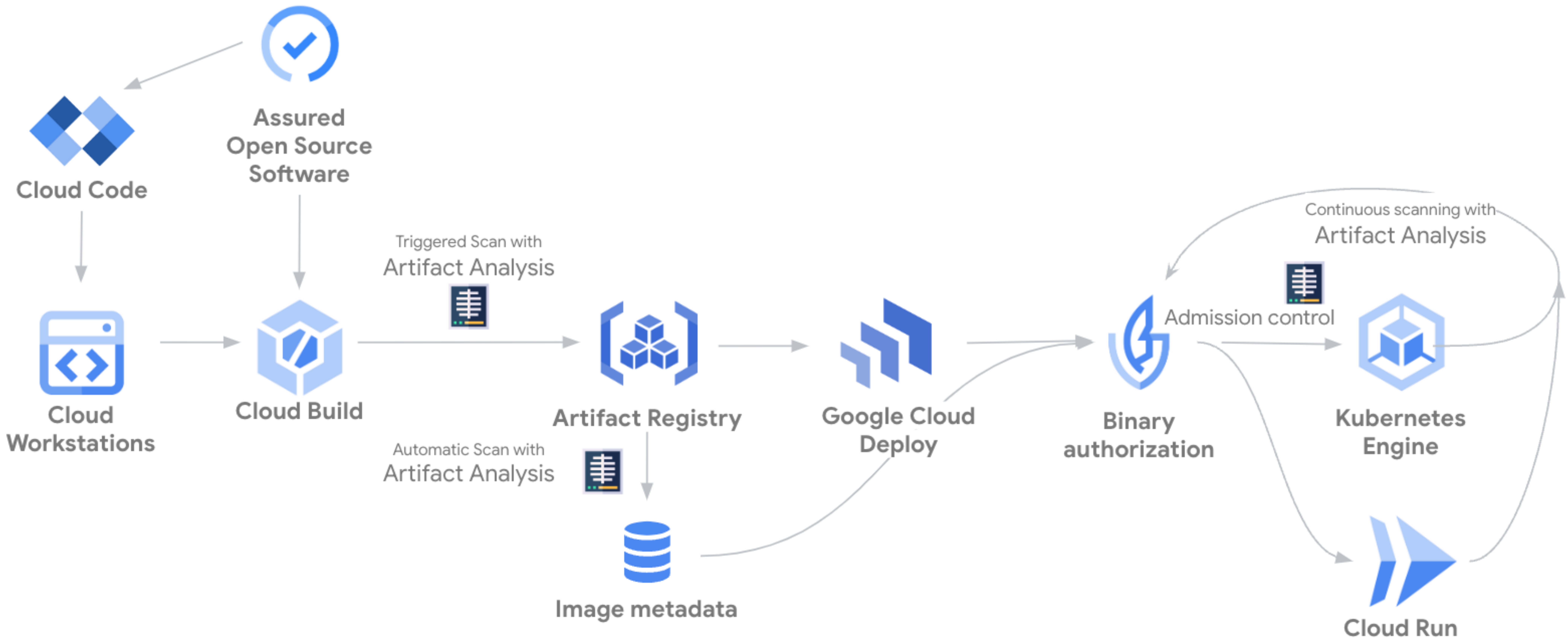
Vulnerability Scanning in IDE



go.dev/s/acmscored

Monitoring the software supply chain

Vulnerability Scanning in Production



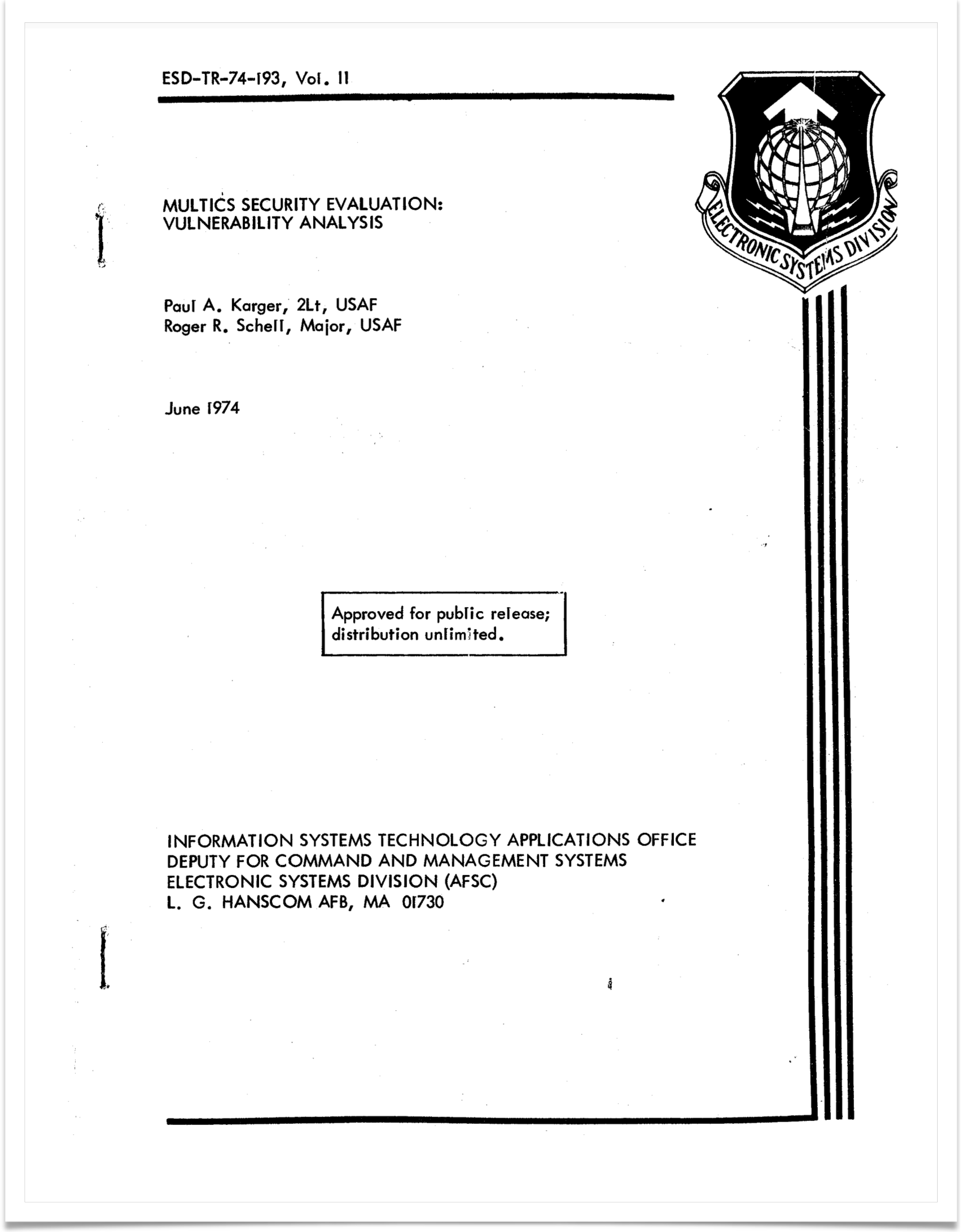
go.dev/s/acmscored

Open-Source Supply Chain Security at Google

Open-Source
Supply Chain Security
Historical Perspective

go.dev/s/acmscored

Air Force review of Multics, 1972–1974



ESD-TR-74-193, Vol. II

MULTICS SECURITY EVALUATION:
VULNERABILITY ANALYSIS

Paul A. Karger, 2Lt, USAF
Roger R. Schell, Major, USAF

June 1974

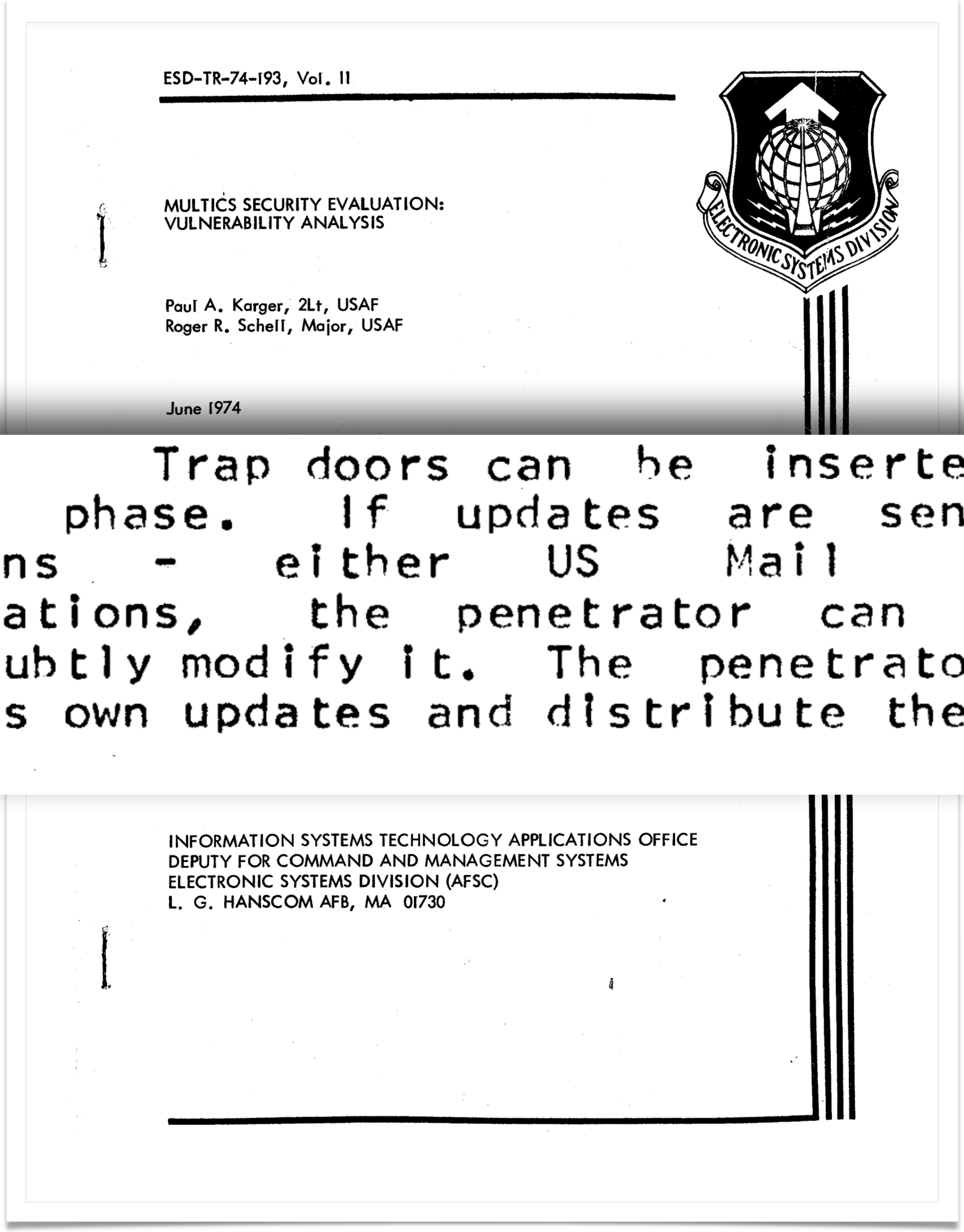
Approved for public release;
distribution unlimited.

INFORMATION SYSTEMS TECHNOLOGY APPLICATIONS OFFICE
DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION (AFSC)
L. G. HANSCOM AFB, MA 01730

ELECTRONIC SYSTEMS DIVISION

go.dev/s/acmscored

Air Force review of Multics, 1972–1974



ESD-TR-74-193, Vol. II

MULTICS SECURITY EVALUATION:
VULNERABILITY ANALYSIS

Paul A. Karger, 2Lt, USAF
Roger R. Schell, Major, USAF

June 1974

INFORMATION SYSTEMS TECHNOLOGY APPLICATIONS OFFICE
DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION (AFSC)
L. G. HANSCOM AFB, MA 01730

Trap doors can be inserted during the distribution phase. If updates are sent via insecure communications - either US Mail or insecure telecommunications, the penetrator can intercept the update and subtly modify it. The penetrator could also generate his own updates and distribute them using forged stationery.

go.dev/s/acmscored

Air Force review of Multics, 1972–1974

ESD-TR-74-193, Vol. II

MULTICS SECURITY EVALUATION:
VULNERABILITY ANALYSIS

Paul A. Karger, 2Lt, USAF
Roger R. Schell, Major, USAF

June 1974

ELECTRONIC SYSTEMS DIVISION

Clearly when a trap door is inserted, it must be well hidden to avoid detection by system maintenance personnel. Trap doors can best be hidden in changes to the binary code of a compiled routine. Such a change is completely invisible on system listings and can be detected only by comparing bit by bit the object code and the compiler listing. However, object code trap doors are vulnerable to recompilations of the module in question.

go.dev/s/acmscored

Air Force review of Multics, 1972–1974

ESD-TR-74-193, Vol. II

It was noted above that while object code trap doors are invisible, they are vulnerable to recompilations. The compiler (or assembler) trap door is inserted to permit object code trap doors to survive even a complete recompilation of the entire system. In Multics, most of the ring 0 supervisor is written in PL/I. A penetrator could insert a trap door in the PL/I compiler to note when it is compiling a ring 0 module. Then the compiler would insert an object code trap door in the ring 0 module without listing the code in the listing. Since the PL/I compiler is itself written in PL/I, the trap door can maintain itself, even when the compiler is recompiled. (38) Compiler trap doors are significantly more complex than the other trap doors described here, because they require a detailed knowledge of the compiler design. However, they are quite practical to implement at a cost of perhaps five times the level shown in Section 3.5. It should be noted that even costs several hundred times larger than those shown here would be considered nominal to a foreign agent.

go.dev/s/acmscored

Ken Thompson's Turing Award Lecture

Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

KEN THOMPSON

INTRODUCTION

I thank the ACM for this award. I can't help but feel that I am receiving this honor for timing and serendip-

programs. I would like to present to you the cutest program I ever wrote. I will do this in three stages and try to bring it together at the end.

go.dev/s/acmscored

Ken Thompson's Actual Code

Extract `nih.a`.

```
% ar xv nih.a
x x.c
x rc
```

Let's read `x.c`*, a C program.*

```
% cat x.c
```

Declare the global variable `nihflg`*, of implied type* `int`.

```
nihflg;
```

Define the function `codenih`*, with implied return type* `int` *and no arguments. The compiler will be modified to call* `codenih` *during preprocessing, for each input line.*

```
codenih()
{
	char *p,*s;
	int i;
```

`cc -p` *prints the preprocessor output instead of invoking the compiler back end. To avoid discovery, do nothing when* `-p` *is used. The implied return type of* `codenih` *is* `int`*, but early C allowed omitting the return value.*

```
	if(pflag)
		return;
```

Skip leading tabs in the line.

```
	p=line;
	while(*p=='\t')
		p++;
```

Look for the line "`name = crypt(pwbuf);`*" from* [`login.c`]*. If not found, jump to* `l1`.

```
	s="namep = crypt(pwbuf);";
	for(i=0;i<21;i++)
		if(s[i]!=p[i])
			goto l1;
```

Do We Learn From History?

1974 Multics report

1983 Thompson lecture

1988 Internet worm

…

Do We Learn From History?

1974 Multics report

1983 Thompson lecture

1988 Internet worm

…

No.

Do We Learn From History?

1974 Multics report

1983 Thompson lecture

1988 Internet worm

…

No.

But why are things not worse?

Hope For the Future

Industry can fix problems when it wants to.

- HTTP to HTTPS
- 2-Factor Auth and Security Keys
- Passkeys?

Maybe we want to fix supply chain security next.
Hopefully we will.

Open-Source Supply Chain Security at Google

Russ Cox (he/him)

ACM SCORED

November 2023

go.dev/s/acmscored