black hat USA 2021

@enjenneer

@SecurityCRob

Securing Open Source Software - End-to-end, At massive scale, Together

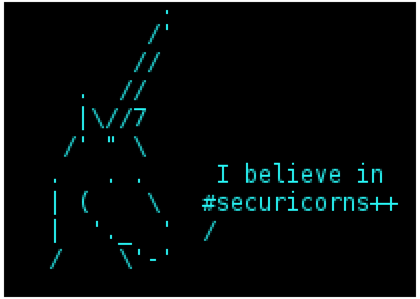
Jennifer Fernick - SVP & Global Head of Research, NCC Group
Christopher Robinson (aka CRob) - Director of Security Communications, Intel

August 5 2021

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

In this talk:

- About us
- Lessons learned from the last 20 years of coordinating OSS vulnerability disclosure
- Why securing the open source ecosystem matters
- Securing OSS at scale has several unique challenges
- Threat modelling the open source ecosystem
- Why the insecurity of the open source ecosystem is accelerating
- Reducing vulns at scale / what actually helps / coordinated approach / OpenSSF
- Q&A



```
I believe in
#securicorns++
```

[us@bh.usa ~]$ ./HelloWorld.sh

Jennifer Fernick

- SVP & Global Head of Research at NCC Group
- Co-founding Governing Board/TAC member for the Open Source Security Foundation
- Prev: Cryptographer (quantum cryptanalysis, post-quantum cryptography, cryptographic standards development, cryptographic architecture)
- Prev: Director of Information Security at a major FI
- Grad school: CS; Engineering (Waterloo); undergrad: Artificial Intelligence (Toronto)

CRob, n, adj, and v

- Pronunciation: U.S. (K-robe)
- Over 25 years of Enterprise-class Architecture, Engineering, Operations, and Security experience
- Ambassador For Intel Product Assurance and Security
- Working Group lead for the OpenSSF Dev Best Practices & Vuln Coordination WGs, FIRST PSIRT TPC WG, and others
- Co-Author FIRST PSIRT Services Framework & others
- Pirate-enthusiast & hat-owner

The thoughts and feels expressed here are personally held or experientially earned, and not necessarily those of our employers

General Members

- accurics
- anchore
- ARDUINO
- okta
- POLYVERSE
- Red Hat
- AURISTOR
- CANONICAL
- CISCO
- RENESAS
- SAMSUNG
- ShiftLeft
- citi
- Codethink
- COMCAST
- snyk
- sonatype
- Spectral
- cybertrust
- debricked
- DevSamurai
- STACKHAWK
- SUSE
- Tencent 腾讯
- Eleven Paths
- facebook
- GitHub
- TIDELIFT
- TRAIL of BITS
- Uber
- GitLab
- Google
- hackerone
- vmware
- WhiteSource
- Hewlett Packard Enterprise
- HUAWEI
- IBM
- iExec
- intel
- JPMorgan Chase & Co.
- Microsoft
- MIRANTIS
- nccgroup

Associate Members

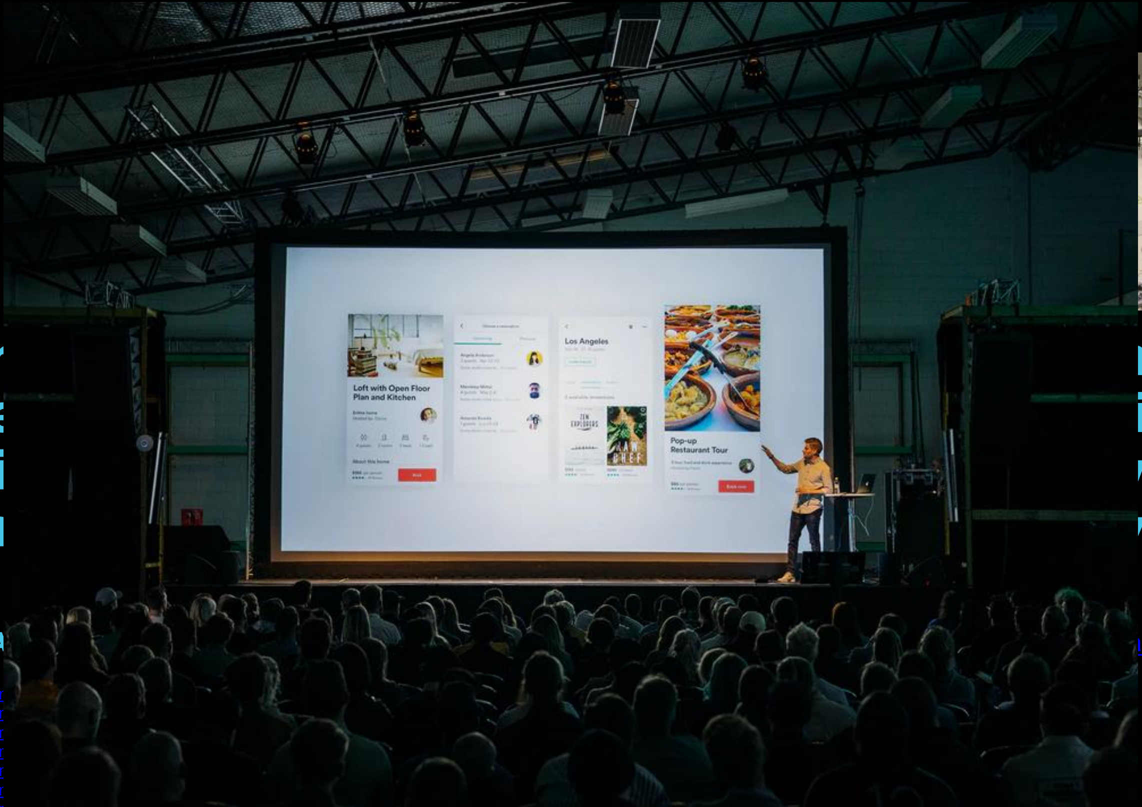
- LISH Laboratory for Innovation Science at Harvard
- OpenUK
- OWASP
- PURDUE UNIVERSITY
- SAFECode

openssf.org

OpenSSF

OPEN SOURCE SECURITY FOUNDATION

Imagine a World....

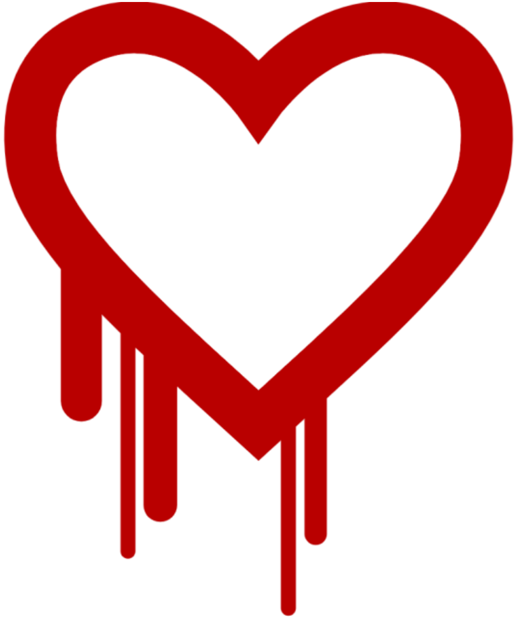


"He em tain the
module inic Tarr
wrote i thing
from m ven use it
anymo

11/26/2018 - A ckdoor

https://images.unsplash.co
https://images.unsplash.co
https://images.unsplash.co
https://images.unsplash.co
https://images.unsplash.co
https://images.unsplash.co
https://images.unsplash.co

CVE-2014-0160



https://heartbleed.com/

Vulnerability in **popular** open source library OpenSSL that could leak sensitive information otherwise thought protected by SSL/TLS encryption

This is a widely-used method to protect communications over TCP-IP-based networks (example - The Internet)

> ”At the time of disclosure, some 17% (around half a million) of the Internet's secure web servers certified by trusted authorities were believed to be vulnerable to the attack, allowing theft of the servers' private keys and users' session cookies and passwords” [7]

Public disclosure - 7 April 2014
Exposure -

- 21 June 2014 - 309,197 public web servers remained vulnerable [8]
- 6 July 2017, the number had dropped to 144,000 [9]
- 11 July 2019, 91,063 devices were vulnerable [10]

At the time, OpenSSL had **TWO** full-time developers to develop, maintain, test, and review 500,000 lines of code [7]

(7) - Heartbleed
(8) - 300k vulnerable to Heartbleed two months later
(9) - Heartbleed’s Heartburn: Why a 5 year Old Vulnerability Continues to Bite
(10) - Heartbleed Report

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Why does securing the open source ecosystem matter?

The Security of OSS =
The Security of EVERYTHING

Problem Overview - Securing the open source ecosystem

- It has been estimated that FOSS constitutes 80-90% of any given piece of modern software [(1)]
- One report found that 84% of these codebases had at least one vulnerability, with the average having 158 per codebase [(2)]
- Other reports discover that average applications contain 118 libraries with roughly ⅓ being active; The average library age was 2.6 years old [(3)]
- Over a 10 year period the volume of vulns has increased over 4 times [as measured with CVE] [(4)]
- Most OSS vulns are discovered in indirect dependencies [(5)]
- A typical vuln can go undetected for 218 weeks, and typically takes 4 weeks to get resolved once the project is alerted to it [(6)]

(1) - State of the Software Supply Chain
(2) - 2021 Open Source Security and Risk Analysis Report
(3) - 2021 State of Open Source Security Report
(4) - 2020 Red Hat Risk Report
(5) - 2020 State of Open Source Security Report
(6) - 2020 State of the Octoverse

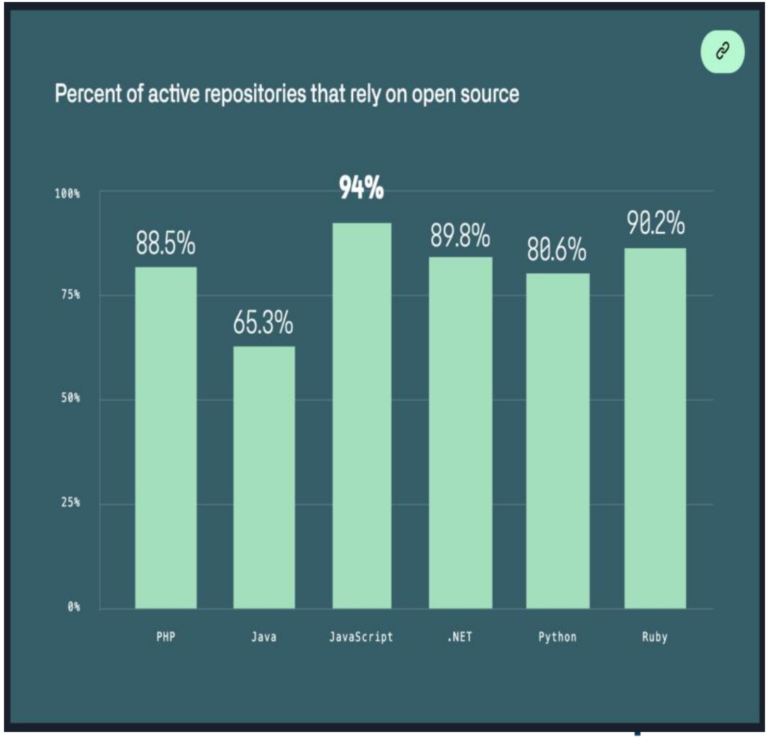


Image - %Rely on OSS

black hat USA 2021

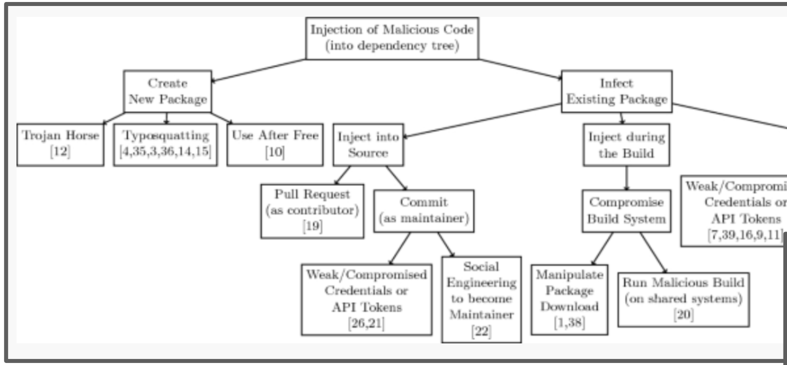
Challenges of Securing Open Source Software at Scale

What makes OSS a unique target for adversaries?

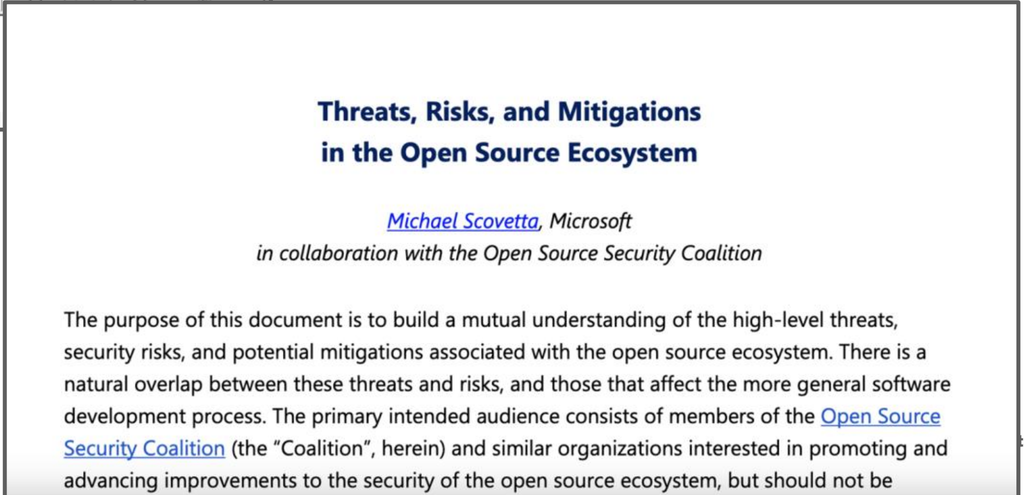
Many of the best things about open source development invite unique security challenges:

- Deobfuscated and **public-facing source code** lowers attacker barrier to entry
- Distributed **community-driven development** with contributions from unknown third-parties
- **Tragedy of the commons** regarding security analysis
- **Lack of consistently-deployed security standards, reviews and tooling**
- (Often) **decreased capacity for vulnerability remediation**
- Lack of resources for monitoring & typical **underpreparedness for incident response**
- **Different economic incentives & feedback loops** than: enterprise devs; threat actors
- In spite of this: many **high-value targets**, foundational to enterprises and the internet itself

OSS security is about more than just vulnerabilities in source code


Injection of Malicious Code (into dependency tree)
Create New Package
Infect Existing Package
Trojan Horse [12]
Typosquatting [4,35,3,36,14,15]
Use After Free [10]
Inject into Source
Inject during the Build
Inject into Repository System
Pull Request (as contributor) [19]
Commit (as maintainer)
Compromise Build System
Weak/Compromised Credentials or API Tokens [7,39,16,9,11]
Exploit Vulnerabilities [1,24,25]
Deploy in Alternative Repository/Mirror
Weak/Compromised Credentials or API Tokens [26,21]
Social Engineering to become Maintainer [22]
Manipulate Package Download [1,38]
Run Malicious Build (on shared systems) [20]

Source: Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks (2020)



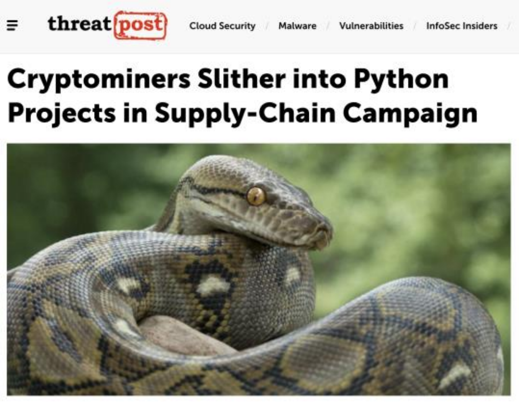
Threats, Risks, and Mitigations in the Open Source Ecosystem

Michael Scovetta, Microsoft
in collaboration with the Open Source Security Coalition

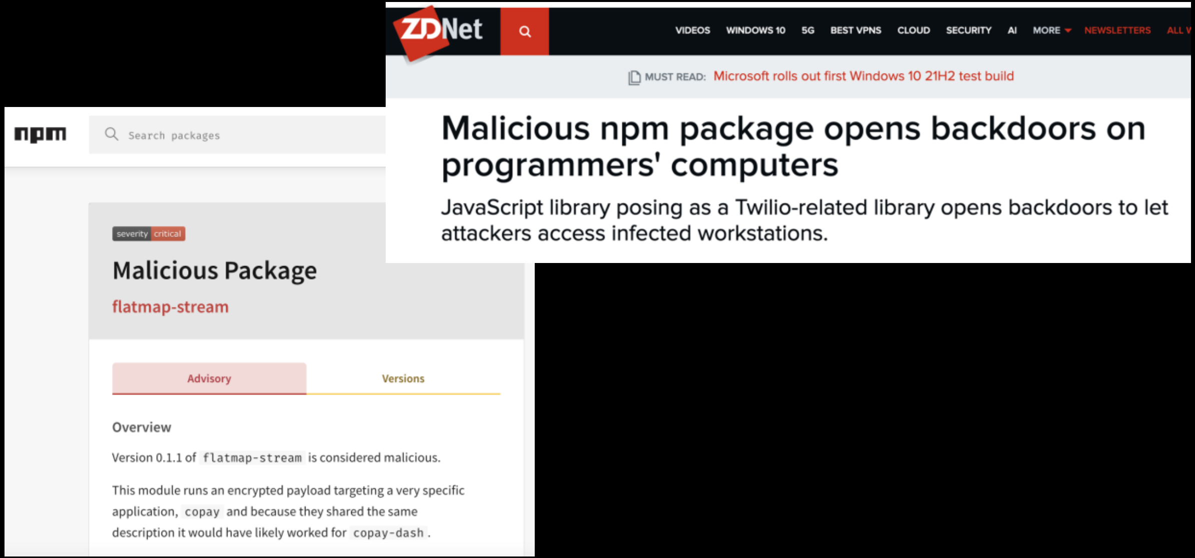
The purpose of this document is to build a mutual understanding of the high-level threats, security risks, and potential mitigations associated with the open source ecosystem. There is a natural overlap between these threats and risks, and those that affect the more general software development process. The primary intended audience consists of members of the Open Source Security Coalition (the "Coalition", herein) and similar organizations interested in promoting and advancing improvements to the security of the open source ecosystem, but should not be

ts

Vulnerabilities in the Open Source Ecosystem



Cryptominers Slither into Python Projects in Supply-Chain Campaign



Malicious Package

flatmap-stream

Overview

Version 0.1.1 of `flatmap-stream` is considered malicious.

This module runs an encrypted payload targeting a very specific application, `copay` and because they shared the same description it would have likely worked for `copay-dash` .

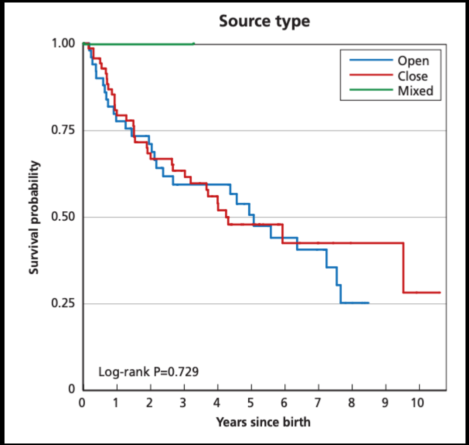
MUST READ: Microsoft rolls out first Windows 10 21H2 test build

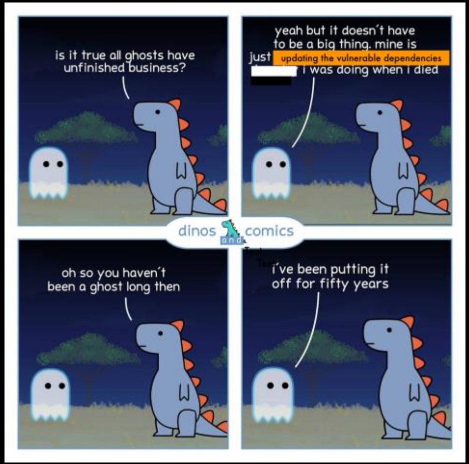
Malicious npm package opens backdoors on programmers' computers

JavaScript library posing as a Twilio-related library opens backdoors to let attackers access infected workstations.

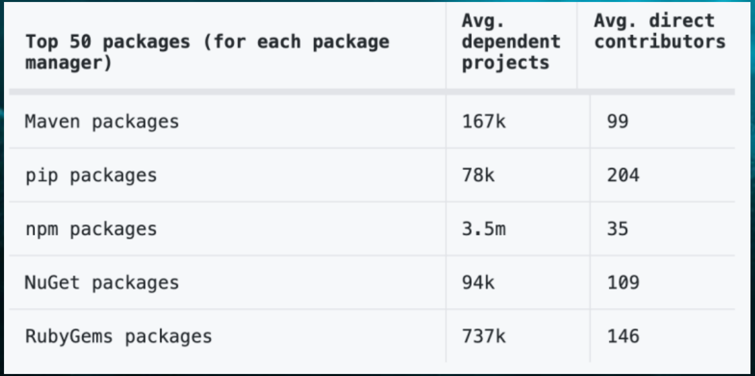
Open source security keeps getting worse & it’s no one’s fault, but it’s everyone’s problem

- **Years to detect:**
  A typical vulnerability on GitHub goes undetected for over 4 years (>2.5y for critical)
- **Mere days to exploit:**
  Days between vulnerability disclosure and exploit creation has gone from 45 days to 3
- **Devs are not getting better at secure coding:**
  “A line of code written in 2020 is just as likely to introduce a security vulnerability as one written in 2016” - GitHub
- **Applications are increasing in complexity**
  & transitive dependency risk seems to be growing




is it true all ghosts have unfinished business?
yeah but it doesn't have to be a big thing. mine is just updating the vulnerable dependencies [redacted] i was doing when i died
dinos and comics
oh so you haven't been a ghost long then
i've been putting it off for fifty years

Source: RAND Corporation, “Zero Days, Thousands of Nights”



Top 50 packages (for each package manager)	Avg. dependent projects	Avg. direct contributors
Maven packages	167k	99
pip packages	78k	204
npm packages	3.5m	35
NuGet packages	94k	109
RubyGems packages	737k	146

Source: Github State of the Octoverse 2019

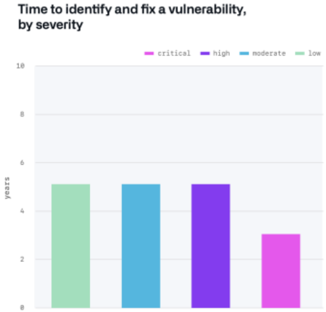
Open source projects have an average of **180 package dependencies**

The top 50 OSS projects with the most downstream dependencies had an average of **3.6 million projects dependent upon them**

Vulns in OSS have been central to major breaches and some of these vulns were **not found until decades** after their creation

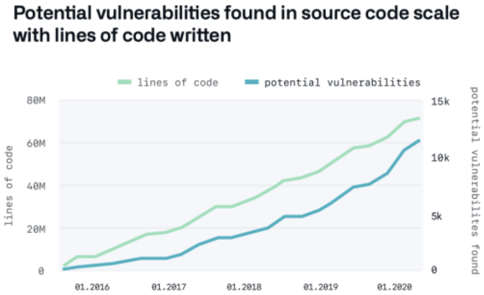
The number of vulnerabilities “in the wild” outpaces the speed at which the security community can patch or even identify them.

And each day, the world contains more lines of source code than it ever has before



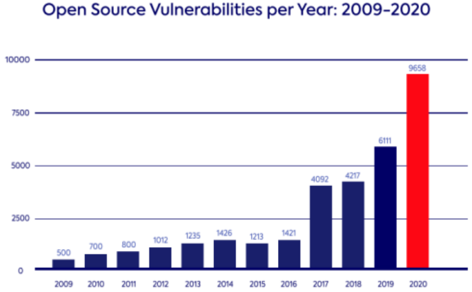
Time to identify and fix a vulnerability, by severity

Source: Github State of the Octoverse 2020



Potential vulnerabilities found in source code scale with lines of code written

Source: Github State of the Octoverse 2020



Open Source Vulnerabilities per Year: 2009-2020

Source: Whitesource State of Open Source Security 2021

Every year, more lines of OSS are written than ever before, but vuln detection lags years behind

+

Vulnerabilities seem to scale with lines of code - but other metrics besides LOC show similar patterns

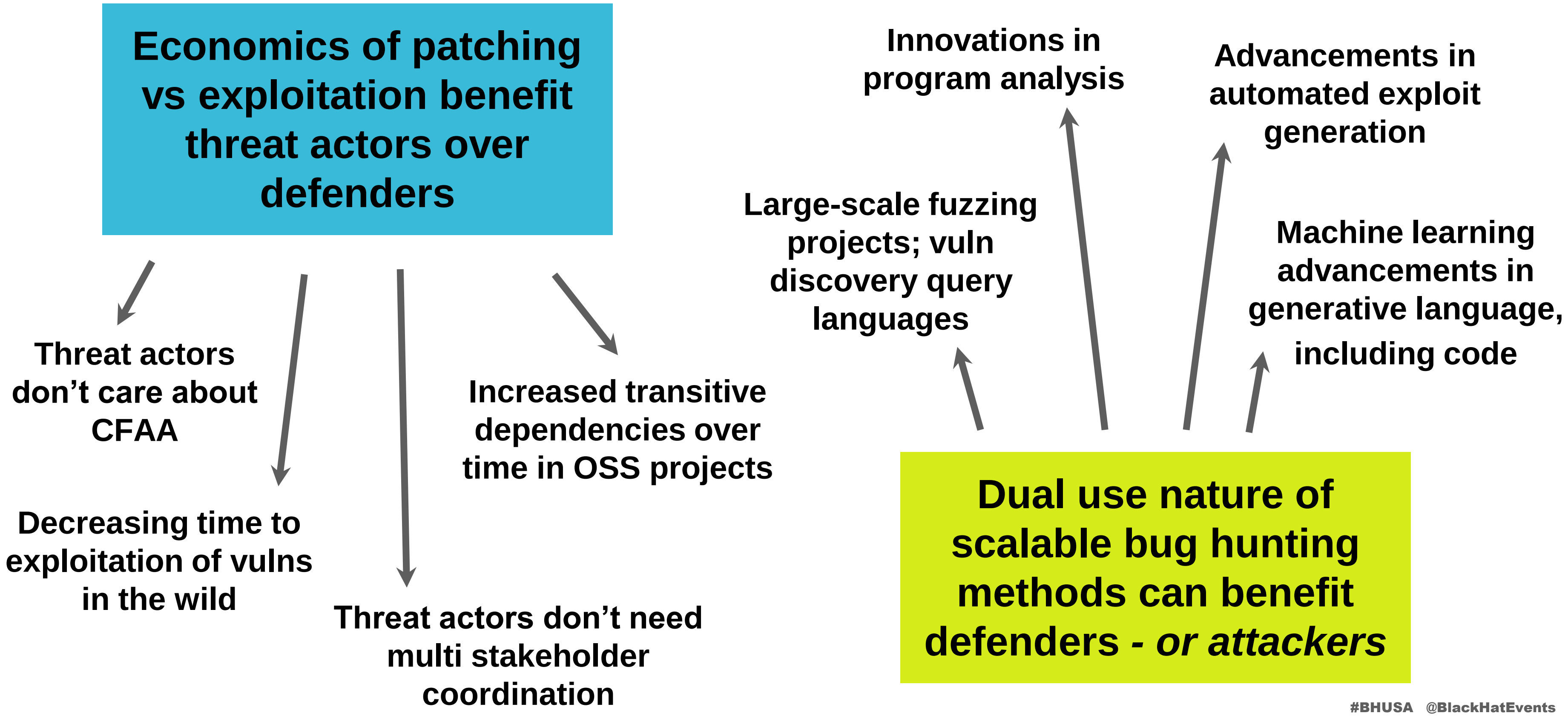
=

The number of reported vulnerabilities in open source codebases is growing each year

The creation of potentially exploitable vulnerabilities increasingly outpaces the rate at which we can search for and remediate them, and this problem is one that only gets worse with time.

Security as it is practiced now does not scale. And we have reasons to believe this will only get worse for defenders.

Uncomfortable truth: Scalable vuln-finding helps adversaries, too



Economics of patching vs exploitation benefit threat actors over defenders

Threat actors don’t care about CFAA

Decreasing time to exploitation of vulns in the wild

Threat actors don’t need multi stakeholder coordination

Increased transitive dependencies over time in OSS projects

Dual use nature of scalable bug hunting methods can benefit defenders - *or attackers*

Large-scale fuzzing projects; vuln discovery query languages

Innovations in program analysis

Advancements in automated exploit generation

Machine learning advancements in generative language, including code

So how do we reduce vulnerabilities at scale?

How do we reduce vulnerabilities at scale?

PREVENT

- Prevent classes of bugs from being possible at all
- Threat model to understand systemic architectural security risks & design with security in mind
- Concentrate resources on securing the most critical libraries, components, and projects

FIND

- Integrate security tooling into your build pipeline (Static analysis, Fuzzing, etc)
- Perform enhanced testing (manual code review, third-party security audit, formal verification) for priority codebases

DETECT

- Improve coordinated vulnerability disclosure
- Software Bill of Materials, security advisories, and CVE improvements

What ACTUALLY helps secure OSS

“Groundbreaking” idea

“If you use software from a project, maybe you could contribute back to that project?”

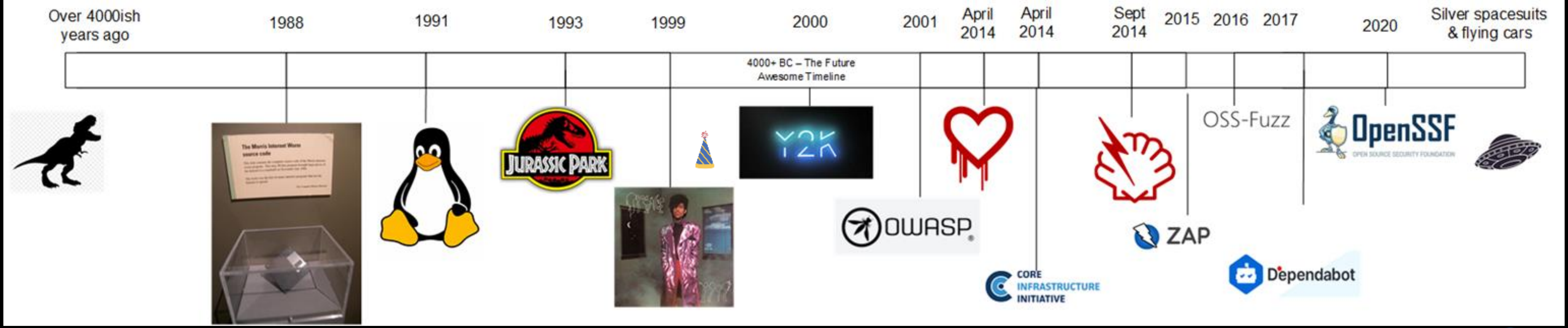
- A very wise person

Image - [Blown Cat]

What is needed to make open source more secure?

- **Threat model** to understand the many places & times at which a project can be compromised.
- Data-driven **identification of the world's most critical open source projects**
- Interventions to **prevent vulnerabilities in the first place**, introduced at various parts of SDLC
- **Preventing inherited security debt** through tools that can help developers obtain and users assess the security of a project (such as the CII Best Practices badge)
- Continued research and open source **tool development** for scalable bug-hunting & remediation
- Investments in **technical security reviews** of critical open source projects,
- **Coordinated patching and incident response support** to respond to high-impact vulnerabilities in OSS
- Better **vulnerability disclosure** processes, response, and workflows.
- **Coordinated, impact-prioritized funding** for security improvements, audits, and research

Timeline: Historical coordination toward securing OSS


Over 4000ish years ago
1988
1991
1993
1999
2000
2001
April 2014
April 2014
Sept 2014
2015
2016
2017
2020
Silver spacesuits & flying cars
4000+ BC – The Future Awesome Timeline
The Morris Internet Worm source code
JURASSIC PARK
Y2K
OWASP
CORE INFRASTRUCTURE INITIATIVE
ZAP
OSS-Fuzz
Dependabot
OpenSSF
OPEN SOURCE SECURITY FOUNDATION

COORDINATED APPROACH - WHY AND HOW

Image - Eek!

OPENSSF* Reference Architecture

(* The Open Source Security Foundation if you're *nasty*)

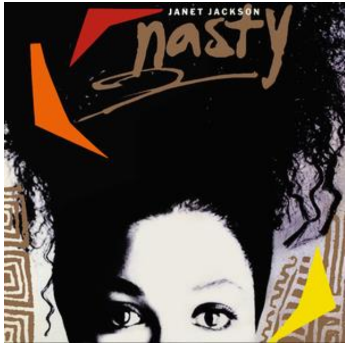
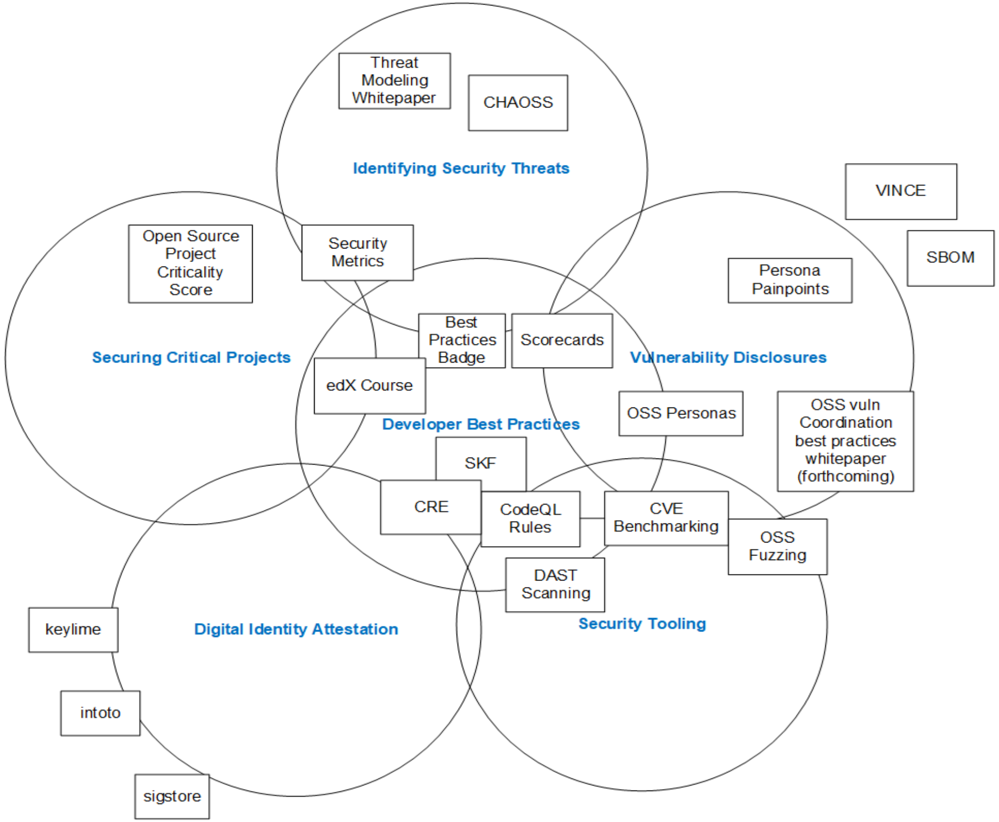
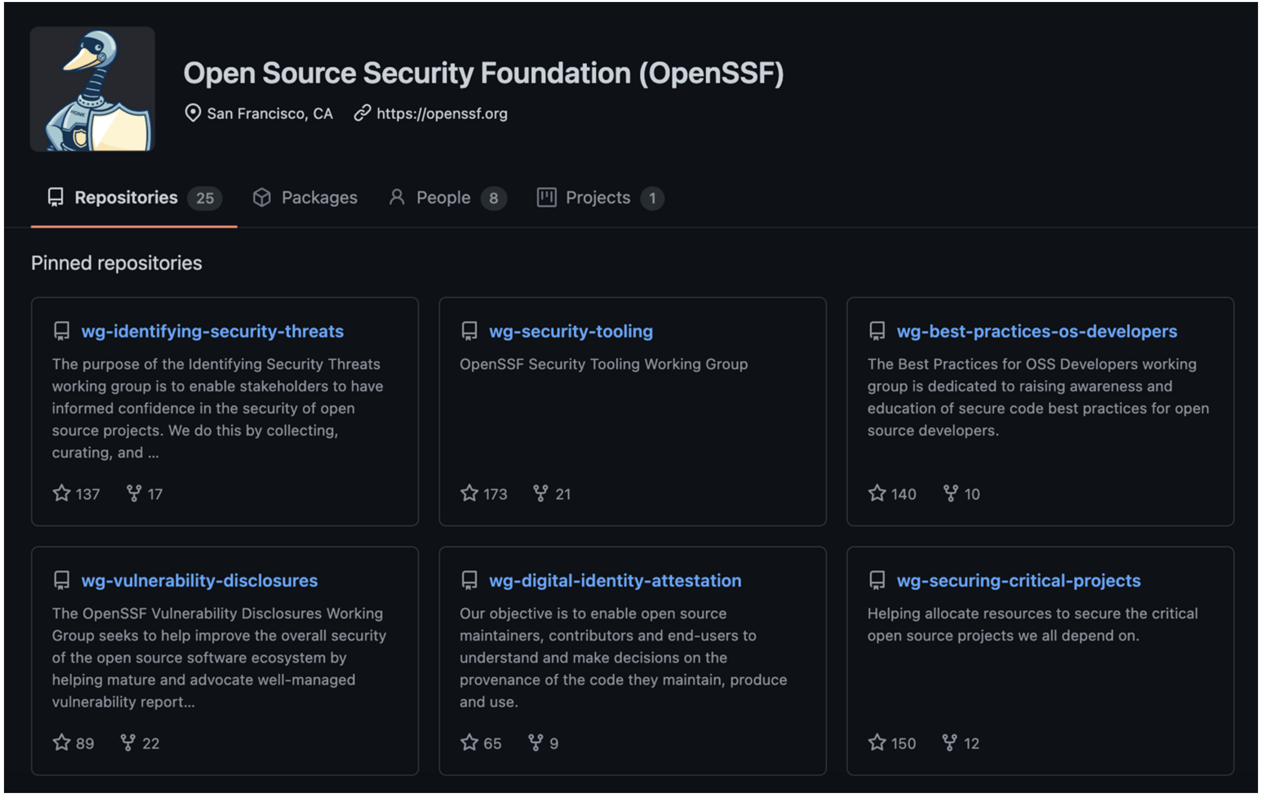


Image - Miss Jackson


Threat Modeling Whitepaper
CHAOSS
Identifying Security Threats
VINCE
SBOM
Open Source Project Criticality Score
Security Metrics
Persona Painpoints
Best Practices Badge
Scorecards
Securing Critical Projects
Vulnerability Disclosures
edX Course
OSS Personas
OSS vuln Coordination best practices whitepaper (forthcoming)
Developer Best Practices
SKF
CRE
CodeQL Rules
CVE Benchmarking
OSS Fuzzing
DAST Scanning
Security Tooling
keylime
Digital Identity Attestation
intoto
sigstore

OpenSSF OPEN SOURCE SECURITY FOUNDATION



Open Source Security Foundation (OpenSSF)

San Francisco, CA https://openssf.org

Pinned repositories

wg-identifying-security-threats
The purpose of the Identifying Security Threats working group is to enable stakeholders to have informed confidence in the security of open source projects. We do this by collecting, curating, and ...
☆ 137 Forks 17

wg-security-tooling
OpenSSF Security Tooling Working Group
☆ 173 Forks 21

wg-best-practices-os-developers
The Best Practices for OSS Developers working group is dedicated to raising awareness and education of secure code best practices for open source developers.
☆ 140 Forks 10

wg-vulnerability-disclosures
The OpenSSF Vulnerability Disclosures Working Group seeks to help improve the overall security of the open source software ecosystem by helping mature and advocate well-managed vulnerability report...
☆ 89 Forks 22

wg-digital-identity-attestation
Our objective is to enable open source maintainers, contributors and end-users to understand and make decisions on the provenance of the code they maintain, produce and use.
☆ 65 Forks 9

wg-securing-critical-projects
Helping allocate resources to secure the critical open source projects we all depend on.
☆ 150 Forks 12

openssf.org/getinvolved
github.com/ossf

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

THE LINUX FOUNDATION

Thank you!

openssf.org/getinvolved
github.com/ossf

@enjenneer
@SecurityCRob

OpenSSF

OPEN SOURCE SECURITY FOUNDATION