Securing Open Source Software

Jennifer Fernick, SVP & Global Head of Research (NCC Group)
David A. Wheeler, Director of Open Source Supply Chain Security (Linux Foundation)
Supply chain security depends on the security of open source software.
Software supply chain attacks increasing

Source: Sonatype 2021 State of the Software Supply Chain
Software is under attack

Malicious npm package opens backdoors on programmers' computers
JavaScript library posing as a Twilio-related library opens backdoors to let attackers access infected workstations.

Malicious npm package opens backdoors on programmers' computers
JavaScript library posing as a Twilio-related library opens backdoors to let attackers access infected workstations.

Reviewed 174 OSS SC attacks
2015-2019, 61% malicious packages use typosquatting

SolarWinds' Orion attacked via a subverted build environment
Vulns hide in codebases for years, on average

- **Years to detect:**
  A typical vulnerability on GitHub goes **undetected for over 4 years** (>2.5 years for critical vulns)

- **Mere days to exploit:**
  Days between vulnerability disclosure and exploit creation has **gone from 45 days down to only 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

---

**The full lifecycle of a vulnerability**

- undetected vulnerability
- alerting users to upgrade
- fixing known vulnerability
- users upgrading to fix version

Open source projects have an average of **180 package dependencies**, but some have >1000

Source: GitHub State of the Octoverse 2019

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

Source: GitHub State of the Octoverse 2019

The **average number of open source components** in an application over the last three years has grown from 298 to **528**

Source: Synopsys Open Source Security and Risk Analysis Report 2021
Attacks on OSS are getting worse & security needs our intervention
Every year, more lines of software are written than ever before, but vulnerability 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 software codebases is growing each year (OSS shown here; true in general).

Source: GitHub State of the Octoverse 2020

Source: GitHub State of the Octoverse 2020

Vulnerability finding tools are dual-use

Developers’ primary goal is to **build core functionality** - not defend against hacking. But attackers’ **entire** goal is to **attack**.

**Economics of patching vs exploitation benefit threat actors over defenders**
- Threat actors don’t care about CFAA
- Decreasing time to exploitation of vuls in the wild

**Increased transitive dependencies over time in OSS projects**

**Innovations in program analysis**
- Large-scale fuzzing projects; vuln discovery query languages

**Advancements in automated exploit generation**
- Machine learning advancements in generative language, including code

**Dual use nature of scalable bug hunting methods can benefit defenders - or attackers**
- Threat actors don’t need multi stakeholder coordination
1. More code developed each year
2. No fewer vulns per unit of code developed, compared to 5 years ago
3. Exponential growth in vulns reported each year
4. We have reason to believe the reported vulns are just the tip of the iceberg
5. It’s not just the # of vulns that are increasing - so are the # of active attacks
6. Exploitation in the wild is happening faster than ever before

Security as practiced today does not scale
So how do we reduce vulnerabilities, at scale?
IDENTIFYING CRITICAL OPEN SOURCE PROJECTS FOR BESPOKE SECURITY IMPROVEMENTS

DEVELOPER SECURITY EDUCATION

Analyze  Design  Implement  Test  Deploy  Maintain

SECURITY BY DESIGN  VULNERABILITY DETECTION TOOLING

THIRD-PARTY SECURITY AUDITS

THREAT MODELING  CERT/INCIDENT RESPONSE SUPPORT

SECURING DEVELOPER WORKSTATIONS & DEPLOYMENT PIPELINES

CODE & ARCHITECTURE REVIEWS  VULNERABILITY DISCLOSURE PROGRAM
What would happen if the technologies built and maintained by the people in this room were to come under adversarial control?
We already know how to make open source more secure, but it requires all of us, together.
Society is noticing & wants action

“Attention [is shifting] to suppliers” - ENISA (EU)

Sources: ENISA Threat Landscape for Supply Chain Attacks; (US) Executive Order on Improving the Nation’s Cybersecurity
LF has many projects working to secure OSS

- sigstore
- in-toto
- OpenSSF
- CHA OSS
- SPDX
- OPENCHAIN
- Act
- OpenChain Reference Tooling

... and many more!
Open Source Security Foundation (OpenSSF)

• Purpose: “Collaborating to secure the open source ecosystem”
• Established August 3, 2020: https://openssf.org/

Open Source Security Foundation Raises $10 Million in New Commitments to Secure Software Supply Chains
2021-10-13
<table>
<thead>
<tr>
<th>Repository Name</th>
<th>Description</th>
<th>Stars</th>
<th>Watchers</th>
</tr>
</thead>
<tbody>
<tr>
<td>wg-identifying-security-threats</td>
<td>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 ...</td>
<td>137</td>
<td>17</td>
</tr>
<tr>
<td>wg-security-tooling</td>
<td>OpenSSF Security Tooling Working Group</td>
<td>173</td>
<td>21</td>
</tr>
<tr>
<td>wg-vulnerability-disclosures</td>
<td>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 reports...</td>
<td>89</td>
<td>22</td>
</tr>
<tr>
<td>wg-digital-identity-attestation</td>
<td>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.</td>
<td>65</td>
<td>9</td>
</tr>
<tr>
<td>wg-best-practices-os-developers</td>
<td>The Best Practices for OSS Developers working group is dedicated to raising awareness and education of secure code best practices for open source developers.</td>
<td>140</td>
<td>10</td>
</tr>
<tr>
<td>wg-securign-critical-projects</td>
<td>Helping allocate resources to secure the critical open source projects we all depend on.</td>
<td>150</td>
<td>12</td>
</tr>
</tbody>
</table>
How OpenSSF Projects Work Together

A. Secure Software Development Fundamentals: edX courses (education)
B. Security Knowledge Framework (SKF): Hands-on course (education), with OWASP
C. CII Best Practices Badge
D. Scorecards
E. Great MFA distribution project
F. Common Requirements Enumeration (CRE)
G. Guide to coordinated vulnerability disclosure for OSS projects; Vulnerability Disclosures Whitepaper
H. osv-schema
I. security-reviews
J. Project-Security-Metrics: Dashboard
K. ossf-cve-benchmark: ??? Location
L. Web Application Definition (tools WG): ??? Location
M. Supply-chain Levels for Software Artifacts (SLSA)
N. allstar
O. package-feeds / package-analysis
P. criticality_score
Q. Harvard study
R. Project Alpha-Omega
**Project Alpha-Omega**

**Alpha**: Deep Security Work, Automated Tooling, Best Practices...

**Omega**: Automated Tooling & Triage
Get Involved!

We all depend on OSS, we need to work together.
OpenSSF: openssf.org/getinvolved
THE LINUX FOUNDATION MEMBER SUMMIT
Software Supply Chain Integrity Map

1. Use, Modify, Build & Verify
   - Select & Acquire
     - Typosquatting, Dependency confusion
     - Signature verification (e.g., sigstore)
   - Local Environment
     - Hardened local/separate environment
     - Insecure install, Exploited
       - Least privilege, memory safety, monitoring
   - Local environment subversion
   - Misunderstanding
     - Education & Training
   - Developers

2. Source & Data Repository ("Forge")
   - Repository ("Forge")
   - Unauthorized change, malicious commit, Source repo subversion

3. Build & Verification
   - Build system subversion
   - Vulnerability ID tools, reproducible builds, CII BP Badge, SBOMs (e.g., SPDX)
   - Improved assessment processes, package monitoring, integrity attestation (e.g., in-toto)
   - Package repo hardening

4. Approve & Release
   - Inadequate/wrong assessment
   - Improved assessment processes (e.g. OpenChain)
   - Package Repository / Distribution Platform

5. Package Assessment
   - Inadequate/wrong assessment
   - Authorization controls, Data integrity, developer activity monitoring, forge hardening

6. Package Assessment
   - Signature verification (e.g., sigstore)
   - Improved assessment processes, package monitoring, integrity attestation (e.g., in-toto)

7. Deploy & Operations
   - Misunderstanding
     - Education & Training
   - Insecure install, Exploited
     - Least privilege, memory safety, monitoring
   - External Receivers

Sample attacks
Sample countermeasures (where applied)

Replication of threats identified
Education & Training

External Services