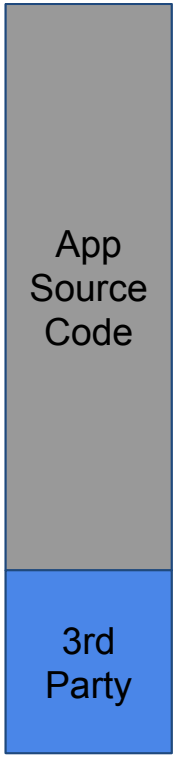
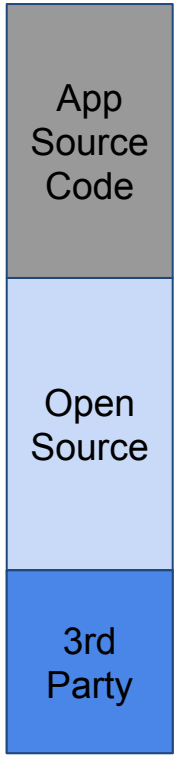
Software Dependency Management

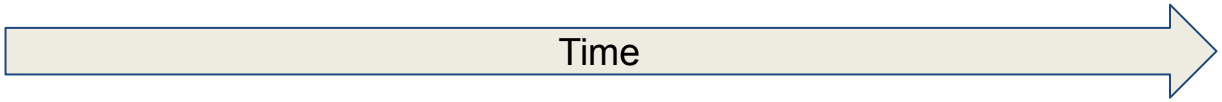
Dependency Management Drivers



- Access
  - Internal Network
  - Proprietary Communication
- Hosting
  - Private Data Center
  - Co-location
- S/W Development
  - Internal development team
  - Manual dependency installs
  - Limited open source footprint



- Access
  - Public Internet
  - HTTP Protocol
- Hosting
  - SaaS
  - Cloud Provider (AWS, Azure, Heroku
- S/W Development
  - Outsourced
  - Contractors
  - Large open Source footprint



State of the Industry

AUBURN

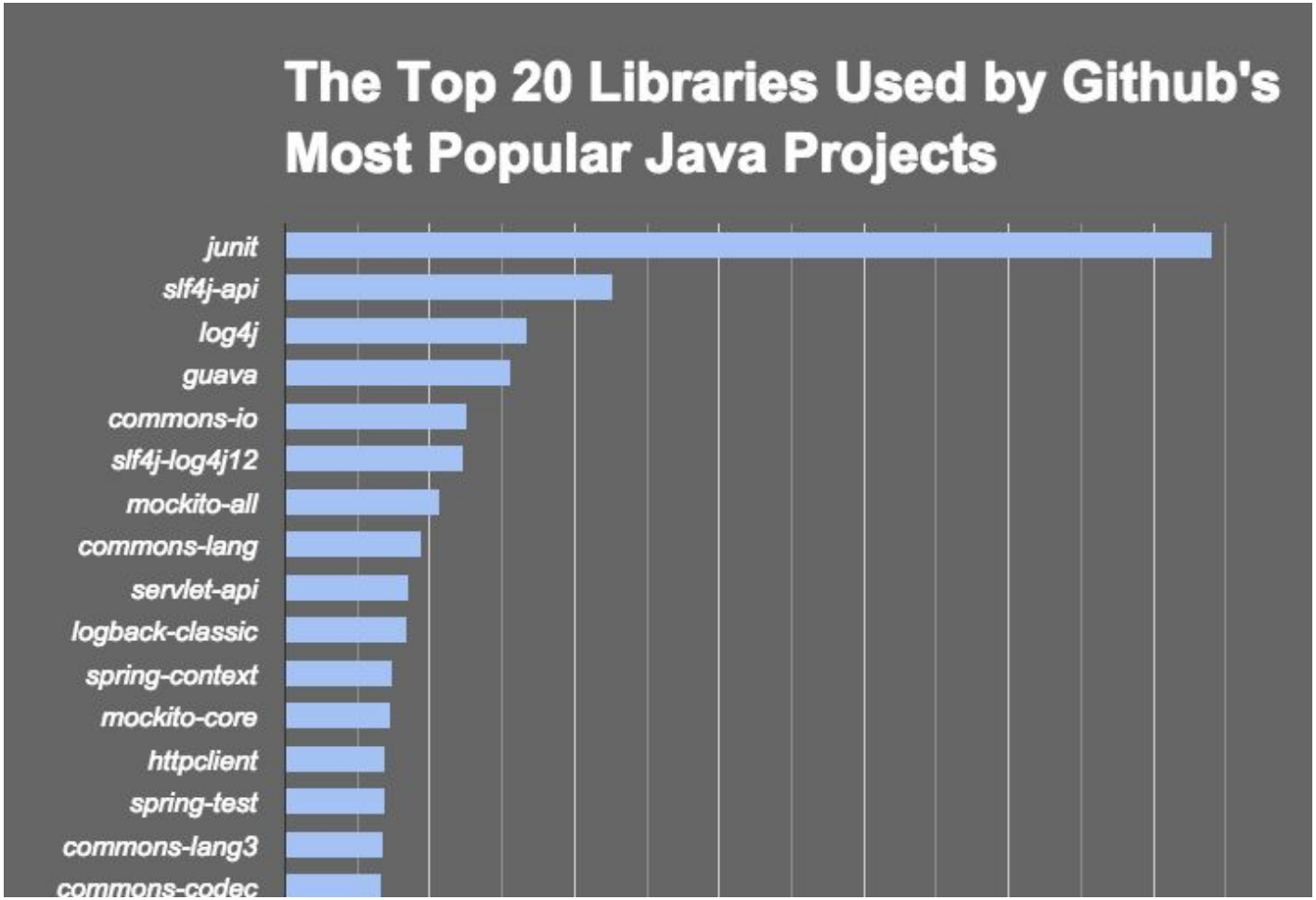
- 96% of the scanned applications contain open source components
- >60% of application source code is open source
- >250 components in large scale applications

Why Use Open Source & 3rd Party

AUBURN

- Most software is primarily composed of open source or 3rd party software
- Driven by several factors:
  - **Common application functionality**
    - Authentication - OAuth, SAML/SSO, Username/Password
    - HTTP Handling - URL Mapping, Parameter Handling
    - Database Access - Object to relational mapping, data retrieval
  - Libraries are **high quality** because of their pervasive use in the industry
    - Apache - Commons, Tomcat, HTTPClient, SLF4J, Maven
    - Spring - Boot, Security, Data, Batch, Web Flow
  - **Increase Productivity, Time to Market, Lower Cost**
    - Allow companies to focus on domain parts of application and not reinvent “boiler plate code”
  - Developing your own solutions and making sure they are secure is difficult
  - Specialized capability requiring domain knowledge and experience

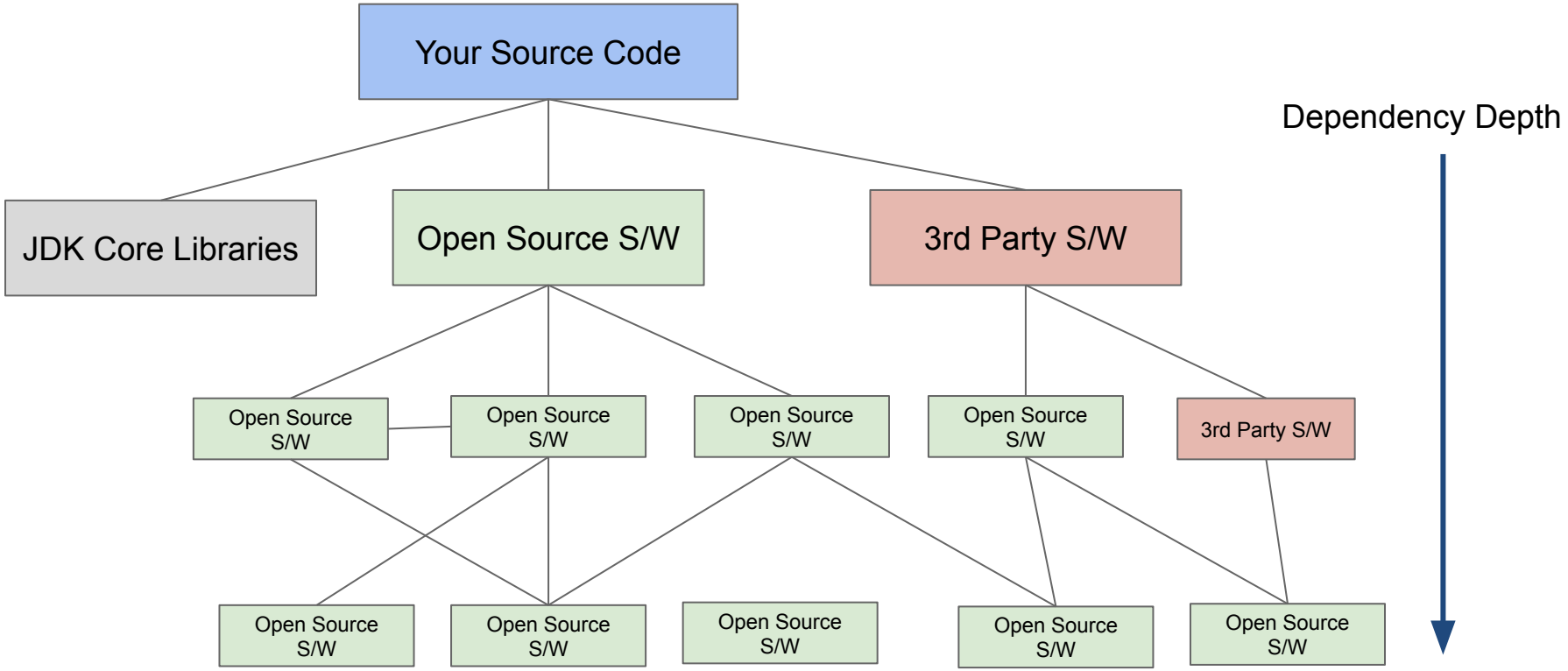
Java Open Source Popularity



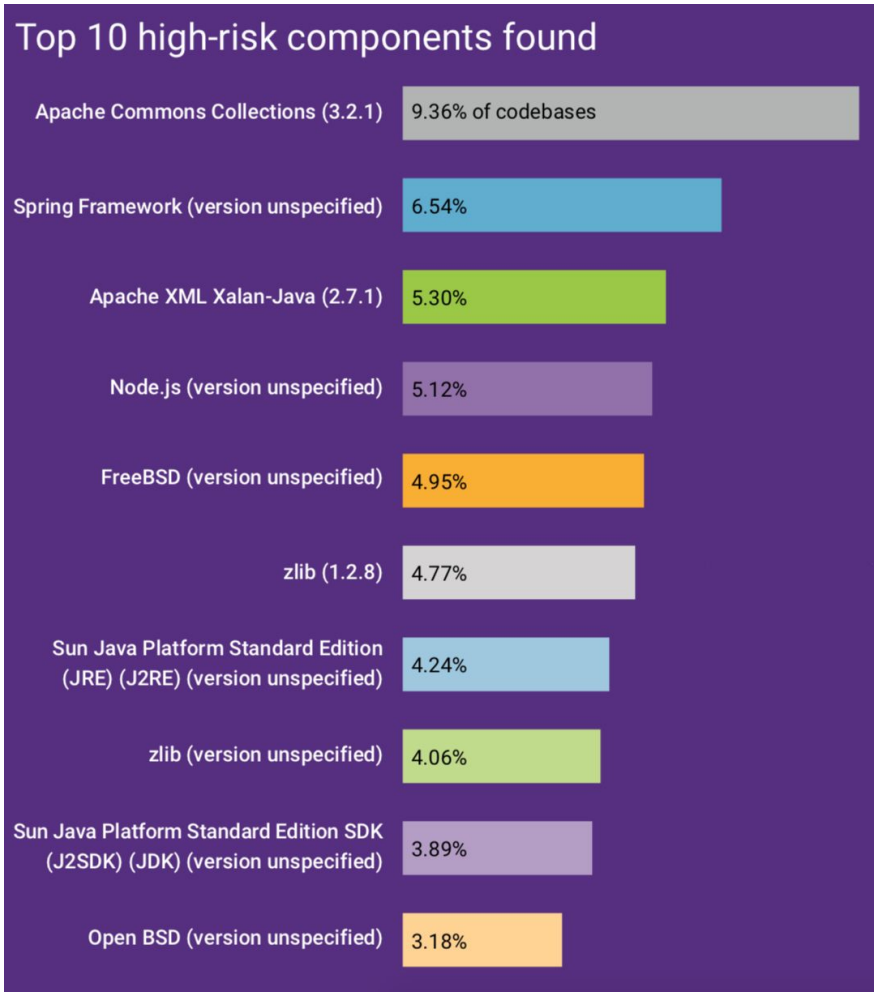
Software Composition

AUBURN

- Most application software utilizes larger amounts of open source and 3rd party software
- Not uncommon for application source code to be <25% of over all source code.



Vulnerabilities in Code Bases

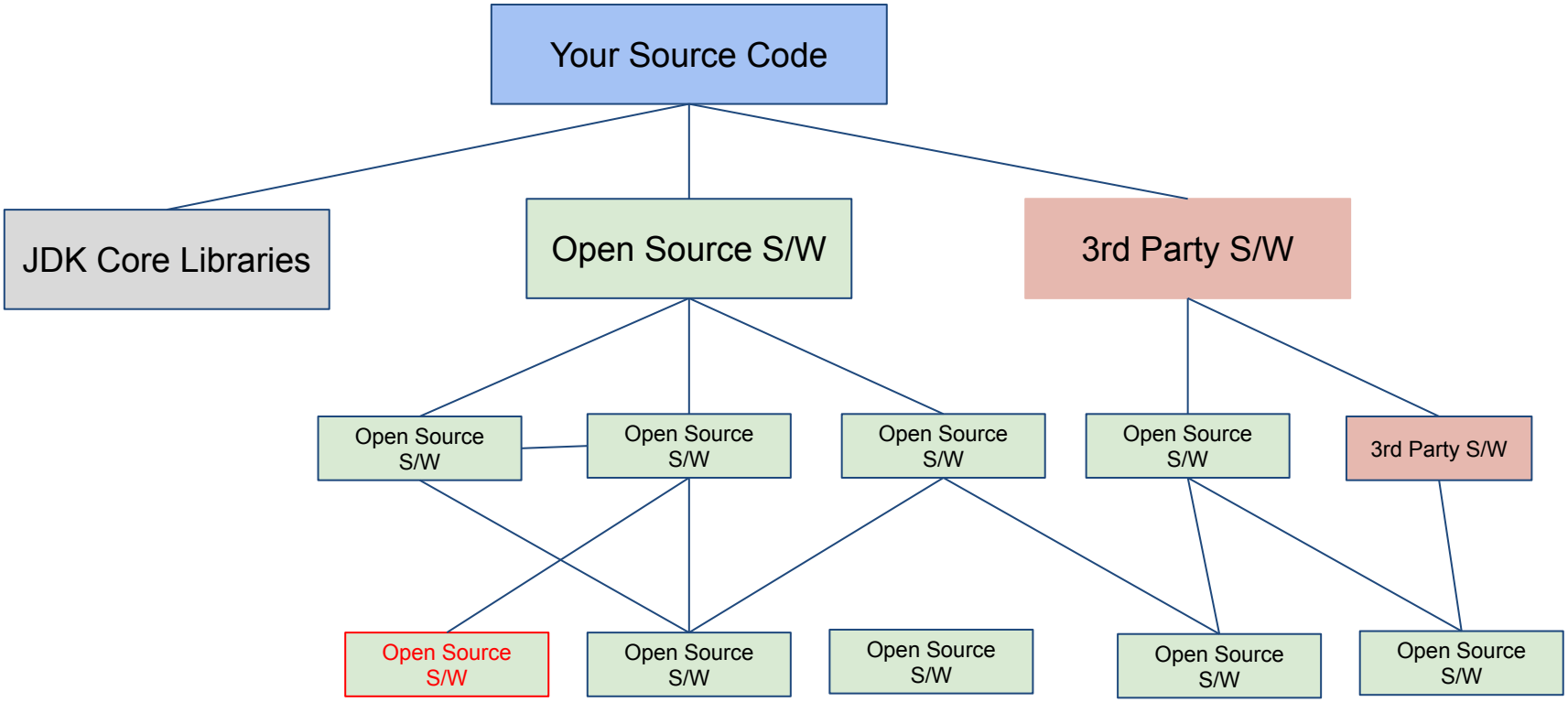


- Analysts estimate >70% of existing code bases using open source software contain vulnerabilities
  - Use of old s/w versions; no process to update
  - Poor visibility into dependencies
  - Lack of dependency mgmt

Software Dependency Challenges

AUBURN

- How do you validate your developers are writing secure source code?
- How do you control the introduction of open source and 3rd party software?
- How are you notified when a vulnerability is uncovered in a software dependency?



What is a software dependency?

AUBURN

- A dependency is any code that your project **depends on**
  - It wont run without it - Java “NoClassDefFoundError”
- Code library or package that is reused in a new piece of software.
  - Open Source libraries
    - Java Apache
    - Java Spring
  - 3rd Party Libraries
    - Database jdbc drivers
    - UI libraries
  - Internal Libraries
    - Packages developed within organizations to share common codebases
    - Examples are: system integration, authentication, database layer
  - For example, a machine learning project might call a Python library to build models. The benefit of software dependencies is that they allow developers to more quickly deliver software by building on previous work.

Dependency Management Challenges

AUBURN

- External dependencies (open source, 3rd party) puts your build and application at risk.
  - Brought in from public hosted repositories
  - Not just runtime, but build and test time components as well
- Awareness of what external libraries are brought it to a build process is important to manage known vulnerabilities and risks
- Risk Mitigation Requirements
  - Dependency verification to avoid integrating compromised dependencies in your project.
  - Alerting when a vulnerability is discovered in a dependency
    - “Zero Day” vulnerability - vulnerability in a system or device that has been disclosed but is not yet patched. Critical/major vulnerabilities require immediate mitigation until a patch is available.
  - Upgrading to latest versions when they are released

Dependency Terminology

AUBURN

- **“Artifact”** - tangible by-products produced during the development of software
  - Project source code
  - Dependencies
  - Binaries
  - Resources

Dependency Management

AUBURN

- Each programming language is different in the way it manages code from third parties.
- "Package Managers" provide a central package-management store where developers can upload their code and retrieve code from others
  - **Java** - Maven or Gradle tool, Maven Central repository
  - **Node.JS** - npm tool, npm public registry repository
  - **Ruby** - RubyGems tool, RubyGems.org repository
  - **Perl** - ppm, CPAN repository
  - **Go** imports and retrieves dependencies directly from their source-code repositories (Github) because it is a natively compiled vs. interpreted language on the host operating system
- Dependencies are typically store locally or in an internal repository to protect against:
  - **Availability** - the remote repository could be offline or not accessible from an internal network
  - **Integrity -** ensure the dependency has not been maliciously change remotely

Best Practices

AUBURN

- **Versioning Pinning**
  - Using a specific dependency version for your application
  - Tests validate version and locking changes ensures reproducibility
  - Risk is it does not allow automatic updates for new releases for security or bug fixes
    - Mitigated by using artifact management tools that provide automatic scanning and checking against newly published versions and security notices
- **Signature and hash verification**
  - Artifact authenticity verified with security controls
  - Hash verification - Compare the hash of a given artifact with a known hash provided by the artifact repository
  - Prevents against man-in-the-middle attack or a compromise of the public artifact repository
    - Must trust hash value is legitimate
  - Signature verification in addition to hashing adds another layer to ensure hash value comes from trusted source

Best Practices

AUBURN

- **Mixing private and public dependencies**
  - Utilize a private repository to manage public and private artifacts
  - Publish internal artifacts as well as pull public artifacts into a centralized artifact repository.
  - Security, governance, and traceability can be implemented on top of management tool
- **Unused Dependency Removal**
  - Similar to unused source code, dependencies should be removed once they are no longer needed.
  - Eliminates the risk of leaving an existing vulnerability that may be still located in an application or accidentally used.
- **Vulnerability scanning**
  - Automatically scan and alert when your dependencies contain vulnerabilities
  - Containers, which contain their own artifacts, can also be scanned.

AUBURN UNIVERSITY