Meet the speaker

Brittany O’Shea
Director of Security PMM @GitHub

Session Agenda

State of Supply Chain Security

Trends driving supply chain attacks

Types of supply chain attacks

Anatomy of a supply chain attack

Best Practices to secure your software supply chain

Securing accounts
Securing code
Securing build systems

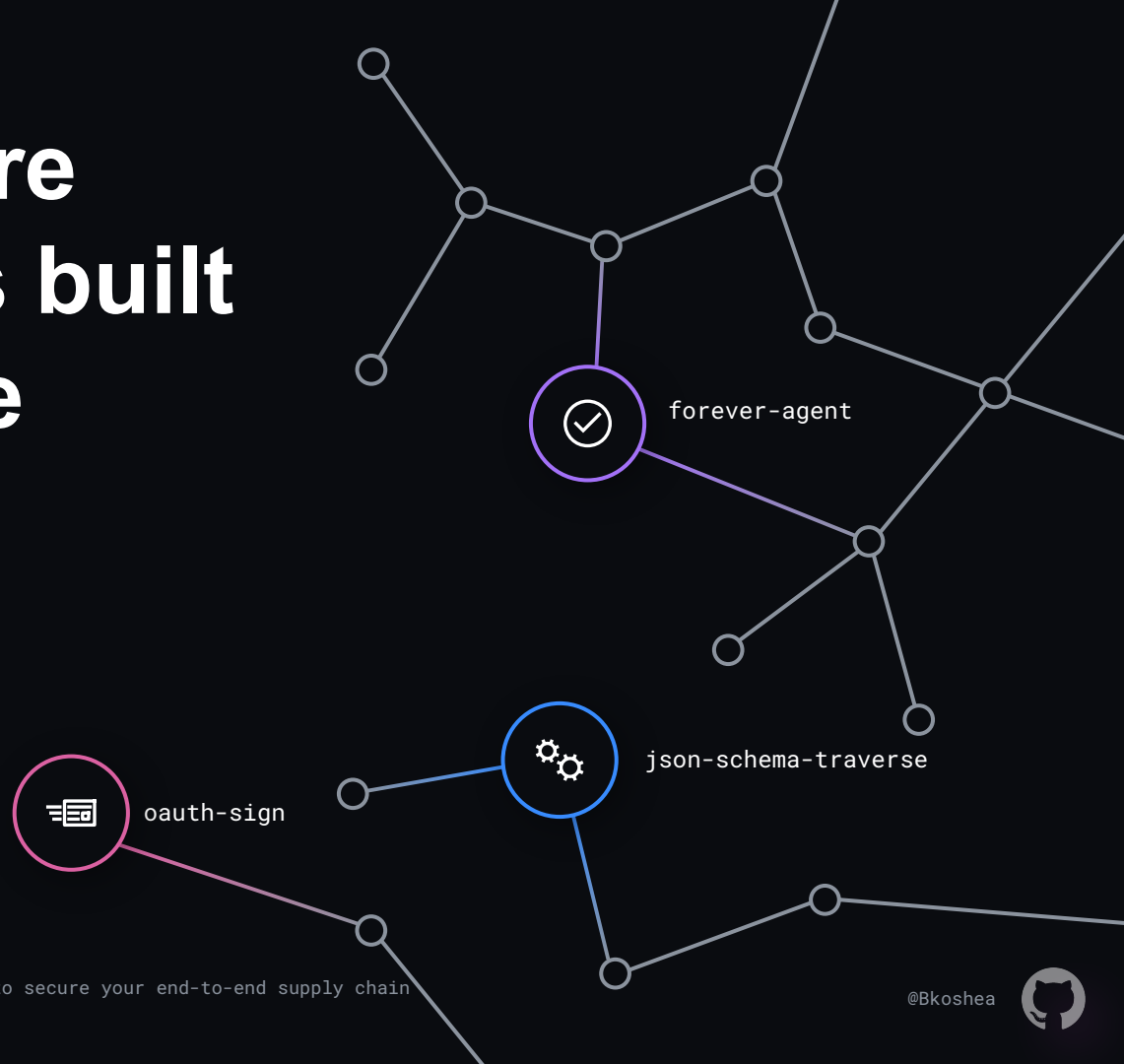
The State of Supply Chain Security is rapidly evolving

Companies must think through and secure a wider attack surface

Modern software development is built on open source

80-90%
open source in new applications

15K+
OSS packages released daily



Enabling open source culture, best practices, and frictionless code reuse inside companies

87%

productivity increase from adopting innersource practices

Source: State of the Octoverse Report, 2021

State of Supply Chain Security

Next-Gen supply chain attacks are up 650%

64% of orgs have been impacted in past 12 months

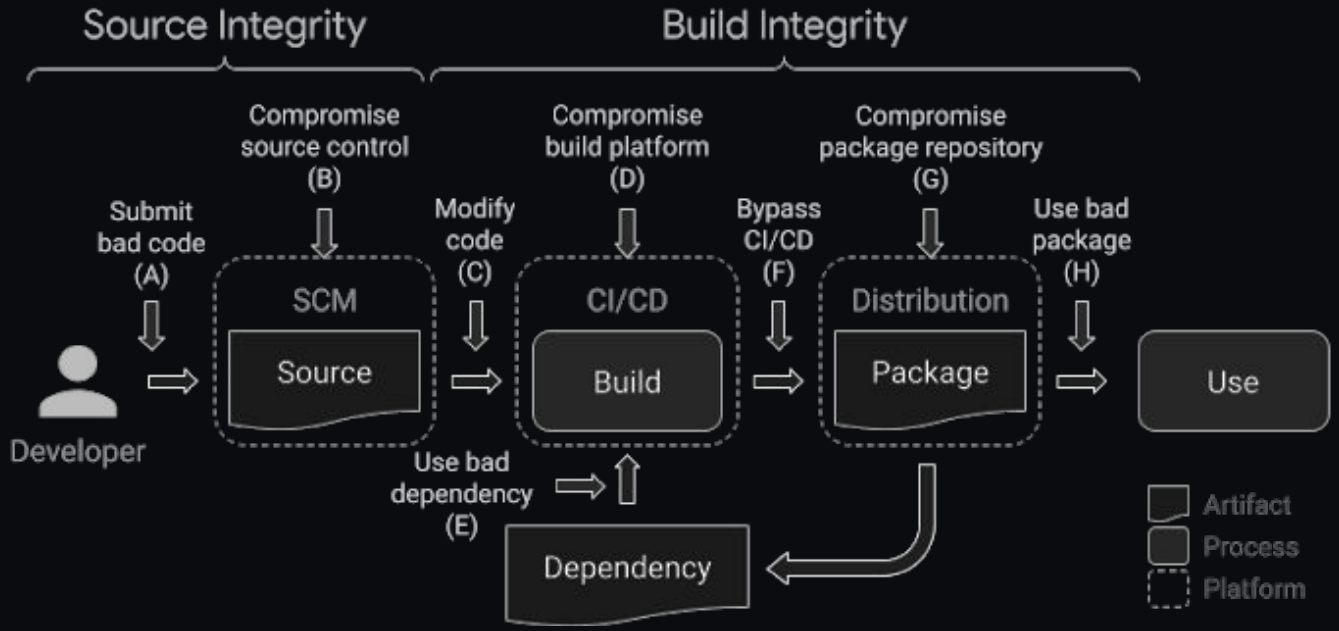
45% of orgs are halfway finished securing their supply chains

Sources: Sonatype, Anchore, Cloudbees

How does a Supply Chain Security attack happen

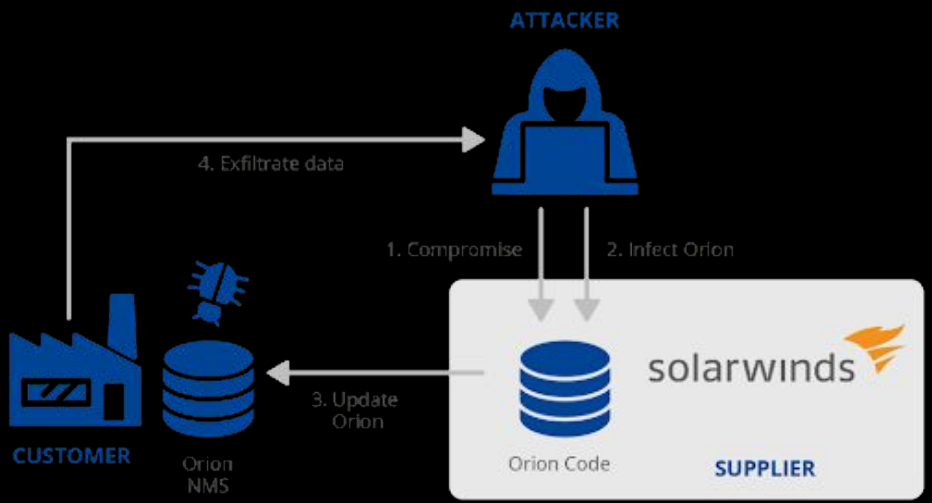
Anatomy of a supply chain attack

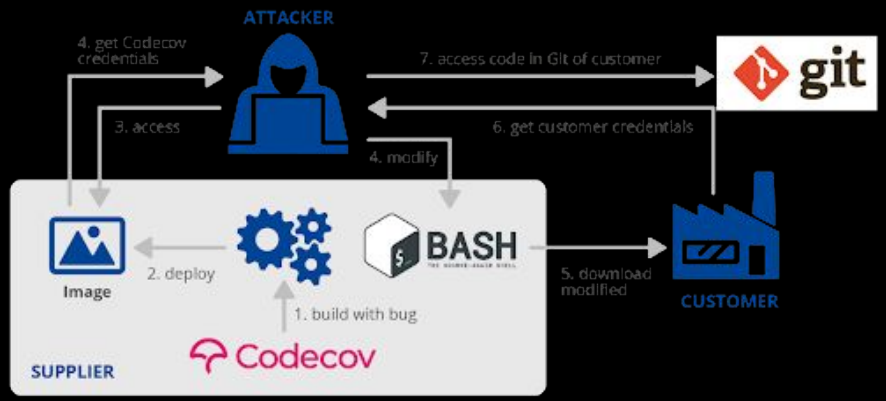
Sample Supply Chain compromise points



Sources: Google

Sample Supply Chain Attacks



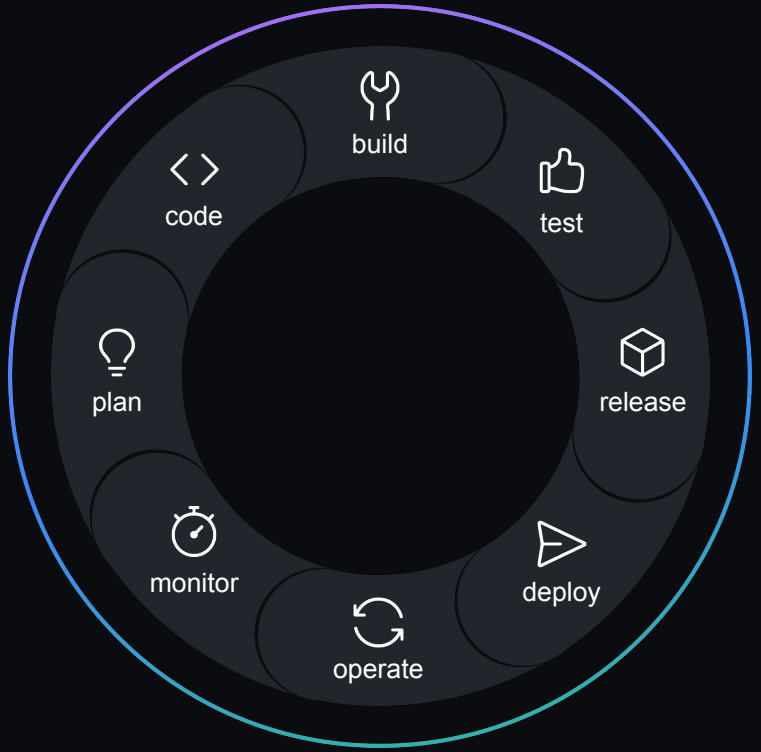


Sources: ENISA Threat Landscape

DevSecOps is what DevOps should have been from the beginning.” –Willis, 2021

Best Practices to secure your software supply chain

End-to-end supply chain security is about making sure the code you distribute hasn't been tampered across the SDLC.



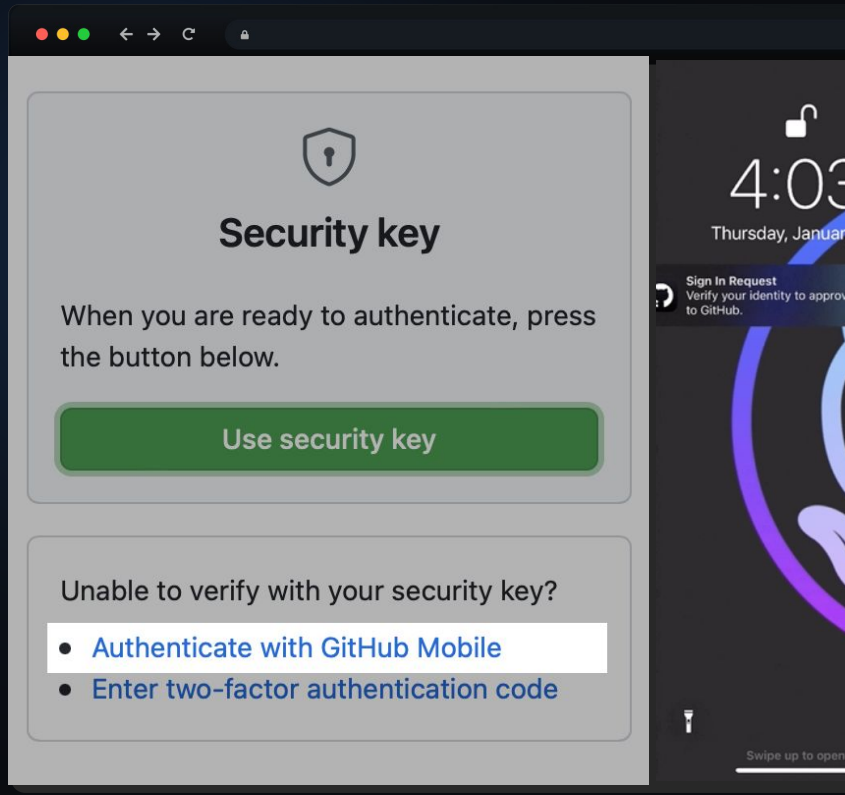
Securing your personal account

Start with configuring two-factor authentication

Two-factor authentication (2FA) is an extra layer of security used when logging into websites or apps. With 2FA, you have to log in with your username and password and provide another form of authentication that only you know or have access to.

Leverage SSH keys securely

If you store your SSH private key on a disk drive, it's a good idea to protect it with a passphrase. Another option is to generate SSH keys on a hardware security key.

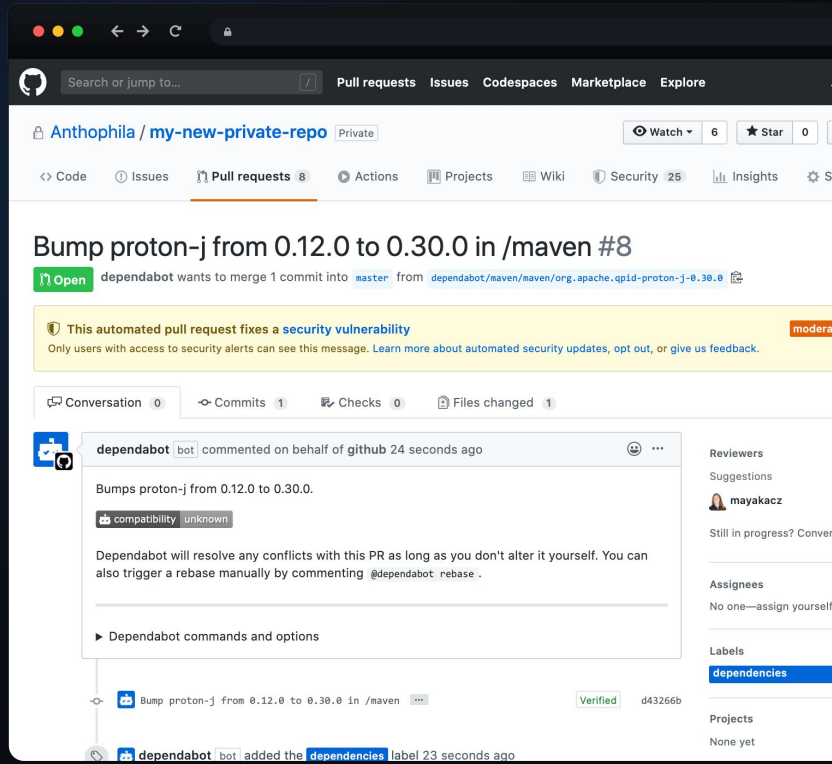


Securing code in your supply chain

Start with a vulnerability management program for dependencies

A vulnerability management program should include:

1. An inventory of your dependencies
2. The ability to know when there is a security vulnerability in a dependency
3. The ability to assess the impact of that vulnerability on your code and decide what action to take.



Securing code in your supply chain

Secure your tokens

Your system needs access to secrets to run, but it's best practice to not include them in your source code. This is especially important for public repositories, but also for private repositories to which many people might have access.

To detect potentially leaked secrets leverage a secret scanning tool and have a revoking process in place where needed.

2 /src/DataModel/LoginHelper/LoginHelper.cs

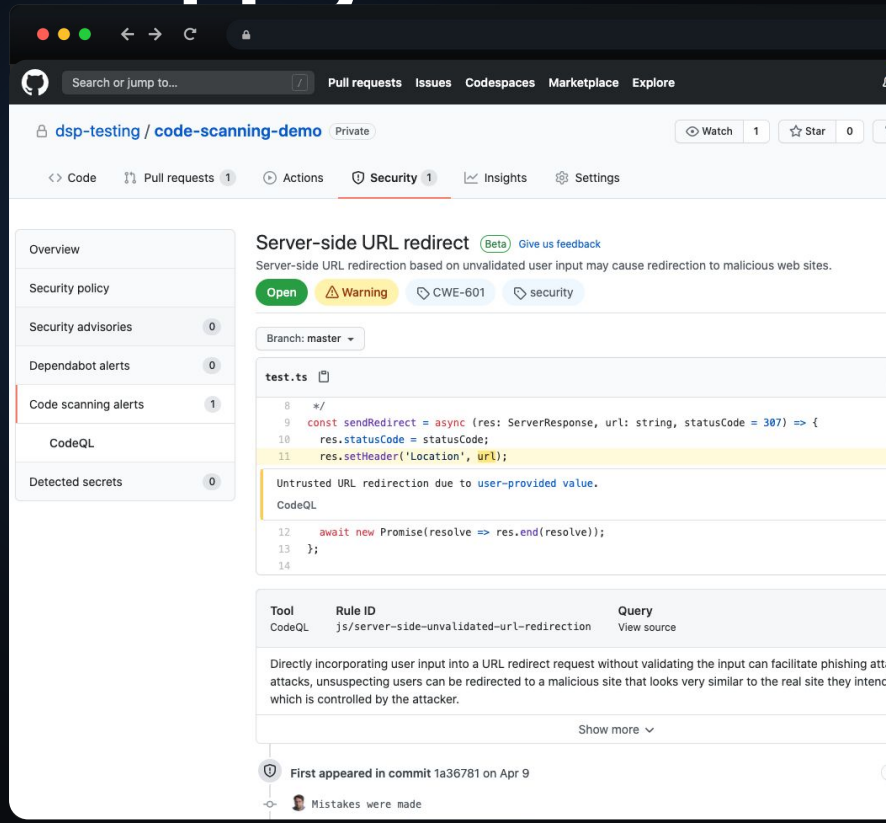
```
1  namespace DataModel
2  {
3      public static class LoginHelper
4      {
5          public static String ServiceUrl = "https://cloud.exam
6          public static String ClientID = "DataModel-0001";
7          public static String ClientSecret = “A002019DRBES$%FA
8          public static String RedirectURL = Windows.Security.A
9
10         /// <summary>
11         /// Handles acquiring all relevant tokens for the app
12         /// </summary>
13         /// <returns>Async progress task.</returns>
14         public static async Task<bool> Authenticate()
15         {
```


Securing code in your supply chain

Keep vulnerable coding patterns out of your repository

Vulnerable code is difficult to spot in manual code reviews. For example, a function that isn't memory-safe, or failing to escaping user input that could lead to an injection vulnerability.

You automatically you can check it for patterns that are associated with security vulnerabilities with SAST tools (static application security testing).



Securing your build system

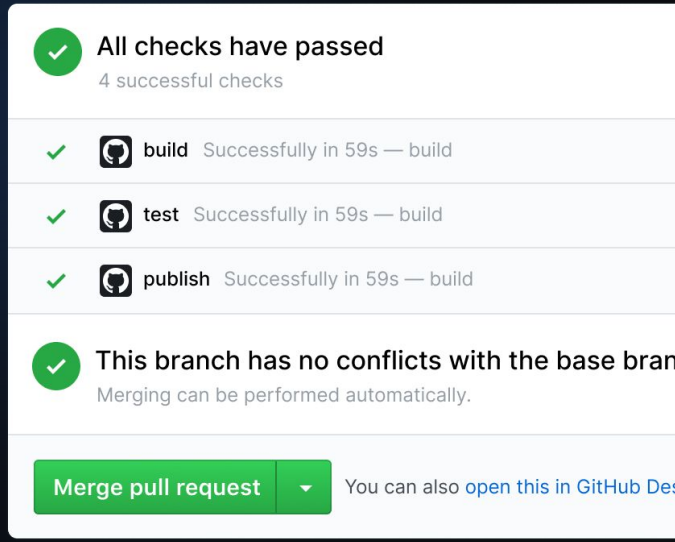
Start with ensuring your build system has security best practices in place

There are several security capabilities a build system should have:

1. The build steps should be clear and repeatable.

2. You should know exactly what was running during the build process.

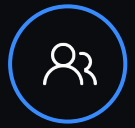
3. Each build should start in a fresh environment, so a compromised build doesn't persist to affect future builds.



Securing your build systems

Sign your builds

To verify your code hasn’t been tampered with you sign your builds. When distributing software publicly, this is often done with a public/private cryptographic key pair. You use the private key to sign the build, and you publish your public key so users of your software can verify the signature on the build before they use it.



Ensure credentials are minimally scoped

Limit who can make changes to your workflows and frequently inspect which users should and can have write access to your repositories

Actions you can take to get started today

Read the end-to- end supply chain security guide

Enable security solutions in your repos

Thank you