



Course material

Course:

Understanding the digital supply chain and its stakes for humanitarian actors

Video:

1.2.3 DevSecOps

Concepts (extracted from automatically generated subtitles):

Software development. Software applications. Testing environments. Internal code. Security team. Build environment. Credentials of a developer. External dependencies. Strict verification of software integrity. Thorough tests. Diverse internal threats. Software bill of material. Software package. Mobile phones. Massive complexity of dependencies.



[to video sequence search](#)

(within Understanding the digital supply chain and its stakes for humanitarian actors.)

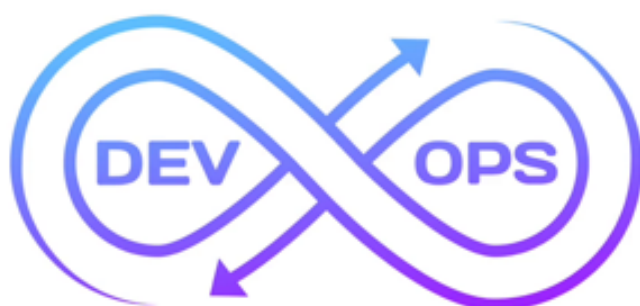


[to video](#)

Center for Digital Education. More educational support material here:

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>
page 1/27

DevSecOps: Bringing Security into DevOps



...

notes

summary

0m 0s



DevSecOps: Bringing Security into DevOps



Develop: create code securely

Build: automate and check

Test: security testing

Deploy: publish images

Operate: integrity checking

Monitor: check for drift

Security team: validates the whole chain

DevSecOps integrates security into this world of chaos across development, build systems, testing environments, deployment, operation and monitoring, with a security team overseeing each step. We will now go into step-by-step details for each of these topics.

notes

summary

0m 9s



DevSecOps: E



Software development faces diverse internal threats that require protection.

notes

summary

0m 24s





Develop

Insider Threats from Developers:

- All internal codes must be checked, reviewed and documented.
- 2-Factor-authentication.

A developer at a company may go rogue and pose an insider threat.

notes

summary

0m 25s





Develop

Insider Threats from Developers:

- All internal codes must be checked, reviewed and documented.
- 2-Factor-authentication.

All internal code must therefore be checked, reviewed, and documented with thorough tests. The credentials of a developer could also be compromised through, for example, a phishing attack.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

summary

0m 37s



.....

.....

.....

.....

.....



Develop

Insider Threats from Developers:

- All internal codes must be checked, reviewed and documented.
- 2-Factor-authentication.

Internal code might be buggy, use weak cryptography or hard-coded secrets, which might turn into vulnerabilities:

- Code Review.

This requires 2-Factor-authentication and careful monitoring.

notes

summary

0m 49s





Develop

Insider Threats from Developers:

- All internal codes must be checked, reviewed and documented.
- 2-Factor-authentication.

Internal code might be buggy, use weak cryptography or hard-coded secrets, which might turn into vulnerabilities:

- Code Review.

Source Code might be tampered:

- Careful validation.

Developed code may be buggy or introduce weak crypto or hard-coded secrets. Some of these bugs will turn into vulnerabilities that could be exploited externally. At this step, code reviews are crucial. Last but not least,

notes

summary

0m 55s





Develop

Insider Threats from Developers:

- All internal codes must be checked, reviewed and documented.
- 2-Factor-authentication.

Internal code might be buggy, use weak cryptography or hard-coded secrets, which might turn into vulnerabilities:

- Code Review.

Source Code might be tampered:

- Careful validation.

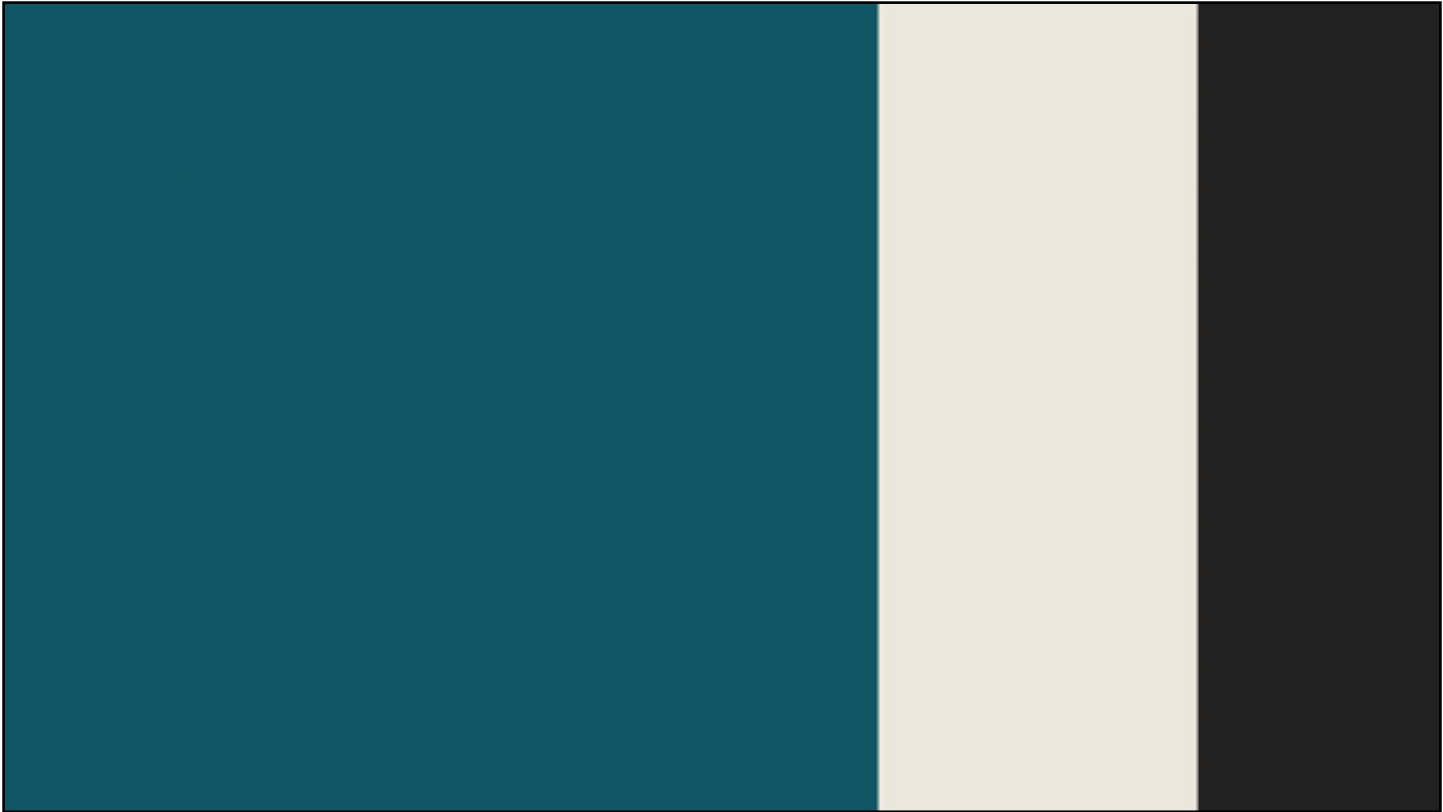
the source code itself could be tampered with before it is built into an application. Careful validation is necessary to ensure that the build environment receives the same code that was developed.

notes

summary

1m 10s





Software applications are built on so-called

notes

.....

.....

.....

.....

.....

.....

.....

.....

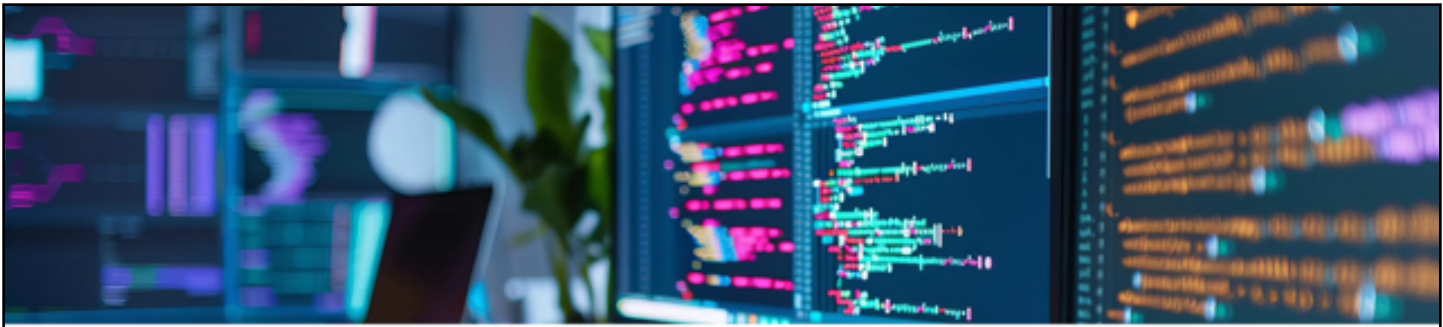
.....

.....

summary

1m 24s





Build

Continuous Integration

Platforms collect code and dependencies to create application packages.

Dependency Risks

External dependencies may contain backdoors or malicious code.

Software Bill of Materials (SBOM)

- Strict verification of software integrity.
- SBOM documents all components used.

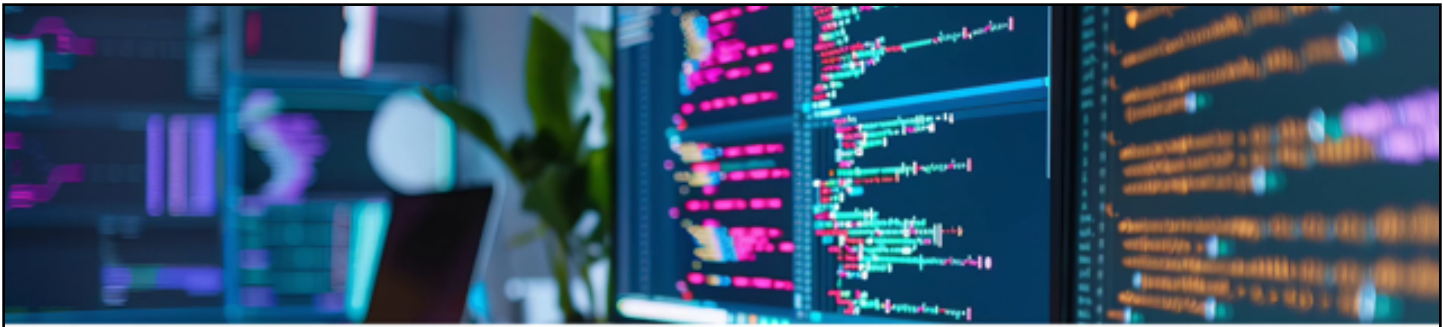
continuous integration platforms. These platforms collect all code and external dependencies and integrate them into an application package,

notes

summary

1m 25s





Build

Continuous Integration

Platforms collect code and dependencies to create application packages.

Dependency Risks

External dependencies may contain backdoors or malicious code.

Software Bill of Materials (SBOM)

- Strict verification of software integrity.
- SBOM documents all components used.

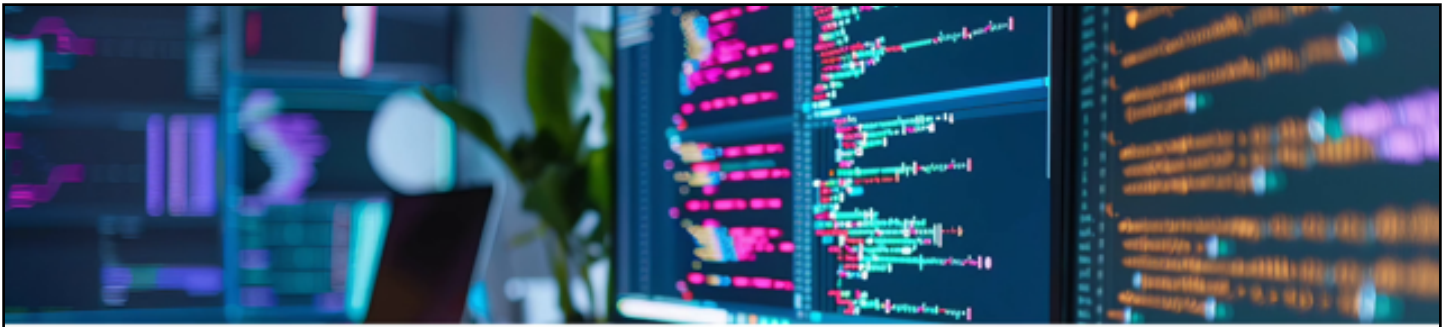
together with the configuration and optimisation settings. This allows the automatic creation of testable and deployable applications whenever the source code changes. Adversaries may control some of these external dependencies that are pulled in at this point, and thereby gain control over the application when it is used,

notes

summary

1m 38s





Build

Continuous Integration

Platforms collect code and dependencies to create application packages.

Dependency Risks

External dependencies may contain backdoors or malicious code.

Software Bill of Materials (SBOM)

- Strict verification of software integrity.
- SBOM documents all components used.

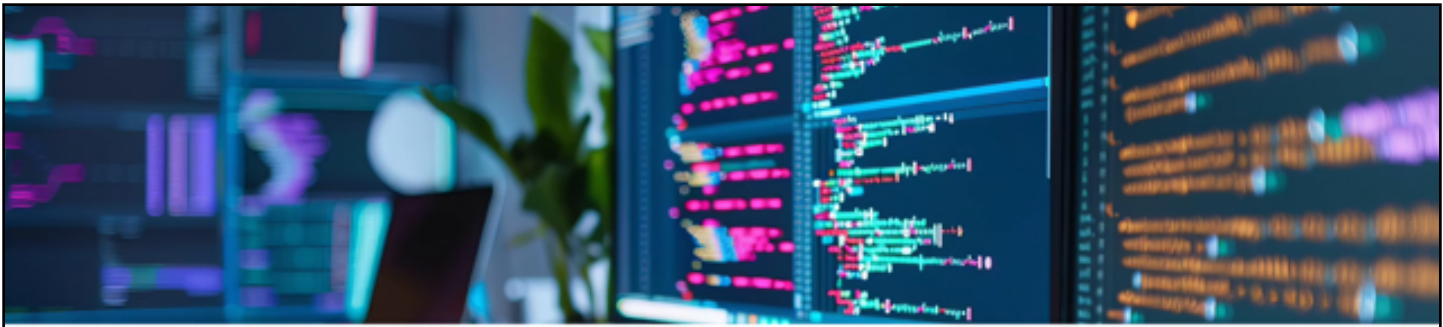
or in the worst case, even of the build environment.

notes

summary

1m 58s





Build

Continuous Integration

Platforms collect code and dependencies to create application packages.

Dependency Risks

External dependencies may contain backdoors or malicious code.

Software Bill of Materials (SBOM)

- Strict verification of software integrity.
- SBOM documents all components used.

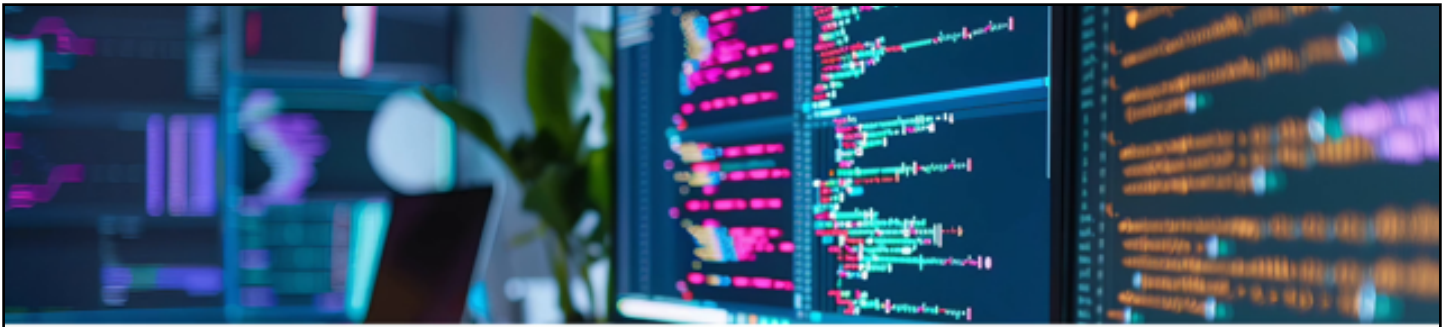
Malicious dependencies may contain backdoors that can only be activated by the attacker. Due to the massive complexity of dependencies,

notes

summary

2m 2s





Build

Continuous Integration

Platforms collect code and dependencies to create application packages.

Dependency Risks

External dependencies may contain backdoors or malicious code.

Software Bill of Materials (SBOM)

- Strict verification of software integrity.
- SBOM documents all components used.

it is extremely challenging to detect these modifications and strict verification of software integrity is necessary and crucial. The build environment ideally appends a software bill of material, generally called an SBOM,

notes

summary

2m 13s





Build

Continuous Integration

Platforms collect code and dependencies to create application packages.

Dependency Risks

External dependencies may contain backdoors or malicious code.

Software Bill of Materials (SBOM)

- Strict verification of software integrity.
- SBOM documents all components used.

to document clearly what components were used.

notes

summary

2m 25s





Component Testing

Thoroughly test all components, including the SBOM.

Regression Testing

Ensure software remains functional.

Security Testing

Always incomplete, and should thus target main attack vectors to cope with computational limitations.

During testing, all components of the software must be thoroughly tested, including that the SBOM has not been tampered with. Some software may not come with tests, or the tests may be incomplete. This requires developers to write thorough tests and to configure them correctly. Malicious tests at this stage could even be part of the attack surface, allowing the attacker some late-stage modifications.

notes

summary

2m 30s





Component Testing

Thoroughly test all components, including the SBOM.

Regression Testing

Ensure software remains functional.

Security Testing

Always incomplete, and should thus target main attack vectors to cope with computational limitations.

At this stage, the software package is carefully analysed

notes

summary

2m 57s





Component Testing

Thoroughly test all components, including the SBOM.

Regression Testing

Ensure software remains functional.

Security Testing

Always incomplete, and should thus target main attacks vectors to cope with computational limitations.

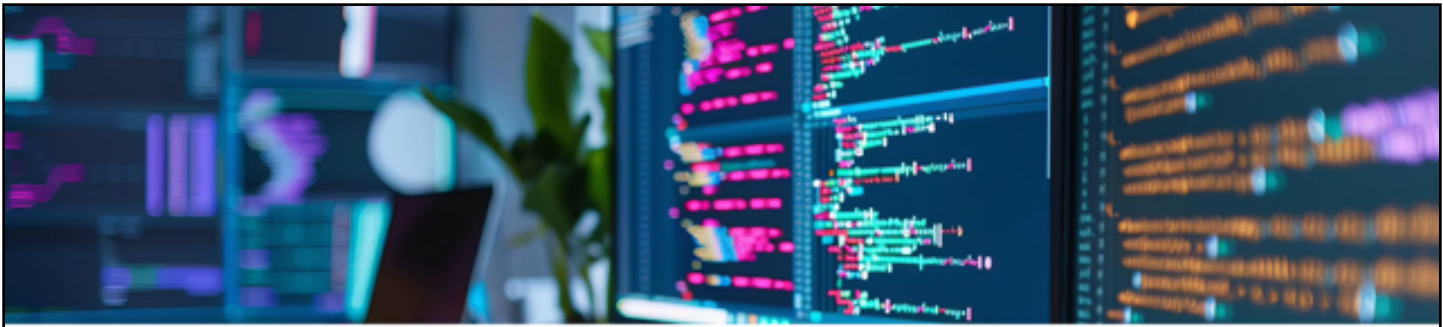
for regression and security. Regression tests ensure that the software remains functional and is usually very straightforward as each feature can simply be tested. Security testing is much more challenging as now, the tests must ensure that there are no bugs, which is, in general, an unsolvable problem due to the exponentially many execution paths through the application which results in computational explosion. Any security testing is therefore incomplete and must carefully target the main attack vectors.

notes

summary

3m 1s





Deploy

Transport Security

Protect software during transport to deployment repositories.

Market Integrity

Be aware of potential tampering by market owners.

Configuration Protection

Safeguard application metadata and environment settings.

The software image could be tampered with,

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

summary

.....

.....

.....

.....

.....

.....


.....

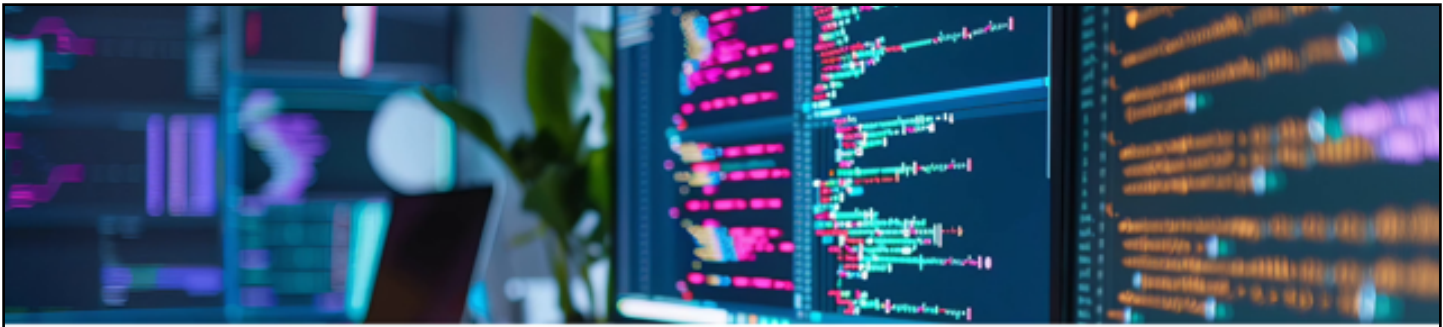
.....

.....

.....

3m 35s





Deploy

Transport Security

Protect software during transport to deployment repositories.

Market Integrity

Be aware of potential tampering by market owners.

Configuration Protection

Safeguard application metadata and environment settings.

when it is transported to the deployed repos, or if the market is malicious by the market owners themselves. The software owner gives partial control over their application to the market owners which deploy their code. In addition, the configuration and environment of the application could be tampered with at the stage of the market,

notes

summary

3m 38s





Deploy

Transport Security

Protect software during transport to deployment repositories.

Marke

Be aware of tampering with the owner's property.

which usually contains some metadata along with the applications. Markets are in an extremely powerful position as they deploy code to the end users and can control which packages are deployed. End users could check that the application was cryptographically protected through a signature of the original software manufacturer, but this is rarely done.

notes

summary

4m 1s





Risks

Software may be misconfigured accidentally.

System might also be compromised via other software.

Continuous monitoring AND new versions must be installed immediately.

Operators and system administrators may misconfigure software that is running on their end. This gives attackers a vector to abuse unintended features. Systems could also be compromised through other software, allowing attackers access to the applications through the system. For example, mobile phones could be compromised through malicious apps that then attack the banking app. When software is being deployed, it must be continuously monitored and checked. When new versions of any component are available, it must be installed immediately after testing.

notes

summary

4m 24s





Best Practices



Integrated Security Team

Full visibility and actionable changes at each step.



Thorough Code Reviews

Track and check component provenance.



Dependency Management

Ensure no drift among products.



Continuous Monitoring

Regularly check all systems and update SBOM.

Let us now summarise this section with a set of best practices. The security team has full visibility and is integrated at each step of the development and deployment process. It is included in operations and allowed to conduct actionable changes.

notes

summary

5m 0s





Best Practices



Integrated Security Team

Full visibility and actionable changes at each step.



Thorough Code Reviews

Track and check component provenance.



Dependency Management

Ensure no drift among products.



Continuous Monitoring

Regularly check all systems and update SBOM.

All code is thoroughly reviewed, and the provenance of components is tracked and checked. The company does thorough dependency management for each component and ensures that there is no drift among products. The SBOM is carefully defined and updated whenever components are available.

notes

summary

5m 17s





Best Practices



Integrated Security Team

Full visibility and actionable changes at each step.



Thorough Code Reviews

Track and check component provenance.



Dependency Management

Ensure no drift among products.



Continuous Monitoring

Regularly check all systems and update SBOM.

All updates are installed immediately and keep the systems up to date. Last but not least, all systems are continuously monitored and checked.

notes

summary

5m 37s





In order of appearance

PICTURE1 : Supply Chain concept by tippapatt from Adobe Stock
PICTURE2 : Devops eight infinity by Svitlana from Noum Project
PICTURE3 : Devops logo isolated on white background by Vector Galaxy from Noum Project
PICTURE4 : DevOps software development by MT.PHOTOSTOCK from Adobe Stock
PICTURE5 : Programmer Coding holographic code by photography09 from Adobe Stock
PICTURE6 : Pensive programmer working on on desktop by Snowing12 from Adobe Stock

As you can see, there are many ways that software supply chain can be prone to attacks. Good security depends on thorough planning ahead of time. Covering all bases by integrating security along the steps of the process and continuously checking if the protocols are followed.

notes

summary

5m 46s