



Course material

Course:

Understanding the digital supply chain and its stakes for humanitarian actors

Video:

1.3.2 Mathias Case Studies

Concepts (extracted from automatically generated subtitles):

Original source code. Commonly used component. Concrete case studies of threats. Code repositories. Solarwinds attack. Software supply chain. Source code. Source control systems. External components. Logging framework. Long time. Access control. Compromised components. Ease of use. Case study.



[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 1/22

Successful Cases of Attacks on the Software Supply Chain



Case 1: Compromised code repository



Case 2: Buggy dependencies



Case 3: Malicious back door

We now cover several concrete case studies of threats and vulnerabilities along the software supply chain to give you a glimpse into where and how attacks can happen. Software is immensely complex and there are ample opportunities all along the chain. We will dive into 3 examples that targeted external components through compromised

notes

summary

0m 0s



Successful Cases of Attacks on the Software Supply Chain



Case 1: Compromised code repository



Case 2: Buggy dependencies



Case 3: Malicious back door

code repositories by maliciously modifying the original source code, buggy dependencies where external modules had severe vulnerabilities, and malicious backdoors

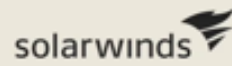
notes

summary

0m 26s



Case n° 1: SolarWinds



- **Source code are stored in systems that track changes.**
- = Systems with highly regulated access control.
- Because risks of source revision corruption and subsequent code modification.

where software was modified on purpose by a maintainer to include backdoors.
Attacking source code to embed backdoors is a prime example of a supply chain attack.

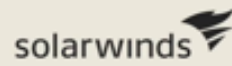
notes

summary

0m 37s



Case n° 1: SolarWinds



- **Source code are stored in systems that track changes.**
- = Systems with highly regulated access control.
- Because risks of source revision corruption and subsequent code modification.

Source code is generally stored in source control systems that track revisions between versions and streamline deployment.

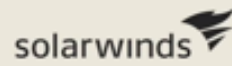
notes

summary

0m 49s



Case n° 1: SolarWinds



- **Source code are stored in systems that track changes.**
- **= Systems with highly regulated access control.**
- **Because risks of source revision corruption and subsequent code modification.**

Access control to these systems must be carefully regulated. Bad or incomplete access control or compromised credentials allow adversaries to corrupt source revision systems and to modify the code that is later used to build applications.

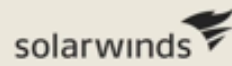
notes

summary

0m 56s



Case n° 1: SolarWinds



In the SolarWinds attack, adversaries discovered an exposed server password in a public GitHub repository and used this password to gain an initial foothold on the company's systems to compromise source code repositories for a privileged component.

notes

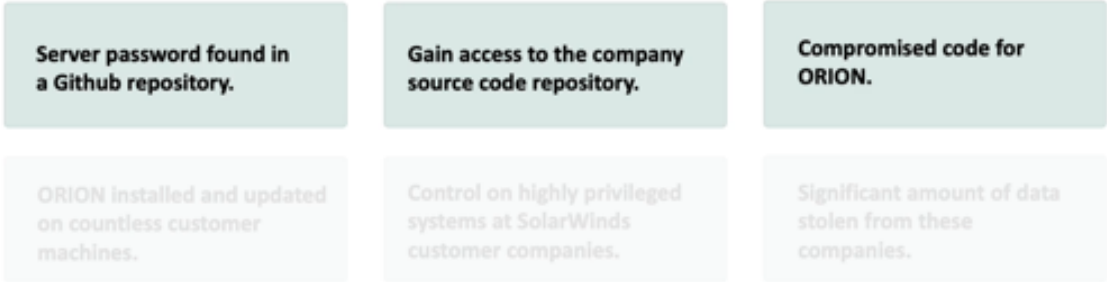
summary

1m 12s





Case n° 1: SolarWinds



After gaining an initial foothold, the attackers compromised the source code for ORION,

notes

summary

1m 29s





Case n° 1: SolarWinds

Server password found in a Github repository.

Gain access to the company source code repository.

Compromised code for ORION.

ORION installed and updated on countless customer machines.

Control on highly privileged systems at SolarWinds customer companies.

Significant amount of data stolen from these companies.

an essential privileged component that was used for system monitoring. SolarWinds ORION was installed on thousands of machines of their customers through the automatic update process, which automatically pushed the compromised components to customers.

notes

summary

1m 37s





Case n° 1: SolarWinds

Server password found in a Github repository.

Gain access to the company source code repository.

Compromised code for ORION.

ORION installed and updated on countless customer machines.

Control on highly privileged systems at SolarWinds customer companies.

Significant amount of data stolen from these companies.

This allowed the adversaries to gain control of highly privileged systems of companies that were customers of SolarWinds. The deep embedding of the malware into this trusted component avoided detection for a long time and enabled the adversaries to steal significant data from SolarWinds customers.

notes

summary

1m 52s



Case n°2: Log4J



SolarWinds was a wake-up call on how trusted components can be used to undermine the security of complex ecosystems. It's best to check all privileged software even if it comes from a trusted vendor as they may be compromised themselves. While the first case focused on exposed credentials allowing adversaries to gain access. This case study focuses on a plain vulnerability

notes

summary

2m 10s



Case n°2: Log4J



Vulnerability in a commonly used component

- Dependencies must be carefully monitored.
- Systems just be patched in time.

in a commonly used component. Bugs in dependencies may be reachable through input from the application. These dependencies have to be carefully monitored over time

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

summary

2m 37s



.....

.....

.....

.....

.....

Case n°2: Log4J



Vulnerability in a commonly used component

- Dependencies must be carefully monitored.
- Systems just be patched in time.

to see if any issues arise. Vendors and customers must carefully monitor what libraries they use and track any issues that come up to patch their systems in time.

notes

summary

2m 50s



Case n°2: Log4J




Log4J is a logging framework that was used ubiquitously in Java software due to its ease of use and simplicity. Many software projects included Log4J and forgot about it.

notes

summary

3m 1s



Product ▾Solutions ▾Resources ▾Open Source ▾Enterprise ▾Pricing

apache / logging-log4j2Public

> log4j-couchdb

> log4j-docker

log4j-fuzz-test

1 /*

2 * Licensed to the Apache Software Foundation (ASF) under one or more

3 * contributor license agreements. See the NOTICE file distributed with

4 * this work for additional information regarding copyright ownership.

Case n°2: Log4J

Log4J = logging framework used ubiquitously (Javs).

Many software project used it but without documenting it.

Many did not even know they were using it.

Log4J contains a bug that allows remote code execution.

Attackers could trigger widespread code execution.

Companies spend significant resources to protect themselves.

Over time, the functionality grew more complex. Software slowly updated to newer versions, but customers largely did not consider track where and how they were including or even using Log4J. Unfortunately, Log4J contained a bug in its functionality that allowed remote code executions.

notes

summary

3m 15s

page 15/22 - 1.3.2 Mathias Case Studies

```
1  /*
2   * Licensed to the Apache Software Foundation (ASF) under one or more
3   * contributor license agreements. See the NOTICE file distributed with
4   * this work for additional information regarding copyright ownership.
```

Companies spend significant resources to protect themselves.

notes




```
1  /*
2   * Licensed to the Apache Software Foundation (ASF) under one or more
3   * contributor license agreements. See the NOTICE file distributed with
4   * this work for additional information regarding copyright ownership.
```

Companies spend significant resources to protect themselves.



Case n°3: XZ



Log4J was a reminder that open source components need to be thoroughly checked and validated. Vulnerabilities are just a matter of time, and it's much easier to address them if one knows where the vulnerable software is used and how. Any dependency must be checked, validated and thoroughly documented. The last case study focused on a common vulnerability. This case study now generalises from vulnerabilities and shows how small open source components can be actively compromised by attackers. Open-source components are broadly used in software, but developers often lack the resources for maintaining these libraries.

notes

summary

4m 8s



Case n°3: XZ



- **Compression utility broadly used in many systems.**
- **Quite stable, so no need for new features.**
- **Maintained by a single developer.**

Targeted attack via open source contributions allowed to slowly gain access...

XZ/liblzma is a common compression utility that was not looking for new features, but was broadly used in diverse systems such as the Linux operating system kernel, OpenSSH, which is a key service to interoperate with servers, or even systemd, which is a core system configuration service. As the library was stable, there were not many new features needed and a single developer worked on the tool to maintain it. A targeted attack by likely a state actor slowly gained access to the source code by contributing test cases and helping the single developer update the software.

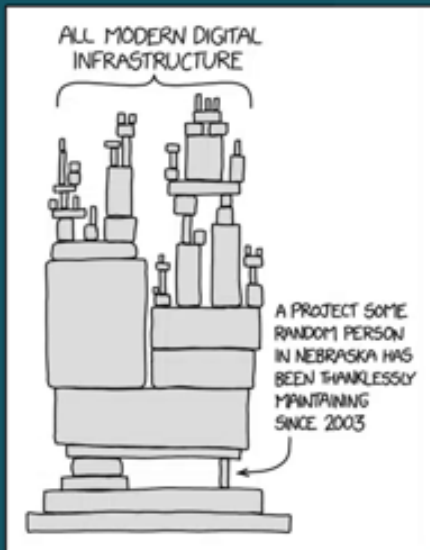
notes

summary

4m 53s



Case n°3: XZ



Over several years, the developer Yatan gained the trust of the main developer and eventually also became a maintainer. After becoming a maintainer, Yatan continued the development, but started slowly integrating a complex backdoor that required a lot of moving components. The backdoor was hidden in test files that were only activated under special conditions when the compression library was built for specific targets. This made the backdoor extremely stealthy but also extremely powerful. Essentially, it allowed an adversary to log into any system through OpenSSH with a special private key. The backdoor was accidentally discovered when a developer saw performance degradation during testing of another component that essentially exposed the backdoor.

notes

summary

5m 37s



Case n°3: XZ



- Any open-source component should be checked.
- Companies should invest in protecting the ecosystem.

Backdooring XZ demonstrated the brittleness of open source. Any component that is used must be checked and companies using these components should invest resources into protecting the ecosystem, potentially by supporting developers.

notes

summary

6m 32s





SolarWinds, Log4J, and the XZ attack are the highest impact software supply chain attacks so far. Each of them demonstrates a slightly different angle of attack against the software supply chain. For SolarWinds, attackers compromise the source code of a supplier to gain access to target systems. For Log4J, a simple programming bug in a core dependency led to disaster. In XZ, attackers gained maintainer access and planted the backdoor. Supply chain attacks can be extremely diverse, and as mentioned before, security must be an integral part of each step in the software supply chain.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

summary

.....

.....

.....

.....

.....

