TRUSTING TRUST REVISITED

Preventing Software Supply Chain Attacks
Using Modern Methods

Florian Sihler

Seminar
February 12, 2021
Institute of Distributed Systems, Ulm University

2

2

~ 18 000 customers[7]
SolarWinds 2020

> 500 million users[1]
XcodeGhost 2015

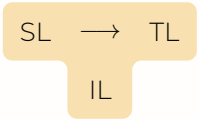
> 100 000 users[2]
Win32/Induc 2009

3 The Attack

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

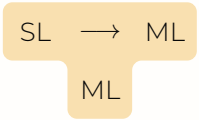
3 The Attack



[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

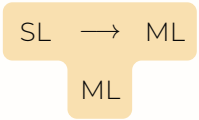
3 The Attack

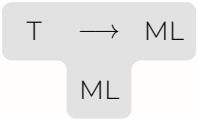


[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

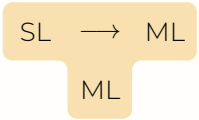


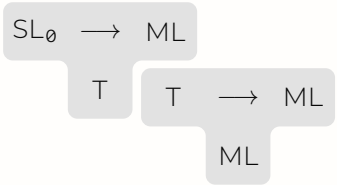


[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

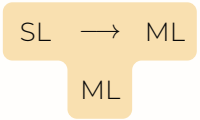


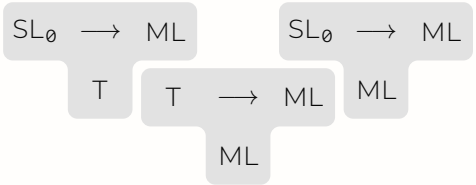


[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

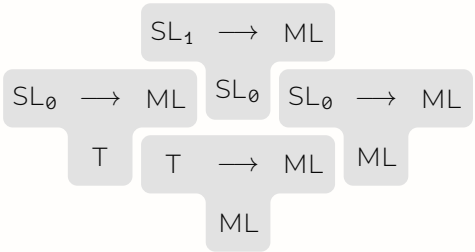


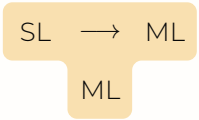


[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

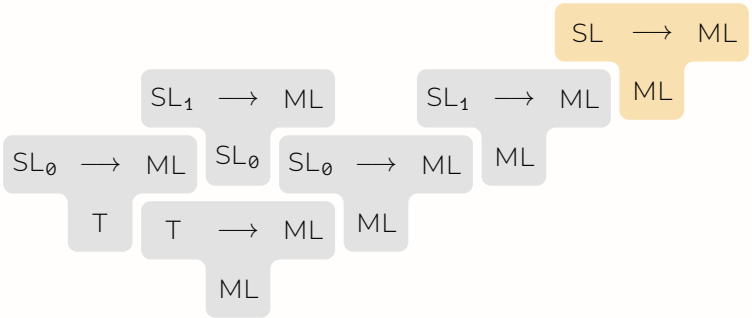




[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

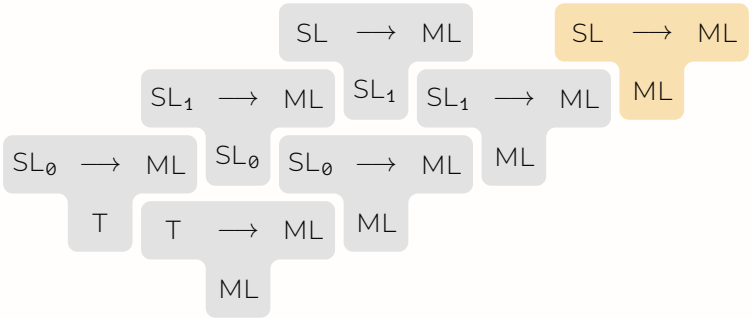
3 The Attack



[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

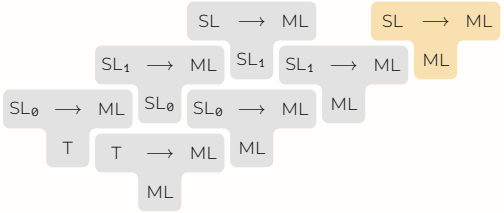
3 The Attack



[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

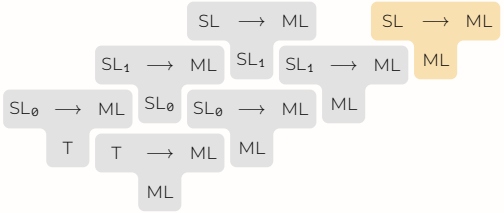
3 The Attack



[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

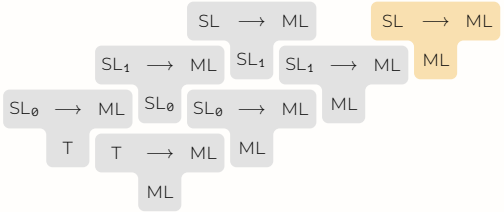



```
compile()
```

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

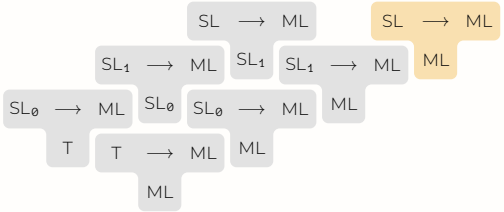



```
inject = 'inject = %c%s%c
if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)'

if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)

compile()
```

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

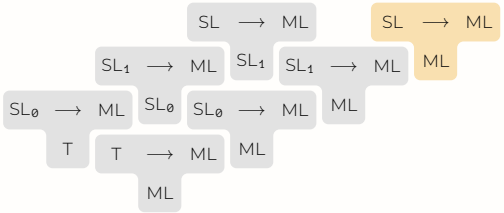


```
inject = 'inject = %c%s%c
if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)'

if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)

compile()
```


[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

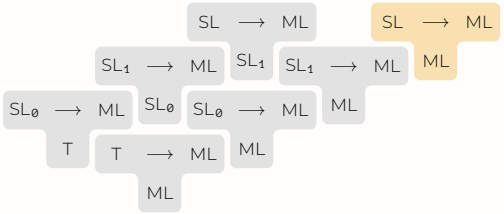


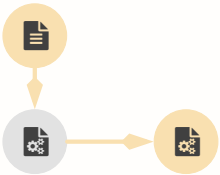
```
inject = 'inject = %c%s%c
if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)'

if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)

compile()
```

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack



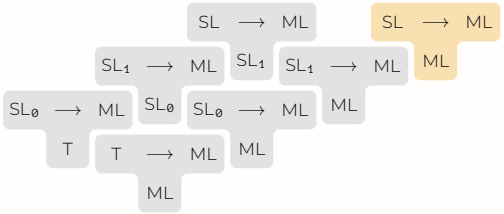


```
inject = 'inject = %c%s%c
if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)'

if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)

compile()
```

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack

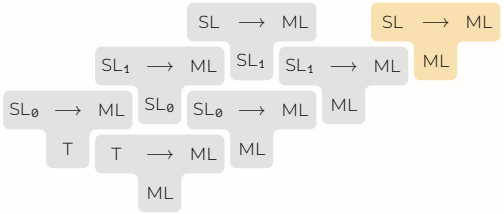


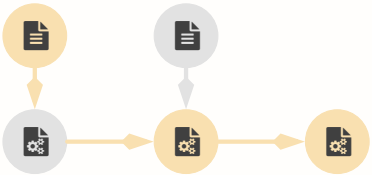
```
inject = 'inject = %c%s%c
if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)'

if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)

compile()
```

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

[6] Thompson: „Reflections on Trusting Trust" 1984

3 The Attack





```
inject = 'inject = %c%s%c
if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)'

if source contains "compile()":
  prepend("compile()", inject % 34, inject, 34)

compile()
```

[3] McKeeman, Horning und Wortman: *A Compiler Generator* 1970

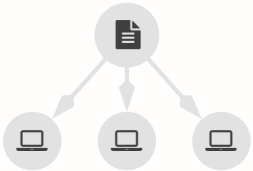
[6] Thompson: „Reflections on Trusting Trust" 1984

4 Diverse Double Compiling

4 Diverse Double Compiling

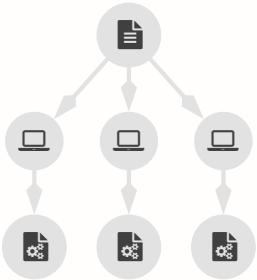
Reproducible

4 Diverse Double Compiling



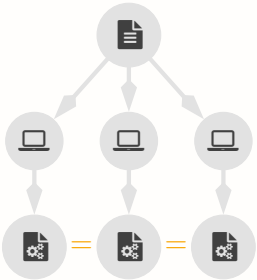
Reproducible

4 Diverse Double Compiling



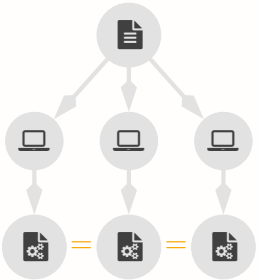
Reproducible

4 Diverse Double Compiling



Reproducible

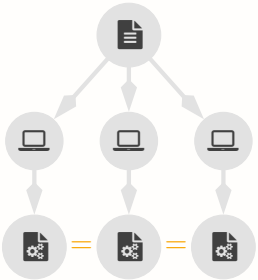
4 Diverse Double Compiling



Reproducible

1. 

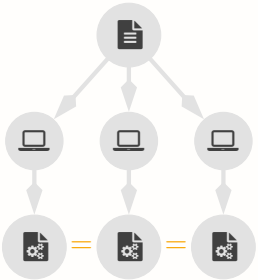
4 Diverse Double Compiling



Reproducible

1. 

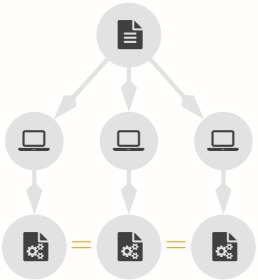
4 Diverse Double Compiling



Reproducible

1.

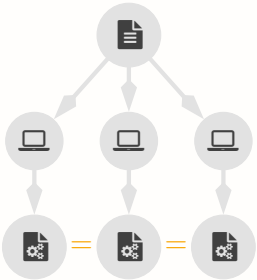
4 Diverse Double Compiling



Reproducible

1.

4 Diverse Double Compiling

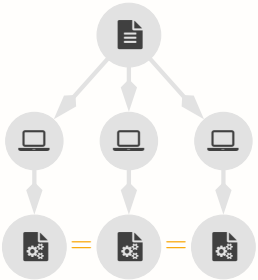


Reproducible

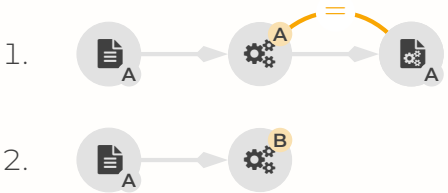
1. 

2. 

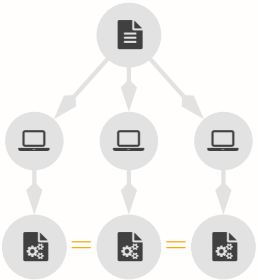
4 Diverse Double Compiling



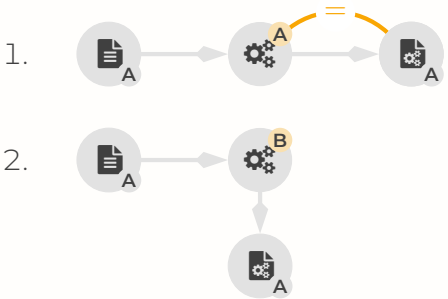
Reproducible



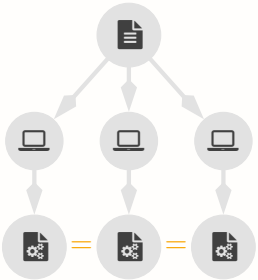
4 Diverse Double Compiling



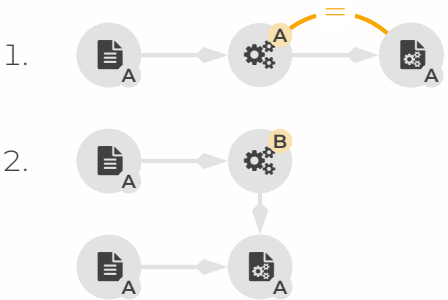
Reproducible



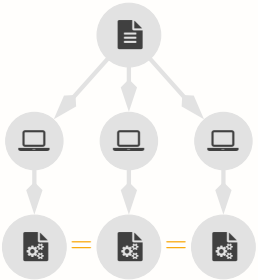
4 Diverse Double Compiling



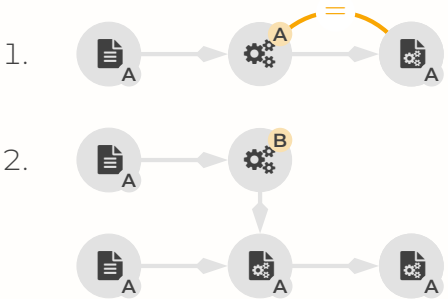
Reproducible



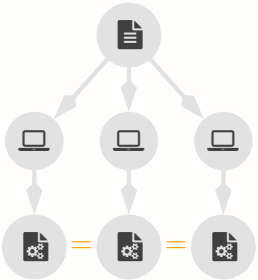
4 Diverse Double Compiling



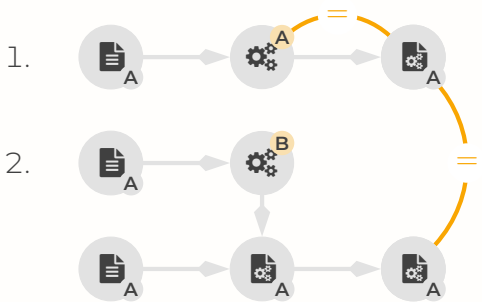
Reproducible



4 Diverse Double Compiling



Reproducible



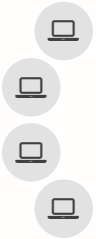
5 CHAINIAC

[4] Nikitin u. a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

5 CHAINIAC

[4] Nikitin u. a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

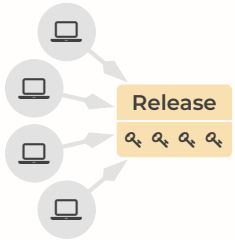
5 CHAINIAC



[4] Nikitin u. a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

5 CHAINIAC

Developers



[4] Nikitin u. a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

5 CHAINIAC

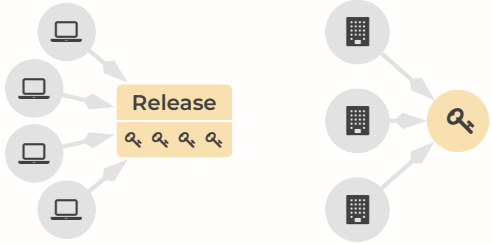
Developers

[4] Nikitin u.a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

5 CHAINIAC

Developers

Cothority

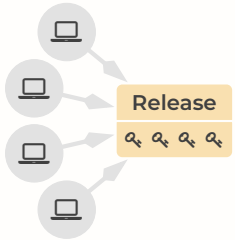


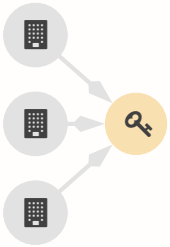
[4] Nikitin u.a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

5 CHAINIAC

Developers

Cothority





[4] Nikitin u.a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

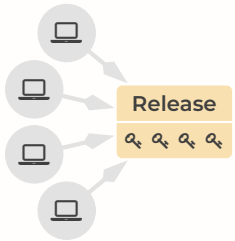
5 CHAINIAC

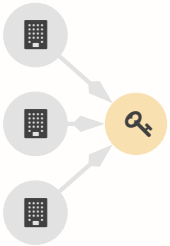
Developers | Cothority | Timeline

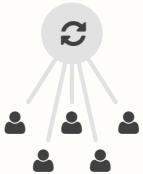
[4] Nikitin u.a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

5 CHAINIAC

Developers | Cothority | Timeline | Mirror



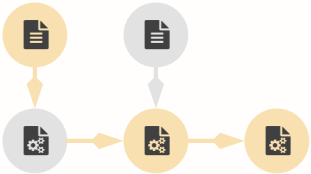




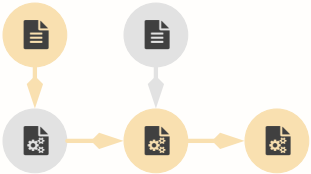
[4] Nikitin u.a.: „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds" 2017

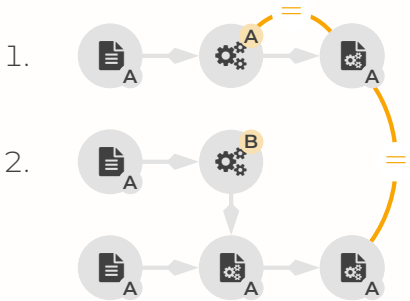
6 Conclusion

6 Conclusion

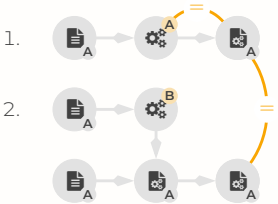


6 Conclusion

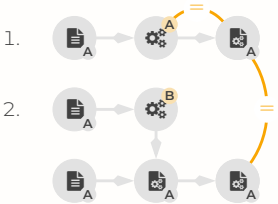


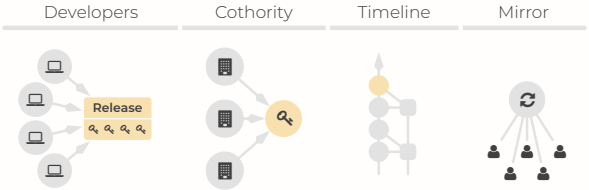


6 Conclusion

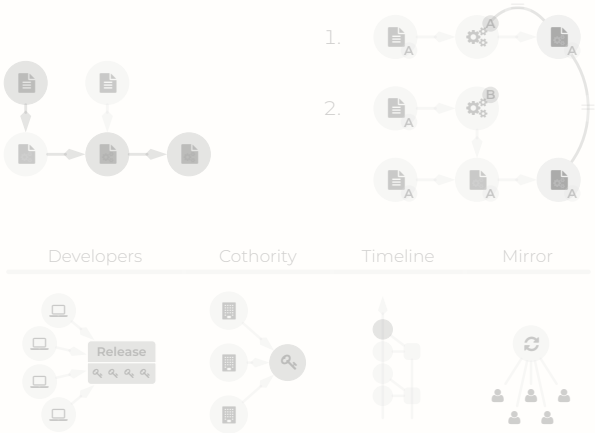


6 Conclusion



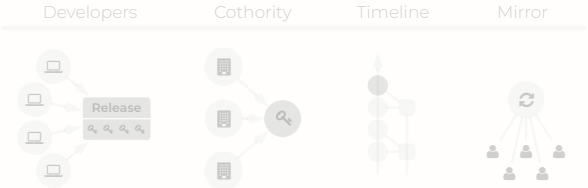


6 Conclusion



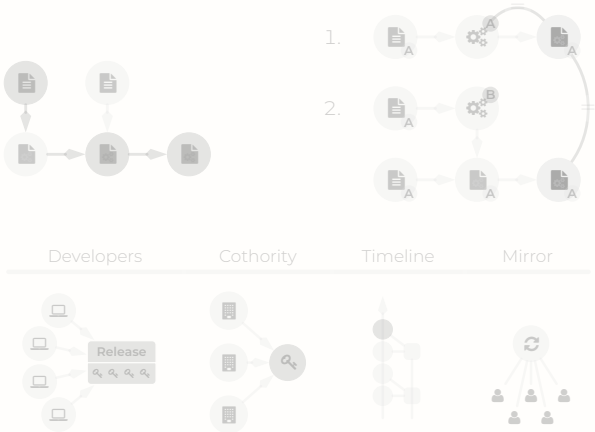
6 Conclusion

› Reproducible Builds



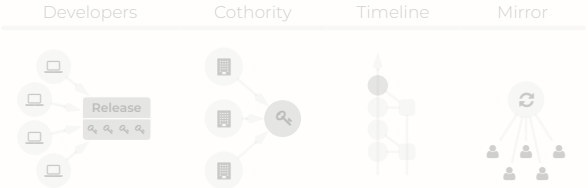
6 Conclusion

- Reproducible Builds
- Tool support



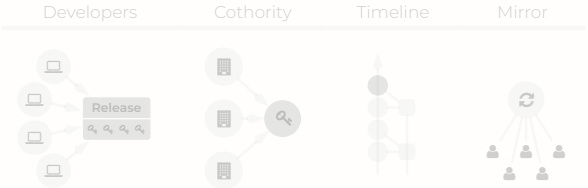
6 Conclusion

- Reproducible Builds
- Tool support
- Trust



6 Conclusion

- Reproducible Builds
- Tool support
- **Trusting Trust**



[1] Dan Gooding. *Apple scrambles after 40 malicious XcodeGhost apps haunt App Store*. Sep. 2015.

[2] Robert Lipovský. „Win32/Induc.C: getting noisier in the library“. Dez. 2011.

[3] William Marshall McKeeman, James J. Horning und David B. Wortman. *A Compiler Generator*. Nov. 1970.

[4] Kirill Nikitin u. a. „CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds“. 2017.

[5] Software Freedom Conservancy. *Reproducible Builds*. Nov. 2020.

[6] Ken Thompson. „Reflections on Trusting Trust“. Aug. 1984.

[7] Olivia von Westernhagen. „Cyberangriffe via SolarWinds-Software – neue Entwicklungen im Überblick“. Dez. 2020.

[8] David A. Wheeler. „Countering trusting trust through diverse double-compiling“. Dez. 2005.

Florian Sihler
Ulm, 13. April 2021
florian.sihler@uni-ulm.de