

VerifyThis

The Long-term Challenge

Marieke Huisman, Raúl Monti,
Mattias Ulbrich, Alexander Weigl

August 12, 2019

Tradition of VerifyThis

- ▶ **VerifyThis** – *“Verification Competition with a Human Factor”*
- ▶ On-site event – *bringing people together, foster discussions*
- ▶ Yearly Workshop at  **ETAPS**
EUROPEAN JOINT CONFERENCES ON
THEORY & PRACTICE OF SOFTWARE
- ▶ Already eight iterations

Peculiarities of VerifyThis!

- ▶ *90 minute slots* – problems must be condensed.
- ▶ *Variety of languages/approaches* – problems must be general
- ▶ *Competitive character* – synergetic effects between approach never identified, let alone exploited

Peculiarities of VerifyThis!

- ▶ *90 minute slots* – problems must be condensed.
- ▶ *Variety of languages/approaches* – problems must be general
- ▶ *Competitive character* – synergetic effects between approach never identified, let alone exploited



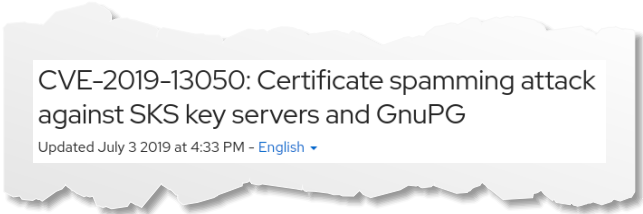
Let's overcome these restrictions

VerifyThis

The Long-term Challenge

- ▶ 6 months time
- ▶ Security / Safety real relevant system
- ▶ Reference implementation, can be reimplemented
- ▶ Requirements given in natural language
- ▶ Various degrees of abstraction possible
- ▶ Collaboration explicitly promoted

Verification Target



CVE-2019-13050: Certificate spamming attack against SKS key servers and GnuPG

Updated July 3 2019 at 4:33 PM - [English](#) ▾

- ▶ Security issues (anyone can upload keys)
- ▶ Denial of service attack (“monster key”)

Solution

The *Verifying* Key Server

Verification Target

CVE-2019-13050: Certificate spamming attack
against SKS key servers and GnuPG

Updated July 3 2019 at 4:33 PM - [English](#) ▾

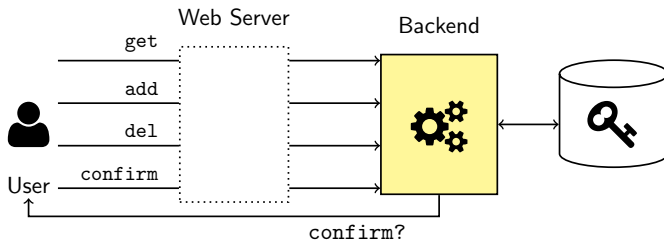
- ▶ Security issues (anyone can upload keys)
- ▶ Denial of service attack (“monster key”)

Solution

The *Verifying* Key Server

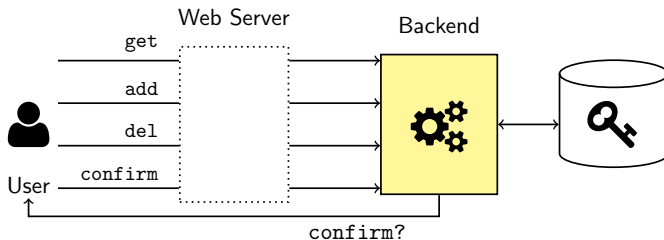
Verification Target

Verification Target: The *Verifying* Key Server



Verification Target

Verification Target: The *Verifying* Key Server



- ▶ Reference Implementation: HAGRID.
- ▶ Deployed as default key server keys.openpgp.org

Challenges (extract)

1. **SAFETY** Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)

Challenges (extract)

1. **SAFETY** Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)
2. **FUNCTIONALITY** Specify and verify that if an e-mail address is queried, the respective key is returned if there is one.

Challenges (extract)

1. **SAFETY** Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)
2. **FUNCTIONALITY** Specify and verify that if an e-mail address is queried, the respective key is returned if there is one.
3. **PRICACY** Specify and verify that if an e-mail address has been deleted from the system, no information about the e-mail address is kept in the server.

Challenges (extract)

1. **SAFETY** Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)
2. **FUNCTIONALITY** Specify and verify that if an e-mail address is queried, the respective key is returned if there is one.
3. **PRICACY** Specify and verify that if an e-mail address has been deleted from the system, no information about the e-mail adress is kept in the server.
4. **THREAD SAFETY** Prove that your implementation is free of data races.

Challenges (extract)

1. **SAFETY** Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)
2. **FUNCTIONALITY** Specify and verify that if an e-mail address is queried, the respective key is returned if there is one.
3. **PRICACY** Specify and verify that if an e-mail address has been deleted from the system, no information about the e-mail adress is kept in the server.
4. **THREAD SAFETY** Prove that your implementation is free of data races.
5. **TERMINATION** Prove that any operation of the server terminates.

Challenges (extract)

1. **SAFETY** Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)
2. **FUNCTIONALITY** Specify and verify that if an e-mail address is queried, the respective key is returned if there is one.
3. **PRICACY** Specify and verify that if an e-mail address has been deleted from the system, no information about the e-mail adress is kept in the server.
4. **THREAD SAFETY** Prove that your implementation is free of data races.
5. **TERMINATION** Prove that any operation of the server terminates.

...

- ▶ Github page is the central place

VerifyThis 2020

Collaborative Long-term Verification Challenge

What is VerifyThis?

The VerifyThis verification competition is a regular event run as

News

Challenge start

Posted on August 12, 2019

<https://verifythis.github.io>

- ▶ Updates and news from other
- ▶ We provide: informal specification in natural language
- ▶ There will be a mailing list
 - ▶ Finding collaboration partners
 - ▶ Share your progress, specification and formalisations

Contribute!

*NOW –
End February 2020*

Challenge active!
Specify! Verify! Collaborate! Submit!

25–26 April 2020

VerifyThis workshop at ETAPS,
presentation of results

After ETAPS

Call for papers for a special issue,
individual solutions and collaboration pa-
pers

Throughout

Advertise! Get the news around!

