A Lingua Franca for Security by Design

Alexander van den Berghe, Koen Yskout, Riccardo Scandariato and Wouter Joosen

IEEE Cybersecurity Development Conference (IEEE SecDev 2018)
01 October 2018
“Improving software security should be an easy sell if your software has a significant number of users; the sheer cost of applying security updates makes it worth getting security, privacy, and reliability right early in the process rather than putting the burden on your customers to apply updates.”

How vicious can a security design flaw be?

Group Policy Remote Code Execution Vulnerability (CVE-2015-0008)\(^1\)

Boils down to the improper use of DNS for authentication

- rated as critical
- impacted multiple Microsoft products
- fix required comprehensive architectural changes (1 year of development)
- older products are not patched due to impact on stability and compatibility

It seems beneficial to start tackling security early on

Requires the ability to create and reason about a high-level security view of the software.
Intermezzo: What is a security view?

Requires the ability to create and reason about a high-level security view of the software.


"architecture view
work product expressing the architecture of a system from the perspective of specific system concerns"

A security view thus expresses only the security-relevant aspects of a system.
It seems beneficial to start tackling security early on.

Requires the ability to create and reason about a high-level security view of the software.

Requires a modelling language "suitable" for security.

What does this mean?
Who uses (security) views and for what?

ISO/IEC/IEEE 42010

- as basis for system design and development activities;
- as input to automated tools for simulation, system generation and analysis;
- communicating among parties involved in the development, production, deployment, operation and maintenance of a system;
- ... 

Our observation is that

- specialised security teams are responsible to assess and harden the security of the designed software; and
- these teams communicate with architects, implementers, managers, ...
A security by design lingua franca needs to reconcile two contradicting forces

Communication to a broader audience

Language that is easily comprehensible

Currently used notations are often ad hoc and informal

Design and analysis of security solutions

Language that is precise and expressive

Proposed languages often go very formal and/or cover few security properties
So we developed our own security modelling language

It essentially consists of data manipulated by processes, which can collaborate by grouping them into networks, each of which can be further refined by assumptions.

Our language is equipped with a graphical notation and is fully formalised using the Coq Proof Assistant.
So we developed our own security modelling language

Built-in types for ciphertexts, cryptographic keys, credentials, ... 

It essentially consists of data manipulated by processes, which can collaborate by grouping them into networks, each of which can be further refined by assumptions
So we developed our own security modelling language

23 pre-defined types (Encrypter, Attacker, Fork, ...)
Each exhibiting well-defined (non-deterministic) behaviour

It essentially consists of data manipulated by processes, which can collaborate by grouping them into networks, each of which can be further refined by assumptions
So we developed our own security modelling language

It essentially consists of data manipulated by processes, which can collaborate by grouping them into networks, each of which can be further refined by assumptions.
So we developed our own security modelling language

It essentially consists of data manipulated by processes, which can collaborate by grouping them into networks, each of which can be further refined by assumptions.

e.g. Attacker cannot obtain original data from a hash value
A security by design lingua franca needs to reconcile two contradicting forces

- Communication to a broader audience
  - Language that is easily comprehensible
- Design and analysis of security solutions
  - Language that is precise and expressive
Evaluating our language with respect to these two forces

<table>
<thead>
<tr>
<th>Comprehension in the large</th>
<th>Creation in the large</th>
<th>Deals with real(istic) designs</th>
</tr>
</thead>
</table>

Deals with building blocks and small designs

105 participants

Created a realistic model of password-based authentication

Security novice, junior software developers
Evaluating our language with respect to these two forces

<table>
<thead>
<tr>
<th>Comprehension in the large</th>
<th>Creation in the large</th>
</tr>
</thead>
</table>

Created a realistic model of password-based authentication

Deals with real(istic) designs
Deals with building blocks and small designs as a prerequisite

Performed a user study with master students in computer science
105 participants
Security novice, junior software developers

Created a realistic model of password-based authentication
Evaluating our language with respect to these two forces

<table>
<thead>
<tr>
<th>Comprehension in the large</th>
<th>Creation in the large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehension in the small</td>
<td>Creation in the small</td>
</tr>
<tr>
<td>Deals with building blocks and small designs as a prerequisite</td>
<td></td>
</tr>
</tbody>
</table>
Evaluating our language with respect to these two forces

<table>
<thead>
<tr>
<th>Comprehension in the large</th>
<th>Creation in the large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehension in the small</td>
<td>Creation in the small</td>
</tr>
</tbody>
</table>

Performed a user study with master students in computer science
Evaluating our language with respect to these two forces

<table>
<thead>
<tr>
<th>Comprehension in the small</th>
<th>Comprehension in the large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation in the small</td>
<td>Creation in the large</td>
</tr>
</tbody>
</table>

- **Comprehension in the large**
  - Deals with realistic designs
  - Deals with building blocks and small designs as a prerequisite

- **Comprehension in the small**
  - Created a realistic model of password-based authentication

- **Creation in the large**
  - Performed a user study with master students in computer science
  - Software architecture course
  - 105 participants
  - Security novice, junior software developers
Research questions

*RQ1.1*: Do participants comprehend the individual building blocks provided by the modelling language?

*RQ1.2*: Do participants comprehend models where multiple building blocks have been tied together?

*RQ1.3*: Do participants comprehend models where security mechanisms and an attacker are intertwined?

*RQ2.1*: Can participants use the language to express an informally described situation?

Not covered in this presentation
Study Design

Tutorial lecture (2h)

Lab session (max. 2.5h)

Data analysis

Background information

Entry survey

Software security introduction
Detailed explanation of language

Exit survey

Individually complete a questionnaire
41 questions (36 multiple choice and 5 open)

Experiences and comments

Entry survey

Exit survey
Multiple choice questions

3. What is valid output data for this Encrypter process?
   - d
   - symk 1
   - enc d (symk 1)
   - enc d (symk 6)
   - Nothing
   - Don’t know

Remarks:

Optional (unless selected “Don’t know”)

Difficulty:
   - Very difficult
   - Rather difficult
   - Rather easy
   - Very easy

Asses perceived difficulty
The data we measured

Scored each answer to multiple choice questions

\[ \text{score} = \max \left( 0, \frac{S^+ - S^-}{N^+} \right) \times 100\% \]

Correctly selected options \( S^+ \), Incorrectly selected options \( S^- \)

Total number of correct options \( N^+ \)

Rationale is manually analysed and coded

A code can, for example, indicate the presence of a common error
### Resulting scores

<table>
<thead>
<tr>
<th>RQ</th>
<th>Topic</th>
<th>Score (%)</th>
<th>Min.</th>
<th>Mean</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1.1</td>
<td>Individual building blocks</td>
<td>41</td>
<td>87</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>RQ1.2</td>
<td>Combined building blocks</td>
<td>33</td>
<td>79</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>RQ1.3</td>
<td>Security aspects</td>
<td>50</td>
<td>77</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>RQ1</td>
<td></td>
<td>47</td>
<td>83</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>

- **Box plot**
  - RQ1.1: Min. 41, Mean 87, Max. 100
  - RQ1.2: Min. 33, Mean 79, Max. 100
  - RQ1.3: Min. 50, Mean 77, Max. 100
  - RQ1: Min. 47, Mean 83, Max. 97
What were the recurring problems?

Non-determinism

Attacker omnipotence

(Complex processes)

(Data equality)
Non-deterministically processing inputs

Roughly 50% of the participants prefer processes to deal with multiple inputs in a certain order.

Significant majority of these participants prefer that “configuration” precedes “functionality”.

Among the participants
- 40% do not prefer order (correct answer)
- 48% configure key before encrypting
- 5% encrypt before configuring key
The challenge of an omnipotent attacker

Our Attacker process can guess any data as well as derive any data from other data it already knows, unless explicitly constrained using assumptions.

What data can be obtained by a given, unconstrained Attacker?

Attacker
{plain 2 5, cred 6, id 5, sid 9, hashed (cred 1), enc (cred 1) (symk 3)}

Replies by 36% of the participants indicate they implicitly constrain the attacker’s guessing and derivation abilities.
Such implicit assumptions are problematic

They remain unchecked with respect to the software under design

Potentially allowing possible attacks to go unnoticed
So what have we learned from this?

We now know what the shortcomings are

Building blocks and small models seem fairly easy to comprehend given limited training (2h) without going into the formal specification

Underlying formal machinery does seem necessary to cover all possibilities e.g. make assumptions on attacker’s abilities explicit
Evaluating our language with respect to these two forces

Comprehension in the large

Creation in the large

Created a realistic model of password-based authentication
What we modelled

Modelled features
   Username-password authentication
   User registration
   Two-factor authentication
   Sessions
   Change password
   Reset password

Followed OWASP’s best practices

Resulted in a large, complex model containing 200+ processes
I won’t dive into it here, excerpts can be found in paper and full model is available online
Experience

❌ Model quickly grows in size and complexity
❌ Defining a comprehensive set of assumptions is laborious
❌ Some processes get cumbersome to work with
❌ Expressing time-based aspects is clumsy

✔️ Pre-defined building blocks to fall back on
✔️ Existing set of process types was sufficient
✔️ Revealed some possibilities we did not think of up-front
e.g. use of password reset to reactivate an account
Wrapping up
It is advantageous to tackle security issues early on in the software development cycle.

This requires a suitable security modelling language:

- Comprehensible by a broad audience
- Allows security experts to design and analyse security solutions
We evaluated our proposed modelling language with respect to these forces

At least to some extend

User study to assess comprehensibility of the building blocks and small models for security novice, junior software developers

Modelled a realistic version of username-password based authentication to assess expressivity

Results are promising and have shown us what the current shortcomings are
A Lingua Franca for Security by Design

Alexander van den Berghe, Koen Yskout, Riccardo Scandariato and Wouter Joosen

IEEE Cybersecurity Development Conference (IEEE SecDev 2018)
01 October 2018