Secure Compilation as Hyperproperties Preservation

Marco Patrignani[1] Deepak Garg[1]

[1]MPI-SWS, Saarbrücken, Germany
first.last@mpi-sws.org

June 2016

Goal of the Talk

1. identify failures of full abstraction for security
2. present TPC, a new notion of secure compilation
3. understand the security relevance of TPC
4. relate TPC and other secure compilation definitions

Background

- setting: reactive language
- any behaviour is described by traces ($\mathsf{TR}(\cdot)$ and $\mathsf{TR}(\cdot)$)
- traces are sequences of input-output actions $\alpha?\alpha!\cdots$
- no nondeterminism, no concurrency

Full-abstraction Failure #1: Input Validation

- source has Bools
- source programs are ids: $\lambda x.x$, $\lambda x.x \vee \texttt{false}$, $\lambda x.x \wedge \texttt{true}$ ...
  - $\mathit{id}(\texttt{true})? \cdot \texttt{ret}(\texttt{true})!$
  - $\mathit{id}(\texttt{false})? \cdot \texttt{ret}(\texttt{false})!$
- target has Nats
- $[\![\texttt{true}]\!]^{\mathcal{S}}_{\mathcal{T}} = \texttt{1}$ and $[\![\texttt{false}]\!]^{\mathcal{S}}_{\mathcal{T}} = \texttt{0}$
- $[\![\cdot]\!]^{\mathcal{S}}_{\mathcal{T}}$ generates $\lambda x.x$

Full-abstraction Failure #1: Input Validation

- source has Bools
- source programs are ids: $\lambda x.x$, $\lambda x.x \vee \texttt{false}$, $\lambda x.x \wedge \texttt{true}$ ...
  - $\mathit{id}(\texttt{true})? \cdot \texttt{ret}(\texttt{true})!$
  - $\mathit{id}(\texttt{false})? \cdot \texttt{ret}(\texttt{false})!$
- target has Nats
- $[\![\texttt{true}]\!]^{\mathcal{S}}_{\mathcal{T}} = \texttt{1}$ and $[\![\texttt{false}]\!]^{\mathcal{S}}_{\mathcal{T}} = \texttt{0}$
- $[\![\,\cdot\,]\!]^{\mathcal{S}}_{\mathcal{T}}$ generates $\lambda x.x$
- Property: "output booleans only" (@ source)
- "output 1 or 0 only" (@ target)

Problems

1. the property is not respected by the compiler

Problems

1. the property is not respected by the compiler
2. how does one translate properties cross language preserving the meaning?

Full-abstraction Failure #2: Declassification

- same languages
- Property: "Do not output the secret until the 10th input"

Full-abstraction Failure #2: Declassification

- same languages
- Property: "Do not output the secret until the 10th input"
- $\overbrace{\mathit{id}(\mathtt{true})? \cdot \mathtt{ret}(\mathtt{true})!}^{\mathit{nine\ \ times}} \cdot \mathit{id}(\cdot)? \cdot \mathtt{ret}(\mathit{secret})! \cdots$

Full-abstraction Failure #2: Declassification

- same languages
- Property: "Do not output the secret until the 10th input"
- $\overbrace{\mathit{id}(\mathtt{true})? \cdot \mathtt{ret}(\mathtt{true})!}^{\textit{nine times}} \cdot \mathit{id}(\cdot)? \cdot \mathtt{ret}(\mathit{secret})! \cdots$
- $\overbrace{\mathit{id}(1)? \cdot \mathtt{ret}(1)!}^{\textbf{less than } \textit{nine times}} \cdot \mathit{id}(2)? \cdot \mathtt{ret}(\mathit{secret})! \cdots$

Trace-Preserving Compilation, Informally

- keep all source-level behaviour

Trace-Preserving Compilation, Informally

- keep all source-level behaviour
- respond to invalid actions in a fresh way:

Trace-Preserving Compilation, Informally

- keep all source-level behaviour
- respond to invalid actions in a fresh way:
- invalid = not related to a source action

Trace-Preserving Compilation, Informally

- keep all source-level behaviour
- respond to invalid actions in a fresh way:

- invalid = not related to a source action
- fresh = add a target-level symbol ($\surd$) like a *visible tau*:

Trace-Preserving Compilation, Informally

- keep all source-level behaviour
- respond to invalid actions in a fresh way:

- invalid = not related to a source action
- fresh = add a target-level symbol ($\surd$) like a *visible tau*:
  1. *opaque*: reveals nothing of the internal state

Trace-Preserving Compilation, Informally

- keep all source-level behaviour
- respond to invalid actions in a fresh way:

- invalid = not related to a source action
- fresh = add a target-level symbol ($\surd$) like a *visible tau*:
    1. *opaque*: reveals nothing of the internal state
    2. *transparent*: does not alter valid program behaviour

Trace-Preserving Compilation, Formally

Informal definition (Trace-preserving compiler)

$\mathsf{TR}(\llbracket C \rrbracket^{\mathcal{S}}_{\mathcal{T}}) = \mathsf{TR}(C) \cup \mathcal{B}_C$.

Trace-Preserving Compilation, Formally

Informal definition (Trace-preserving compiler)

$\mathsf{TR}(\llbracket C \rrbracket^{\mathcal{S}}_{\mathcal{T}}) = \mathsf{TR}(C) \cup \mathcal{B}_C$.

Definition (Invalid traces)

$\mathcal{B}_C \stackrel{\text{def}}{=} \{\overline{\alpha}\alpha?\surd \mid \exists\overline{\alpha} \in \mathsf{TR}(C).\overline{\alpha} \approx \overline{\alpha} \land \nexists\alpha? \in \mathsf{TR}(C).\alpha? \approx \alpha?\}$

Invalid Traces

- $\surd$ can be implemented in various forms:
  1. halt
  2. diverge
  3. ignore
- right now it was mostly halting

TPC Security

Why is TPC secure?

TPC Security

Why is TPC secure?

because it preserves (some) hyperproperties

Hyperproperties (HP)

- HP formalise any program property

Hyperproperties (HP)

- HP formalise any program property
- they are sets of sets of traces

Hyperproperties (HP)

- HP formalise any program property
- they are sets of sets of traces
- they capture security properties including safety, liveness and non interference in all of its forms

Hyperproperties Preservation

- why are source HP meaningful at the target?

Hyperproperties Preservation

- why are source HP meaningful at the target?
- *Challenge*: how to describe the "same idea" of a source property in the target language?

Hyperproperties Preservation

- why are source HP meaningful at the target?
- *Challenge*: how to describe the "same idea" of a source property in the target language?
- assume a relation between source and target actions ($\approx$)
  1. all that is related is ok
  2. target actions that are not related are invalid
  3. unless they're $\surd$, in which case they're ok

Safety Preservation

- Standard definition of Safety:
- $\forall\overline{\alpha},\ \text{if } \overline{\alpha} \notin S \text{ then } (\exists\overline{m} \le \overline{\alpha} \text{ and } \forall\overline{\alpha'} \text{ if } \overline{m} \le \overline{\alpha'} \text{ then } \overline{\alpha'} \notin S)$

Safety Preservation

- Standard definition of Safety:
- $\forall \overline{\alpha}$, if $\overline{\alpha} \notin S$ then $(\exists \overline{m} \leq \overline{\alpha}$ and $\forall \overline{\alpha'}$ if $\overline{m} \leq \overline{\alpha'}$ then $\overline{\alpha'} \notin S)$
- Equivalent, alternative definition:
- if $\widehat{\overline{m}} :: S$ then $\overline{\alpha} \notin S$ iff $\exists \overline{m} \in \widehat{\overline{m}}. \overline{m} \leq \overline{\alpha}$

Safety Preservation

- Standard definition of Safety:
- $\forall\overline{\alpha}$, if $\overline{\alpha} \notin S$ then $(\exists\overline{m} \leq \overline{\alpha}$ and $\forall\overline{\alpha'}$ if $\overline{m} \leq \overline{\alpha'}$ then $\overline{\alpha'} \notin S)$
- Equivalent, alternative definition:
- if $\widehat{\overline{m}} :: S$ then $\overline{\alpha} \notin S$ iff $\exists\overline{m} \in \widehat{\overline{m}}.\overline{m} \leq \overline{\alpha}$
- given a source safety property
- add all invalid traces to the set of bad prefixes
- and obtain its target-level equivalent

Safety Preservation Theorem

Theorem (Safety preservation)

- $\forall S, \widehat{\overline{m}}.\ S :: \widehat{\overline{m}}, \forall \overline{\alpha}.\ \textit{if } \overline{\alpha} \notin S \textit{ then } \exists \overline{m} \in \widehat{\overline{m}}. \overline{m} \leq \overline{\alpha}$
- $\forall S, \widehat{\overline{m}}.\ S :: \widehat{\overline{m}}, \forall \overline{\alpha}.\ \textit{if } \overline{\alpha} \notin S \textit{ then } \exists \overline{m} \in \widehat{\overline{m}}. \overline{m} \leq \overline{\alpha}$

$\forall C.\ \textit{if } \widehat{\overline{m}} \approx \widehat{\overline{m}} \textit{ and } [\![\cdot]\!]^{\mathcal{S}}_{\mathcal{T}} \textit{ is TPC and } \mathrm{TR}(C) = S \textit{ then}$
$\mathrm{TR}([\![C]\!]^{\mathcal{S}}_{\mathcal{T}}) = S.$

Where $\widehat{\overline{m}} \approx \widehat{\overline{m}}$ *is defined as:*

$$\widehat{\overline{m}} = \{\overline{\alpha} \mid \exists \overline{\alpha} \in \widehat{\overline{m}}, \overline{\alpha} \approx \overline{\alpha}\} \cup$$
$$\{\overline{\alpha}\alpha?\alpha! \mid \exists \overline{\alpha} \approx \overline{\alpha} \textit{ and } \nexists \alpha? \approx \alpha? \textit{ and } \alpha! \neq \surd\}$$

Hypersafety Preservation

- generalise the previous idea: capture all possible systems that are invalid
- add a set of uni-sets of traces, each with a possible bad trace (invalid action - tick)

Limitations

- liveness / hyperliveness / arbitrary HP cannot be preserved

Limitations

- liveness / hyperliveness / arbitrary HP cannot be preserved
- what does it mean to preserve a generic HP?

TPC and Existing Secure Compilation Statements

- TPC $\Rightarrow$ FAC
- FAC $\not\Rightarrow$ TPC

TPC and Existing Secure Compilation Statements

- TPC $\Rightarrow$ FAC
- FAC $\not\Rightarrow$ TPC
- TPC $\Rightarrow$ NIPC
- NIPC $\not\Rightarrow$ TPC

TPC and Existing Secure Compilation Statements

- TPC $\Rightarrow$ FAC
- FAC $\not\Rightarrow$ TPC
- TPC $\Rightarrow$ NIPC
- NIPC $\not\Rightarrow$ TPC   TPC implies TSNI, NIPC achieves TINI

TPC and Existing Secure Compilation Statements

- TPC $\Rightarrow$ FAC
- FAC $\not\Rightarrow$ TPC
- TPC $\Rightarrow$ NIPC
- NIPC $\not\Rightarrow$ TPC   TPC implies TSNI, NIPC achieves TINI
- FAC $\Rightarrow$ SCC (with full definedness and the notion of compartment interfaces)

Questions

Thank you!

Qs ?

Hypersafety Preservation Theorem

Theorem (Hypersafety preservation)

- $\forall \mathfrak{S}, \mathfrak{M}.\ \mathfrak{S} :: \mathfrak{M}, \forall \widehat{\overline{\alpha}}.$ *if* $\widehat{\overline{\alpha}} \notin \mathfrak{S}$ *then* $\exists \widehat{\overline{m}} \in \mathfrak{M}. \widehat{\overline{m}} \leq \widehat{\overline{\alpha}}$
- $\forall \mathfrak{S}, \mathfrak{M}.\ \mathfrak{S} :: \mathfrak{M}, \forall \widehat{\overline{\alpha}}.$ *if* $\widehat{\overline{\alpha}} \notin \mathfrak{S}$ *then* $\exists \widehat{\overline{m}} \in \mathfrak{M}. \widehat{\overline{m}} \leq \widehat{\overline{\alpha}}$

$\forall C.$ *if* $\mathfrak{M} \approx \mathfrak{M}$ *and* $[\![\cdot]\!]^{\mathcal{S}}_{\mathcal{T}} \in TP^{\mathcal{P}}$ *and* $\mathrm{TR}(C) \in \mathfrak{S}$ *then* $\mathrm{TR}([\![C]\!]^{\mathcal{S}}_{\mathcal{T}}) \in \mathfrak{S}$.
Where $\mathfrak{M} \approx \mathfrak{M}$ *is defined as:*

$$\mathfrak{M} = \{\widehat{\overline{\alpha}} \mid \exists \widehat{\overline{\alpha}} \in \mathfrak{M}, \widehat{\overline{\alpha}} \approx \widehat{\overline{\alpha}}\} \cup$$
$$\{\{\overline{\alpha}\alpha?\alpha!\} \mid \exists \overline{\alpha} \approx \overline{\alpha} \text{ and } \nexists \alpha? \approx \alpha? \text{ and } \alpha! \neq \surd\}$$

And $\widehat{\overline{\alpha}} \approx \widehat{\overline{\alpha}}$ *is defined as:*

$$\forall \overline{\alpha} \in \widehat{\overline{\alpha}}, \exists \overline{\alpha} \in \widehat{\overline{\alpha}}. \overline{\alpha} \approx \overline{\alpha} \text{ and } \forall \overline{\alpha} \in \widehat{\overline{\alpha}}, \exists \overline{\alpha} \in \widehat{\overline{\alpha}}. \overline{\alpha} \approx \overline{\alpha}$$