Language-Based Technology for Security - 28, May 2021

In this homework, we will reason about Control Flow Analysis and about its application to distributed systems and IoT scenarios and, in particular, to the verification of security properties.

Learning objectives Goals of the homework are

- to appreciate the main ideas of CFA applied to process algebras;
- to understand the flexibility of this static technique in verifying different properties by using the same core analysis of the behaviour of a system;
- to reason about possible variations with respect to the discussed framework and properties.

The assignment In this homework, you will first answer to some questions on CFA, and afterwards you will be asked to reason about new security properties in the IoT scenario.

1 Questions

CFA and its application to Pi Calculus

1. What kind of information the components ρ e κ in the CFA applied to the pi calculus collect? (Che cosa raccolgono le componenti ρ e κ nella CFA vista del Pi calcolo?)
2. $$P \mid Q \mid R = (\underbrace{\overline{a}d.P' + \overline{a}b.P''}_{P} \mid R \mid \underbrace{a(w).\overline{w}c.Q'}_{Q})$$
   - Which could be the result (in terms of the ρ e κ components) of the analysis in the above example? (Quale potrebbe essere il risultato dell'analisi nell'esempio sopra nelle componenti ρ e κ?)
   - Assuming that $SecretNames = \{a, d\}$ and that $PublicNames = \{b, c\}$. What hints the CFA would give us on the discussed secrecy property? (Se $SecretNames = \{a, d\}$ e $PublicNames = \{b, c\}$. Cosa ci dice l'analisi della proprietà di segretezza discussa a lezione?)

CFA and its application to LySa and IoT-LySa

1. Which is the aim of the component Ψ? (A cosa serve la componente Ψ?)
2. Which are the main differences between the CFA estimates of LySa and those of IoT LySa? (Quali sono le maggiori differenze tra le soluzioni CFA di LySa e quelle di IoT LySa?)
3. What *abstract values* allow us to track? (Che cosa ci permettono di tracciare i *valori astratti*?)

2 Reason about security properties in the IoT scenario

Starting from the version of IoT LySa with encryption and decryption, you can choose to work on one of the following lines.

2.1 Taint and Resilience Analysis a là Quality

Taking into account the CFA presented in *Bodei, Galletta. Tracking sensitive and untrustworthy data in IoT. ITA-SEC 2017* (see Material) illustrated in one of our lectures, and the following considerations, try to reason about how we can statically evaluate the impact of a certain amount of tamperable data, e.g. by answering to question like: how many sensors can be compromised without dramatically impacting on the overall behaviour of the system? Try to reason about way of combining quality and taint information.

Taint information To formally introduce our abstract values and operators through which we can combine and manipulate them, we first define the set of *taint labels* $\mathcal{B}$ (ranged over by $b, b_1, ...$) whose elements (the colours in the pdf should help the intuition) are:

$\diamond$ untainted $\quad$ $\diamond$ sensitive $\quad$ $\diamond$ tamperable $\quad$ $\diamond$ sensitive & tamperable

The idea is that these labels mark our abstract values with information about their sources. Formally, abstract values are pairs in $\hat{\mathcal{V}}$ defined as follows, where $b \in \mathcal{B}$.

$\hat{\mathcal{V}} \ni \hat{v} ::=$	*abstract terms*	
	$(\top, b)$	abstract value denoting cut (see below)
	(ν, b)	abstract value for clear data
	$(f(\hat{v}_1, \cdots, \hat{v}_n), b)$	abstract value for aggregated data
	$(\{\hat{v}_1, \cdots, \hat{v}_n\}_{k_0}, b)$	abstract value for encrypted data

Taint information propagate with aggregation functions. We can define simple propagation policies for function applications and encryptions, e.g., in the case of function application, imposing that a single tainted argument turns to tainted the result of the application. Moreover, encryption protects its sensitive data, but preserves tamperable taint information.

It is obviously possible to consider different policies for propagation, e.g. we could consider a set of functions that produce sensitive data independently from the taint information of its arguments. Moreover, we could deal with anonymisation functions that process their arguments to remove sensitive information. Consider e.g. blurring the faces of people in surveillance videos to protect their privacy. A policy for these functions is similar to the one for the encryption: the resulting taint label is $\diamond$, if no argument is also tamperable, otherwise is $\diamond$.

Finally, we could extend the whole presented schema to record also the source nodes of tainted information. For more details, see Appendix.

Resilience Analysis a là Quality The above taint analysis can be further pushed in order to consider resilience issues, by answering to question like "Given an aggregation function, how many values must be correct for giving the right answer?" The aim is to ensure a certain level of reliability of actuations even in the presence of unreliable data. For instance, consider the check whether the given temperature average, used to trigger the fire suppression system, is greater than a certain threshold, and suppose that the temperature sensors in the room now are eight.

$$gt(avg(1_{E_{11}}, 1_{E_{21}}, 1_{E_{31}}, 1_{E_{41}}, 1_{E_{51}}, 1_{E_{61}}, 1_{E_{71}}, 1_{E_{81}}), T_{RT})$$

A possible detailed question here may be: "What if $1_{E_{11}}$ and $1_{E_{61}}$, give the wrong results?" and "Does the overall result amortise these weaknesses?" Maybe it does, but "what does it happen if additionally, also $1_{E_{41}}$ fails? " In other words, the overall result can also be altered because of more local conditions. To handle these issues, we can embed quality logical predicates, in the style of *H. Riis Nielson, F. Nielson, R. Vigo. A calculus of quality for robustness against unreliable communication. Journal of Logical and Algebraic Methods in Programming Volume 84, Issue 5, September 2015, Pages 611-639* (see Material) inside our abstract terms:

$$\&_\exists(\hat{v}_1, \cdots, \hat{v}_r) \text{ and } \&_\forall(\hat{v}_1, \cdots, \hat{v}_r)$$

where $\&_\exists$ means that it suffices to have at least one correct value, and $\&_\forall$ means that all the values are necessary. More complex nested logical structures like $f(\&_\forall(\hat{v}_1, \&_\exists(\hat{v}_2, \hat{v}_3))$ are possible, as well. This allows us to encode the dependence of our actuation on a suitable combination of reliable data. For example, the following formula expresses that the reliable data are $1_{E_{11}}$, $1_{E_{21}}$, $1_{E_{51}}$, $1_{E_{61}}$, one between $1_{E_{31}}$ and $1_{E_{41}}$, and one between $1_{E_{71}}$ and $1_{E_{81}}$:

$$gt(avg(\&_\forall(1_{E_{11}}, 1_{E_{21}}, \&_\exists(1_{E_{31}}, 1_{E_{41}}), 1_{E_{51}}, 1_{E_{61}}, \&_\exists(1_{E_{71}}, 1_{E_{81}}))), T_{RT})$$

Note that this technique can be used starting from the estimates of our CFA as they are, because certain quality predicates can be applied directly to the resulting abstract values in order to reason about resilience issues.

$\otimes$	$\diamond$	$\diamond_H$	$\diamond_T$	$\diamond_{HT}$
$\diamond$	$\diamond$	$\diamond_H$	$\diamond_T$	$\diamond_{HT}$
$\diamond_H$	$\diamond_H$	$\diamond_H$	$\diamond_{HT}$	$\diamond_{HT}$
$\diamond_T$	$\diamond_T$	$\diamond_{HT}$	$\diamond_T$	$\diamond_{HT}$
$\diamond_{HT}$	$\diamond_{HT}$	$\diamond_{HT}$	$\diamond_{HT}$	$\diamond_{HT}$

$\otimes$	$\diamond$	$\diamond_{H,L}$	$\diamond_{T,L}$	$\diamond_{HT,L}$
$\diamond$	$\diamond_L$	$\diamond_{H,L}$	$\diamond_{T,L}$	$\diamond_{HT,L}$
$\diamond_{H,L'}$	$\diamond_{H,L'}$	$\diamond_{H,L\cap L'}$	$\diamond_{HT,L\cap L'}$	$\diamond_{HT,L\cap L'}$
$\diamond_{T,L'}$	$\diamond_{T,L'}$	$\diamond_{HT,L\cap L'}$	$\diamond_{T,L\cap L'}$	$\diamond_{HT,L\cap L'}$
$\diamond_{HT,L'}$	$\diamond_{HT,L'}$	$\diamond_{HT,L\cap L'}$	$\diamond_{HT,L\cap L'}$	$\diamond_{HT,L\cap L'}$

Table 1: The operator $\otimes$, version 1 (on the left) and version 2 (on the right)

2.2 Tracking trajectories

Try to reason about a way of extending the main CFA for IoT Lysa to track the trajectories of data, seen through the flows of abstract data in the CFA,

$\hat{v}$ ::=	i^{ℓ}	data from sensor i from the node ℓ
	v^{ℓ}	constant in node ℓ
	$f^{\ell}(\hat{v}_1, \cdots, \hat{v}_n)$	function on abstract data in node ℓ
	$\{\hat{v}_1, \cdots, \hat{v}_n\}^{\ell}_{k_0}$	encryption on abstract data
	$\top^{\ell}$	term of depth greater than d

Trajectories can be obtained, starting from a set of micro-trajectories (describing single hops) and by suitably composing them in order. In turn possibly micro-trajectories may be derived from investigating the κ component. For instance, from $\kappa(\ell_a) \ni (\ell_{cp}, \langle\!\langle noiseRed^{\ell_{cp}}(1^{\ell_{cp}})\rangle\!\rangle)$ we can derive the micro-trajectory from to N_{cp} to N_a of the abstract data represented by $noiseRed^{\ell_{cp}}(1^{\ell_{cp}})$.

This extension can allow us to state new security properties, such as ensuring that some (aggregated) data pass through an untrusted node before contributing to take a critical decision. The notion of trajectories can be extended also by associating a score to each node with label, representing some quantitative and logical information. Trajectories can be therefore compared on the basis of their overall score.

Appendix

More details on taint analysis

For simplicity, hereafter we write them as $\top^b, \nu^b, \{\hat{v}_1, \cdots, \hat{v}_n\}^b_{k_0}$, and indicate with $\downarrow_i$ the projection function on the i^{th} component of the pair representing an abstract value. We naturally extend the projection to sets, i.e. $\hat{V}_{\downarrow_i} = \{\hat{v}_{\downarrow_i} | \hat{v} \in \hat{V}\}$, where $\hat{V} \subseteq \hat{\mathcal{V}}$. In the abstract value ν^b, ν abstracts the concrete value from sensors or computed by a function in the concrete semantics, while the first value of the pair $\{\hat{v}_1, \cdots, \hat{v}_n\}^b_{k_0}$ abstracts encrypted data. Since the dynamic semantics may introduce encrypted terms with an arbitrarily nesting level, we have the special abstract values $\top^b$ that denote all the terms with a depth greater than a given threshold d.

Definition 2.1 (Data classification)**.** *Given the set of* sensitive sensors $\mathcal{S}_\ell$*, and the set of the* tamperable sensors and variables $\mathcal{T}_\ell$*, the* taint assignment function τ *is defined as follows:*

$$\tau(y,\ell) = \begin{cases} \diamond_H & \text{if } y \in \mathcal{S}_\ell \\ \diamond_T & \text{if } y \in \mathcal{T}_\ell \\ \diamond_{HT} & \text{if } y \in \mathcal{S}_\ell \cap \mathcal{T}_\ell \\ \diamond & o.w. \end{cases} \qquad \text{where } y \in \mathcal{I}_\ell \cup \mathcal{X} \qquad \tau(v,\ell) = \diamond$$

The above function classifies only the data source; to propagate the taint information we resort to the *taint combination operator* $\otimes : \mathcal{B}\times\mathcal{B} \to \mathcal{B}$ defined in Table 1, on the left part. Note that $\otimes$ works as a join operator making our abstract values a lattice similar to the 4-point lattice LL,LH,HL,HH, used in the information flow literature, which combines confidentiality and integrity. This operator naturally extends to abstract values: $b \otimes \hat{v} = \hat{v}_{\downarrow_1}^{b'}$ where $\hat{v} \in \hat{\mathcal{V}}$ and $b' = b \otimes \hat{v}_{\downarrow_2}$; and to sets of abstract values: $b \otimes \hat{V} = \{b \otimes \hat{v} | \hat{v} \in \hat{V} \subseteq \hat{\mathcal{V}}\}$.

Now we need to specify how taint information propagate with aggregation functions. Below we define two simple propagation policies for function applications and encryptions. In the case of function application, the idea is that a single tainted argument turns to tainted the result of the application. Encryption protects its sensitive data, but preserves tamperable taint information. This means that encryption as a structure is marked with label $\diamond$, unless one of its component is marked with $\diamond_T$ or $\diamond_{HT}$; in these cases, encryption is labelled with $\diamond_T$.

Definition 2.2 (Taint propagation policies)**.** *Given the combination operator* $\otimes : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$*, the taint resulting by the application of*

- *a function* f *is* $F_\tau(f, \hat{v}_1, \ldots, \hat{v}_r) = \otimes(\hat{v}_{1\downarrow_2}, \ldots, \hat{v}_{r\downarrow_2})$
- *an encryption function is* $Enc_\tau(\hat{v}_1, \ldots, \hat{v}_r) = \begin{cases} \diamond & \text{if } \forall i.\hat{v}_{i\downarrow_2} \in \{\diamond, \diamondsuit\} \\ \diamondsuit & o.w. \end{cases}$

It is obviously possible to consider different policies for propagation, e.g. we could consider a set of functions that produce sensitive data independently from the taint information of its arguments. Moreover, we could deal with anonymisation functions that process their arguments to remove sensitive information. Consider e.g. blurring the faces of people in surveillance videos to protect their privacy. A policy for these functions is similar to the one for the encryption: the resulting taint label is $\diamond$, if no argument is also tamperable, otherwise is $\diamondsuit$.

Finally, we could extend the whole presented schema to keep also the source nodes of tainted information. To do that, we could introduce in our abstract values also the labels of nodes from which the taint information derives. To deal with this new information, the taint assignment function τ returns pairs (b, L), written b_L (b_ℓ when $L = \{\ell\}$), our combinator operator becomes the one shown on the right part in Table 1, and the encryption operator takes ℓ as argument and uses it to annotate the resulting taint label.

CFA We specify our CFA in a logical form through a set of inference rules expressing the validity of the analysis results. The analysis result is a triple $(\hat{\Sigma}, \kappa, \Theta)$ (a pair $(\hat{\Sigma}, \Theta)$ when analysing a term), called *estimate* for N (for E), where $\hat{\Sigma}$, κ, and Θ are the following *abstract domains*:

- the union $\hat{\Sigma} = \bigcup_{\ell \in \mathcal{L}} \hat{\Sigma}_\ell$ of the super-sets $\hat{\Sigma}_\ell : \mathcal{X} \cup \mathcal{I}_\ell \to 2^{\widehat{\mathcal{V}}}$ of abstract values that may possibly be associated to a given location in $\mathcal{I}_\ell$ or a given variable in $\mathcal{X}$,
- a super-set $\kappa : \mathcal{L} \to \mathcal{L} \times \bigcup_{i=1}^{k} \widehat{\mathcal{V}}^i$ of the messages that may be received by the node ℓ, and
- a super-set $\Theta : \mathcal{L} \to \mathcal{A} \to 2^{\widehat{\mathcal{V}}}$ of the taint information of the actual values computed by each labelled term M^a in a given node ℓ, at run time.

Here, we show some examples of CFA clauses. The judgements for labelled terms have the form $(\hat{\Sigma}, \Theta) \vDash_\ell M^a$. For each term M^a occurring in the node ℓ, the corresponding judgement requires that $\Theta(\ell)(a)$ includes all the abstract values $\hat{v}$ associated to M^a. Consider, e.g. the following clause for the variable x^a.

$$\frac{\tau(x, \ell) \otimes \hat{\Sigma}_\ell(x) \subseteq \Theta(\ell)(a)}{(\hat{\Sigma}, \Theta) \vDash_\ell x^a}$$

In this case, an estimate is valid if $\Theta(\ell)(a)$ includes the abstract values resulting from the combination (through the operator $\otimes$) of the taint information associated to the variable x (via $\tau(x, \ell)$), and the abstract values bound to x in $\hat{\Sigma}_\ell$. This combination allows us to propagate the tamperable taint if x belongs to the set of tamperable variables.

The judgements for nodes have the form $(\hat{\Sigma}, \kappa, \Theta) \vDash N$. The rule for a single node $\ell : [B]$ requires that its internal components B are analysed with judgements $(\hat{\Sigma}, \kappa, \Theta) \vDash_\ell B$. As examples of clauses, we consider the clauses for communication.

$$\frac{\begin{array}{c} \bigwedge_{i=1}^{k} (\hat{\Sigma}, \Theta) \vDash_\ell M_i^{a_i} \ \wedge \ (\hat{\Sigma}, \kappa, \Theta) \vDash_\ell P \ \wedge \\ \forall \hat{v}_1, \cdots, \hat{v}_r : \ \bigwedge_{i=1}^{r} \hat{v}_i \in \Theta(\ell)(a_i) \ \Rightarrow \forall \ell' \in L : (\ell, \langle\!\langle \hat{v}_1, \cdots, \hat{v}_r \rangle\!\rangle) \in \kappa(\ell') \end{array}}{(\hat{\Sigma}, \kappa, \Theta) \vDash_\ell \langle\!\langle M_1^{a_1}, \cdots, M_r^{a_r} \rangle\!\rangle \rhd L.\, P}$$

$$\frac{\begin{array}{c} \bigwedge_{i=1}^{j} (\hat{\Sigma}, \Theta) \vDash_\ell M_i^{a_i} \ \wedge \\ \forall (\ell', \langle\!\langle \hat{v}_1, \cdots, \hat{v}_r \rangle\!\rangle) \in \kappa(\ell) : \ Comp(\ell', \ell) \Rightarrow \left(\bigwedge_{i=j+1}^{r} \hat{v}_i \otimes \tau(x_i, \ell) \in \hat{\Sigma}_\ell(x_i) \ \wedge \ (\hat{\Sigma}, \kappa, \Theta) \vDash_\ell P \right) \end{array}}{(\hat{\Sigma}, \kappa, \Theta) \vDash_\ell (M_1^{a_1}, \cdots, M_j^{a_j}; x_{j+1}^{a_{j+1}}, \cdots, x_r^{a_r}).\, P}$$

An estimate is valid for *multi-output* if it is valid for the continuation of P and the set of messages communicated by the node ℓ to each node ℓ' in L, includes all the messages obtained by the evaluation of the r-tuple $\langle\!\langle M_1^{a_1}, \cdots, M_r^{a_r} \rangle\!\rangle$. More precisely, the rule (i) finds the sets $\Theta(\ell)(a_i)$ for each term $M_i^{a_i}$, and (ii) for all tuples of values $(\hat{v}_1, \cdots, \hat{v}_r)$ in $\Theta(\ell)(a_1) \times \cdots \times \Theta(\ell)(a_r)$ it checks if they belong to $\kappa(\ell')$ for each $\ell' \in L$. Symmetrically, the rule for *input* requires that the values inside messages that can be sent to the node ℓ, passing the pattern matching, are included in the estimates of the variables $x_{j+1}, \cdots, x_r$. More in detail, the rule analyses each term $M_i^{a_i}$, and requires that for any message that it can receive, i.e. $(\ell', \langle\!\langle \hat{v}_1, \cdots, \hat{v}_j, \hat{v}_{j+1}, \ldots, \hat{v}_r \rangle\!\rangle)$ in $\kappa(\ell)$ and $Comp(\ell', \ell)$, $\hat{v}_{j+1}, \ldots, \hat{v}_r$ are included and combined with the estimates of $x_{j+1}, \cdots, x_r$.

Our analysis respects the operational semantics of IoT-LySA.

Checking taint We now show that by inspecting the results of our CFA, we detect whether a variable receives a value coming from a tamperable source, and hence is not trustworthy.

In the following, we denote with $N \xrightarrow{M_1^{a_1},\ldots,M_r^{a_r}}_{\ell} N'$ when all the terms $M_i^{a_i}$ are evaluated inside node ℓ, and with $N \xrightarrow{\langle\!\langle v_1,\ldots,v_r \rangle\!\rangle}_{\ell_1,\ell_2} N'$ when the message $\langle\!\langle v_1,\ldots,v_r \rangle\!\rangle$ is sent from the node ℓ_1 to the node ℓ_2.

First we characterise when a variable x of a node ℓ is *untrustworthy*, i.e. when it stores a value propagated from a tamperable source, and therefore with taint label in $\{\diamondsuit, \oplus\}$. To statically check this property we can simply inspect the abstract store $\widehat{\Sigma}$ and the taint labels of the corresponding abstract values.

Definition 2.3. *Let N be a system of nodes with labels in $\mathcal{L}$, and $\mathcal{T} = \{\mathcal{T}_\ell \mid \ell \in \mathcal{L}\}$ be the set of tamperable sources. Then, the variable x of a node $\ell \in \mathcal{L}$* is untrustworthy *w.r.t. $\mathcal{T}$, if for all derivatives N' s.t. $N \rightarrow^* N'$ it holds that $\Sigma'^{i}_{\ell}(x)\downarrow_2 \in \{\diamondsuit, \oplus\}$ where Σ'^{i}_{ℓ} is the store of ℓ in N'.*

Theorem 2.4. *Let N be a system of nodes with labels in $\mathcal{L}$, and let $\mathcal{T} = \{\mathcal{T}_\ell \mid \ell \in \mathcal{L}\}$ the set of tamperable sources. Then a variable x of the node ℓ is untrustworthy w.r.t. $\mathcal{T}$ if $(\widehat{\Sigma}, \kappa, \Theta) \vDash N$, and $\widehat{\Sigma}_\ell(x)_{\downarrow_2} \subseteq \{\diamondsuit, \oplus\}$.*

We define the confidentiality property in terms of sensitive taint information. There are *no leaks* when messages do *not* expose values with taint label in $\{\ominus, \oplus\}$. We statically verify it, by inspecting the labels of the corresponding abstract values in the component κ.

Definition 2.5. *Let N be a system of nodes with labels in $\mathcal{L}$, and $\mathcal{S} = \{\mathcal{S}_\ell \mid \ell \in \mathcal{L}\}$ the set of its sensitive sensors. Then N has* no leaks *w.r.t. $\mathcal{S}$ if $N \rightarrow^* N'$ and, for all $\ell_1, \ell_2 \in \mathcal{L}$, there is no transition $N' \xrightarrow{\langle\!\langle v_1,\ldots,v_n \rangle\!\rangle}_{\ell_1,\ell_2} N''$ such that $v_{i_{\downarrow_2}} \in \{\ominus, \oplus\}$ for some i.*

Theorem 2.6. *Let N be a system of nodes with labels in $\mathcal{L}$, and $\mathcal{S} = \{\mathcal{S}_\ell \mid \ell \in \mathcal{L}\}$ the set of its sensitive sensors. Then N has no leaks w.r.t. $\mathcal{S}$ if $(\widehat{\Sigma}, \kappa, \Theta) \vDash N$, and $\forall \ell_1, \ell_2 \in \mathcal{L}$ such that $(\ell_2, \langle\!\langle \hat{v}_1, \cdots, \hat{v}_r \rangle\!\rangle) \in \kappa(\ell_2)$ we have that $\forall i.\ \hat{v}_{i\downarrow_2} \in \{\diamondsuit, \oplus\}$.*

The last property characterises when computations considered security critical are not directly or indirectly affected by tainted data i.e. they are reached by untrustworthy data. We assume that the designer identify a set $\mathcal{P} \subseteq \mathcal{A}$ of *critical points* in the application, i.e. points where possibly tainted data should flow, unless they are checked for validity. We statically verify this property, by inspecting the taint labels of the values in Θ for each critical point.

Definition 2.7. *Let N be a system of nodes with labels in $\mathcal{L}$, $\mathcal{S} = \{\mathcal{S}_\ell \mid \ell \in \mathcal{L}\}$ the set of its sensitive sensors, $\mathcal{T} = \{\mathcal{T}_\ell \mid \ell \in \mathcal{L}\}$ the set of its tamperable sources, and $\mathcal{P}$ a set of program critical points. Then N* does not use tainted values in a critical point *if $N \rightarrow^* N'$ and there is no transition $N' \xrightarrow{M_1^{a_1},\ldots,M_r^{a_r}}_{\ell} N''$ s.t. $a_i \in \mathcal{P}$ and $([\![M_i^{a_i}]\!]_{\Sigma'^{i}_{\ell}})_{\downarrow_2} \subseteq \{\ominus, \diamondsuit, \oplus\}$ for some $i \in \{1,\ldots,r\}$ and $\ell \in \mathcal{L}$.*

Theorem 2.8. *Let N be a system of nodes with labels in $\mathcal{L}$, $\mathcal{S} = \{\mathcal{S}_\ell \mid \ell \in \mathcal{L}\}$ the set of its sensitive sensors, $\mathcal{T} = \{\mathcal{T}_\ell \mid \ell \in \mathcal{L}\}$ the set of its tamperable sources, and $\mathcal{P}$ a set of program critical points. Then N does not use tainted values in a program critical point if $(\widehat{\Sigma}, \kappa, \Theta) \vDash N$, and $\Theta(\ell)(a)_{\downarrow_2} = \{\diamondsuit\}$ for all labels $a \in \mathcal{P}$ and $\ell \in \mathcal{L}$.*