About this talk: Keywords

- Network Security Policy
- Stateful Firewalls
- Isabelle/HOL

Directed Security Policies: A Stateful Network Implementation

3rd International Workshop on Engineering Safety and Security Systems (ESSS)

Cornelius Diekmann[†] Lars Hupel[‡] Georg Carle[†]

[†] Chair for Network Architectures and Services [‡] Chair for Logic and Verification
Technische Universität München
Munich, Germany

Motivation

- A directed security policy

$$Alice \rightarrow Bob$$
$$Alice \rightarrow Carl$$
$$\dots$$

- A policy rule

$$A \rightarrow B$$

Motivation

- A policy rule

$$A \rightarrow B$$

- Policy implementation (Linux netfilter iptables firewall)
- Version A

```
iptables -A INPUT -s A -d B -j ACCEPT
iptables -A INPUT -j DROP
```

- Version B

```
iptables -A INPUT -s A -d B -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
iptables -A INPUT -j DROP
```

Version A

```
iptables -A INPUT -s A -d B -j ACCEPT
iptables -A INPUT -j DROP
```

- $A \rightarrow B$ is "can send packets to"
- Direct translation of policy
- Example: Smart meter *A* reports data to billing gateway *B*

Version B

```
iptables -A INPUT -s A -d B -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
iptables -A INPUT -j DROP
```

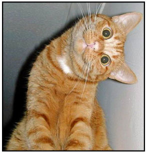
- $A \rightarrow B$ is "can initiate connections to"
- Lax interpretation of policy, also allow packets from *B* to *A*
- Example: Alice *A* requests cat pictures from website *B*

Versio
iptabl
iptabl

- A
- D
- E ... way *B*

Versio
iptabl
iptabl
iptabl

$$A \xrightarrow{\text{give me cat pictures}} B$$



$$A \longleftarrow B$$

- A
- La
- Example: Alice *A* requests cat pictures from website *B*

Overall Problem Statement

Directed Policy: $A \to B$

Stateless Implementation Version A

```
iptables -A INPUT -s A -d B -j ACCEPT
iptables -A INPUT -j DROP
```

Security ++
Network functionality --

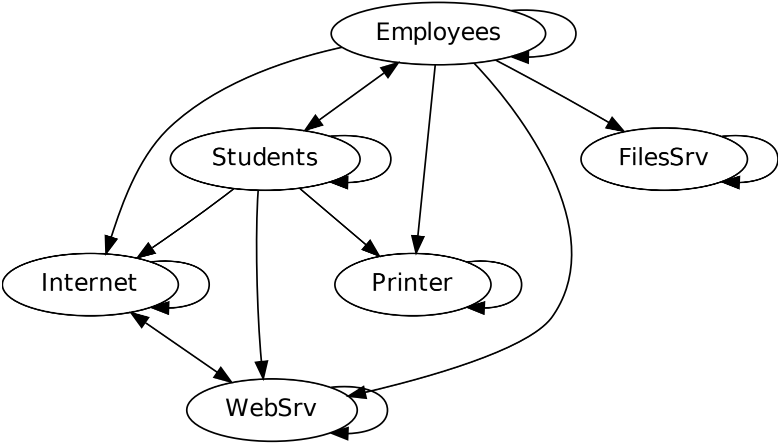
Stateful Implementation Version B

```
iptables -A INPUT -s A -d B -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
iptables -A INPUT -j DROP
```

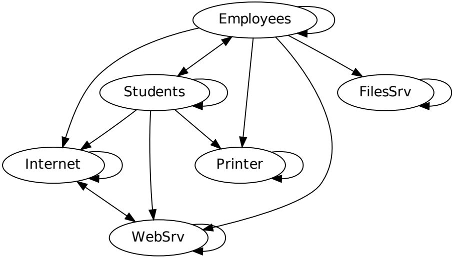
Security -
Network functionality ++

Agenda

Example: Policy of a university department network



Security Invariants: Access Control

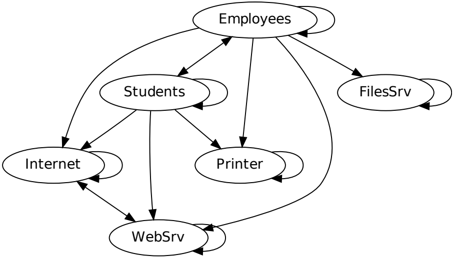


ACS 1 *Printer* only accessible by *Employees* and *Students*

ACS 2 *FileServer* only accessible by *Employees*

ACS 3 *Employees* and *Students* are in a joint subnet
(can collaborate, are protected from accesses from the outside)

Security Invariants: Information Flow



IFS 1 *FileServer* has confidential data. Only *Employees* have the necessary security clearance and are trusted (i. e. can declassify).

IFS 2 *Printer* is an information sink

(no data, e. g., exams, must be retrievable from the printer)

Directed Policy

- Directed graph $G = (V,\ E)$
- Example
  - $A \rightarrow B$
  - $G = (\{A, B\},\ \{(A, B)\})$

Security Invariant

see [DPN+14]

- Total function m of type $\mathcal{G} \Rightarrow \mathbb{B}$
- $m\ G \quad \longleftrightarrow$ policy G fulfills invariant m
- Monotonicity: "prohibiting more is more or equally secure"
  $m\,(V, E)\ \wedge\ E' \subseteq E \Longrightarrow m\,(V, E')$
- m's security strategy
  - ACS: Access Control Strategy
  - IFS: Information Flow Strategy

Offending Flows

$G = (V, E)$

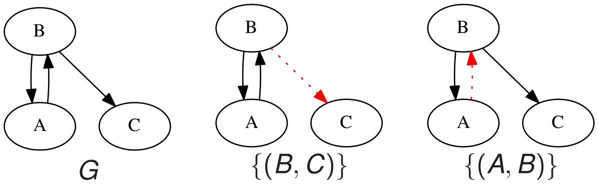
- If a security invariant is violated $\neg\ m\ G$
- Flows $F \subseteq E$ responsible for the violation
- *offending_flows* $m\ G$ is the set of all such minimal F

Offending Flows

$G = (V,\ E)$

- If a security invariant is violated $\neg\ m\ G$
- Flows $F \subseteq E$ responsible for the violation
- $\mathit{offending\_flows}\ m\ G$ is the set of all such minimal F

- Example: m is that A must not access C transitively



- $\mathit{offending\_flows}\ m\ G = \{\{(B, C)\},\ \{(A, B)\}\}$

Stateful Policy

- $T = (V,\ E_\tau,\ E_\sigma)$
  - $E_\tau \approx E$
  - $E_\sigma \subseteq E_\tau$ flows "upgraded" to stateful flows

- Mapping α between T and G

$$\alpha\ T = (V,\ E_\tau \cup \overleftarrow{E_\sigma})$$

- where $\overleftarrow{E_\sigma} = \{(r, s) \mid (s, r) \in E_\sigma\}$ backflows
- Example: $\alpha\ \underbrace{(V,\ E,\ \emptyset)}_{T_{\text{stateless}}} = G$

When does T comply with G?

IFS Compliance

- Protect against information leakage
  - side channels: ACKs, SEQ nr, ... timing channels, ...
- All IFS invariants must be fulfilled

$$\forall m \in \mathit{getIFS}\ M.\ \ m\ (\alpha\ T) \tag{1}$$

ACS Compliance

- Requirements can be relaxed
- If A accesses B, A expects an answer from B for its request
- Example
  - Alice $\rightarrow$ CatPictures

Assumption: no (higher layer) software vulnerability at A

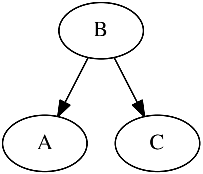
ACS Compliance

- Violations due to backflows are tolerable (expected answers)

ACS Compliance

- Violations due to backflows are tolerable (expected answers)
- As long as they cause no negative side effect!

- Example
  - m is that A must not access C transitively
  - $G = (\{A, B, C\},\ \{(B, A), (B, C)\})$
  - $T = (\{A, B, C\},\ \{(B, A), (B, C)\},\ \underbrace{\{(B, A)\}}_{E_\sigma})$
  - $\alpha\ T = (\{A, B, C\},\ \{(B, A), (B, C), (A, B)\})$
  - *offending_flows m G* $=$
    $\{\{(B, C)\},\ \{(A, B)\}\} = \{\{(B, C)\},\ \overleftarrow{E_\sigma}\}$



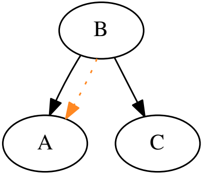
G

ACS Compliance

- Violations due to backflows are tolerable (expected answers)
- As long as they cause no negative side effect!

- Example
  - m is that A must not access C transitively
  - $G = (\{A, B, C\},\ \{(B, A), (B, C)\})$
  - $T = (\{A, B, C\},\ \{(B, A), (B, C)\},\ \underbrace{\{(B, A)\}}_{E_\sigma})$
  - $\alpha\ T = (\{A, B, C\},\ \{(B, A), (B, C), (A, B)\})$
  - $\mathit{offending\_flows}\ m\ G\ =$

    $\{\{(B, C)\},\ \{(A, B)\}\} = \{\{(B, C)\},\ \overleftarrow{E_\sigma}\}$



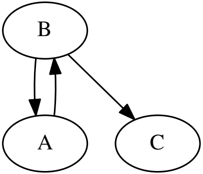
T

ACS Compliance

- Violations due to backflows are tolerable (expected answers)
- As long as they cause no negative side effect!

- Example
  - m is that A must not access C transitively
  - $G = (\{A, B, C\},\ \{(B, A), (B, C)\})$
  - $T = (\{A, B, C\},\ \{(B, A), (B, C)\},\ \underbrace{\{(B, A)\}}_{E_\sigma})$
  - $\alpha\ T = (\{A, B, C\},\ \{(B, A), (B, C), (A, B)\})$
  - *offending_flows m G* $=$

    $\{\{(B, C)\},\ \{(A, B)\}\} = \{\{(B, C)\},\ \overleftarrow{E_\sigma}\}$

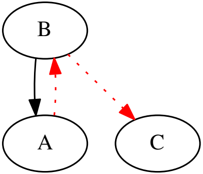


$\alpha\ T$

ACS Compliance

- Violations due to backflows are tolerable (expected answers)
- As long as they cause no negative side effect!

- Example
  - m is that A must not access C transitively
  - $G = (\{A, B, C\},\ \{(B, A), (B, C)\})$
  - $T = (\{A, B, C\},\ \{(B, A), (B, C)\},\ \underbrace{\{(B, A)\}}_{E_\sigma})$
  - $\alpha\ T = (\{A, B, C\},\ \{(B, A), (B, C), (A, B)\})$
  - $offending\_flows\ m\ G\ =$
    $\{\{(B, C)\},\ \{(A, B)\}\} = \{\{(B, C)\},\ \overleftarrow{E_\sigma}\}$



$\bigcup$ *offending*

ACS Compliance

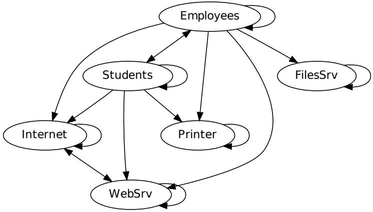
- Violations due to backflows are tolerable (expected answers)
- As long as they cause no negative side effect!
- Approach: for all subsets of $\overleftarrow{E_\sigma}$, verify that violations are exactly in this subset

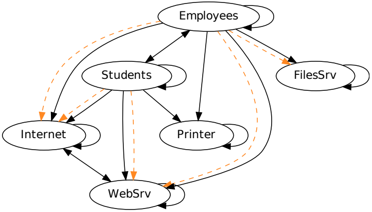
Contributions

- Verify that T is compliant with G and the security invariants
  - in linear time $\mathrm{O}(|E|)$
- Generate T (in particular E_σ) from G and the security invariants
  - in $\mathrm{O}(|E|^2)$

For linear time security invariants

Example





- Everything can be stateful – except at the *Printer*
- Special cases
  - Aircrafts, critical infrastructure, SCADA systems, smart meters, ...

Case Study

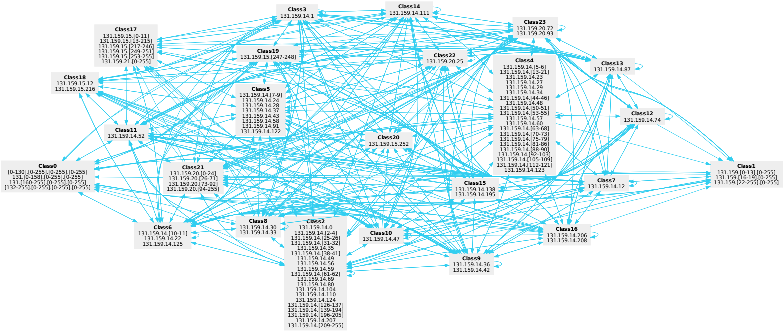


Figure : Firewall TUM Chair for Network Architectures and Services

- Instant results
- $E_\sigma =$ all unidirectional flows and α $T = (V,\ E \cup \overleftarrow{E})$

Conclusion

- Stateful filtering: often overlooked in previous work
- Need for: formally verified translation of network device configurations to formally accessible objects
  - E.g. firewall rule sets, SDN flow tables, routing tables, ... to graphs
- Directed policy vs. stateful firewall implementation ✓
- Fully machine-verified with Isabelle/HOL ✓

  Thys: `https://github.com/diekmann/topoS`
  Data: `https://github.com/diekmann/net-network`

Thanks for your attention!

Questions?

Cornelius Diekmann, Stephan-A. Posselt, Heiko Niedermayer, Holger Kinkelin, Oliver Hanka, and Georg Carle.
Verifying Security Policies using Host Attributes.
In *Proc. FORTE*, Berlin, Germany, June 2014. Springer.
to appear.

Backup Slides

ACS Compliance

- Approach: for all subsets of $\overleftarrow{E_\sigma}$, verify that violations are exactly in this subset

$$\forall X \subseteq \overleftarrow{E_\sigma}. \quad \forall F \in \mathit{get\_offending\_flows}\ (\mathit{getACS}\ M)\ (V,\ E_\tau \cup E_\sigma \cup X).\ F \subseteq X \tag{2}$$

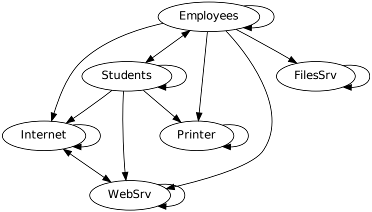
$$\bigcup \mathit{get\_offending\_flows}\ (\mathit{getACS}\ M)\ (\alpha\ T) \subseteq \overleftarrow{E_\sigma} \tag{3}$$

$$\forall (r, s) \in \overleftarrow{E_\sigma}. \quad \bigcup \mathit{get\_offending\_flows}\ (\mathit{getACS}\ M)\ (V,\ E_\tau \cup E_\sigma \cup \{(r, s)\}) \subseteq \{(r, s)\} \tag{4}$$

- Obviously (2) $\Longrightarrow$ (3) and (2) $\Longrightarrow$ (4)

ACS Compliance

- Approach: for all subsets of $\overleftarrow{E_\sigma}$, verify that violations are exactly in this subset
- Exponential complexity
- New formula: All violations must only be due to the *newly added* backflows $\overleftarrow{E_\sigma} \setminus E_\tau$
- Sufficient to show lack of side effects!



ACS Compliance

- Formula (2): For all subsets of $\overleftarrow{E_\sigma}$, verify that violations are exactly in this subset
- New Formula (5): All violations must only be due to the *newly added* backflows $\overleftarrow{E_\sigma} \setminus E_\tau$

$$\bigcup \mathit{get\_offending\_flows}\ \ (\mathit{getACS}\ \ M)\ \ (\alpha\ \ T) \quad \subseteq \quad \overleftarrow{E_\sigma} \setminus E_\tau \quad (5)$$

- Theorem: (5) $\Longrightarrow$ (2)