Semantics-Based Design for Secure Web Services. Massimo Bartoletti et al.

Aleksander Pohl

Katedra Informatyki, Akademia Górniczo-Hutnicza

19. stycznia 2009

Plan prezentacji

Plan prezentacji

Problematyka

- Zastosowanie metod formalnych[1] w paradygmacie SOC.
- Modelowanie aspektów bezpieczeństwa z wykorzystaniem notacji zbliżonej do UML-a.
- Automatyczna, statyczna weryfikacja polityk bezpieczeństwa.
- Określanie bezpiecznych planów wykonania usługi dla warunków normalnych.
- Określanie bezpiecznych strategii wykonania usługi dla warunków nieprawidłowych (niedostępność serwisu).

[1] Semantyka *teoriomodelowa*, a nie *sieć* semantyczna

Service Oriented Computing

- Paradygmat tworzenia aplikacji rozproszonych
- Gotowe rozwiązania tworzone poprzez kompozycję oraz koordynację dostępnych serwisów
- Serwisy – niezależne moduły obliczeniowe dostępne w sieci rozproszonej poprzez ustandaryzowane mechanizmy
- *Otwartość* serwisów – budowane przy założeniu brak (pełnej) wiedzy na temat otaczającego środowiska, klientów, itp.
- Serwisy webowy oparte o technologię XML – najbardziej rozpowszechnione aplikacje SOC

Cel i założenia metodologii

- Projektowanie skomplikowanych aplikacji, z określonymi nie-funkcjonalnymi wymaganiami (w tym wypadku – bezpieczeństwa)
- Podstawa – rachunek λ^{req} wykorzystywany do opisywania, wyboru oraz bezpiecznej kompozycji serwisów
- *Bezpieczeństwo bazujące na historii (history-based security)* – istotne akcje (np. otwarcie pliku) są logowane. Mechanizm bezpieczeństwa określa poprawność następnej akcji na podstawie historii wcześniejszych akcji.
- *Wywołanie zgodne z kontraktem (call by contract)* – wybór zdalnego serwisu na podstawie algorytmu dopasowującego globalne wymogi bezpieczeństwa do jego charakterystyki

Ogólny schemat działania

- Rozszerzenie języka WSDL o informacje związane zachowaniem serwisu (w kontekście bezpieczeństwa)
- Publikowany interfejs serwisu jest annotowanym typowanym systemem funkcyjnym postaci: $\tau_1 \xrightarrow{H} \tau_2$
  - τ_1 – typ argumentu
  - τ_2 – typ zwracanej wartości
  - H – gramatyka bezkontekstowa opisująca wszystkie możliwe historie wykonania serwisu
- Dla żądania klienta $req_r\tau$, którego typ $\tau = \tau_1 \xrightarrow{\varphi} \tau_2$ dopasowywane są serwisy o typie $\tau_1 \xrightarrow{H} \tau_2$, których abstrakcyjne zachowanie H odpowiada polityce φ

Plan prezentacji

Bezpieczeństwo bazujące na historii

- Obserwacje aktywności istotnych dla bezpieczeństwa (np. otwarcie socketu, czytanie i zapis do pliku, dostęp do krytycznego obszaru pamięci, itp.) zwane są *zdarzeniami*
- *Historie* są sekwencjami zdarzeń
- Polityki bezpieczeństwa związane są z pewnymi własnościami historii
- Istotne własności bezpieczeństwa są wyrażane przez automaty skończone
- Typowy mechanizm gwarancji bezpieczeństwa realizowany jest przez *monitor referencyjny*, który obserwuje wykonanie programu i przerywa je, jeśli miałoby złamać zadaną politykę bezpieczeństwa

Serwisy stanowe (ω) i bezstanowe (1)

- Serwis stanowy śledzi historie pomiędzy kolejnymi wywołaniami, natomiast serwis bezstanowy tylko w ramach danego wywołania
- Serwisy stanowe mogą realizować bardziej skomplikowane polityki bezpieczeństwa, np.:
  - ograniczenie liczby wywołań serwisu dla poszczególnych klientów
  - zapamiętywanie zdarzenia zalogowania do serwisu i korzystanie z tej informacji w ramach danej sesji
  - ograniczenie liczby nieudanych logowań do trzech
- Pomimo mniejszej ekspresywności serwisy bezstanowe również mogą być poddawane użytecznej, statycznej analizie

Historie globalne (G) i lokalne (L)

- Jeśli historie są *lokalne* dla serwisu, weryfikując politykę bezpieczeństwa, bazuje on wyłącznie na własnej historii
- W przypadku historii *globalnych* serwis podejmuje decyzję na podstawie pełnej historii określonego żądania
- Historie lokalne mogą być wykorzystywane jeśli dany serwis nie ufa informacjom dystrybuowanym przez inne serwisy
- Historie globalne pozwalają na realizację bardziej skomplikowanych polityk bezpieczeństwa
- Przykład (hist. globalna): dystrybuowana *black-lista* – grupa ufających sobie serwisów tworzy wspólną listę serwisów, które nie będą wykorzystywane

Żądania pierwszego (F) i wyższych rzędów (H)

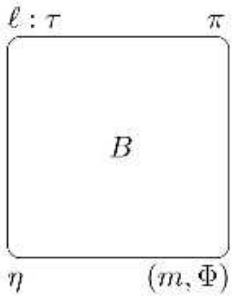
- Żądanie typu $\tau \xrightarrow{\varphi} \tau'$ jest żądaniem pierwszego rzędu, jeśli zarówno τ jak i τ' są typami podstawowymi (book, int, etc.)
- W przeciwnym razie żądanie jest żądaniem rzędu wyższego
- Przykładowo, jeśli τ jest funkcją znaczy to, że klient wysyła jako parametr pewien kod, który ma zostać wykonany
- Symetrycznie, jeśli τ' jest funkcją, to serwis wywołany zwraca pewien kod do klienta

Wątki zależne (D) i niezależne (I)

- Historie wątków niezależnych nie dzielą żadnych informacji – każdy wątek posiada osobną historię
- Historie wątków zależnych mogą dzielić część lub całą historię
- Wątki zależne mogę wpływać na swoje wykonanie (w aspekcie bezpieczeństwa), zaś wątki niezależne – nie
- Przykładowo, jeśli serwis realizuje politykę *one-shot* – tzn. może być wywołany tylko raz, nie może być ona modelowana z użyciem wątków niezależnych

Plan prezentacji

Serwis



- $\ell : \tau$ – lokalizacja serwisu ℓ + interfejs τ
- π – plan wykonania
- η – historia
- (m, Φ) – flaga monitora (on/off) oraz lista aktywnych polityk
- B – kod serwisu

Plan wykonania

- Plan formalizuje w jaki sposób żądania serwisu $req_r\tau$, wobec których określono politykę bezpieczeństwa, są zamieniana na aktualne wywołania do konkretnych serwisów w konkretnych lokalizacjach
- $\pi ::=$
  - 0 – pusty
  - $r[\ell]$ – wybór serwisu
  - $r[?]$ – wybór nieprzypisany
  - $\pi|\pi'$ – kompozycja
- Wybór planu polega na przypisaniu dla każdego żądania r odpowiadającego mu miejsca ℓ
- Plan jest kompletny jeśli wszystkie żądania mają przypisane miejsca (tzn. nie występują w nim wyrażenia $r[?]$)

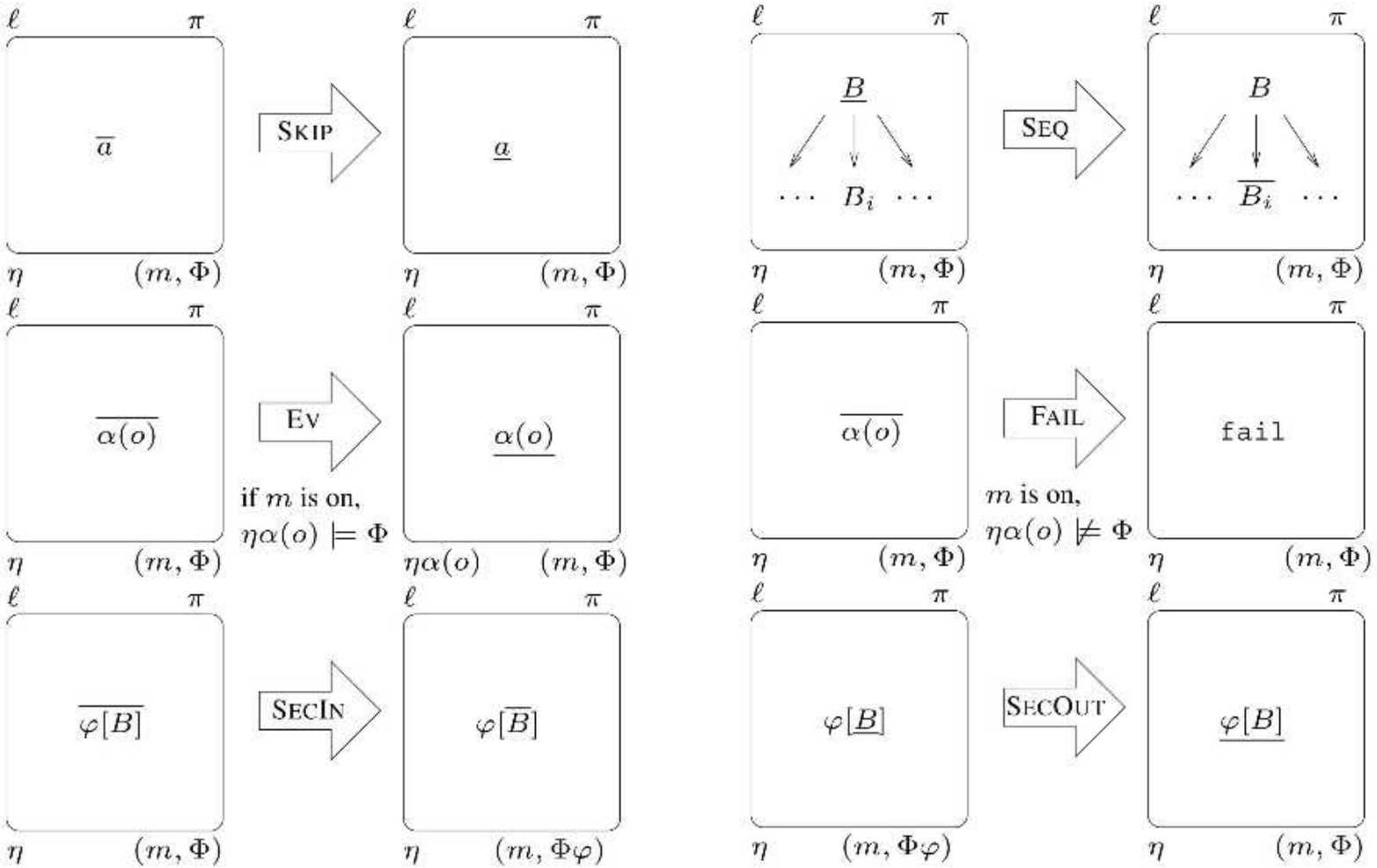
Zachowanie serwisu

- Blok B opisuje zachowanie serwisu. Jest diagramem, którego węzłami mogą być następujące elementy:
  - *aktywność podstawowa* a – np. obliczenie matematyczne, które nie ma związku z bezpieczeństwem serwisu
  - *zdarzenie* $\alpha(o)$ – zdarzenie α związane z obiektem o, które jest związane z politykami bezpieczeństwa (np. odczytanie pliku)
  - *żądanie* $\mathrm{req}_r\,\tau$ – żądanie, które ma zostać wysłane do zdalnego serwisu, identyfikowane przez r
  - *blok bezpieczeństwa* $\varphi[B]$ – polityka φ musi być przestrzegana wewnątrz bloku B
  - *blok planowania* $\{B\}$ – konstruuje plan wykonania dla bloku B

Graf aktywności

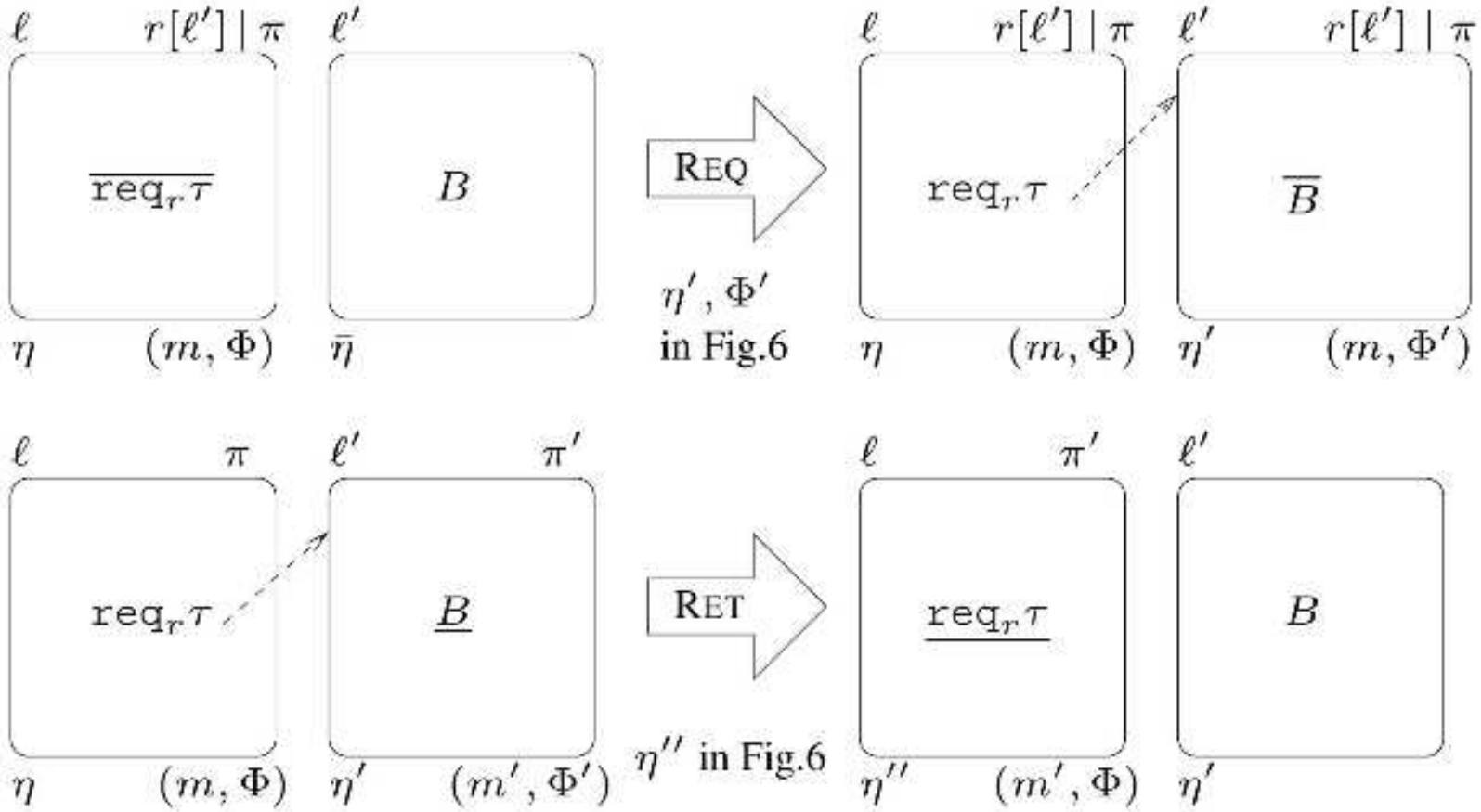
- Zachowanie serwisów jest formalnie opisywane za pomocą semantyki przekształcającej grafy
- Analizując graf aktywności serwisów można określić, czy istnieje dla niego plan, który nie prowadzi do naruszenia polityk bezpieczeństwa
- Ponadto, wykorzystana aparatura formalna pozwala znaleźć taki plan w sposób całkowicie automatyczny
- Plan wykonania serwisu zwany jest *żywotnym*, jeśli przy założeniu, że występujące w nim serwisy są dostępne nie prowadzi do zatrzymania obliczeń, tzn. nie narusza żadnej polityki bezpieczeństwa

Semantyka (1)



Semantics-Based Design for Secure Web Services. Massimo Bartoletti et al.

Semantyka (2)



Semantics-Based Design for Secure Web Services. Massimo Bartoletti et al.

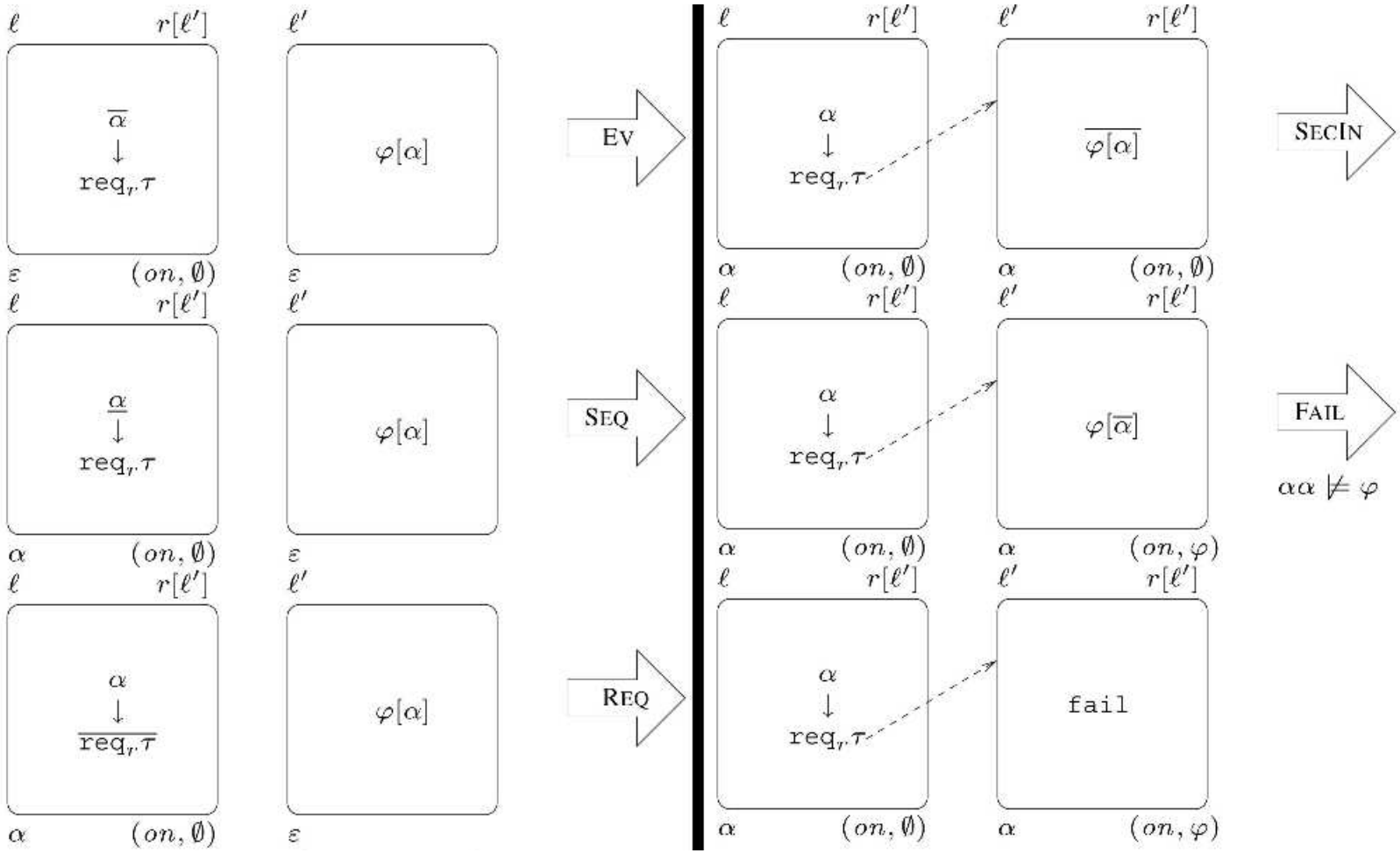
Semantyka (3)

	Stateless service ℓ'	
	Local histories	Global histories
REQ	$\eta' = \varepsilon$ $\Phi' = \varepsilon$	$\eta' = \eta$ $\Phi' = \Phi$
RET	$\eta'' = \eta$	$\eta'' = \eta$
	Stateful service ℓ'	
	Local histories	Global histories
REQ	$\eta' = \bar{\eta}$ $\Phi' = \varepsilon$	$\eta' = \eta$ $\Phi' = \Phi$
RET	$\eta'' = \eta$	$\eta'' = \eta'$

Semantyka (4)

- *Skip* – pominięcie nieistotnego elementu obliczenia
- *Ev* – zdarzenie dopisywane do historii serwisu
- *SecIn* – wejście do sekcji z określoną polityką bezpieczeństwa
- *SecOut* – wyjście z sekcji bezpiecznej
- *Seq* – wybór jednej z ścieżek realizacji programu
- *Fail* – zaprzestanie wykonywania programu z powodu naruszenia polityki bezpieczeństwa
- *Req* – wywołanie zdalnego serwisu
- *Ret* – powrót ze zdalnego serwisu
- $\overline{a}$ – aktywność, która ma zostać wykonana
- $\underline{a}$ – aktywność, która właśnie została wykonana

Przykład przekształceń



Polityki bezpieczeństwa

- Opisywane za pomocą automatów skończonych
- Badają regularne własności historii zdarzeń
- Stany końcowe automatów są stanami naruszającymi polityki bezpieczeństwa.
- Są to automaty parametryzowane, tzn. występują w nich krawędzie postaci $q \xrightarrow{x} q'$ oraz $q \xrightarrow{\overline{x}} q'$, gdzie x to parametr:
  - x – oznacza, że krawędź będzie aktywowana dla wartości związanej ze zmienną x
  - $\overline{x}$ oznacza, że krawędź będzie aktywowana dla każdej innej wartości niż wartość związana ze zmienną x
- Dla wszystkich par (q, σ) nie określonych *explicite* istnieją pętle $(q, \sigma) \rightarrow (q, \sigma)$

Strategie na wypadek błędu

W wypadku niedostępności jednego z serwisów określonych w planie, można przyjąć jedną z 4 strategii:

- **Pies Greyfriara** – czekaj tak długo, aż system będzie dostępny
- **Patch** – spróbuj zastąpić niedostępne serwisy przez (nowo odkryte lub niewykorzystane) inne, zweryfikuj ich zgodność z planem
- **Piaskownica** – włącz monitor i wykorzystuj serwisy, które zgodne są z planem z pominięciem ograniczenia H, polityki bezpieczeństwa nie zostaną złamane, ale system może utknąć jeśli monitor zasygnalizuje naruszenie polityki w kolejnym kroku
- **Ponowne planowanie** – skonstruuj zupełnie nowy plan na podstawie dostępnych serwisów (może być czasochłonne)

Strategia dla pojedynczego żądania *r*

STRATEGY	UPDATE	CASE	CONDITION
Greyfriars Bobby	π Φ	all	The current plan π has a choice for r
Patch	$\pi \mid r[\ell_i]$ Φ	IFL1	$\varphi[H_i]$ is valid
		IFLω	$\varphi[H_i]$ is valid, and $\ell_i \notin \pi$
		IFG1	$\eta\varphi[H_i]$ is valid
		DFL1	$\varphi[H_i]$ is valid
Sandbox	$\pi \mid r[\ell_i]$ $(on, \Phi\varphi)$	all	The service ℓ_i has type $\tau \to \tau'$
Replan	π' $(off, \Phi\varphi)$	all	The new plan π' has a choice for r

Własności modelu

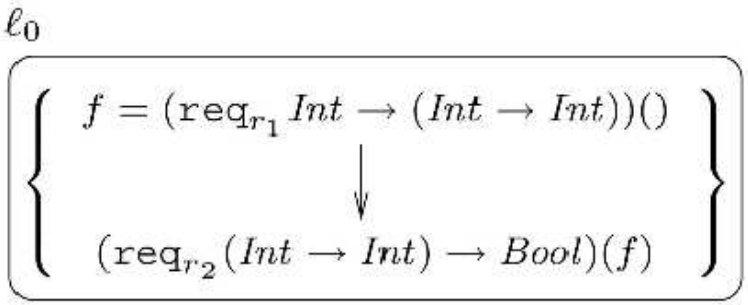
Statyczna analiza modelu pozwala na formułowanie zdań postaci: $H \Rightarrow B : \tau$, gdzie H to historie serwisu reprezentowanego przez B o typie τ. Posiada ona następujące własności

- *Poprawność* – efekt końcowy nad-aproksymuje wszystkie możliwe historie wykonania serwisu
- *Zgodność typu* – poprawne efekty uzyskiwane są przez obliczenia, które nigdy nie prowadzą do naruszenia polityk bezpieczeństwa
- *Sprawdzanie modelu* – poprawność historii jest weryfikowalna w modelu

Aktualny algorytm sprawdzania poprawności modelu oblicza wszystkie poprawne plany, dzięki czemu system może być wykorzystywany do automatycznego komponowania serwisów.

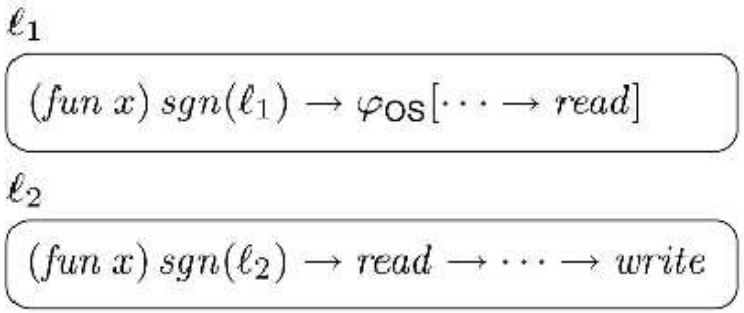
Plan prezentacji

Wywołanie zdalnego kodu



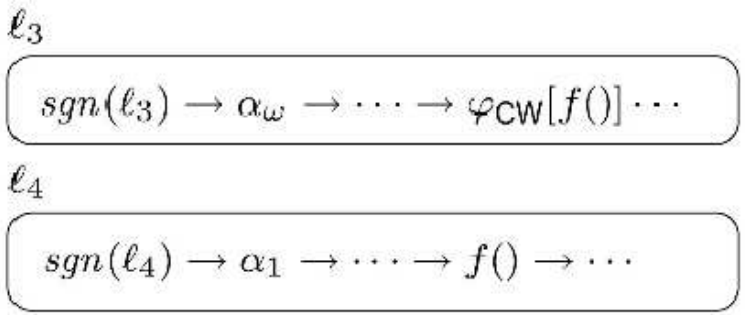
- Klient w lokalizacji ℓ_0 jest urządzeniem o ograniczonych zasobach obliczeniowych
- Klient chce uruchomić kod pobrany z lokalizacji ℓ_1 lub ℓ_2 w lokalizacji ℓ_3 lub ℓ_4
- Klient tworzy dwa żądania wyższego rzędu:
  1. req_{r_1} pobranie kodu (z ℓ_1 lub ℓ_2) akceptującego parametr typu całkowitego, zwracającego wartość typu całkowitego
  2. reg_{r_2} wysłanie pobranego kodu (do ℓ_3 lub ℓ_4) w celu jego wykonania

Dostarczyciele kodu



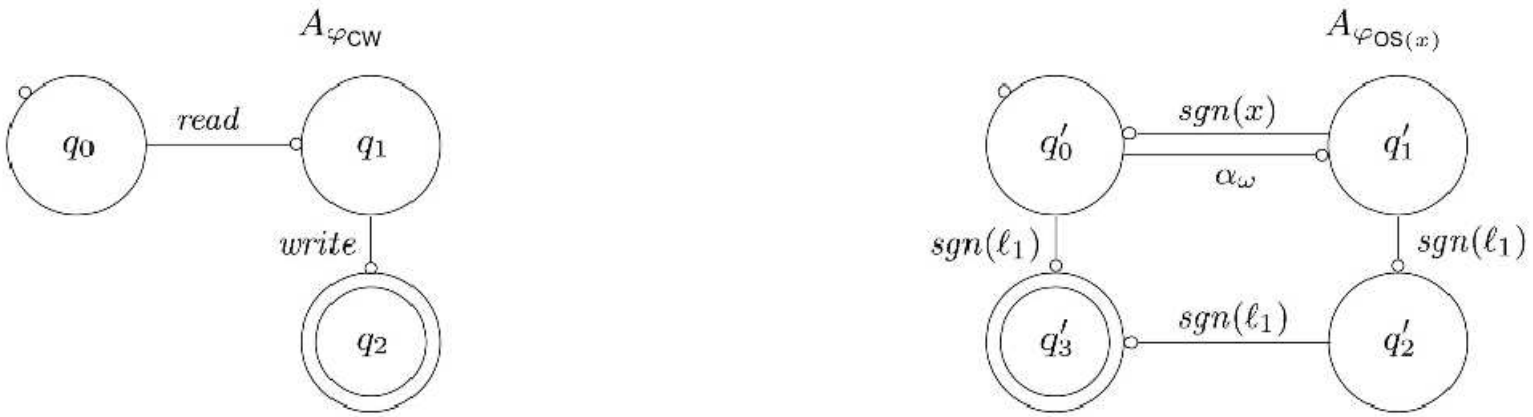
- Serwis ℓ_1 dostarcza funkcję, która może być użyta tylko raz – φ_{OS} („one-shot”)
- Wykonanie tej funkcji zawiera dwa istotne zdarzenia: podpisanie sygnatury serwisu ($sgn(\ell_1)$) oraz odczytanie pliku (*read*)
- Serwis ℓ_2 dostarcza funkcję, która nie definiuje żadnych restrykcji bezpieczeństwa
- Wykonanie tej funkcji zawiera dwa istotne zdarzenia: czytanie (*read*) oraz zapisanie (*write*) pliku

Serwisy obliczeniowe



- Serwis ℓ_3 jest serwisem stanowym i uruchamia kod pod nadzorem polityki „chińskiego muru” φ_{CW}, wymagającej aby dane, które zostały odczytane, nie mogły być ponownie zapisane
- Serwis ℓ_4 jest serwisem bezstanowym i nie nakłada żadnych restrykcji bezpieczeństwa

Polityki bezpieczeństwa



Rysunek: Diagramy stanów dla polityki „chińskiego muru” (CW) oraz „jednokrotnego uruchomienia” (OS)

Analiza statyczna (1)

- Zachowanie sieci serwisów jest opisywane przez następujące wyrażanie:
  $H = \{r_2[\ell_3] \triangleright \ell_3 :$
  $sgn(\ell_3) \cdot \alpha_\omega \cdot \varphi_{CW}[\{r_1[\ell_1] \triangleright sgn(\ell_1) \cdot \varphi_{OS}[read],$
  $r_1[\ell_2] \triangleright read \cdot write\}],$
  $r_2[\ell_4] \triangleright \ell_4 : sgn(\ell_4) \cdot \alpha_1 \cdot \{r_1[\ell_1] \triangleright sgn(\ell_1) \cdot \varphi_{OS}[read],$
  $r_1[\ell_2] \triangleright read \cdot write\}\}$
- Problem, który pozostaje przy analizie tej historii to fakt, że ze względu na występowanie wywołań wyższych rzędów, wybór konkretnego serwisu nie odpowiada na pytanie, czy wymagana polityka bezpieczeństwa będzie zrealizowana

Analiza statyczna (2)

- Można go rozwiązać przekształcając H na równoważne wyrażenie H':
  $H' = \{r_1[\ell_1]|r_2[\ell_3] \triangleright \ell_3 : sgn(\ell_3) \cdot \alpha_\omega \cdot \varphi_{CW}[sgn(\ell_1) \cdot \varphi_{OS}[read]],$
  $r_1[\ell_2]|r_2[\ell_4] \triangleright \ell_4 : sgn(\ell_4) \cdot \alpha_1 \cdot sgn(\ell_2) \cdot read \cdot write,$
  $r_1[\ell_1]|r_2[\ell_4] \triangleright \ell_4 : sgn(\ell_4) \cdot \alpha_1 \cdot sgn(\ell_1) \cdot \varphi_{OS}[read],$
  $r_1[\ell_2]|r_2[\ell_3] \triangleright \ell_3 : sgn(\ell_3) \cdot \alpha_\omega \cdot \varphi_{CW}[sgn(\ell_2) \cdot read \cdot write]\}$
- W powyższym wyrażeniu opis plan realizacji jest ściśle oddzielony o abstrakcyjnego opisu historii
- Dzięki temu można statyczne zweryfikować poprawność każdego z planów
- Widać, że plany $r_1[\ell_1]|r_2[\ell_4]$ oraz $r_1[\ell_2]|r_2[\ell_3]$ nie są poprawne, a pozostałe dwa plany mogą być realizowane, bez naruszenia ograniczeń bezpieczeństwa.

- Bartoletti M., Degano P., Ferrari G. L., Zunino R. „Semantic based Design for Secure Web Services” *IEEE Transactions on Software Engineering*, vol. 34, No. 1, January/February 2008