UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di laurea magistrale in tecnologie dell'informazione e della comunicazione

Race condition in applicazioni web

Relatore: Prof. Danilo Bruschi
Correlatore: Dott. Roberto Paleari

Tesi di laurea di: **Davide Marrone**
Matricola: 685167

Sommario

1. Il problema
2. La soluzione proposta
3. I risultati ottenuti
4. Conclusioni

Introduzione

Applicazioni web

- Molte applicazioni adottano il paradigma web: un modello client-server che utilizza il protocollo HTTP
- Nei server web sono presenti dei moduli che permettono l'esecuzione di codice lato server

Introduzione

Applicazioni web

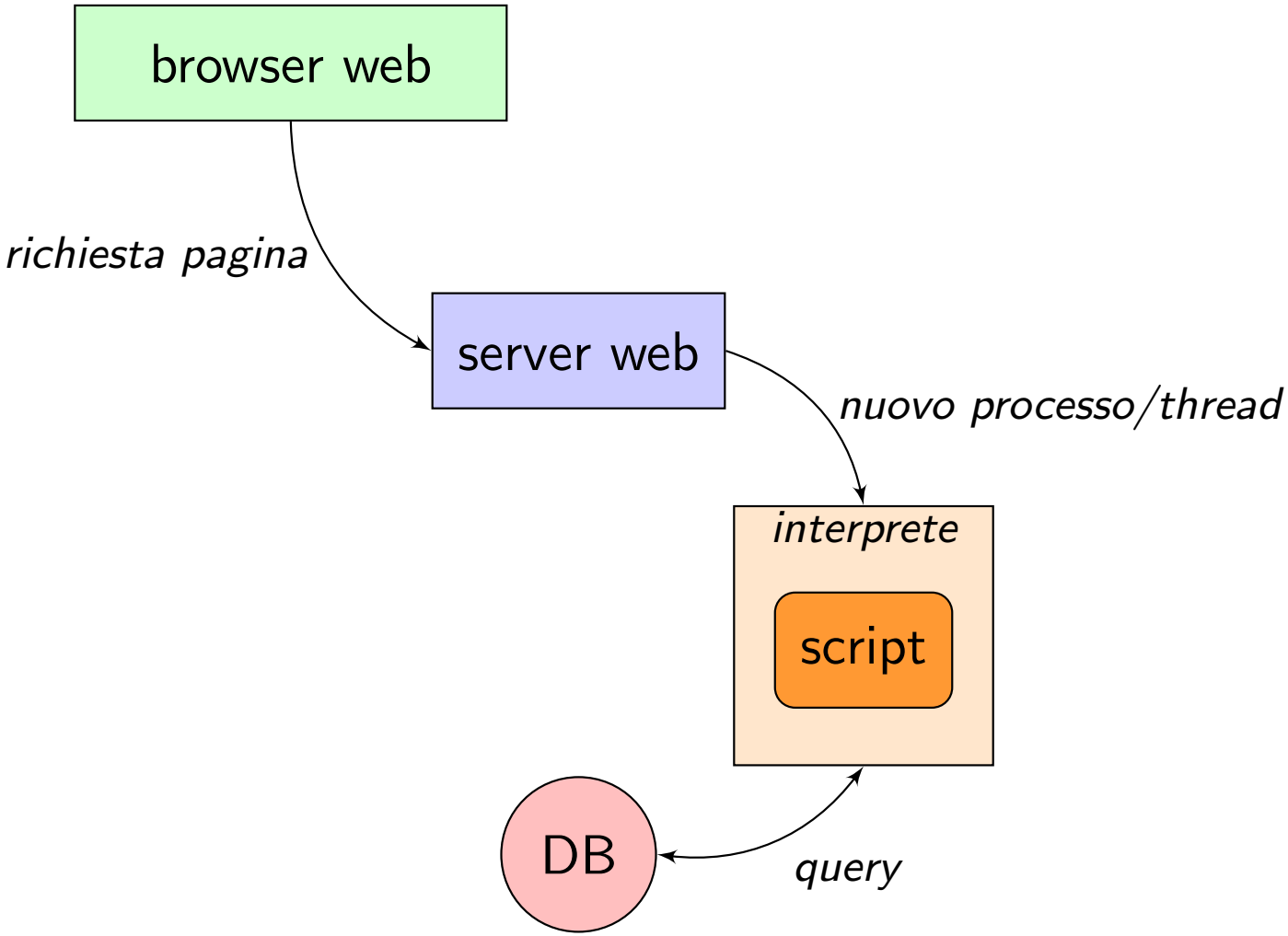
- Molte applicazioni adottano il paradigma web: un modello client-server che utilizza il protocollo HTTP
- Nei server web sono presenti dei moduli che permettono l'esecuzione di codice lato server

Problemi di sicurezza

- Le applicazioni web sono notoriamente soggette a diversi tipi di attacchi (*es.*, SQLI e XSS)
- $\sim$ 60% delle vulnerabilità sul software sono relative ad applicazioni web
- Le race condition sono un problema ben noto ma il loro impatto sulle applicazioni web non è stato esplorato a fondo

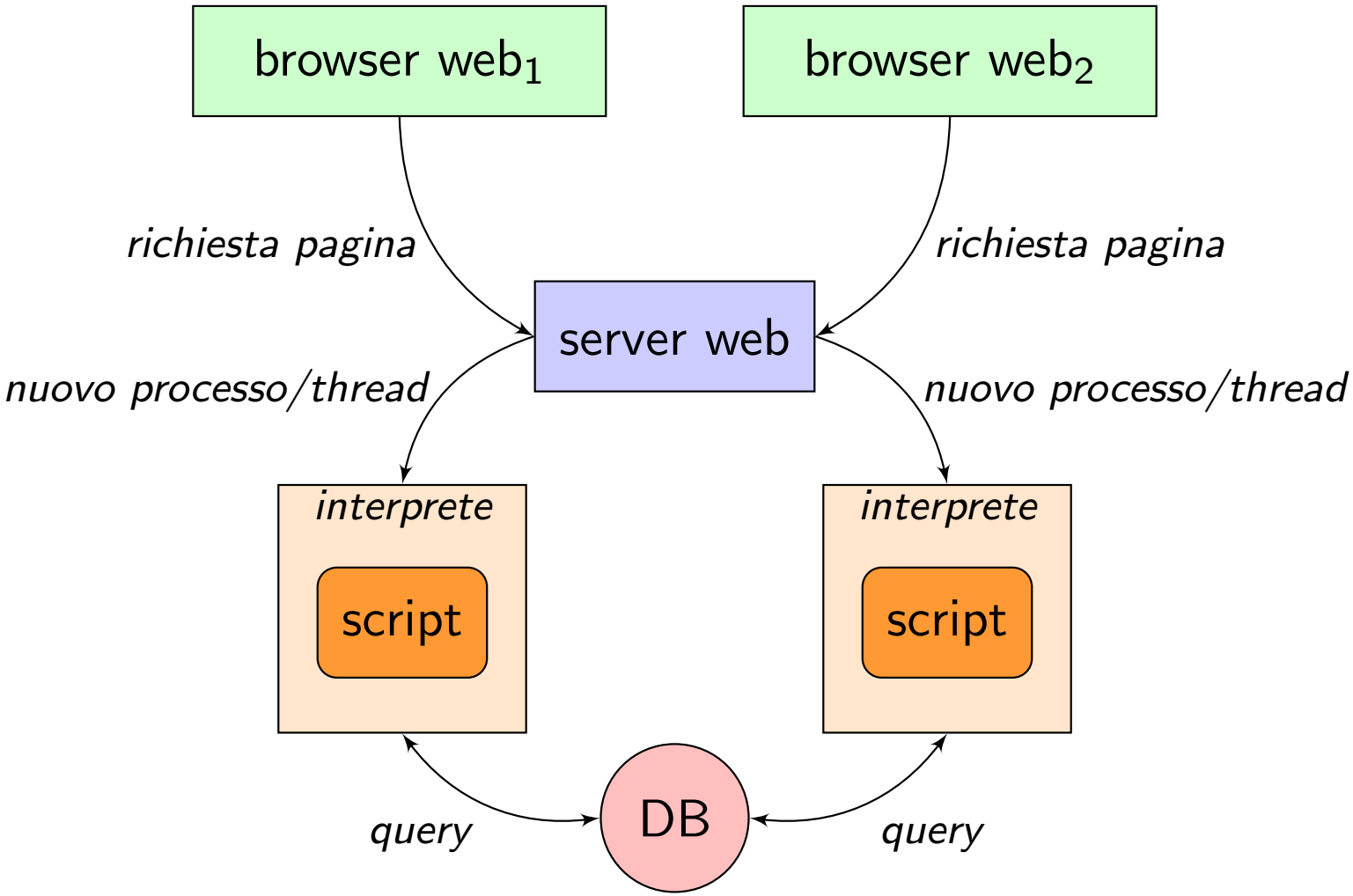
Funzionamento delle applicazioni web

Richiesta singola



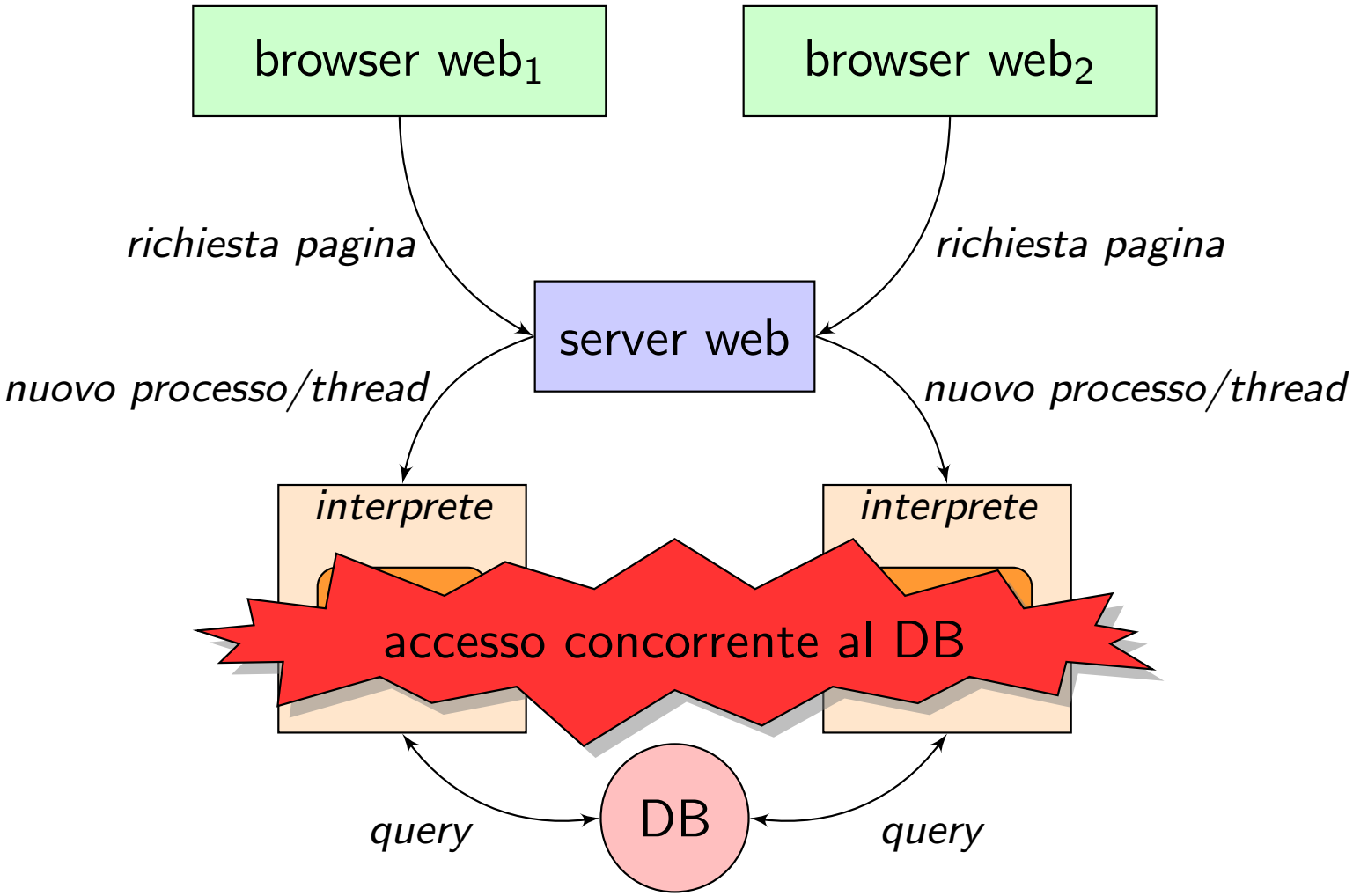
Funzionamento delle applicazioni web

Richieste multiple in parallelo



Funzionamento delle applicazioni web

Richieste multiple in parallelo



Analisi del problema

```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*

Database	
ID	*Credito*
50	450
12	100
96	750
35	110
...	...

Analisi del problema

```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*
2	(id: 12, credito: 100)	$\perp$	$\perp$

Database	
ID	*Credito*
50	450
12	**100**
96	750
35	110
...	...

Analisi del problema

```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*
2	(id: 12, credito: 100)	$\perp$	$\perp$
4	(id: 12, credito: 100)	1	$\perp$

Database	
ID	*Credito*
50	450
12	100
96	750
35	110
...	...

Analisi del problema

```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*
2	(id: 12, credito: 100)	$\perp$	$\perp$
4	(id: 12, credito: 100)	1	$\perp$
4	(id: 12, credito: 100)	2	(id: 12, credito: 100)

Database	
ID	*Credito*
50	450
12	**100**
96	750
35	110
...	...

Analisi del problema

```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*
2	(id: 12, credito: 100)	$\perp$	$\perp$
4	(id: 12, credito: 100)	1	$\perp$
4	(id: 12, credito: 100)	2	(id: 12, credito: 100)
5	(id: 12, nuovo: 20)	4	(id: 12, credito: 100)

Database	
ID	*Credito*
50	450
12	100
96	750
35	110
...	...

Analisi del problema

```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*
2	(id: 12, credito: 100)	$\perp$	$\perp$
4	(id: 12, credito: 100)	1	$\perp$
4	(id: 12, credito: 100)	2	(id: 12, credito: 100)
5	(id: 12, nuovo: 20)	4	(id: 12, credito: 100)
6	(id: 12, nuovo: 20)	4	(id: 12, credito: 100)

Database	
ID	*Credito*
50	450
12	**20**
96	750
35	110
...	...

Analisi del problema

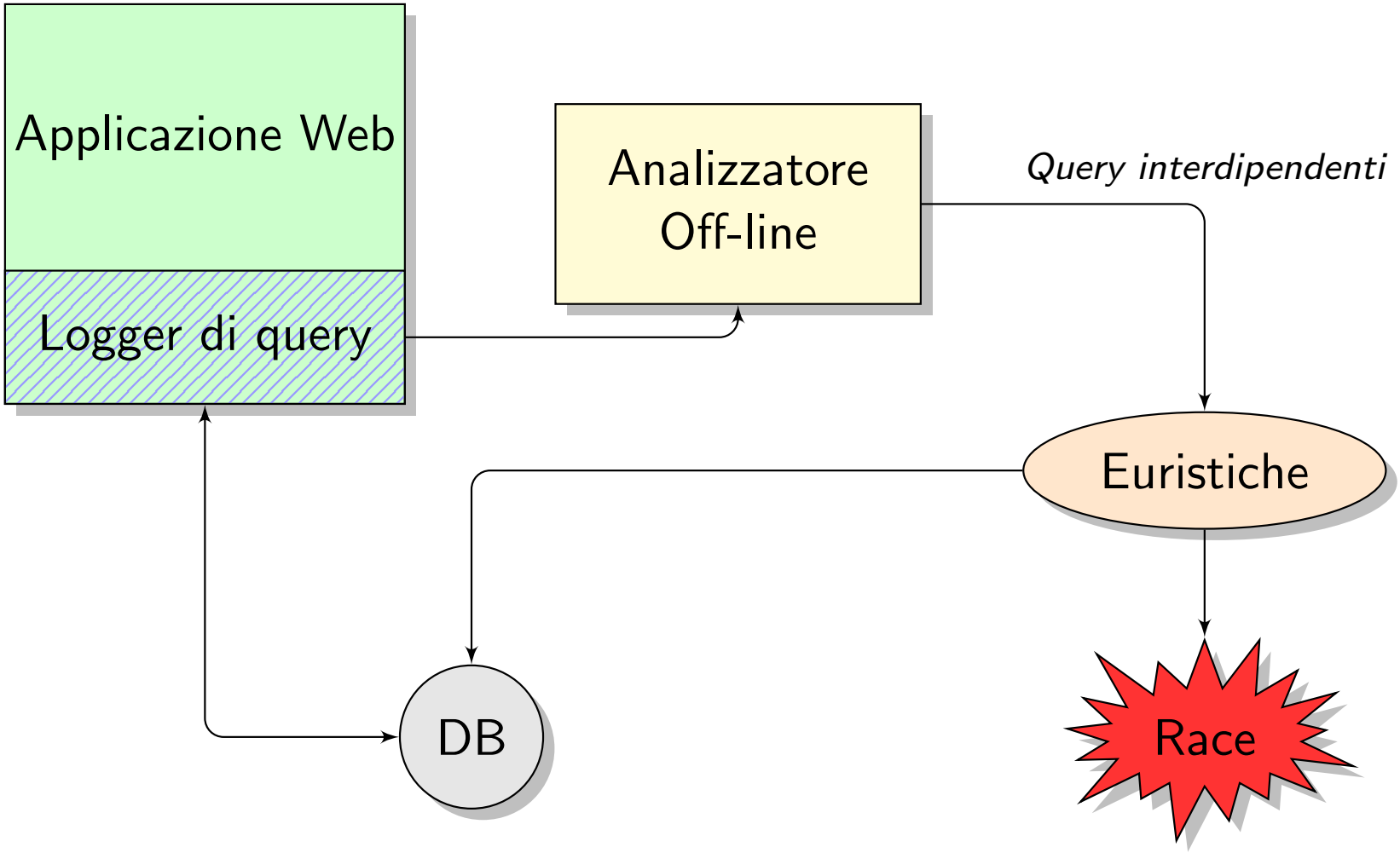
```
1  $query = mysql_query('SELECT credito FROM Utente '
                        .'WHERE id = '. $id);
2  $riga = mysql_fetch_assoc($query);
3  if($riga['credito'] >= 80) {
4     <esecuzione dell'operazione di trasferimento credito>
5     $nuovoCredito = $riga['credito'] - 80;
6     mysql_query('UPDATE Utente SET credito = ' . $nuovoCredito
                  .' WHERE id = ' . $id);
7  }
```

P_1		P_2	
Linea	*Dati*	*Linea*	*Dati*
2	(id: 12, credito: 100)	$\perp$	$\perp$
4	(id: 12, credito: 100)	1	$\perp$
4	(id: 12, credito: 100)	2	(id: 12, credito: 100)
5	(id: 12, nuovo: 20)	4	(id: 12, credito: 100)
6	(id: 12, nuovo: 20)	4	(id: 12, credito: 100)
$\perp$	$\perp$	6	(id: 12, nuovo: 20)

Database	
ID	*Credito*
50	450
12	**20**
96	750
35	110
...	...

Architettura per la rilevazione di race condition

Individuazione di race condition relative al database



Analizzatore off-line

Algoritmo per l'identificazione di query interdipendenti

- Query registrate: $Q = \langle q_1, q_2, \ldots, q_n \rangle$
- $\forall q \in Q$ calcolo degli insiemi $\mathrm{use}(q)$ e $\mathrm{def}(q)$
- Identificazione query *interdipendenti*:
  $(q_i, q_j) \in Q^2 : \mathrm{use}(q_i) \cap \mathrm{def}(q_j) \neq \emptyset \wedge i < j$

Analizzatore off-line

Algoritmo per l'identificazione di query interdipendenti

- Query registrate: $Q = \langle q_1, q_2, \ldots, q_n \rangle$
- $\forall q \in Q$ calcolo degli insiemi $\mathrm{use}(q)$ e $\mathrm{def}(q)$
- Identificazione query *interdipendenti*:
  $(q_i, q_j) \in Q^2 : \mathrm{use}(q_i) \cap \mathrm{def}(q_j) \neq \emptyset \wedge i < j$

Un semplice esempio

Query$_1$
SELECT credito
FROM Utente
WHERE id = 12;

Query$_2$
UPDATE Utente
SET credito = 20
WHERE id = 12;

= *usati*

= *definiti*

Euristiche

Alcune condizioni presenti nelle clausole **WHERE** possono portare a falsi positivi:

Query$_1$
SELECT id
FROM Sessione
WHERE *scadenza* <= *123*;

Query$_2$
DELETE
FROM Sessione
WHERE *scadenza* > *123*;

■ = *usati*
■ = *definiti*

Euristiche

Alcune condizioni presenti nelle clausole **WHERE** possono portare a falsi positivi:

Query$_1$
SELECT id
FROM Sessione
WHERE *scadenza* <= *123*;

Query$_2$
DELETE
FROM Sessione
WHERE *scadenza* > *123*;

= *usati*

= *definiti*

Soluzioni

- Interrogazione dinamica del DB intersecando le due condizioni presenti nelle clausole WHERE ($\rightarrow$ efficiente ma *non* completo)
- Constraint solver ($\rightarrow$ risposta completa ma poco efficiente e non supporta tutti i costrutti SQL)

Il prototipo realizzato

Caratteristiche

- Implementato per applicazioni PHP, il modulo di analisi è indipendente dal linguaggio
- Sono analizzate le interazioni tra istanze multiple dello *stesso* script

Il prototipo realizzato

Caratteristiche

- Implementato per applicazioni PHP, il modulo di analisi è indipendente dal linguaggio
- Sono analizzate le interazioni tra istanze multiple dello *stesso* script

Race condition trovate

Applicazione	**Categoria**	**Query**	**Race**
Drupal 7.0	CMS	13416	63 (3)
phpBB 3.0.8	forum	3136	58 (4)
WordPress 3.1	blog/CMS	3729	82 (3)
Magento 1.5.0.1	carrello virtuale	9453	61 (2)

Race condition relative alla sicurezza

Tipo di vulnerabilità

Dipendenti dalle applicazioni, alcuni esempi:

- Risorse limitate
- Voti multipli su sondaggi
- Raggiro delle protezioni contro il brute forcing
- Elusione dei controlli anti-flooding

Race condition relative alla sicurezza

Tipo di vulnerabilità

Dipendenti dalle applicazioni, alcuni esempi:

- Risorse limitate
- Voti multipli su sondaggi
- Raggiro delle protezioni contro il brute forcing
- Elusione dei controlli anti-flooding

Race condition in applicazioni closed-source

- Individuazione dei pattern di programmazione e di race condition dall'interfaccia pubblica
- Analisi di due applicazioni reali per l'invio di SMS, entrambe sono risultate vulnerabili

Conclusioni

Contributi

- Studio dell'impatto delle race condition in applicazioni web
- Nuova tecnica di rilevamento
- Realizzato un prototipo sperimentale per applicazioni PHP
- Individuate diverse race condition in programmi open-source
- Progettata una tecnica per massimizzare la probabilità di riuscita di un attacco su un'applicazione remota

Conclusioni

Contributi

- Studio dell'impatto delle race condition in applicazioni web
- Nuova tecnica di rilevamento
- Realizzato un prototipo sperimentale per applicazioni PHP
- Individuate diverse race condition in programmi open-source
- Progettata una tecnica per massimizzare la probabilità di riuscita di un attacco su un'applicazione remota

Sviluppi futuri

- Considerare le interazioni tra diversi script
- Utilizzare tecniche di analisi statica per estrarre più informazioni riguardo la logica delle applicazioni

Race condition in applicazioni web

Grazie per l'attenzione!

Davide Marrone