Studi *Web Race Condition* Sebagai Acuan Pembuatan Modul Praktikum

Tio Hana Lolita B.L.T.[1)], Herman Kabetta[2)]
(1) *Rekayasa Kriptografi, Politeknik Siber dan Sandi Negara, tio.hana@poltekssn.ac.id*
(2) *Politeknik Siber dan Sandi Negara, herman.kabetta@poltekssn.ac.id*

Abstrak

Serangan race condition dapat dilakukan pada aplikasi yang berjalan secara paralel maupun aplikasi runtunan. Race condition pada aplikasi web dapat memberikan dampak yang merugikan, namun studi terkait race condition pada web masih terbilang minim. Pada penelitian ini, dilakukan serangan race condition pada aplikasi runtunan berupa aplikasi transaksi perbankan sederhana yang memiliki sebuah basis data. Serangan dilakukan dengan mengirimkan shell script yang berisi perintah curl secara paralel. Dengan menggunakan konsep time-of-check time-of-use (TOCTOU), serangan berhasil mengeksploitasi delay waktu dan menyebabkan aplikasi menghasilkan output yang tidak sesuai. Aplikasi kemudian diberikan mitigasi berupa table-level locking dan row-level locking. Hasil studi kemudian dijadikan suatu modul praktikum untuk membantu mahasiswa memahami bagaimana cara kerja web race condition dan langkah mitigasinya. Modul disimulasikan dengan menggunakan metode pra-eksperimen one-group pretest-posttest pada sejumlah responden mahasiswa. Hasil simulasi menunjukkan bahwa simulasi berhasil meningkatkan pemahaman mahasiswa mengenai race condition pada web. Berdasarkan survey yang dilakukan, terdapat peningkatan nilai posttest rata-rata sebesar 2,3 poin dari nilai pretest.

Kata kunci: aplikasi web, *locking*, modul praktikum, *one-group pretest-posttest*, *race condition*, TOCTOU.

1 PENDAHULUAN

Internet telah menjadi bagian yang tidak dapat terlepas dari kehidupan sehari-hari manusia. Penggunaan internet yang sangat terintegrasi dengan kehidupan manusia dan sering dipercayakan untuk mengelola aset penting membuatnya tidak lepas dari ancaman.

Menurut *Open Web Application Security Project* (OWASP), beberapa ancaman yang dialami aplikasi *web* adalah *SQL Injection*, *cross site Scripting* (XSS), *Credential Stuffing*, *Race Condition* dan lain-lain [1]. Salah satu ancaman yang sering tidak diperhatikan adalah *race condition* [2]. *Race condition* merupakan suatu *error* saat terdapat dua alur kerja aplikasi yang memanipulasi *shared data* secara bersamaan tanpa dilakukannya sinkronisasi [3]. *Race condition* dapat menyebabkan program *crash*, kehilangan data atau menghasilkan *output* yang tidak diinginkan [4].

Josip Franjković pada tahun 2015 berhasil melakukan serangan *race condition* pada Facebook dan DigitalOcean. Pada DigitalOcean, Franjković menggunakan *race condition* untuk mendapatkan uang dengan menggunakan satu kode promo yang sama berkali-kali. Hal ini menunjukkan bahwa *race condition* dapat dieksploitasi menjadi ancaman yang serius, baik secara materiil maupun finansial [5].

Salah satu mata kuliah di Perguruan Tinggi XYZ yang mempelajari terkait *web* beserta metode pemrogramannya adalah Pemrograman Web. Namun, mata kuliah Pemrograman Web masih belum mempelajari terkait serangan *race condition* pada *web*. Hal ini menyebabkan mahasiswa Perguruan Tinggi XYZ tidak memiliki pengetahuan yang memadai terkait serangan tersebut. Maka dari itu diperlukan suatu modul praktikum untuk membantu mahasiswa dalam mempelajari dan mempersiapkan diri dalam menghadapi serangan *race condition* pada *web*.

Peneliti melakukan studi terkait serangan dan mitigasi *race condition* pada *web*. Hasil studi yang disusun dapat dijadikan suatu modul praktikum bagi mahasiswa. Modul praktikum bertujuan untuk memberikan gambaran kepada mahasiswa mengenai cara kerja *web race condition* dan langkah mitigasinya.

2 LANDASAN TEORI

2.1 *Race Condition*

Race condition adalah salah satu permasalahan utama pada sistem paralel dan terdistribusi [6]. *Race condition* terjadi ketika program dieksekusi, lokasi memori yang sama diakses 2 kali pada waktu yang sama [7]. *Race condition* dapat terjadi pada sistem *web* karena adanya *delay* pada pemrosesan *asynchronous* (tidak terjadi bersamaan) dan waktu transfer pada jaringan, sementara perilaku sistem dapat berubah-ubah tergantung pada urutan eksekusi dari proses yang bekerja bersamaan [6].

2.2 CWE (*Common Weakness Enumeration*)

Level abstraksi awal terkait *race condition* tercatat pada CWE-691: *Insufficient Control Flow Management*. Tercatat bahwa kelemahan ini berasal dari kode yang tidak mengatur secara baik *control flow* selama proses eksekusi sehingga *control flow*

dapat dimodifikasi secara tak terduga. Kelemahan ini kemudian diturunkan pada CWE-362: *Concurrent Execution using Shared Resource with Improper Synchronization* ('*Race Condition*'). Menurut CWE-362, kelemahan ini terjadi karena pada program terdapat suatu urutan kode yang dapat dijalankan secara *concurrent* dan kode tersebut membutuhkan akses sementara dan ekslusif pada suatu *shared resource*, namun kode lain dapat memodifikasi *shared resource* karena adanya suatu *timing window*. CWE-362 mencatat bahwa *race condition* dapat memberikan dampak pada *availability* (ketersediaan), *confidentiality* (kerahasiaan), dan *integrity* (keutuhan).

Kelemahan pada CWE-362 kemudian dispesifikasikan dan diturunkan kembali pada CWE-367 *Time-of-check Time-of-use* (TOCTOU) *Race Condition*. CWE-367 mencatat bahwa kelemahan ini terjadi ketika program memeriksa keadaan *resource* sebelum menggunakan *resource*, tapi keadaan tersebut dapat berubah antara waktu ketika diperiksa dan ketika digunakan. Hal tersebut dapat menyebabkan program menjalankan proses yang *invalid* dikarenakan *resource* dalam keadaan yang tidak terduga. Kelemahan ini dapat terjadi program dengan *multi-thread*. CWE-367 mencatat bahwa kelemahan ini dapat memberikan dampak pada *integrity* (keutuhan) dan *non-repudiation* (nirsangkal)

2.3 CAPEC (*Common Attack Pattern Enumeration and Classification*)

CAPEC-26: *Leveraging Race Condition* membahas mengenai serangan *race condition*, baik pada program aplikasi maupun pada *web*. CAPEC-26 diberikan *level* tinggi (*high*) pada kemungkinan serangan dan tingkat dampak. CAPEC-26 menyebutkan bahwa *race condition* dapat menyerang layanan *confidentiality* (kerahasiaan), *access control* (kontrol akses), *authorization* (otorisasi), dan *integrity* (keutuhan). CAPEC-26 kemudian diturunkan pada CAPEC-29: *Leveraging Time-of-Check and Time-of-Use* (TOCTOU) *Race Conditions*. Serangan ini dapat dilakukan ketika suatu *resource* dapat diakses atau dimodifikasi secara *concurrent* oleh beberapa proses. Penyerang dapat mengubah *resource* di antara waktu pengecekan *resource* dengan waktu penggunaan.

2.4 *One-group Pretest-Posttest*

Salah satu cara bagi suatu peneliti untuk mengumpulkan data pada suatu eksperimen adalah melakukan suatu pra-eksperimen. Pada tahap pra-eksperimen akan ada suatu tes yang melibatkan sejumlah partisipan. Partisipan akan diobservasi setelah diberikan suatu intervensi atau perlakuan yang diasumsikan akan menyebabkan perubahan. *One-group pretest-posttest* merupakan salah satu metode pra-eksperimen, pada metode ini terdapat satu grup partisipan yang akan melakukan tes sebanyak dua kali [8]. Tes yang pertama dilakukan sebelum eksperimen dilakukan dan disebut dengan *pretest*. Tes yang kedua adalah *posttest* yang dilakukan setelah eksperimen dilakukan. *Pretest* dan *posttest* dibuat sama agar perubahan yang diamati peneliti dapat terlihat jelas. Peneliti tidak mengontrol variabel eksternal selain eksperimen yang mungkin turut mengubah hasil.

3 METODE PENELITIAN

Metode penelitian dilakukan dalam tiga tahap, yaitu tinjauan literatur, eksperimen, dan penyusunan modul praktikum.

3.1 Tinjauan Literatur

Tahap yang pertama adalah tinjauan literatur terkait *race condition* pada *web* yang mengacu pada Baumeister [9]. Tinjauan literatur mencakup beberapa kegiatan, yaitu:

a. Pengumpulan referensi yang memuat topik *race condition* pada *web*;
b. Mengekstraksi informasi terkait metode serangan yang pernah dilakukan dan bentuk kerentanan aplikasi *web* yang diserang dari referensi;
c. Mengintegrasikan hasil ekstraksi informasi menjadi suatu model *web* yang rentan akan serangan *race condition.*

3.2 Eksperimen

Tahap kedua adalah eksperimen terkait *race condition* pada *web*. Pada tahap kedua ini, metode yang dilakukan diadopsi dari penelitian Rob J. van Emous terkait pembangunan alat *testing* untuk *race condition* pada *web* [2]. *Pada* tahap eksperimen, dilakukan beberapa kegiatan, yaitu:

a. Membuat aplikasi *web* yang rentan akan serangan *race condition* berdasarkan model yang disusun pada tahapan tinjauan literatur;
b. Melakukan serangan *race condition* pada aplikasi *web*;
c. Menganalisis hasil serangan *race condition* pada aplikasi *web*;
d. Menyusun metode pemrograman yang dapat memitigasi serangan *race condition* pada *web*;
e. Melakukan validasi terhadap metode mitigasi yang telah disusun.

Serangan *race condition* pada *web* akan dilakukan menggunakan curl. Curl digunakan untuk mengirimkan data dari atau ke suatu *server* [10]. Perintah yang memuat informasi untuk mengubah data pada aplikasi *web* yang akan diserang kemudian dimasukkan ke dalam curl untuk dieksekusi. Suatu *shell script* akan dibuat untuk menjalankan curl sebanyak mungkin untuk kemudian menyebabkan *race condition.*

3.3 Penyusunan Modul Praktikum

Tahap yang terakhir adalah penyusunan modul praktikum mengenai serangan *race condition* pada *web* berdasarkan hasil pada tahap studi literatur dan

tahap eksperimen yang terdokumentasi. Modul praktikum disusun dengan mengacu pada susunan modul praktikum *race condition* milik SEED Labs [11]. Modul praktikum yang telah disusun kemudian disimulasikan dan dievaluasi. Evaluasi praktikum dilakukan melalui pretest/posttest pengetahuan menggunakan kuesioner.

4 HASIL DAN PEMBAHASAN

4.1 Perancangan Aplikasi

Race condition dapat terjadi pada dua keadaan. Keadaan yang pertama adalah ketika suatu aplikasi web memang dibuat dalam bentuk paralel. Keadaan kedua adalah ketika suatu aplikasi *web* yang disusun dengan kode sekuensial namun dapat dieksekusi secara paralel atau bersamaan [12]. Pada penelitian ini, aplikasi transaksi bank yang akan diserang merupakan suatu aplikasi *web* dengan kode sekuensial. Maka dari itu, serangan akan dijalankan secara paralel.

Serangan *race condition* pada penelitian ini akan difokuskan untuk mengubah jumlah saldo. Saldo akan disimpan dalam basis data. Maka dari itu, aplikasi yang akan diserang harus memiliki basis data untuk menyimpan informasi. Untuk memastikan serangan dapat dilakukan, basis data tidak boleh menggunakan mekanisme keamanan tambahan apapun.

Berdasarkan CWE 367, *race condition* menggunakan konsep *time-of-check time-of-use* (TOCTOU). Pada dasarnya, konsep TOCTOU terjadi ketika suatu program memeriksa keadaan *resource* sebelum digunakan, namun keadaan *resource* dapat berubah di antara waktu ketika diperiksa dengan ketika digunakan. Apabila *resource* berubah, hasil dari eksekusi juga akan berubah atau bahkan invalid. Maka dari itu, pada penelitian ini perlu digunakan dua PC antara aplikasi dengan penyerang, untuk menciptakan *delay* waktu dalam pengecekan dengan penggunaan nilai pada basis data. *Delay* waktu itu akan dieksploitasi ketika serangan dijalankan secara paralel.

Attack surface dari serangan *web race condition* pada praktikum ini adalah basis data yang menyimpan seluruh informasi. Serangan akan mengeksploitasi *delay* waktu saat pengecekan saldo dengan pemindahan saldo. *Attack vector* yang digunakan pada *payload* adalah *request* untuk mengirimkan saldo. *Threat model* berupa *data flow diagram* dapat dilihat pada Gambar 1.

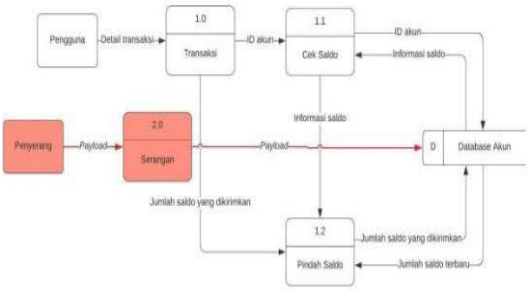
4.2 Pembuatan Aplikasi

Aplikasi yang akan diserang adalah aplikasi sederhana mengenai transaksi pada suatu bank. Aplikasi terdiri dari satu tampilan yang dapat dilihat pada Gambar 2. Fungsi utama dari aplikasi adalah untuk mengirimkan sejumlah saldo dari satu rekening ke rekening lain pada suatu akun bank yang telah terdaftar.

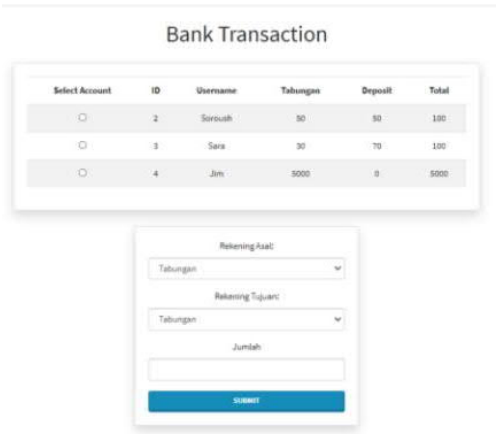
Langkah-langkah untuk mengirimkan saldo adalah sebagai berikut:

a. Pengguna menekan *radio button* untuk memilih akun yang akan digunakan.
b. Pengguna memilih Rekening Asal. Rekening Asal memiliki dua pilihan, yaitu Tabungan dan Deposit.
c. Pengguna memilih Rekening Tujuan. Rekening Tujuan tidak boleh sama dengan Rekening Asal yang telah dipilih.
d. Pengguna memasukkan jumlah saldo yang akan dikirimkan.
e. Pengguna menekan tombol Submit dan proses pengiriman selesai.

Pada saat pengguna menekan tombol Submit, sistem akan memeriksa apakah saldo yang terdapat pada rekening asal mencukupi untuk dikirim sesuai jumlah yang dimasukkan oleh pengguna. Apabila saldo mencukupi untuk dikirim maka sistem akan memindahkan saldo ke Rekening Asal ke Rekening Tujuan.



Gambar 1. *Data Flow Diagram* dari Serangan *Web Race Condition*



Gambar 2. Tampilan aplikasi yang akan diserang

4.3 Lingkungan Implementasi

Penelitian ini membutuhkan 2 perangkat keras yang dibagi menjadi PC 1 dan PC 2. PC 1 bertugas sebagai penyedia aplikasi dan basis data, sedangkan PC 2 sebagai penyerang yang akan menjalankan *shell script.* Spesifikasi yang digunakan masing -masing PC dapat dilihat pada Tabel 1.

Tabel 1. Spesifikasi PC 1 dan PC 2

Spesifikasi	PC 1	PC 2
Model	ASUS X455LA	HP Convertible x360 11-ab0XX
Processor	Intel(R) Core(TM) i3-4030U CPU @1.90GHz (4 CPUs)	Intel(R) Celeron(R) CPU N3060 @1.60GHz (2 CPUs)
RAM	4096 MB	4096 MB
Sistem Operasi	Windows 8.1 Pro 64-bit (6.3, Build 9600)	Windows 10 Home 64-bit (10.0, Build 17763)
Aplikasi	- XAMP 7.4.3 - MySQL 8 - Apache 2.4.41	- XAMP 7.4.3 - Apache 2.4.41 - Cygwin 3.1.4

4.4 Serangan dan Analisis

Suatu *shell script* membagi ke beberapa *thread* yang akan menjalankan beberapa perintah curl secara berurutan. Perintah curl yang digunakan dalam serangan adalah transaksi untuk mengirimkan saldo dari Rekening Asal ke Rekening Tujuan.

Berdasarkan aplikasi transaksi bank yang akan diserang, pengguna menentukan ID, Rekening Asal, Rekening Tujuan, dan jumlah saldo dengan mengisi form pada laman. Untuk memimik cara kerja aplikasi tersebut, perintah curl akan menggunakan opsi –d. Laman dokumentasi curl menjelaskan bahwa opsi –d akan mengirimkan data dalam *request* POST ke *server* HTTP seperti apa yang aplikasi lakukan ketika pengguna mengisi form pada laman dan menekan tombol Submit. Apabila data yang dikirimkan berjumlah lebih dari satu, data bisa digabung dengan simbol & sebagai pemisah .

Skenario yang digunakan adalah ada 10 *thread* yang akan mengirimkan 5 perintah pengiriman saldo sebesar 7. Saldo dikirimkan dari rekening tabungan ke rekening deposit milik akun dengan ID 4.

```
#! /bin/bash
for y in `seq 1 10` do
(for x in `seq 1 5`
   do curl –d
   "acct=4&from=tabungan&to=deposit&
   amount=7" [IP Address]/bank/
   done)&
done
```

Pada proses penelitian ini, serangan dilakukan sebanyak 5 kali. Saldo rekening deposit yang seharusnya didapatkan dalam kondisi normal adalah 350. Namun, saldo rekening deposit yang didapatkan setelah serangan tidak ada yang berjumlah 350. Hal ini dapat dilihat pada Tabel 2.

Tabel 2. Hasil Saldo Rekening Deposit dan Tabungan Setelah Serangan

Serangan	Saldo Rekening Deposit	Saldo Rekening Tabungan
I	154	4.846
II	280	4.720
III	259	4.741
IV	245	4.755
V	266	4.734

Tahap analisis yang pertama adalah memeriksa *access log* dari *server* Apache. Dari *log* yang didapat, *log* yang berasal dari *shell script* serangan menyatakan *referer* kepada perintah curl yang digunakan. Berdasarkan Tabel 3, terdapat 50 *request* dengan *referer curl*, sesuai dengan jumlah *request* yang dikirimkan. Hal ini menyatakan bahwa seluruh *request* diterima dan dieksekusi oleh Apache. Namun, hasil pada Tabel 2 menyatakan bahwa tidak semua *request* tereksekusi, sehingga mengakibatkan saldo rekening deposit yang berjumlah kurang dari 350. Maka dari itu, perlu dilakukan analisis lebih lanjut.

Tabel 3. *Access Log* dari Apache

No.	Alamat IP	Waktu	*Request*	*Referer*
6.	192.168.10.106	06/Apr/2020 20:22:03	POST/race/ HTTP/1.1	curl/7.55.1
7.	192.168.10.106	06/Apr/2020 20:22:03	POST/race/ HTTP/1.1	curl/7.55.1
⋮	⋮	⋮	⋮	⋮
53.	192.168.10.106	06/Apr/2020 20:22:08	POST /race/ HTTP/1.1	curl/7.55.1
54.	192.168.10.106	06/Apr/2020 20:22:09	POST /race/ HTTP/1.1	curl/7.55.1
55.	192.168.10.106	06/Apr/2020 20:22:09	POST /race/ HTTP/1.1	curl/7.55.1

File *log* dari MySQL didapatkan pada tabel `general_log` dalam basis data `mysql`. Untuk memfokuskan penyimpanan *log* kepada serangan spesifik yang dijadikan sampel, penyimpanan dilakukan persis sebelum serangan dilakukan. Pengaktifan *logging* dilakukan dengan menjalankan *query* SQL berikut.

```
SET global general_log = 1;
SET global log_output = „table“;
```

Setelah serangan dilakukan, *logging* dapat dinonaktifkan untuk mengurangi penyimpanan data yang tidak diperlukan. Penonaktifan *logging* dapat dilakukan dengan menjalankan query SQL berikut.

```
SET global general_log = 0;
```

Petikan *Log* MySQL yang berjalan normal dapat dilihat pada Tabel 4, dan petikan *Log* MySQL dengan *thread* berkolisi pada Tabel 5.

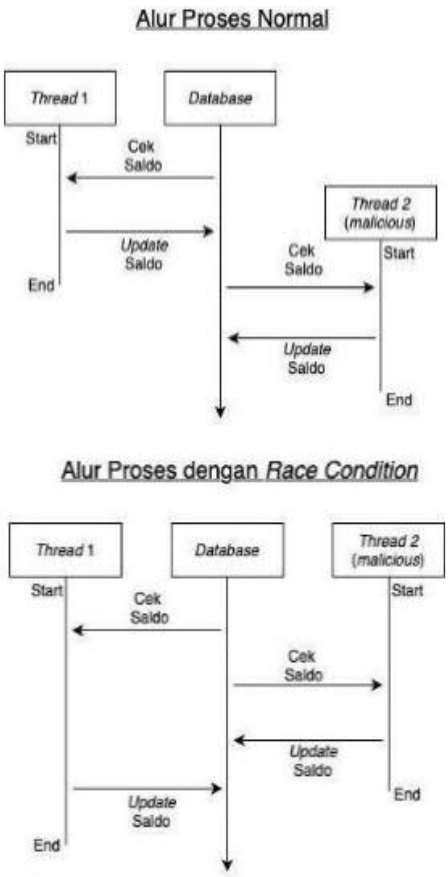
Terdapat perbedaan alur proses normal dan alur proses dengan *Race Condition* seperti terlihat pada Gambar 3. Setiap *thread* akan membuat suatu koneksi TCP untuk mengirimkan data. Setelah mendapatkan balasan, *thread* akan menutup koneksi untuk kemudian dibuka kembali saat akan mengirimkan data. Semua data memang dikirimkan oleh tiap *thread* secara bersamaan, namun *request* HTTP tidak akan sampai secara bersamaan, walaupun hanya sepersekian detik. Hal ini dapat disebabkan karena *server web* seperti Apache akan memroses *query* secara *asynchronous*. Ketika *thread* sampai pada basis data, tidak ada pengaturan yang mencegah dua atau lebih *thread* mengakses nilai yang sama sehingga akhirnya menyebabkan saling bertabrakan.

Tabel 4. Petikan *Log* MySQL yang Berjalan Normal

event_time	user_host	thread _id	server _id	command _type	argument
06/04/2020 20:22:03.5	[root]@localhost[::1]	380	1	Connect	root@localhost as anonymous on bankk
06/04/2020 20:22:03.5	[root]@localhost[::1]	380	1	Query	SELECT * FROM account WHERE id = 4
06/04/2020 20:22:03.5	root[root]@localhost[::1]	380	1	Query	UPDATE `account` SET tabungan = 4993, deposit = 7 WHERE `account`,`id` = 4
06/04/2020 20:22:03.7	root[root]@localhost[::1]	380	1	Query	SELECT * FROM account
06/04/2020 20:22:03.7	root[root]@localhost[::1]	380	1	Quit	

Tabel 5. Petikan *Log* MySQL dengan *Thread* Berkolisi

event_time	user_host	thread _id	server _id	command _type	argument
06/04/2020 20:22:05.4	[root]@localhost[::1]	394	1	Connect	root@localhost as anonymous on bankk
06/04/2020 20:22:05.4	root[root]@localhost[::1]	394	1	Query	SELECT * FROM account WHERE id = 4
06/04/2020 20:22:05.4	[root]@localhost[::1]	395	1	Connect	root@localhost as anonymous on bankk
06/04/2020 20:22:05.4	root[root]@localhost[::1]	395	1	Query	SELECT * FROM account WHERE id = 4
06/04/2020 20:22:05.4	root[root]@localhost[::1]	394	1	Query	UPDATE `account` SET tabungan = 4930, deposit = 70 WHERE `account`,`id` = 4
06/04/2020 20:22:05.4	root[root]@localhost[::1]	395	1	Query	UPDATE `account` SET tabungan = 4930, deposit = 70 WHERE `account`,`id` = 4
06/04/2020 20:22:05.4	root[root]@localhost[::1]	395	1	Query	SELECT * FROM account
06/04/2020 20:22:05.4	root[root]@localhost[::1]	395	1	Quit	
06/04/2020 20:22:05.5	root[root]@localhost[::1]	395	1	Query	SELECT * FROM account



Gambar 3. Perbedaan Alur Proses Normal dan Alur Proses dengan *Race Condition*

4.5 Mitigasi

Serangan *race condition* yang dilakukan terjadi pada bagian basis data. Tidak ada pengaturan pada basis data yang mencegah dua atau lebih *thread* mengakses nilai yang sama dan menyebabkan kolisi. Untuk mencegah hal tersebut, dapat dilakukan *locking* pada basis data. *Locking* merupakan suatu mekanisme untuk mengunci suatu *file* atau *record* [13]. *Locking* pada basis data dapat dilakukan dalam 2 cara, *table-level* dan *row-level.*

4.5.1 *Table-level Locking*

Locking dapat dilakukan pada level tabel. Ketika *thread* mengakses tabel, *thread* akan mengunci tabel agar tidak bisa diakses oleh *thread* lain. *Thread* lain dapat mengakses tabel tersebut setelah kunci pada tabel dibuka. Laman dokumentasi MySQL menyatakan bahwa terdapat 2 jenis *table-level locking*, yaitu mode WRITE dan READ [14]. Untuk *lock* READ, *session* lain yang juga sedang mengakses tidak bisa memodifikasi tabel namun dapat membaca isi tabel. Untuk *lock* WRITE, *session* lain tidak dapat membaca maupun memodifikasi tabel.

Locking dan *unlocking* harus dilakukan oleh tiap-tiap *thread*. Proses tersebut dapat dilakukan melalui *query* SQL yang dieksekusi oleh aplikasi. Ketika aplikasi membuka koneksi, *lock* WRITE diaktifkan.

```
$conn = new mysqli($servername,
$username, $password, $dbName);
if ($conn->connect_error) {
die("Connection failed: " .
$conn->connect_error);}
$conn->query("lock table account
write");
```

Setelah *lock* diaktifkan, tabel akan secara eksklusif dapat diakses oleh *thread* dengan *session* yang sedang aktif. Operasi pemindahan saldo pun dapat dilakukan tanpa intervensi dari *thread* lain. Ketika operasi telah selesai dilakukan, *lock* harus dinonaktifkan.

```
if ($conn->query($sql) === TRUE)
{array_push($success, 'Transaction
Success!'); } else
{array_push($error, 'Transaction
Error!');}
$conn->query("unlock table");
```

4.5.2 *Row-level Locking*

Ketika dilakukan *table-level locking*, keseluruhan tabel tidak akan bisa diakses oleh *thread* lain. Hal ini dapat mengurangi performa dari aplikasi. Apabila suatu *thread* hendak mengakses suatu baris yang sebenarnya tidak sedang diakses oleh *thread* lain, proses akan tertunda. Skenario tersebut dapat terjadi pada saat 2 pengguna menggunakan aplikasi transaksi bank secara bersamaan. Pengguna 1 hendak bertransaksi dengan ID 2, sedangkan pengguna 2 bertransaksi dengan ID 3. Transaksi yang dilakukan oleh pengguna 1 tidak akan mempengaruhi transaksi yang dilakukan oleh pengguna 2. Namun, dikarenakan keseluruhan tabel dikunci maka transaksi pengguna 2 akan tertunda oleh karena transaksi yang dilakukan pengguna 1.

Salah satu jenis *locking* yang dapat dilakukan pada basis data dengan skala yang lebih kecil adalah *row-level locking.* Kunci hanya akan digunakan pada baris yang sedang diakses oleh *thread* dengan *session* yang aktif. Baris yang sedang tidak dikunci masih bisa diakses dengan bebas oleh *thread* lain.

Proses diawali dengan membuat suatu *transaction* baru. Pembuatan *transaction* baru dilakukan setelah aplikasi membuka koneksi kepada basis data.

```
$conn = new mysqli($servername,
$username, $password, $dbName);
if ($conn->connect_error)
{die("Connection failed: " .
$conn->connect_error);}
$conn->query(“BEGIN”);
```

Locking dilakukan ketika aplikasi mengambil informasi terkait akun dengan ID terpilih dengan menambahkan parameter FOR UPDATE. *Thread*

aktif kemudian dapat memodifikasi akun tersebut, dan *thread* dari operasi lain dapat memodifikasi akun yang lain.

```
$sql = "SELECT * FROM account WHERE id = " . $_POST['acct'] . "FOR UPDATE";
$result = $conn->query($sql);
```

Setelah keseluruhan transaksi selesai, perlu dilakukan penetapan transaksi sehingga perubahan apapun yang dilakukan selama transaksi dapat ditetapkan dan diterapkan untuk session selanjutnya menggunakan perintah *commit*.

```
$conn->query("COMMIT");
```

4.6 Pengujian Mitigasi

4.6.1 *Table-level Locking*

Ketika *table-level locking* diaktifkan, hanya 1 *thread* yang dapat mengakses tabel dalam satu waktu. *Locking* dapat dilakukan melalui eksekusi *query* SQL yang dikirimkan sistem pada aplikasi. Pada saat proses pemindahan saldo dilakukan, tabel *account* akan dikunci terlebih dahulu sebelum nilai dibaca dan dimodifikasi. Setelah saldo berhasil dipindahkan dan nilai pada tabel dimodifikasi, barulah kunci dibuka sehingga tabel dapat diakses oleh *thread* selanjutnya. Tabel 6 menyajikan petikan data *Log* MySQL setelah mitigasi *Table-level Locking.*

4.6.2 *Row-level Locking*

Sama halnya dengan *table-level locking*, ketika ada dua atau lebih *thread* mengakses akun yang sama, maka *thread* yang lebih dulu melakukan *locking* yang diperbolehkan melakukan modifikasi terlebih dahulu. Setelah saldo berhasil dipindahkan dan nilai pada tabel dimodifikasi, barulah kunci dibuka sehingga tabel dapat diakses oleh *thread* selanjutnya. Tabel 7 menyajikan petikan data *Log* MySQL setelah mitigasi *Row-level Locking*.

5 PEMBUATAN DAN SIMULASI MODUL PRAKTIKUM

5.1 Pembuatan Modul Praktikum

Hasil studi yang telah dilakukan pada serangan *race condition* pada *web* akan dijadikan acuan dalam membuat suatu modul praktikum. Modul praktikum bertujuan untuk memberikan gambar mengenai serangan *web race condition*. Mahasiswa akan dipandu untuk melakukan serangan dan menganalisis dampak yang diberikan, serta menerapkan langkah - langkah yang dapat memitigasi serangan.

Tabel 6. Petikan Data *Log* MySQL Setelah Mitigasi *Table-level Locking*

event_time	user_host	thread _id	server _id	command _type	argument
09/04/2020 16:54:15.5	[root]@localhost[::1]	213	1	Connect	root@localhost as anonymous on bankk
09/04/2020 16:54:15.5	root[root]@localhost[::1]	213	1	Query	lock table account write
09/04/2020 16:54:15.5	root[root]@localhost[::1]	214	1	Connect	root@localhost as anonymous on bankk
09/04/2020 16:54:15.5	root[root]@localhost[::1]	213	1	Query	SELECT * FROM account WHERE id = 4
09/04/2020 16:54:15.5	root[root]@localhost[::1]	214	1	Query	lock table account write
09/04/2020 16:54:15.5	root[root]@localhost[::1]	213	1	Query	UPDATE `account` SET tabungan = 4720, deposit = 280 WHERE `account`.`id` = 4
09/04/2020 16:54:15.5	root[root]@localhost[::1]	213	1	Query	SELECT * FROM account
09/04/2020 16:54:15.5	root[root]@localhost[::1]	214	1	Query	SELECT * FROM account WHERE id = 4
09/04/2020 16:54:15.5	root[root]@localhost[::1]	214	1	Query	UPDATE `account` SET tabungan = 4713, deposit = 287 WHERE `account`,`id` = 4
09/04/2020 16:54:15.5	root[root]@localhost[::1]	214	1	Query	`account` unlock table
09/04/2020 16:54:15.6	root[root]@localhost[::1]	214	1	Query	SELECT * FROM account
09/04/2020 16:54:15.6	root[root]@localhost[::1]	213	1	Quit	
09/04/2020 16:54:15.6	root[root]@localhost[::1]	214	1	Quit	

Tabel 7. Petikan Data *Log* MySQL Setelah Mitigasi *Row-level Locking*

event_time	user_host	thread_id	command_type	argument
09/04/2020 16:25:43.7	root[root]@localhost [::1]	354	Connect	root@localhost as anonymous on bankk
09/04/2020 16:25:43.7	root[root]@localhost [::1]	354	Query	BEGIN
09/04/2020 16:25:43.7	root[root]@localhost [::1]	354	Query	SELECT * FROM account WHERE id = 4 FOR UPDATE
09/04/2020 16:25:43.9	root[root]@localhost [::1]	356	Connect	root@localhost as anonymous on bankk
09/04/2020 16:25:43.9	root[root]@localhost [::1]	356	Query	BEGIN
09/04/2020 16:25:43.9	root[root]@localhost [::1]	356	Query	SELECT * FROM account WHERE id =4 FOR UPDATE
09/04/2020 16:25:44.0	root[root]@localhost [::1]	354	Query	UPDATE `account` SET tabungan = 4741, deposit = 259 WHERE `account`.`id` = 4
09/04/2020 16:25:44.0	root[root]@localhost [::1]	354	Query	COMMIT
09/04/2020 16:25:44.0	root[root]@localhost [::1]	356	Query	UPDATE `account` SET tabungan = 4734, deposit = 266 WHERE `account`.`id` = 4
09/04/2020 16:25:44.0	root[root]@localhost [::1]	356	Query	COMMIT

Simulasi modul praktikum akan dilakukan dengan menggunakan dua PC, PC fisik dan PC virtual. PC fisik digunakan untuk menjalankan aplikasi yang diserang, sehingga membutuhkan layanan *web server* dan basis data. Layanan yang digunakan adalah XAMPP. PC virtual digunakan untuk menjalankan *shell script* berupa *payload*.

Karena *payload* berupa perintah curl yang merupakan perintah dasar dari sistem operasi Linux, maka untuk mempermudah partisipan, PC virtual yang dijalankan adalah Ubuntu Server. Ubuntu Server dipilih karena memiliki ukuran yang lebih kecil daripada sistem operasi berbasis Linux lainnya.

5.2 Simulasi Modul Praktikum

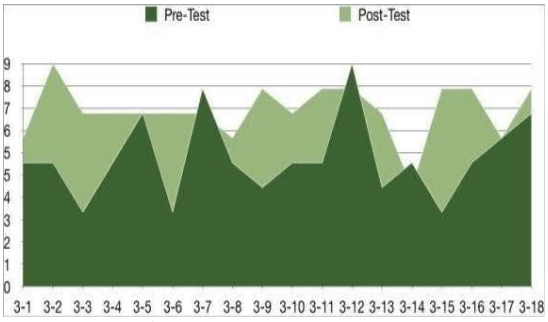
Modul praktikum dibuat dengan mengacu pada modul praktikum serangan *race condition* milik SEED Labs. Modul praktikum terdiri dari tiga bagian utama. Bagian pertama adalah Pendahuluan. Pada bagian ini, diberikan penjelasan singkat terkait materi yang akan dipraktikan. Bagian kedua adalah Persiapan. Bagian Persiapan menjelaskan hal-hal apa saja yang dibutuhkan mahasiswa dalam melakukan praktikum. Bagian terakhir adalah Praktikum yang berisikan langkah-langkah yang perlu dilakukan selama praktikum.

Modul praktikum yang telah disusun kemudian disimulasikan kepada mahasiswa Perguruan Tinggi XYZ. Karena fokus pada materi adalah pada bagian pemrograman, maka partisipan pada simulasi ini dibatasi pada mahasiswa dengan bidang minat Rekayasa Perangkat Lunak Kripto. Simulasi diikuti oleh 35 orang mahasiswa yang dibagi menjadi 2 kelompok. Kelompok 1 sebanyak 18 orang mengerjakan simulasi terlebih dahulu, yang kemudian diikuti oleh kelompok 2 sebanyak 17 orang.

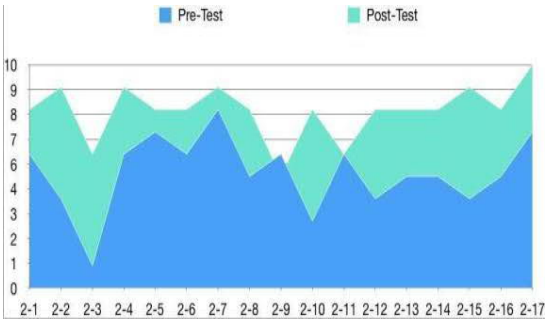
Simulasi dilakukan dengan menggunakan pra-eksperimen. Metode pre-eksperimen yang digunakan adalah *one-group pretest-posttest.* Sebelum diberikan modul praktikum, mahasiswa diberikan lembar *pretest* yang harus diisi tanpa bantuan apapun. Setelah *pretest* dilakukan dan dikumpulkan, mahasiswa diberikan modul praktikum dan memulai simulasi. Jika modul praktikum sudah selesai dikerjakan, mahasiswa mengerjakan lembar *posttest* yang berisi pertanyaan yang sama dengan lembar *pretest*.

5.3 Hasil dan Analisis

Berdasarkan perbandingan rata-rata dari skor *pretest* dan *posttest*, terjadi peningkatan rata-rata skor sebanyak 2,3 poin. Pada Gambar 4 dan Gambar 5 juga dapat dilihat bahwa terdapat 28 orang dari keseluruhan 35 orang partisipan mengalami peningkatan pada skor *posttest* dari skor *pretest.* Skor tertinggi *posttest* hingga mencapai nilai 10 pada kelompok 2. Hal ini menunjukkan bahwa modul praktikum yang telah disimulasikan berhasil membantu 80% dari partisipan dalam memahami materi terkait serangan *race condition* pada *web*.



Gambar 4. Grafik Perbandingan Skor Pretest dan Posttest Kelompok 1



Gambar 5. Grafik Perbandingan Skor Pretest dan Posttest Kelompok 2

6 KESIMPULAN

Race condition dapat terjadi pada aplikasi yang memang berjalan secara paralel maupun beruntun namun dapat dieksploitasi secara paralel. Dengan memanfaatkan konsep *Time-of-Check Time-of-Use* (TOCTOU), penyerang dapat mengeksploitasi *delay* waktu antara pengecekan *resource* dengan penggunaan *resource* untuk mengubah kondisi sumberdaya yang kemudian menyebabkan aplikasi menghasilkan *output* yang tidak diinginkan. Dalam penelitian ini, aplikasi yang diserang adalah aplikasi transaksi perbankan sederhana yang memiliki basis data untuk menyimpan informasi terkait akun dan saldo masing-masing rekening. Serangan *race condition* melalui suatu *shell script* yang mengirimkan perintah curl secara paralel. Serangan yang dilakukan menghasilkan jumlah saldo yang diterima tidak sesuai dengan yang dikirimkan. Untuk menghindari serangan tersebut, dilakukan mitigasi berupa *table-level locking* dan *row-level*.

Hasil studi terhadap serangan *race condition* pada *web* kemudian dijadikan acuan dalam pembuatan suatu modul praktikum untuk mahasiswa. Modul praktikum kemudian disimulasikan dengan menggunakan metode pra-eksperimen *one-group pretest-posttest*. Setelah dilakukan simulasi, hasil skor *pretest* dan *posttest* menunjukkan peningkatan rata-rata sebesar 2,3 poin. Hal ini menandakan bahwa modul praktikum yang dibuat berhasil membantu mahasiswa dalam memahami materi terkait *race condition* pada *web*.

REFERENSI

[1] OWASP, "Testing for Race Conditions (OWASP-AT-010)," [Online]. Available: www.owasp.org/index.php/Testing_for_Race_Conditions_(OWASPAT-010).

[2] R. J. van Emous, "Towards Systematic Black-box Testing for Exploitable Race," *Master's Thesis, University of Twente,* 2019.

[3] M. Abadi, C. Flanagan and S. N. Freund, "Types for Safe Locking," in *ACM Transactions on Programming Languages and Systems*, 2006, pp. 207-255.

[4] B. Petrov, M. Vechev, M. Sridharan and J. Dolby, "Race Detection for Web Applications," in *ACM SIGPLAN Notices*, 2012, pp. 251-262.

[5] J. Franjković, "Race conditions on Facebook, DigitalOcean and others (fixed)," 2015. [Online].https://josipfranjkovic.blogspot.com/2015/04/race-conditions-on facebook .html.

[6] K. Sathiyamurthy, S. A. Sophia, A. Asthalatha and P. Sujitha, "Automated Reasoning Tool for the Detection of Race Conditions of Web Services," in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, 2007.

[7] C. Flanagan and S. N. Freund, "FastTrack," in *Communications of the ACM*, 2010.

[8] N. J. Salkind, Encyclopedia of Research Design, Los Angeles, California: SAGE, 2010.

[9] R. F. Baumeister, "Writing a Literature Review," in *The Portable Mentor*, New York, Kluwer Academic/Plenum Publishers, 2012, pp. 119-132.

[10] D. Stenberg, "Everything curl," Alibris UK, 2018.

[11] Wenliang Du, "SEED Labs - Race Condition Vulnerability Lab," 2016. [Online]. Available: http://www.cis.syr.edu/~wedu/seed/Labs_12.04 /Software/Race_Condition/.

[12] R. Paleari, D. Marrone, D. Bruschi and M. Monga, "One Race Vulnerabilities in Web Applications," *Detection of Intrusions and Malware, and Vulnerability Assessment Lecture Notes in Computer Science*, pp. 126-142, 2008.

[13] J. Gray and A. Reuter, Distributed Transaction Processing: Concepts and Techniques, Morgan Kauffman, 1993.

[14] "MySQL 8.0 Reference Manual : 13.3.6 LOCK TABLES and UNLOCK TABLES Statements," [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/lock-tables.html.