Hints for exercises

Step1

Query 1.1: get the session of attack from application log

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-webapp.log")
| parse content, "'[' TIMESTAMP('dd/MMM/yyyy:HH:mm:ss.S'):event_time"
| fields event_time, content
| sort event_time asc
| filter event_time >= toTimestamp("2023-01-16 10:40:00")
```

Open *Logs* in new browser tab for next query

Query 1.2: what was the effective sql-statement used to verify end user?

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-sql.log")
| parse content, "TIMESTAMP('yyyy-MM-dd HH:mm:ss.S'):event_time ' 3 '
LD:statement"
| fields event_time, statement
| sort event_time asc
| filter event_time >= toTimestamp("2023-01-16 10:40:28") and event_time
<= toTimestamp("") //paste the end time here
```

Open *Logs* in new browser tab for next query

Query 1.3: find the ip-address of the attacker

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-access.log")
| parse content, "IPADDR:client_ip LD HTTPDATE:event_time LD DQS LD DQS '
' DQS ' ' LD:session_id EOS"
| fields event_time, client_ip, session_id, content
| sort event_time asc
| filter contains(content, "") //paste here the session_id from query 1.1
```

Step 2.

Query 2.1: collect session_id's of all successful sqli exploited authentications

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-webapp.log")
| parse content, "'[' TIMESTAMP('dd/MMM/yyyy:HH:mm:ss.S'):event_time LD '}
- ' LD:session_id ' '"
| fields event_time, session_id, content
| sort event_time asc
| filter contains(content, "retrieved matching list of size 7") //gets all
user queries returning > 1 rows
```

Query 2.2: get all successful sqli exploited sessions

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-webapp.log")
| parse content, "'[' TIMESTAMP('dd/MMM/yyyy:HH:mm:ss.S'):event_time LD '}
- ' LD:session_id ' '"
| fields event_time, session_id, content
| sort event_time asc
| filter in(session_id, "", ...)    //copy-paste session_id's from
previous query result
```

Query 2.3: check database to evaluate total financial loss

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-sql.log")
| parse content, "TIMESTAMP('yyyy-MM-dd HH:mm:ss.S'):event_time ' 3 '
LD:statement"
| fields event_time, statement
| sort event_time asc
| filter (event_time >= toTimestamp("") and event_time <= toTimestamp(""))
OR
         (event_time >= toTimestamp("") and event_time <= toTimestamp(""))
//paste session begin/end timestamps here
```

Query 2.4: check application logs to find out the beginning of sqli attacks

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-webapp.log")
| parse content, "'[' TIMESTAMP('dd/MMM/yyyy:HH:mm:ss.S'):event_time LD '}
- ' LD:session_id ' ' 'Starting findUsersByUsernameAndPassword of user: '
LD:username EOS"
| fields event_time, session_id, username, content
| filter contains(content, "Starting findUsersByUsernameAndPassword of
```

```
user:")
| filter contains(username, " ")        //sqli often contains multiple
words
      OR contains(username, "--")       //sqli often contains sql-style
comment
      OR contains(username, "'")        //sqli often contains injected
single quote)
| sort event_time asc
```

Query 2.5: find out attacker ip-addresses

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-access.log")
| parse content, "IPADDR:client_ip LD HTTPDATE:event_time LD DQS LD DQS '
' DQS ' ' LD:session_id EOS"
| fields event_time, client_ip, session_id, content
| sort event_time asc
| filter in(session_id, "", ...)    //copy-paste session_id's from query
2.1 result
| summarize count(), by:client_ip
```

Step 3: Assess threat from the attacker

Query 3.1: find out attacker ip-addresses

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-access.log")
| parse content, "IPADDR:client_ip LD HTTPDATE:event_time LD DQS LD DQS '
' DQS ' ' LD:session_id EOS"
| fields event_time, client_ip, session_id, content
| sort event_time asc
| summarize total=count(), failed=countIf(contains(content, "/insecure-
bank/login?authenticationFailure=true")), by:client_ip
| sort failed desc
```

Query 3.2: visualize attacker failed login attempts timing

```
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-access.log")
| parse content, "IPADDR:client_ip LD HTTPDATE:event_time LD DQS LD DQS '
' DQS ' ' LD:session_id EOS"
| fields event_time, client_ip, session_id, content
| filter contains(content,"") //paste here from previous results the ip-
```

```
address deviating the most by failed queries
| summarize failed=countIf(contains(content, "/insecure-bank/login?
authenticationFailure=true")), by:bin(event_time, 1m)
```

Query 3.3 find targeted users

```
//Query 2: extract users with failed logins
fetch logs, from:-90d, samplingRatio:1, scanLimitGBytes:-1
| filter dt.entity.host == "HOST-D866B6DD5365DD5B"
| filter contains(log.source, "insecure-bank-webapp.log")
| filter contains(content, "No users found")
| parse content, "LD 'username: ' LD:username"
| summarize failed=count(), by:username
| sort failed desc
```