Topics in Privacy & Security

Trust-Enhanced Reputation Metrics

Y1481702

March 10, 2018

Contents

i Description

ii Analysis

1

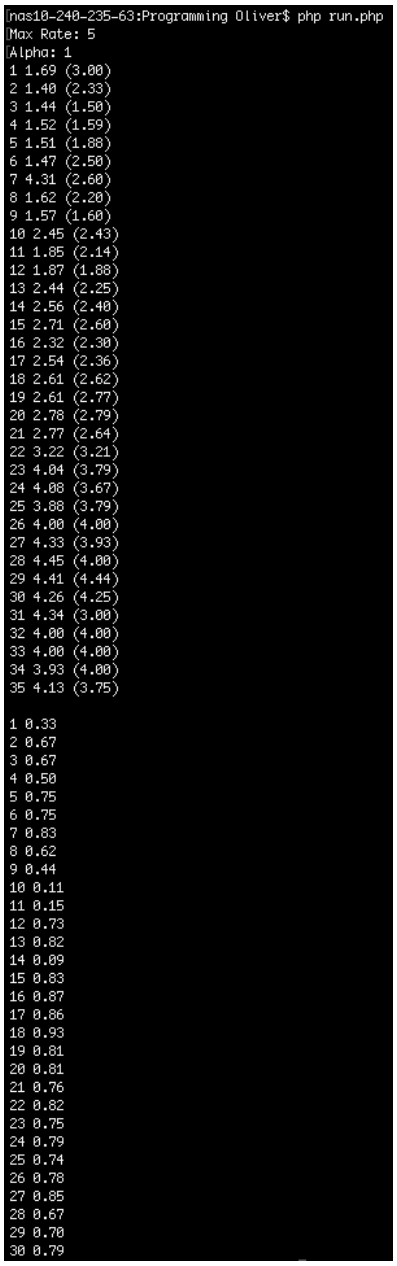
iii Simulated Attacks

Self-promoting attacks of varying sizes have been simulated on product #4, this is shown by Fig. 2a. Slander attacks of varying sizes have been simulated on product #29, this is shown by Fig. 2b.

iv Results

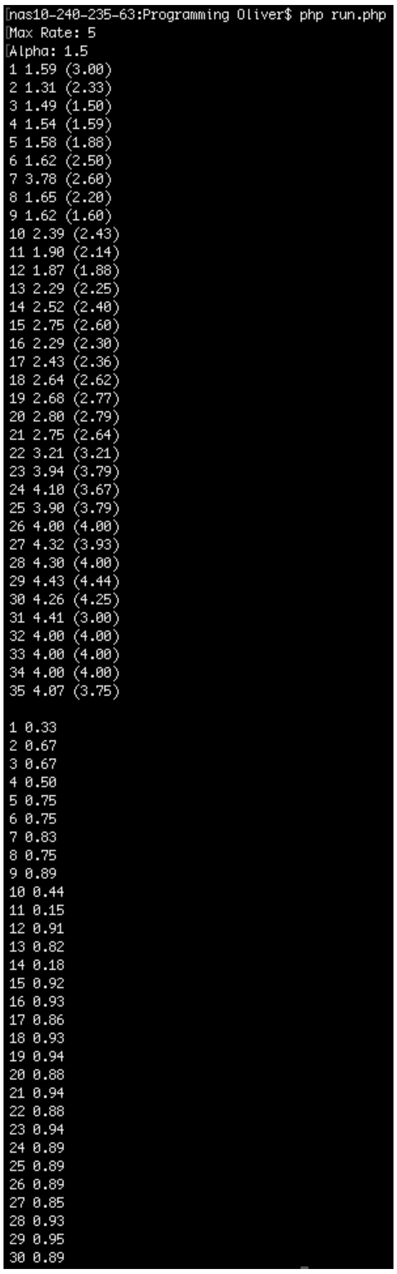
The results show that the system is least susceptible to attack when an α value of 2 is used. Using values of alpha that ignore new reviews may prevent genuine customer reviewers' opinions from being heard.

v Appendix



```
nas10-240-235-63:Programming Oliver$ php run.php
Max Rate: 5
Alpha: 1
1 1.69 (3.00)
2 1.40 (2.33)
3 1.44 (1.50)
4 1.52 (1.59)
5 1.51 (1.88)
6 1.47 (2.50)
7 4.31 (2.60)
8 1.62 (2.20)
9 1.57 (1.60)
10 2.45 (2.43)
11 1.85 (2.14)
12 1.87 (1.88)
13 2.44 (2.25)
14 2.56 (2.40)
15 2.71 (2.60)
16 2.32 (2.30)
17 2.54 (2.36)
18 2.61 (2.62)
19 2.61 (2.77)
20 2.78 (2.79)
21 2.77 (2.64)
22 3.22 (3.21)
23 4.04 (3.79)
24 4.08 (3.67)
25 3.88 (3.79)
26 4.00 (4.00)
27 4.33 (3.93)
28 4.45 (4.00)
29 4.41 (4.44)
30 4.26 (4.25)
31 4.34 (3.00)
32 4.00 (4.00)
33 4.00 (4.00)
34 3.93 (4.00)
35 4.13 (3.75)

1 0.33
2 0.67
3 0.67
4 0.50
5 0.75
6 0.75
7 0.83
8 0.62
9 0.44
10 0.11
11 0.15
12 0.73
13 0.82
14 0.09
15 0.83
16 0.87
17 0.86
18 0.93
19 0.81
20 0.81
21 0.76
22 0.82
23 0.75
24 0.79
25 0.74
26 0.78
27 0.85
28 0.67
29 0.70
30 0.79
```

(a) $\alpha = 1$

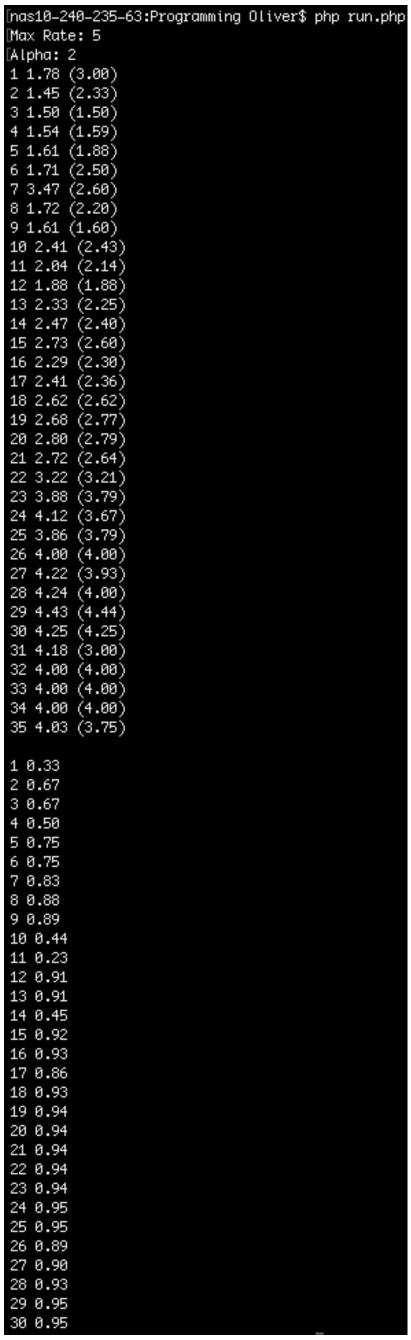


```
nas10-240-235-63:Programming Oliver$ php run.php
Max Rate: 5
Alpha: 1.5
1 1.59 (3.00)
2 1.31 (2.33)
3 1.49 (1.50)
4 1.54 (1.59)
5 1.58 (1.88)
6 1.62 (2.50)
7 3.78 (2.60)
8 1.65 (2.20)
9 1.62 (1.60)
10 2.39 (2.43)
11 1.90 (2.14)
12 1.87 (1.88)
13 2.29 (2.25)
14 2.52 (2.40)
15 2.75 (2.60)
16 2.29 (2.30)
17 2.43 (2.36)
18 2.64 (2.62)
19 2.68 (2.77)
20 2.80 (2.79)
21 2.75 (2.64)
22 3.21 (3.21)
23 3.94 (3.79)
24 4.10 (3.67)
25 3.90 (3.79)
26 4.00 (4.00)
27 4.32 (3.93)
28 4.30 (4.00)
29 4.43 (4.44)
30 4.26 (4.25)
31 4.41 (3.00)
32 4.00 (4.00)
33 4.00 (4.00)
34 4.00 (4.00)
35 4.07 (3.75)

1 0.33
2 0.67
3 0.67
4 0.50
5 0.75
6 0.75
7 0.83
8 0.75
9 0.89
10 0.44
11 0.15
12 0.91
13 0.82
14 0.18
15 0.92
16 0.93
17 0.86
18 0.93
19 0.94
20 0.88
21 0.94
22 0.88
23 0.94
24 0.89
25 0.89
26 0.89
27 0.85
28 0.93
29 0.95
30 0.89
```

(b) $\alpha = 1.5$

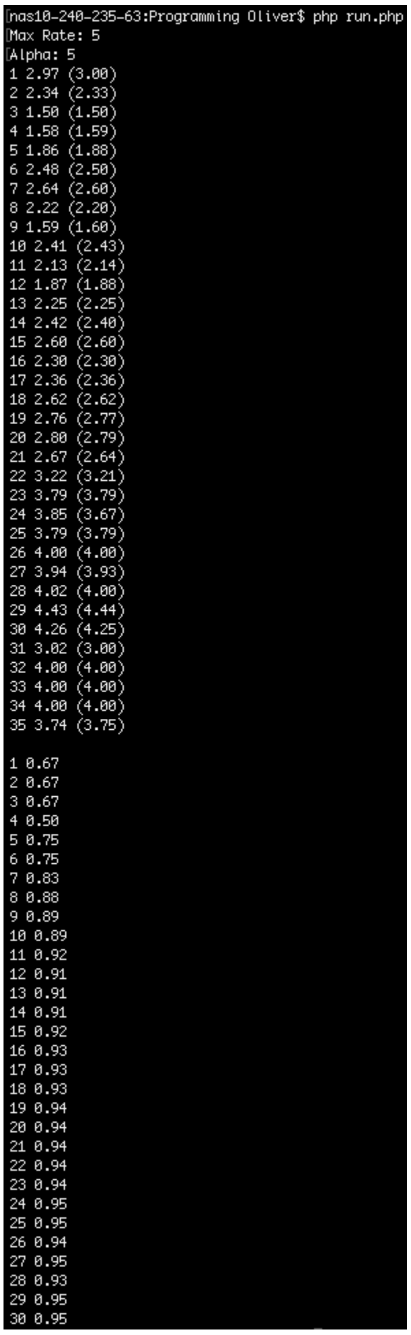
v.1 run.php

```
<?php
require_once("setup.php");
require_once("process.php");
```



```
nas10-240-235-63:Programming Oliver$ php run.php
Max Rate: 5
Alpha: 2
1 1.78 (3.00)
2 1.45 (2.33)
3 1.50 (1.50)
4 1.54 (1.59)
5 1.61 (1.88)
6 1.71 (2.50)
7 3.47 (2.60)
8 1.72 (2.20)
9 1.61 (1.60)
10 2.41 (2.43)
11 2.04 (2.14)
12 1.88 (1.88)
13 2.33 (2.25)
14 2.47 (2.40)
15 2.73 (2.60)
16 2.29 (2.30)
17 2.41 (2.36)
18 2.62 (2.62)
19 2.68 (2.77)
20 2.80 (2.79)
21 2.72 (2.64)
22 3.22 (3.21)
23 3.88 (3.79)
24 4.12 (3.67)
25 3.86 (3.79)
26 4.00 (4.00)
27 4.22 (3.93)
28 4.24 (4.00)
29 4.43 (4.44)
30 4.25 (4.25)
31 4.18 (3.00)
32 4.00 (4.00)
33 4.00 (4.00)
34 4.00 (4.00)
35 4.03 (3.75)

1 0.33
2 0.67
3 0.67
4 0.50
5 0.75
6 0.75
7 0.83
8 0.88
9 0.89
10 0.44
11 0.23
12 0.91
13 0.91
14 0.45
15 0.92
16 0.93
17 0.86
18 0.93
19 0.94
20 0.94
21 0.94
22 0.94
23 0.94
24 0.95
25 0.95
26 0.89
27 0.90
28 0.93
29 0.95
30 0.95
```

(c) $\alpha = 2$



```
nas10-240-235-63:Programming Oliver$ php run.php
Max Rate: 5
Alpha: 5
1 2.97 (3.00)
2 2.34 (2.33)
3 1.50 (1.50)
4 1.58 (1.59)
5 1.86 (1.88)
6 2.48 (2.50)
7 2.64 (2.60)
8 2.22 (2.20)
9 1.59 (1.60)
10 2.41 (2.43)
11 2.13 (2.14)
12 1.87 (1.88)
13 2.25 (2.25)
14 2.42 (2.40)
15 2.60 (2.60)
16 2.30 (2.30)
17 2.36 (2.36)
18 2.62 (2.62)
19 2.76 (2.77)
20 2.80 (2.79)
21 2.67 (2.64)
22 3.22 (3.21)
23 3.79 (3.79)
24 3.85 (3.67)
25 3.79 (3.79)
26 4.00 (4.00)
27 3.94 (3.93)
28 4.02 (4.00)
29 4.43 (4.44)
30 4.26 (4.25)
31 3.02 (3.00)
32 4.00 (4.00)
33 4.00 (4.00)
34 4.00 (4.00)
35 3.74 (3.75)

1 0.67
2 0.67
3 0.67
4 0.50
5 0.75
6 0.75
7 0.83
8 0.88
9 0.89
10 0.89
11 0.92
12 0.91
13 0.91
14 0.91
15 0.92
16 0.93
17 0.93
18 0.93
19 0.94
20 0.94
21 0.94
22 0.94
23 0.94
24 0.95
25 0.95
26 0.94
27 0.95
28 0.93
29 0.95
30 0.95
```

(d) $\alpha = 5$

Figure 1: Tool Output for Varying Alpha

```
while(!feof($myfile)){
    //FOR EACH RATING.. update the database:
    $data = explode("␣", fgets($myfile));
    if(count($data) != 3){break;}
    $customer_id = $data[0];
    $product_id = $data[1];
    $rating = $data[2];

    log_new_rating($db, $data[0], $data[1], $data[2]);
}
fclose($myfile);

if (basename(__FILE__) == basename($_SERVER["SCRIPT_FILENAME"])) {
    //Only run output if file was run DIRECTLY from console,
    //NOT included in another file: i.e. attack.php
    $base_output = "output/Alpha_" . strval(ALPHA) . "_";
    require_once("output.php");
}
```

v.2 attack.php

```
<?php

$attack_type = readline("Attack␣Type␣(slander/promote):␣");
require_once("run.php");

for($j = 0; $j < 5; $j++){
    for($i = 0; $i < 5; $i++){
        //Rating is 0 if slander, MAX_RATE if self-promoting
        $rating = ($attack_type == "slander") ? 0: MAX_RATE;
        //Null customer rating- creates new customer id
        //product id = 29, as stated in question
        log_new_rating($db, null, 4, $rating);
    }

    $base_output = "output/" . ucfirst($attack_type) . "_" .
        strval($attack * ($j + 1)) . "_Alpha_" . strval(ALPHA) . "_";
    require("output.php");
}
```

v.3 setup.php

```
<?php
$filename = readline("Input␣File:␣");
$myfile = fopen($filename, "r");

define("MAX_RATE", intval(readline("Max␣Rate:␣")));
define("ALPHA", floatval(readline("Alpha:␣")));


$db = new mysqli("localhost", "psec", "password");
$db->query("DROP_DATABASE␣psec_assessment;");
$table_setup = "
```

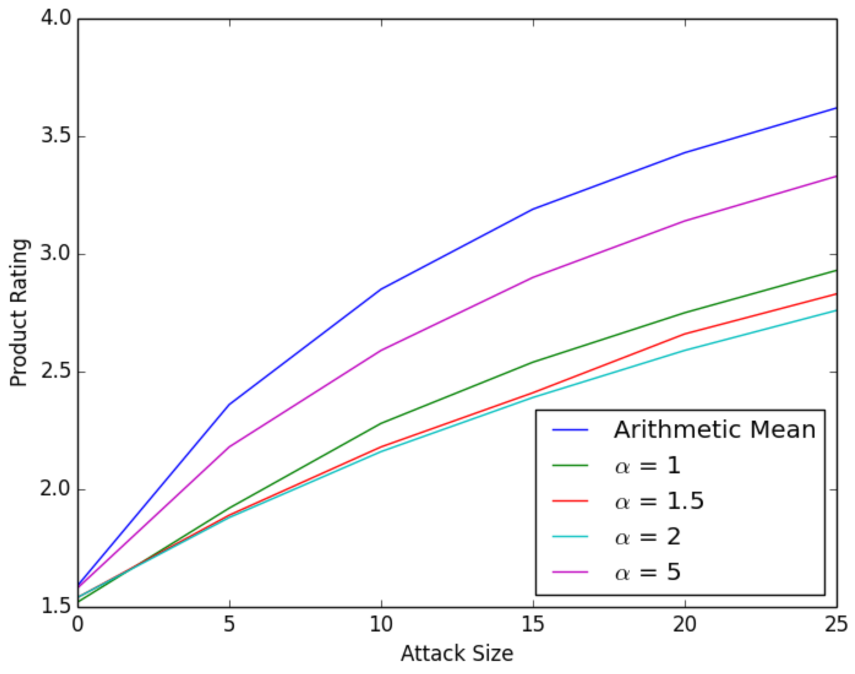
```
   CREATE DATABASE psec_assessment;
   USE psec_assessment;
   CREATE TABLE ratings(
      id INT AUTO_INCREMENT PRIMARY KEY,
      customer_id INT,
      product_id INT,
      rating INT
   );
   CREATE TABLE customers(
      id INT AUTO_INCREMENT PRIMARY KEY,
      trust_level FLOAT
   );
   CREATE TABLE products(
      id INT AUTO_INCREMENT PRIMARY KEY,
      rating FLOAT
   );
";
$db->multi_query($table_setup);
while($db->more_results()){
    $res = $db->next_result();
}
```

v.4 process.php

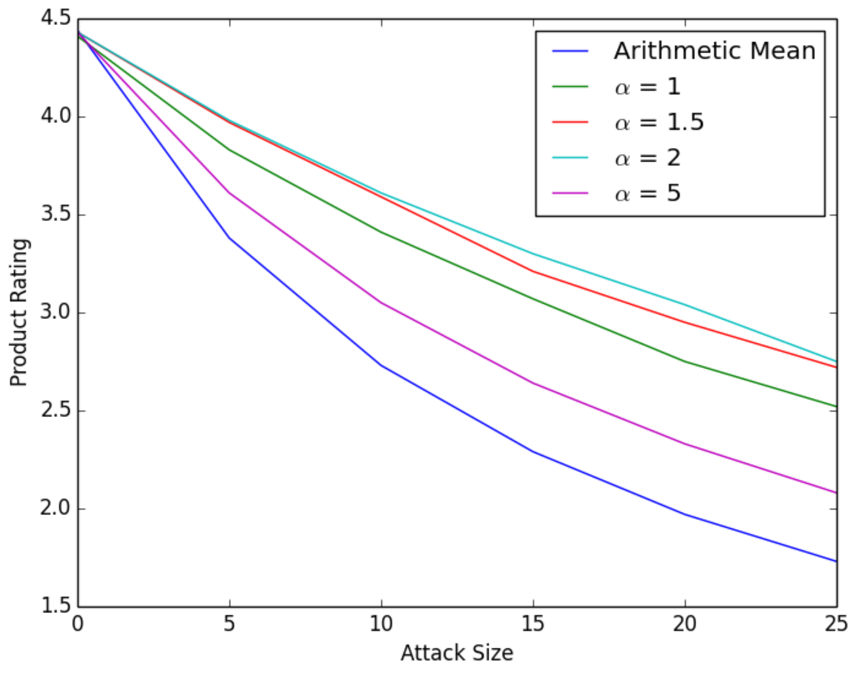
```
<?php
function log_new_rating($db, $customer_id, $product_id, $rating){
    //Check if this is a new user:
    if($customer_id == null){
        //This is a simulated attack:
        //completely new customer ID must be created:
        $trust = trust_index([]);
        $stmt = $db->prepare(
            "INSERT INTO customers (trust_level) VALUES(?);"
        );
        $stmt->bind_param("s", $trust);
        $stmt->execute();

        $customer_id = $db->insert_id;
    }else{
        $stmt = $db->prepare(
            "SELECT COUNT(*) FROM customers where id=?;"
        );
        $stmt->bind_param("s", $customer_id);
        $stmt->execute();
        if($stmt->get_result()->fetch_assoc()["COUNT(*)"] == 0){
            //initialise trust level if new customer: This is 0.5
            if($stmt =
                $db->prepare(
                    "INSERT INTO customers VALUES (?, ?);"
                )){
                $trust = trust_index([]);
                $stmt->bind_param("ss", $customer_id, $trust);
                $stmt->execute();
            }
        }
    }


    //LOG THE NEW RATING:
    $stmt = $db->prepare(
        "INSERT INTO ratings (customer_id, product_id, rating)
        VALUES(?, ?, ?);"
    );
    $stmt->bind_param("sss", $customer_id, $product_id, $rating);
    $stmt->execute();

    //Calculate overall product rating
    $stmt = $db->prepare("SELECT COUNT(*) FROM products where id=?;");
    $stmt->bind_param("s", $product_id);
    $stmt->execute();
    //If NEW product
    if($stmt->get_result()->fetch_assoc()["COUNT(*)"] == 0){
        //initialise rating with the rating of the NEW customer:
        if($stmt = $db->prepare("INSERT INTO products VALUES (?, ?);")){
            $stmt->bind_param("ss", $product_id, floatval($rating));
            $stmt->execute();
        }
        //NEW PRODUCT, nothing left to update?
        return;
    }//IF EXISTING product:

    //Update trust levels of all customers who bought this product
    $stmt = $db->prepare(
        "SELECT customer_id FROM ratings WHERE product_id=?;"
    );
    $stmt->bind_param("s", $product_id);
    $stmt->execute();
    $result = $stmt->get_result();
    while($row = $result->fetch_assoc()){
        update_trust($db, $row["customer_id"]);

    }

    //Recalculate all products other than project j
    //LIMIT THIS TO PRODUCTS THAT HAVE BEEN AFFECTED!:
    $stmt = $db->prepare(
        "SELECT DISTINCT product_id FROM ratings
        WHERE customer_id IN
        (SELECT customer_id FROM ratings WHERE product_id = ?)"
    );
    $stmt->bind_param("s", $product_id);
    $stmt->execute();
    $result = $stmt->get_result();
    while($row = $result->fetch_assoc()){
        update_rating($db, $row["product_id"]);
    }
}

function update_rating($db, $product_id){
    $stmt = $db->prepare(
        "SELECT␣rating,␣trust_level␣FROM␣ratings,␣customers
␣␣␣␣␣␣WHERE␣customer_id␣=␣customers.id␣AND␣product_id␣=␣?;"
    );
    $stmt->bind_param("s", $product_id);
    $stmt->execute();
    $result = $stmt->get_result();

    $numerator = 0;
    $denominator = 0;

    foreach($result as $rating){
        $numerator += $rating["rating"] * $rating["trust_level"];
        $denominator += $rating["trust_level"];
    }

    $stmt = $db->prepare("UPDATE␣products␣SET␣rating␣=␣?␣WHERE␣id␣=␣?;");
    $rating = $numerator / $denominator;
    $stmt->bind_param("ss", $rating, $product_id);
    $stmt->execute();
}

function update_trust($db, $customer_id){
    $fetch_products = "SELECT
␣␣␣␣␣␣ratings.rating␣AS␣customer_rating,
␣␣␣␣␣␣products.rating␣AS␣overall_rating
␣␣␣␣␣␣FROM␣ratings,␣products
␣␣␣␣␣␣WHERE␣product_id=products.id␣AND␣customer_id=?;";

    $stmt = $db->prepare($fetch_products);
    $stmt->bind_param("s", $customer_id);
    $stmt->execute();
    $result = $stmt->get_result();

    $stmt = $db->prepare(
        "UPDATE␣customers␣SET␣trust_level␣=␣?␣WHERE␣id␣=␣?;"
    );
    $tl = trust_index($result);
    $stmt->bind_param("ss", $tl, $customer_id);
    $stmt->execute();
}

function trust_index($products){
    $numerator = 1;
    $denominator = 2;

    foreach($products as $product){
        $numerator += is_trusted(
            $product["overall_rating"],
            $product["customer_rating"]
        );

        $denominator++;
    }

    return $numerator / $denominator;
}

function is_trusted($overall_rating, $customer_rating){
    if(abs($overall_rating - $customer_rating) <= ALPHA){
        return 1;
    }
    return 0;
}
```

v.5 output.php

```
<?php
//Output to file or console:
$customers = fopen($base_output . "Customers.txt", "w");
$products = fopen($base_output . "Products.txt", "w");

$rep_based = $db->query("SELECT * FROM products;");
$average = $db->query(
    "SELECT AVG(rating) rating FROM ratings
    GROUP BY product_id ORDER BY product_id;"
);
foreach($rep_based as $product){
    $avg = $average->fetch_assoc();

    $out_str = sprintf("%u %0.2f (%0.2f)\n",
        $product["id"],
        $product["rating"],
        $avg["rating"]
    );

    echo $out_str;
    fwrite($products, $out_str);
}
echo "\n";
$result = $db->query("SELECT * FROM customers;");
foreach($result as $customer){
    $out_str = sprintf("%u %0.2f\n",
        $customer["id"],
        $customer["trust_level"]
    );

    echo $out_str;
    fwrite($customers, $out_str);
}
```



(a) Effect of Self-Promotion Attacks of Varying Sizes on Product 4



(b) Effect of Slander Attacks of Varying Sizes on Product 29

Figure 2: Attacks on the Online Store Rating System