Infrastructure as a Code and Its Security

TIC 4302 - Information Security Practicum II

Adopted from CS398 - ACC by Prof. Robert J. Brunner, ALPACKED.IO , NEWSTACK.IO , SYNK.IO

Infrastructure as a Code

Problem Statement

- Distributed applications…
    - Are sensitive to how they are configured
- Needs of a database server will be different than an web server
    - Are updated continuously
- New code and patches are deployed daily, if not hourly
    - Will be operated by teams of humans
- Possibility of “operator error”
    - Run on tens/hundreds/thousands of nodes

How do we deploy our Cloud infrastructure?

- Setup everything manually!
    - Does this scale? Clearly no.
- Custom scripts
    - Use your cloud provider’s API to create machines
    - Programmatically SSH into the machine to do tasks
    - Does this scale? Maybe... but why reinvent the wheel?
- Infrastructure as Code
    - Declare your infrastructure setup in a specific format
    - Your IaC framework deploys/updates your cloud infrastructure!
    - Does this scale? Yes!

What is Infrastructure as a Code?

- **Kief Morris** in his book "**Infrastructure as Code, 2d edition**" defines IaC as follows:

  "an approach to infrastructure automation based on practices from software development. It emphasizes consistent, repeatable routines for provisioning and changing systems and their configuration".

- Infrastructure as a Code becomes a key enabler of **DevOps culture** establishment, since it helps **automate the routine tasks and infrastructure management**, allowing teams focus on frequent updates and features release.

Infrastructure as a Code Benefits

- **Faster modernization and ability** to build reliable, safe and low cost systems.
- **Measurable metrics** to make the "figures"-driven decisions.
- **One-size-fits-all** solution that can be applied to any system (cloud, VMs, servers).
- **Decreased time on infrastructure maintenance**, so have more time on updates and features
- **Reduced risks and downtimes** because the manual changes are eliminated.
- Users can **get the needed resources** they need, when they need.
- **Full control** over security, governance, user roles.
- **Faster troubleshooting** because many bottlenecks are eliminated on the step of implementation and testing stage will uncover where the bug popped up.
- **Documented infrastructure** description which is described as desired state of the infrastructure so any user can read the JSON / YAML file and easily understand how the system must work.
- **Version control** means that any change is recorded, know when to rollback if the action was undesired, and team members can make an audit and suggest improvements.
- **System consistency** because the code built the same way all the time so it can predict on how the system will behave and make testing even more efficient.

Infrastructure as a Code Disadvantages

- **Challenging implementation**
  No matter if you are an experienced player or a startup, infrastructure as code implementation will be quite painful staff for both at the beginning.
- **Resistance from the side of other teams**
  Yes, you have to prepare to objections like "We don't make changes so often, so it would be better and cheaper to go to AWS console and click a few buttons than to write the templates or code".
- **Additional tooling**
  You will need to implement configuration management systems like Ansible, Puppet, CHEF, Salt (however, Ansible is used in 90% cases).

Infrastructure as Code Ideas

Approaches to “writing down” cloud configuration:

- **Declarative:** Define the target state of your cloud. What should the eventual cloud deployment look like?
- **Imperative:** Define how the configuration system should setup the cloud. How should the system deploy your application?
- **Intelligent:** Define relationships and constraints between services, and the system will figure out how and what to update.

Infrastructure as Code Ideas

Approaches to updating cloud configuration:

- **Push:** A central server tells child servers their configuration
- **Pull:** Child servers request configuration from a central server

Tool	Tool type	Infrastructure	Architecture	Approach	Language
CHEF	Config management	Mutable	Pull	Declarative & Imperative	Ruby
puppet	Config management	Mutable	Pull	Declarative	DSL & ERB
SALTSTACK	Config management	Mutable	Push & pull	Declarative & Imperative	YAML
AWS CloudFormation	Provisioning	Immutable	Push	Declarative	JSON & YAML
aws Cloud Development Kit	Provisioning	Immutable	Push	Declarative	TS, JS, Python, Java, C#/.Net
ANSIBLE	Config management	Mutable	Push	Declarative & Imperative	YAML
Terraform	Provisioning	Immutable	Push	Declarative	HashiCorp Configuration Language
pulumi Cloud Native Infrastructure as Code	Provisioning	Immutable	Push	Declarative	JS, TS, Python, Go, NET language, incl C#, F#, and VB

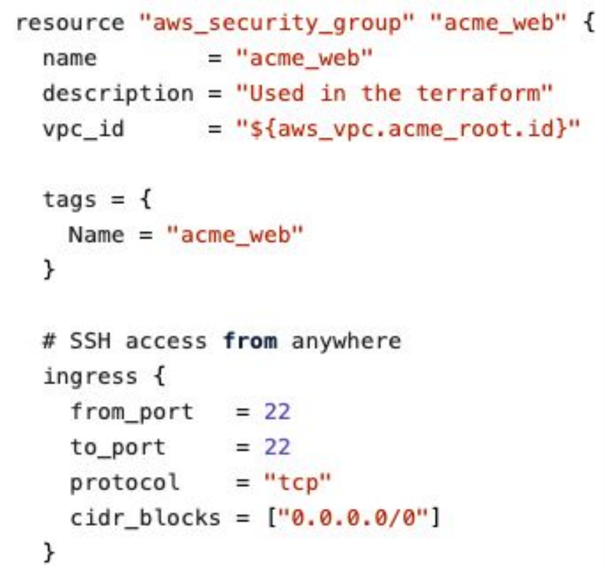
Infrastructure as a Code Demo

Infrastructure as a Code Security

What is Infrastructure as a Code Security?

Infrastructure as Code security is the discipline of ensuring that **security best practices are built into the IaC declarative scripts**. These best practices include the principle of **least privilege, network segmentation** so that the resources and their related dependencies are all secured within **a private subnet, and the encryption** of data in-transit and at-rest. The consequences of IaC scripts that **do not uphold security principles** can be severe, including **unsecured storage buckets** that expose sensitive data or an **instance that is inadvertently publicly accessible** and acts as an attack vector for hackers.

IaC Security Risk - Network Exposures



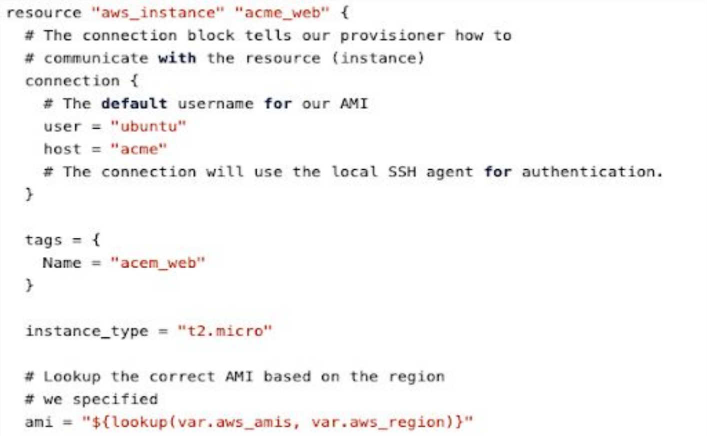
```
resource "aws_security_group" "acme_web" {
  name        = "acme_web"
  description = "Used in the terraform"
  vpc_id      = "${aws_vpc.acme_root.id}"

  tags = {
    Name = "acme_web"
  }

  # SSH access from anywhere
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

Insecure IaC configurations can expand the attack surface that **enables reconnaissance, enumeration,** or even the **deliver cyberattacks**

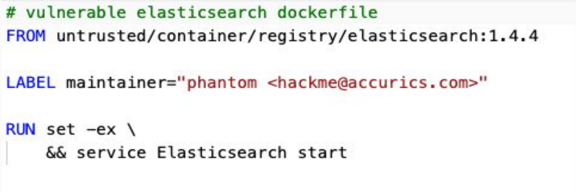
Configuring open Security Groups for **public access for cloud storage, ssh access, and databases** are examples of common IaC misconfigurations

Tip: Perform static security analysis of IaC and eliminate risks early is highly cost-effective and reduces your residual risks.

IaC Security Risk - Vulnerabilities



```
resource "aws_instance" "acme_web" {
  # The connection block tells our provisioner how to
  # communicate with the resource (instance)
  connection {
    # The default username for our AMI
    user = "ubuntu"
    host = "acme"
    # The connection will use the local SSH agent for authentication.
  }

  tags = {
    Name = "acem_web"
  }

  instance_type = "t2.micro"

  # Lookup the correct AMI based on the region
  # we specified
  ami = "${lookup(var.aws_amis, var.aws_region)}"
```



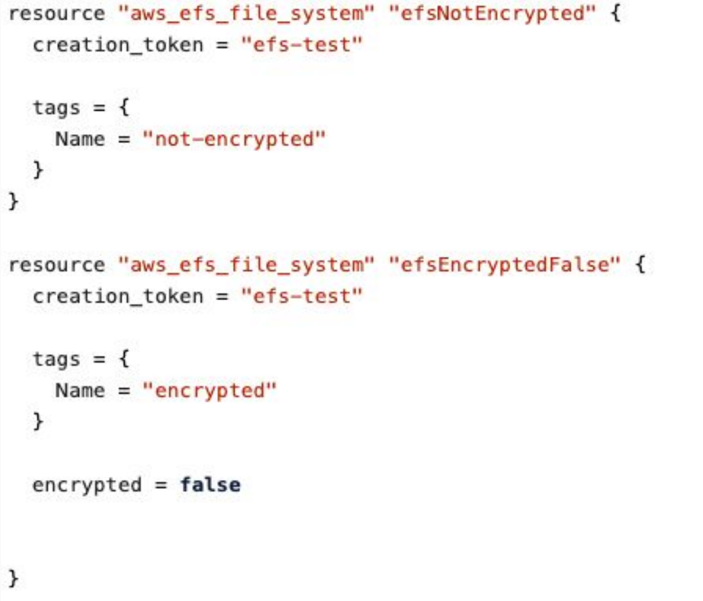
```
# vulnerable elasticsearch dockerfile
FROM untrusted/container/registry/elasticsearch:1.4.4

LABEL maintainer="phantom <hackme@accurics.com>"

RUN set -ex \
    && service Elasticsearch start
```

IaC templates are used to provision compute and containerized instances by including **base images stored in trusted registries**.

Detect vulnerabilities in such base images early and dramatically reduce the cost of remediation.

Tip: Perform vulnerability assessment of images referred to within IaC files and detect vulnerabilities early in the development lifecycle.

IaC Security Risk - Data Exposures



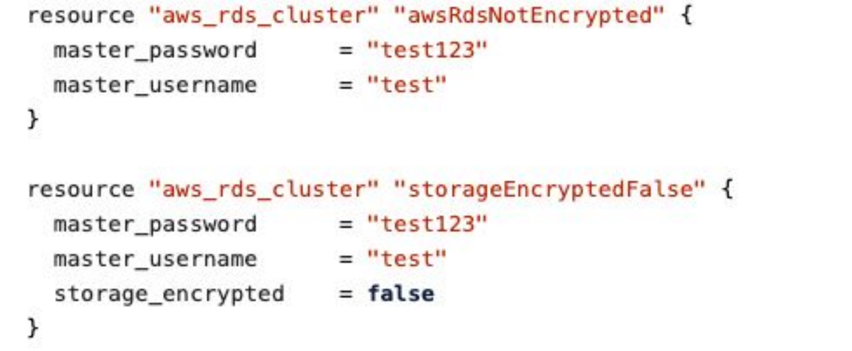
```
resource "aws_efs_file_system" "efsNotEncrypted" {
  creation_token = "efs-test"

  tags = {
    Name = "not-encrypted"
  }
}

resource "aws_efs_file_system" "efsEncryptedFalse" {
  creation_token = "efs-test"

  tags = {
    Name = "encrypted"
  }

  encrypted = false


}
```

Databases or cloud storage services that are created **without enabling encryption can pose risks**.

Encryption is only just one aspect of data security, there are a number of other misconfigurations that can create data exposures in the cloud.

Tip: Assess data security-related configurations in infrastructure as code and remediate them early in the development cycle.

IaC Security Risk - Hardcoded Secret



```
resource "aws_rds_cluster" "awsRdsNotEncrypted" {
  master_password       = "test123"
  master_username       = "test"
}

resource "aws_rds_cluster" "storageEncryptedFalse" {
  master_password       = "test123"
  master_username       = "test"
  storage_encrypted     = false
}
```

Hardcoded secrets or credentials is a common malpractice that involves storing plain text credentials within source code.

This enable **unauthorized privilege escalation and lateral movement and it is very difficult to trace** and contextualize hardcoded secrets in runtime environments.

Provisioning and managing infrastructure **through code makes it easier to hardcode secrets** within it.

Tip: Scan infrastructure as code for hard coded secrets and remediate issues before cloud infrastructure is provisioned.

Thank You!