SETH VARGO
@sethvargo
SECRET MANAGEMENT
WHAT IS "SECRET"?
SECRET VS. SENSITIVE
SECRET

DB CREDENTIALS
SSL CA/CERTIFICATES
CLOUD ACCESS KEYS
ENCRYPTION KEYS
WIFI PASSWORDS
SOURCE CODE

SENSITIVE
SECRET

DB CREDENTIALS
SSL CA/CERTIFICATES
CLOUD ACCESS KEYS
ENCRYPTION KEYS
WIFI PASSWORDS
SOURCE CODE

SENSITIVE

PHONE NUMBERS
MOTHER'S MAIDEN NAME
EMAIL ADDRESSES
DATACENTER LOCATIONS
CUSTOMER PII
EMAIL/CHAT
SECRET

DB CREDENTIALS
SSL CA/CERTIFICATES
CLOUD ACCESS KEYS
ENCRYPTION KEYS
WIFI PASSWORDS
SOURCE CODE

SENSITIVE

PHONE NUMBERS
MOTHER'S MAIDEN NAME
EMAIL ADDRESSES
DATACENTER LOCATIONS
CUSTOMER PII
EMAIL/CHAT
ANYTHING THAT MAKES THE NEWS
ASHLEY MADISON®
Life is short. Have an affair.®

Get started by telling us your relationship status:

Please Select

See Your Matches »

Over 32,875,000 anonymous members!
HOW DO I DISTRIBUTE SECRETS?

▼ How do applications get secrets?

▼ How do humans acquire secrets?

▼ How are secrets updated?

▼ How is a secret revoked?
secure master cat config.son

{
    "mysql_user": "root",
    "mysql_pass": "s3(Ret"
}

WHY NOT CONFIG MANAGEMENT?

- Centrally stored
- Eventually consistent
- No access control
- No auditing
- No revocation
WHY NOT (ONLINE) DATABASES?

- RDBMS, Consul, ZooKeeper, etc
- Not designed for secrets
- Limited access controls
- Typically plaintext storage
- No auditing or revocation abilities
HOW TO HANDLE SECRET SPRAWL?

- Secret material is distributed
- Who has access?
- When were secrets used?
- What is the attack surface?
- What do we do in the event of a compromise?
STATE OF THE WORLD 1.0

- Secret sprawl
- Decentralized keys
- Limited visibility
- Poorly defined “break glass” procedures
SECRET MANAGEMENT 2.0
VAULT
MODERN SECRETS MANAGEMENT
VAULT GOALS

- Single source for secrets
- Programmatic application access (Automated)
- Operator access (Manual)
- Practical security
- Modern data center friendly
VAULT FEATURES

- Secure secret storage (in-memory, Consul, file, postgres, and more)
- Dynamic secrets
- Leasing, renewal, and revocation
- Auditing
- Rich ACLs
- Multiple client authentication methods
SECURE SECRET STORAGE

- Data is encrypted in transit and at rest
- 256bit AES in GCM mode
- TLS 1.2 for clients
- No HSM required
Success! Data written to: secret/foo
Key            Value
lease_id       secret/foo/2a798f6f-00da-8d48-659a-ef1c969f23ed
lease_duration 2592000
lease_renewable false
bar            bacon
DYNAMIC SECRETS

- Never provide “root” credentials to clients
- Provide limited access credentials based on role
- Generated on demand when requested
- Leases are enforceable via revocation
- Audit trail can identify point of compromise
Successfully mounted 'postgresql' at 'postgresql'!
## DESCRIPTION

The PostgreSQL backend dynamically generates database users.

After mounting this backend, configure it using the endpoints within the "config" path.

## PATHS

The following paths are supported by this backend. To view help for any of the paths below, use the help command with any route matching the path pattern. Note that depending on the policy of your auth token, you may or may not be able to access certain paths.

^config/connection$
vault write postgresql/config/connection
value="user=hashicorp password=hashicorp database=hashicorp"

Success! Data written to: postgresql/config/connection
vault write postgresql/roles/production name=production

Success! Data written to: postgresql/roles/production
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lease_id</td>
<td>postgresql/creds/production/2d483e34-2d82-476...</td>
</tr>
<tr>
<td>lease_duration</td>
<td>3600</td>
</tr>
<tr>
<td>lease_renewable</td>
<td>true</td>
</tr>
<tr>
<td>password</td>
<td>80e6ffa5-d6e9-beb1-e630-9af0c41299bb</td>
</tr>
<tr>
<td>username</td>
<td>vault-root-1432058168-8081</td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lease_id</td>
<td>postgresql/creds/production/a99b952e-222c-6eb...</td>
</tr>
<tr>
<td>lease_duration</td>
<td>3600</td>
</tr>
<tr>
<td>lease_renewable</td>
<td>true</td>
</tr>
<tr>
<td>username</td>
<td>vault-root-1432058254-7887</td>
</tr>
<tr>
<td>password</td>
<td>17a21ba7-8726-97e4-2088-80b7a756702b</td>
</tr>
</tbody>
</table>
DYNAMIC SECRETS

▼ Pluggable Backends

▼ AWS, Consul, PostgreSQL, MySQL, Transit, Generic

▼ Grow support over time
LEASING, RENEWAL, AND REVOCATION

- Every Secret has a Lease*
- Secrets are revoked at the end of the lease unless renewed
- Secrets may be revoked early by operators
  - “Break Glass” procedure
- Dynamic Secrets make leases enforceable
  - Not possible for arbitrary secrets
  - Not possible for transit backend
AUDITING

▼ Pluggable Audit Backends

▼ Request and Response Logging

▼ Prioritizes Safety over Availability

▼ Secrets Hashed in Audits

▼ Searchable, but not reversible
RICH ACLS

- Role Based Policies
- Restrict access to “need to know”
- Default Deny, must be explicitly allowed
FLEXIBLE AUTH

- Pluggable Backends
- Tokens, GitHub, AppID, User/Pass, TLS Certs
- Machine-Oriented vs Operator-Oriented
HIGH AVAILABILITY

- Consul used for leader election
- Active/Standby
- Automatic failover
UNSEALING THE VAULT

- Data in Vault encrypted
- Vault requires encryption key
- Must be provided *online*
Sealed: true
Key Shares: 10
Key Threshold: 7
Unseal Progress: 6

High-Availability Enabled: false
secure master vault unseal

Key (will be hidden):
Key (will be hidden):

Sealed: false
Key Shares: 10
Key Threshold: 7
Unseal Progress: 0
WATCHING THE WATCHMEN

- Master Key is the “key to the kingdom”
- All data could be decrypted
- Protect against insider attack
- Two-Man Rule
SHAMIR SECRET SHARING

- Protect Encrypt Key with Master Key
- Split Master Key into \( N \) shares
- \( T \) shares to recompute Master
- Quorum of key holders required to unseal
- Default \( N:5, T:3 \)
SUMMARY

▼ Solves the “Secret Sprawl Problem”

▼ Protects against external threats (Cryptosystem)

▼ Protects against internal threads (ACLs and Secret Sharing)
BUILDING ON VAULT
SECURITY FOUNDATION

- Base of Trust
- Core Infrastructure
- Flexible Architecture
- Foundation for Security Infrastructure
PERSONALLY IDENTIFIABLE INFORMATION

- PII information is everywhere
  - SSN, CC#, OAuth Tokens, etc.
  - Email? Physical address?
- Security of storage?
- Scalability of storage?
- Audibility of access?
PII WITH VAULT

- “transit” backend in Vault
- Encrypt/Decrypt data in transit
- Avoid secret management in client applications
- Builds on Vault foundation
TRANSIT BACKEND

- Web server has no encryption keys
- Requires two-factor compromise (Vault + Datastore)
- Decouples storage from encryption and access control
CERTIFICATE AUTHORITY

- Vault acts as Internal CA
- Vault stores root CA keys
- Dynamic secrets - generates signed TLS keys
- No more tears
MUTUAL TLS FOR SERVICES

- Dynamic CA allows all services to generate keys
- All internal service communication can use mutual TLS
- End-to-End encryption inside the datacenter
VAULT IN PRACTICE
USING VAULT

- API Driven
- JSON/HTTPS
- Rich CLI for humans and scripts
- Rich client libraries
APPLICATION INTEGRATION

▼ Vault-aware

▼ Native client libraries (go, ruby, rails, python, node, and more)

▼ Secrets only in-memory

▼ Safest but high-touch
CONSUL TEMPLATE INTEGRATION

- Secrets templatized into application configuration
- Vault is transparent
- Lease management is automatic
- Non-secret configuration still via Consul
{{ with $secret := vault "postgresql/creds/production" }}

---

production:
  adapter: postgresql
  database: postgres.service.consul
  username: {{$.secret.Data.username}}
  password: {{$.secret.Data.password}}
  pool: {{key "production/postgres/pool"}}
{{ end }}
REMEMBER TO RATE THIS SESSION
CREET @SETHVARGO FOR QUESTIONS TO
THANK YOU!

QUESTIONS?