The Problem with Passwords -- Humans

HashiCorp Vault - Dynamic Passwords

John Dohoney, Jr.
September 7, 2018

Agenda

- ❏ Who am I
- ❏ Overview -- Passwords as a Security Problem
- ❏ Solution - Secrets Management and HashiCorp Vault
- ❏ How we stack up to alternatives
- ❏ Questions

Who am I

Professionally

Turbo C++ IDE

≡ File Edit Search Run Compile Debug Project Options Window Help

About

Turbo C++

Version 3.0

Copyright (c) 1990, 1992 by

Borland International, Inc.

OK

F1 Help | Accept the settings in this dialo

SMALLTALK-80

THE LANGUAGE

API

DC/OS

docker SWARM

kubernetes

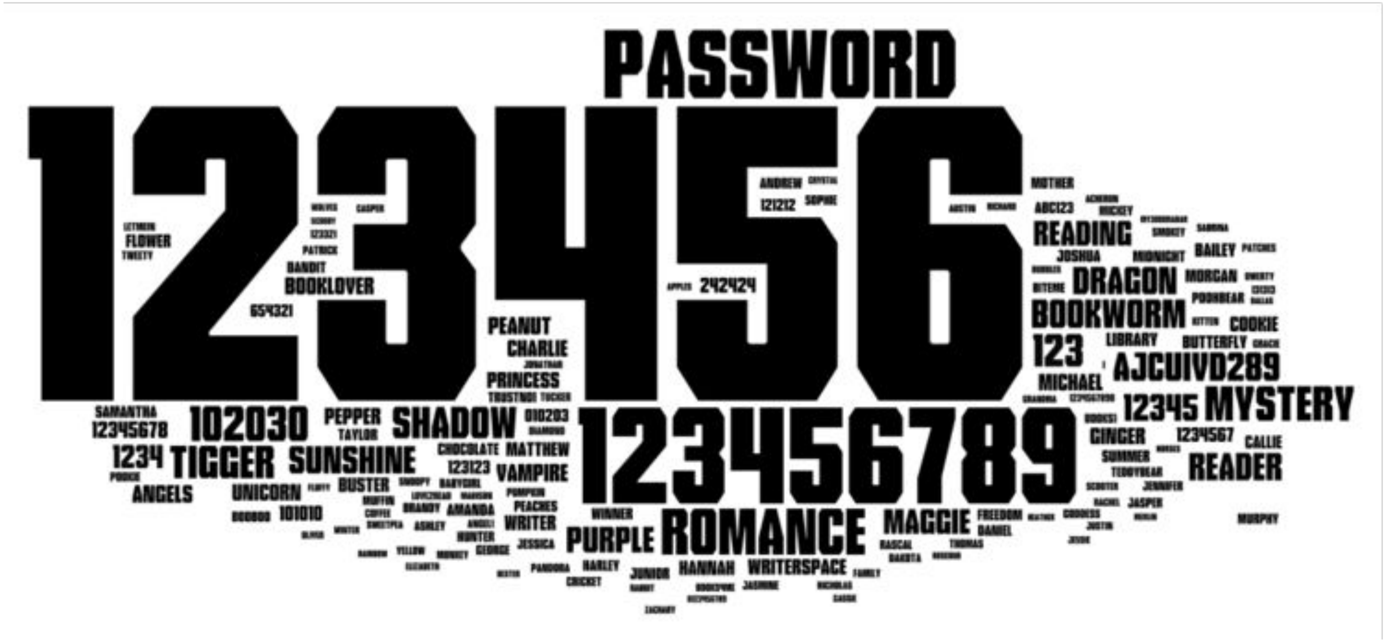
TechGeek

Home

THE HOME DEPOT

adafruit

How original



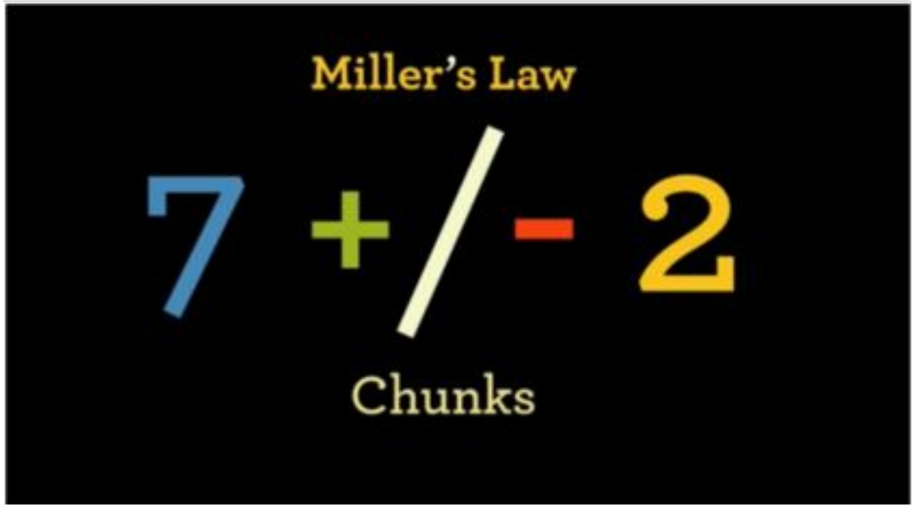
Add Password Policies = Post-It Notes

Bad Stuff Starts to happen...

Social Engineering
by Nagasahas
Damn! His password is encrypted with 256 bit algorithm I guess it is too difficult to crack
Buy him a beer dude, he will reveal the password and other information too :P
Boy! This was soo much more easier than what I had thought. Social Engineering really works!
IMAGES © 2014 PIXTON.COM

"DUMPSTER DIVING"
PERSONAL INFORMATION IS DISCARDED AND THEIFS REMOVE IT FROM THE TRASH.

Cognitive Science has an explanation



Our brain is a machine that recognize patterns, chunk things together & then process the chunks to gain understanding. According to Miller, our minds can easily process somewhere between 5-9 chunks of information at any given time. Any more than that and our brains are overloaded. Think about your passwords...

For evil to prevail, good men need do nothing

Gentoo Hack (June 2018) - https://www.networkworld.com/article/3287973/linux/the-aftermath-of-the-gentoo-github-hack.html

“The most obvious take-home was that the admin's password was guessed because it too closely related to one that had been captured on another system. This might be like your using "Spring2018" on one system and "Summer2018" on another.”

Tweeter Hack (May 2018) - https://blog.twitter.com/official/en_us/topics/company/2018/keeping-your-account-secure.html

“Due to a bug, passwords were written to an internal log before completing the hashing process. We found this error ourselves, removed the passwords, and are implementing plans to prevent this bug from happening again.”

Github (May 2018) -
https://www.bleepingcomputer.com/news/security/github-accidentally-recorded-some-plaintext-passwords-in-its-internal-logs/

GitHub says that normally, passwords are secure, as they are hashed with the bcrypt algorithm. The company blamed a bug for plaintext passwords ending up in its internal logs. Only users who've recently reset passwords were affected.

Solution: Secrets Management Approach

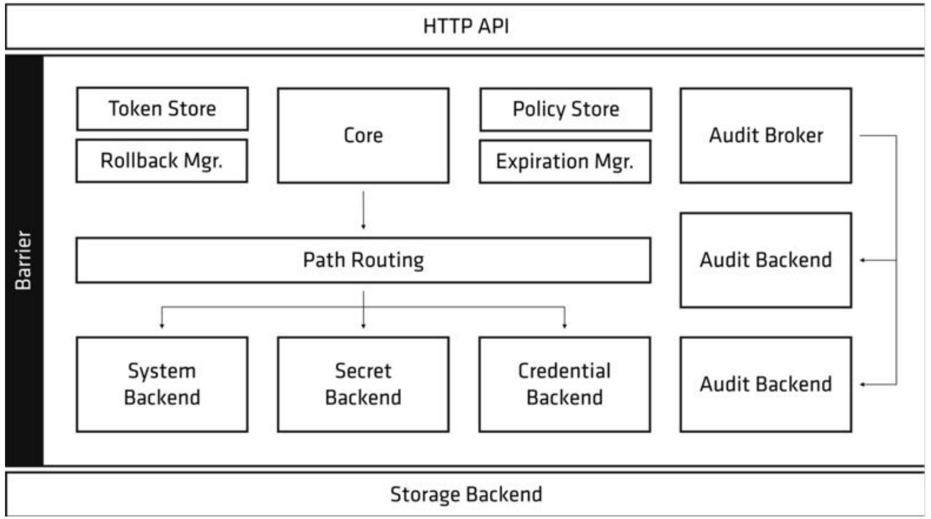
Managing a set of different credentials, both Authorization (AuthZ) and Authentication (AUTHN) in nature that includes:

- DB Credentials
- API Token
- TLS Certificates
- Username and Password

The result of not managing these are what is called “Secret Prawl” - the AUTHN and AUTHZ data is everywhere in the enterprise. Compounding, this are the questions of who is authorized to see this information, and if they did, how could we audit this access? Furthermore, most secret management approaches are usually static, not rotated, and not given an expiration on usage.

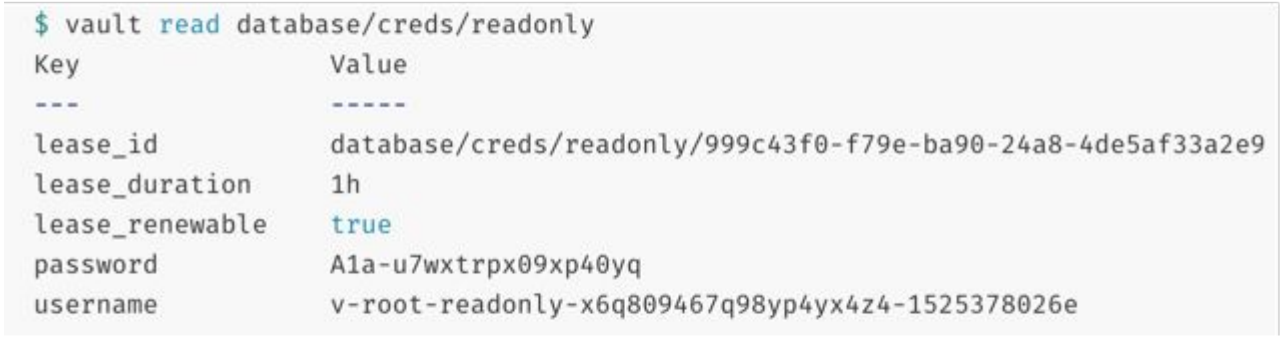
Design Pattern for Secret Sprawl Remediation

- Centralizes access
- Heavy Use of Encryption
  - Encrypted at rest
  - Encrypted in transit
  - Inside the system.
- Specific Access
  - fine grained role based access
- Auditable
  - Not only who, but what credential and the outcome of access request



The need for Ephemeral Secrets

- Why?
  - Traditional Hard coding of config files with passwords
  - Today, Kubernetes and Marathon -- DC/OS only uses Static secrets; K8 has a Vault Operator by CoreOS
  - Logs out to STDOUT -- See Slide 8 - **Tweeter Hack (May 2018)**
  - Exceptions spit out "secret" information to monitoring system
- Bounded credentials, reduces Attack Surface area, even if they do leak, usage is limited - "Moving Target"
- Credentials are unique to a client, and easier to isolate due to unique credentials
- Easier Revocation and isolates a leak



```
$ vault read database/creds/readonly
Key                Value
---                -----
lease_id           database/creds/readonly/999c43f0-f79e-ba90-24a8-4de5af33a2e9
lease_duration     1h
lease_renewable    true
password           A1a-u7wxtrpx09xp40yq
username           v-root-readonly-x6q809467q98yp4yx4z4-1525378026e
```

Application Storage

Cryptography is hard to implement properly - if you need it, it will also get audited

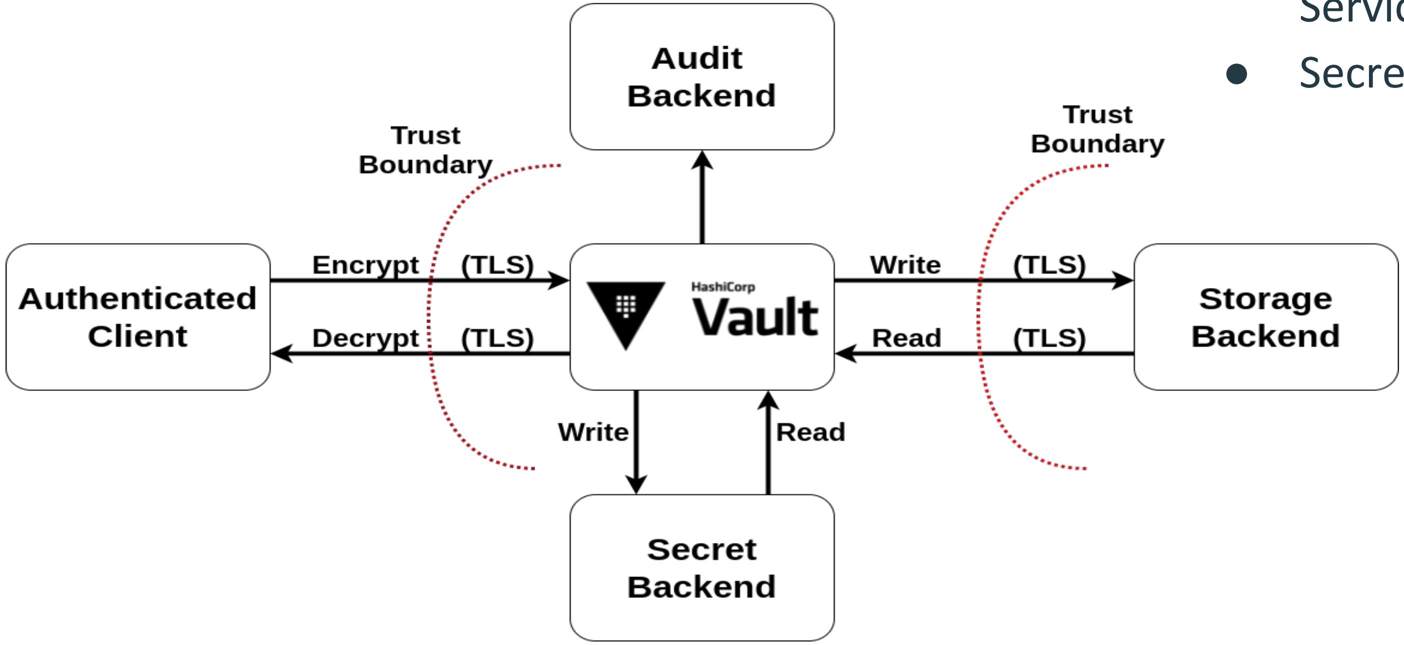
Creates Named Keys -- examples: PII, SSN, Credit Cards

HIgh Level API's -- Encrypt, Decrypt, Sign, Verify as example

Hides actual implementation (Information Hiding) example: HMAC256, SHA512

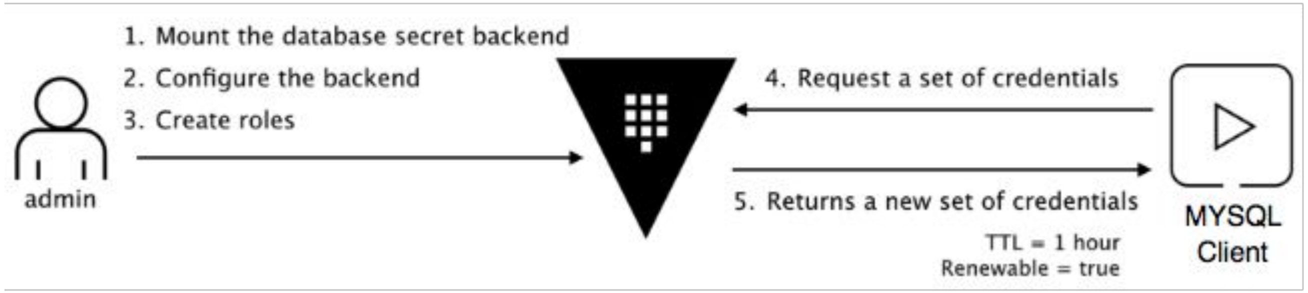
Offloads Key Management -- Key Creation, Key Versioning, Key Rotation, and Key Decommissioning - jast hard to do properly

Solution: Hashicorp Vault



- Secret Store
- Dynamic Secrets
- Encryption as a Service
- Secrets Audit Trail

Dynamic Secrets Demo


1. Mount the database secret backend
2. Configure the backend
3. Create roles
admin
4. Request a set of credentials
5. Returns a new set of credentials
TTL = 1 hour
Renewable = true
MYSQL
Client

Are there Choices?

HSM - hardware Security Module -- wins the war in Complexity and Cost, seems to be advantageous when there is a FIPS compliance requirement. Vault does complement an HSM, for example, adds Dynamic Secrets capability.

Chef and Puppet -- Weak, one password away from being decrypted. TIed to the respective system, Vault has no vendor lock-in. CHef's data bag and Puppets" Heira area feel good product, no dynamic secrets, No secrets as a service, and no RBAC

AWS KMS -- The AWS KMS is back by HSM. AWS KMS is solid for encryption key management as well as supporting Crypto-Ops and is FIPS 140-2 compliant. Vault is superior in that it handles any type of secret data, including database credentials, API keys, PKI keys, and encryption keys. Vault also supports dynamic secrets, generating credentials on-demand for fine-grained security controls, auditing, and non-repudiation. Vault is your best option for Multi-Cloud and Hybrid Cloud Ops.

Are there Choices continued?

Azure and GCE -- Upgrade to Enterprise Pro, both are supported. Enterprise Pro is your best dollar, for multi-cloud and avoiding platform lock-in. Both have a KMS, Azure Key Vault went head to head with Amazon is FIPS 140-2 level 2 compliant backed by HSMs. Google CLoud KMS is similar to both AWS and Azure, they up the ante on with FIPS 140-2 level 3 device compliance. As with AWS, Vault is superior in that it handles any type of secret data, including database credentials, API keys, PKI keys, and encryption keys. Vault also supports dynamic secrets, generating credentials on-demand for fine-grained security controls, auditing, and non-repudiation. Vault is your best option for Multi-Cloud and Hybrid Cloud Ops.

Keywhiz is a secret management solution built by Square. Vault and Keywhiz are on parity in many ways. Where vault is differentiated is how it forces a mandatory lease contract with clients. All secrets read from Vault have an associated lease which enables operators to audit key usage, perform key rolling, and ensure automatic revocation. Vault provides multiple revocation mechanisms to give operators a clear "break glass" procedure after a potential compromise.

Questions