Managing server secrets at scale with a vaultless password manager

Ignat Korchagin
$ whoami

● Platform engineer at Cloudflare

● Passionate about security and crypto

● Enjoy low level programming
Disclaimer

There is no cloud
it's just someone else's computer
So you have a server
So you have a server

You need:

- server SSH key
So you have a server

You need:

● server SSH key
● configuration management key
So you have a server

You need:

- server SSH key
- configuration management key
- disk encryption key
So you have a server

You need:

● server SSH key
● configuration management key
● disk encryption key
● some server credentials
So you have a server

You need:

● server SSH key
● configuration management key
● disk encryption key
● some server credentials
● probably more...
So you have a server

You need:

- server SSH key
- configuration management key
- disk encryption key
- some server credentials
- probably more...

... so at least 5 keys... and that’s per server
So you have a datacentre
So you have a datacentre(s)
So you have a datacentre(s)
Cloudflare network today
So you have a datacentre(s)
Where do keys live?
Where do keys live?
Where do keys live?
Keys in configuration management

```
#!/yaml|gpg

root-password: |
    -----BEGIN PGP MESSAGE-----
    Version: GnuPG v1

    hQIMA221up1ZYLmdARAAB583Z4oZwWzK8yUYJKBEmMQD/i+RRn7A0+h8SEmosovq
    QkUxgeaCWfI5pRpCpVOKISWZGi0dzWkWe1DeNisawv5X/VUG3d5ej1xtAD4kB
    TyaAzcNf7qIsV8c+jguHYGITU++pFVAgEdGrb09mf6SEDaAGJhOq01BmHccw0Pat
    rBH/+gvD155F7sxM/BBQwL25ZjtC+8jUsplbUcTQVofsy6kTVRNSS4h04UNtMuMQ
    hYf6UApaJv3PhFXYYyu0tEp2THZVT1UtTjyKAZrNiKyRkC/0exbJjJMqkymmUG9r
    yPlCvubJnmHda2u42981dK3pz5T1leo4MrBry6yvnN0TJfXwn1nt7YMVatiiViQb9
    UK5NDbjVKBBE6Kkn28ksJtsTkCOM7+RztjLdf+7ZWzwxFV5EkM+2SLPiHqFCMjRG
    ...
```
Keys in configuration management
Keys in configuration management

- Bootstrap configuration management
Keys in configuration management

- Bootstrap configuration management
- Does not scale for unique keys
Keys in configuration management

- Bootstrap configuration management
- Does not scale for unique keys

```yaml
root-password: |
{% if grains['hostname'] == 'baredog' %}
    -----BEGIN PGP MESSAGE-----
    ...
{% elif grains['hostname'] == 'cheesyonion' %}
    -----BEGIN PGP MESSAGE-----
    ...
{% elif ... %}
```
Keys on local disk
# super nitty startup script, fully automated !!!
if [ ! -f /etc/server_key ]; then
    dd if=/dev/urandom of=/etc/server_key bs=1 count=32
fi

# don’t forget SSH
if [ ! -f /etc/ssh/id_rsa ]; then
    ssh-keygen -f /etc/ssh/id_rsa
fi
WHAT IF I TOLD YOU

ENTROPY IS LOW RIGHT
AFTER SYSTEM BOOT
Keys on local disk
Keys on local disk

- Not suitable for some key types
Keys on local disk

- Not suitable for some key types
  - root passwords
Keys on local disk

- Not suitable for some key types
  - root passwords
- Does not play well with disk encryption
Keys on local disk

- Not suitable for some key types
  - root passwords
- Does not play well with disk encryption
  - decrypt configuration management key
Keys on local disk

- Not suitable for some key types
  - root passwords

- Does not play well with disk encryption
  - decrypt configuration management key

- What about diskless/stateless systems?
Encrypted disks

- server
  - store
  - encrypt
- disk 1
- disk 2
- disk N
Unified Extensible Firmware Interface
Unified Extensible Firmware Interface

- Aka BIOS 2.0
Unified Extensible Firmware Interface

- Aka BIOS 2.0
- Standard pre-OS environment
Unified Extensible Firmware Interface

- Aka BIOS 2.0
- Standard pre-OS environment
- Extensible (you can write your own apps)
Unified Extensible Firmware Interface

- Aka BIOS 2.0
- Standard pre-OS environment
- Extensible (you can write your own apps)
- Supported by most major OSes
Unified Extensible Firmware Interface

- Aka BIOS 2.0
- Standard pre-OS environment
- Extensible (you can write your own apps)
- Supported by most major OSes
- Provides many advanced features
Unified Extensible Firmware Interface

- Aka BIOS 2.0
- Standard pre-OS environment
- Extensible (you can write your own apps)
- Supported by most major OSes
  - UEFI variables
- Provides many advanced features
UEFI variables

- Backed by flash memory on platform firmware chip
UEFI variables

- Backed by flash memory on platform firmware chip
- Can store standard and custom (OEM/user) data
UEFI variables

- Backed by flash memory on platform firmware chip
- Can store standard and custom (OEM/user) data
- Can be accessed after OS kernel booted
UEFI variables

- Backed by flash memory on platform firmware chip
- Can store standard and custom (OEM/user) data
- Can be accessed after OS kernel booted
- Have built-in support in Linux
# not needed for systemd-based Linux distributions
mount -t efivarfs efivarfs /sys/firmware/efi/efivars

# need to prepend data with 4 byte attr and put an “owner” GUID

```bash
cat <(printf "\x07\x00\x00\x00") <(cat mydata.bin) > /sys/firmware/efi/efivars/mydata-<some GUID>
```
# not needed for systemd-based Linux distributions
mount -t efivarfs efivarfs /sys/firmware/efi/efivars

# need to prepend data with 4 byte attr and put an "owner" GUID

cat <(printf "\x07\x00\x00\x00") <(cat mydata.bin) > /
/sys/firmware/efi/efivars/mydata-<some GUID>

● always available
UEFI variables in Linux

- always available
- can be accessed in early boot stages

# not needed for systemd-based Linux distributions

```
mount -t efivarfs /sys/firmware/efi/efivars
```

# need to prepend data with 4 byte attr and put an “owner” GUID

```
cat <(printf \"\x07\x00\x00\x00\") <(cat mydata.bin) > /sys/firmware/efi/effvars/mydata-<some GUID>
```
UEFI variables in Linux

- always available
- can be accessed in early boot stages
- however, may have limited storage

# not needed for systemd-based Linux distributions
mount -t efivarfs efivarfs /sys/firmware/efi/efivars

# need to prepend data with 4 byte attr and put an “owner” GUID
cat <(printf \"\x07\x00\x00\x00\") <(cat mydata.bin) > /sys/firmware/efi/efivars/mydata-<some GUID>
Keys in cryptography
Keys in cryptography

derive
Keys in cryptography
Key derivation functions

master key

KDF
Key derivation functions
Key derivation functions

- string1
- master key
- KDF
Key derivation functions

- string1
- master key
- KDF
- key1
Key derivation functions

- `string1` → `KDF` → `key1`
- `master key` → `KDF` → `string2`
Key derivation functions

- string1
- master key
- KDF
- string2
- key1
- key2
Generating key pairs
Generating key pairs

seed1 → Deterministic CSPRNG
Generating key pairs

seed1

Deterministic CSPRNG

keypair1
Generating key pairs

seed1

Deterministic CSPRNG

seed2

keypair1
Generating key pairs

- seed1
- Deterministic CSPRNG
- seed2
- keypair1
- keypair2
Introducing gokey tool

- master seed
- realm

For example, "ssh", "saltstack", "disk encryption" etc
Introducing gokey tool

For example “ssh”, “saltstack”, “disk encryption” etc.
Introducing gokey tool

HKDF

Deterministic CSPRNG

master seed

realm

For example “ssh”, “saltstack”, “disk encryption” etc
Introducing gokey tool

HKDF

Deterministic CSPRNG

ssh key

master seed

realm

For example “ssh”, “saltstack”, “disk encryption” etc
Introducing gokey tool

- Master seed
- Deterministic CSPRNG
- HKDF
- Deterministic CSPRNG
- SSH key
- Realm

For example, “ssh”, “saltstack”, “disk encryption” etc.
Introducing gokey tool

 master seed

 Deterministic CSPRNG

 saltstack key

 HKDF

 Deterministic CSPRNG

 ssh key

 realm

 For example “ssh”, “saltstack”, “disk encryption” etc
Key management

- Provisioning process ensures a master seed is generated and stored in UEFI on first boot
Key management

- Provisioning process ensures a master seed is generated and stored in UEFI on first boot
- Startup scripts “recover” (derive from master seed) configuration management credential (key)
Key management

● Provisioning process ensures a master seed is generated and stored in UEFI on first boot

● Startup scripts “recover” (derive from master seed) configuration management credential (key)

● Configuration management “recovers” all other keys
Key management

root-password: |
{% if grains['hostname'] == 'baredog' %}
    -----BEGIN PGP MESSAGE-----
    ...
{% elif grains['hostname'] == 'cheesyonion' %}
    -----BEGIN PGP MESSAGE-----
Key management

```plaintext
root-password: |
{% if grains['hostname'] == 'baredog' %}
    -----BEGIN PGP MESSAGE-----
    ...
{% elif grains['hostname'] == 'cheesyonion' %}
    -----BEGIN PGP MESSAGE-----

root-password: {% gokey('root-password') %}
ssh-key: {% gokey('ssh') %}
```
Key management

root-password: |  
{% if grains['hostname'] == 'baredog' %}  
-----BEGIN PGP MESSAGE-----  
...  
{% elif grains['hostname'] == 'cheesyonion' %}  
-----BEGIN PGP MESSAGE-----  

root-password: {% gokey('root-password') %}  
ssh-key: {% gokey('ssh') %}

ssh-key: {% gokey('ssh-v2') %}
Encrypted disks (previously)

- Server
  - Disk 1
  - Disk 2
  - Disk N

- Store
- Encrypt
Encrypted disks

server

encrypt

disk 1

disk 2

disk N
Conclusions

- Decouple key storage from regular storage
Conclusions

● Decouple key storage from regular storage
● Decouple key contents from key management
Conclusions

● Decouple key storage from regular storage
● Decouple key contents from key management
● Easy to add new or rotate existing keys
Conclusions

● Decouple key storage from regular storage
● Decouple key contents from key management
● Easy to add new or rotate existing keys
● Better security guarantees
Conclusions

- Decouple key storage from regular storage
- Decouple key contents from key management
- Easy to add new or rotate existing keys
- Better security guarantees

https://github.com/cloudflare/gokey
Thank you