MAY 21 , 2020

Crypto Fails and How to Tackle Them in Go

Anna-Katharina Wickert

About Me

Anna-Katharina Wickert

Work: PhD student

Organizes local chapters Go and Go Bridge user group

@akwickert

Outline

What to expect in this talk

- Why care about crypto?
- 6 common issues in-the-wild
- Repeatable in Go?
- Code Examples

Critical vulnerability patched in Schneider Electric car charging station

Schneider Electric is warning users of multiple vulnerabilities in the EVLink Parking product including a "Critical" vulnerability.

Schneider Electric is warning users of multiple vulnerabilities in the EVLink Parking product including a "critical" vulnerability.

The critical vulnerability is caused by hard-coded credentials that allows an attacker to gain access to the device, according to a Dec. 20 security notification issued by the firm.

Hard-coded credential

https://www.scmagazine.com/home/security-news/schneider-electric-is-warning-users-of-multiple-vulnerabilities-in-the-evlink-parking-product-including-a-critical-vulnerability/
20th of May 2020

SAYING THEY DON'T DECRYPT IT AT ANY POINT DOES NOT MEAN THAT THEY CANNOT DECRYPT IT AT ANY POINT

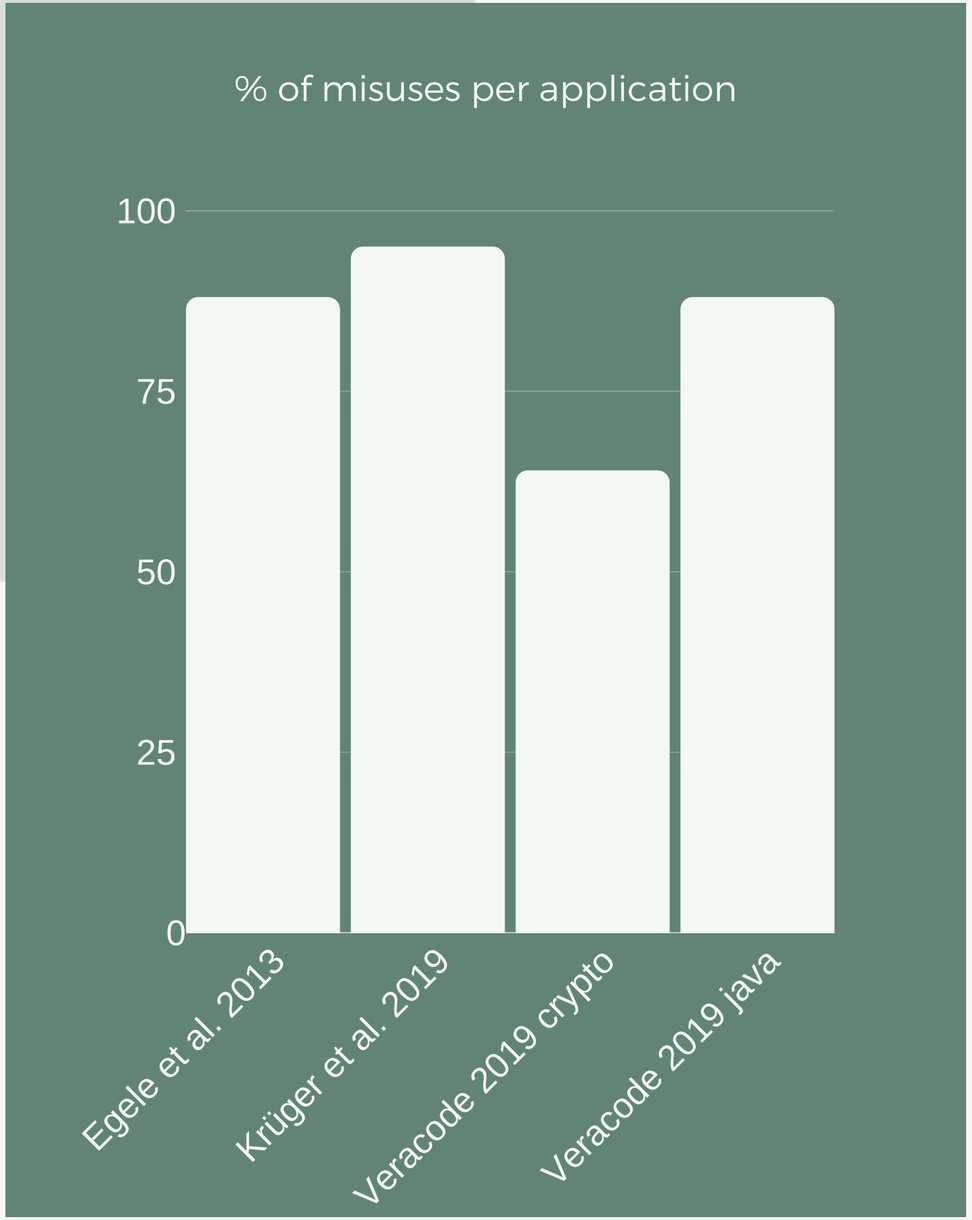
SENY KAMARA, BROWN UNIVERSITYS

https://www.wired.com/story/zoom-security-encryption/
22th of May

Crypto Misuses

Selected recent studies

Conducted in Academia (mostly Android Apps) and Industry



6 Crypto Misuses

Do not use ECB mode for encryption

Do not use a non-random IV for CBC encryption

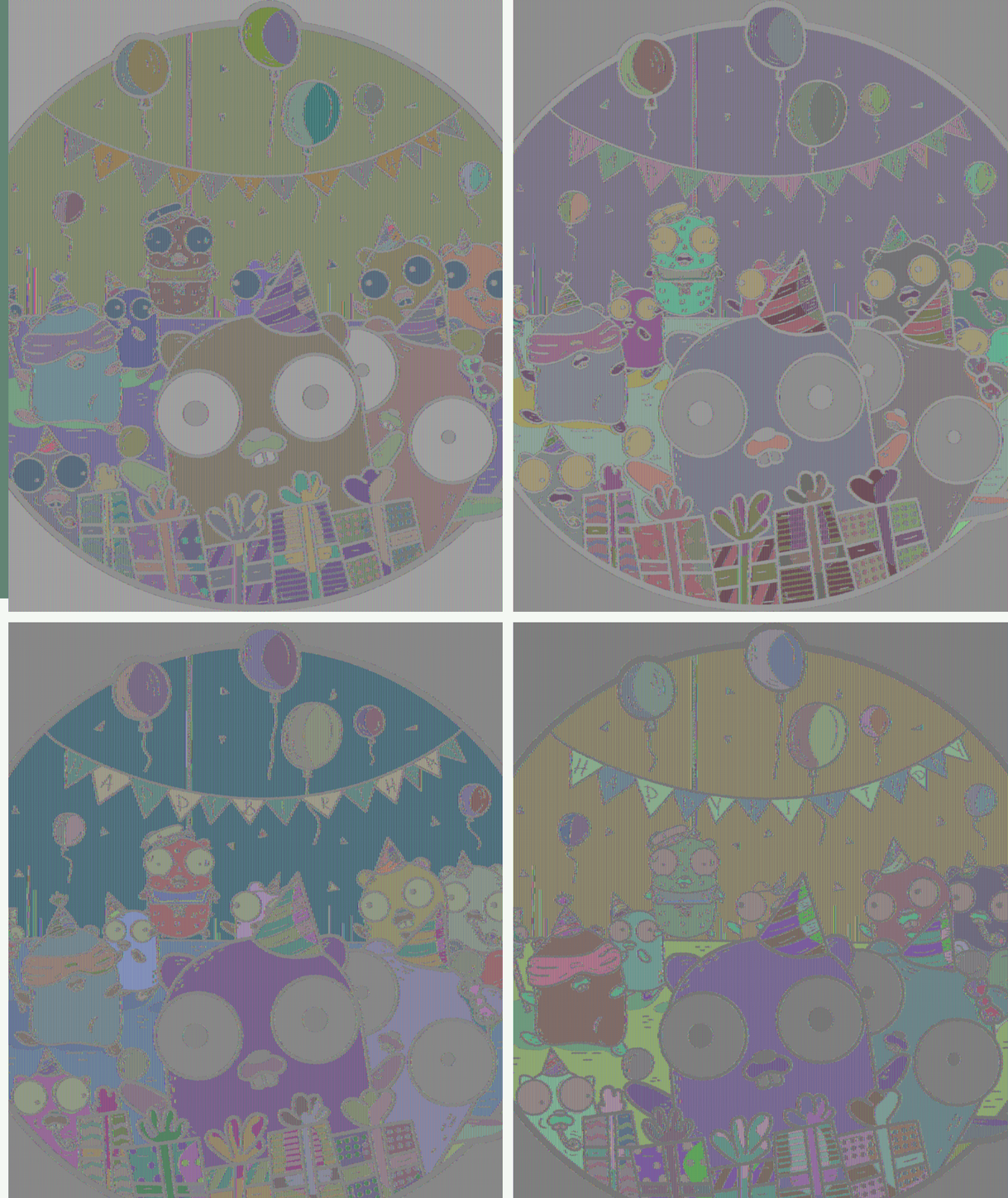
Do not use constant encryption keys

Do not use constant salts for PBE

Do not use fewer than 1000 iterations for PBE

Do not use static seeds to seed SecureRandom(.)

ECB Mode



Electronic Codebook

Splits the message into blocks
Encrypt each block separately

CBC Mode Constant IV

Cipher Block Chaining

XOR cipher text with previous cipher

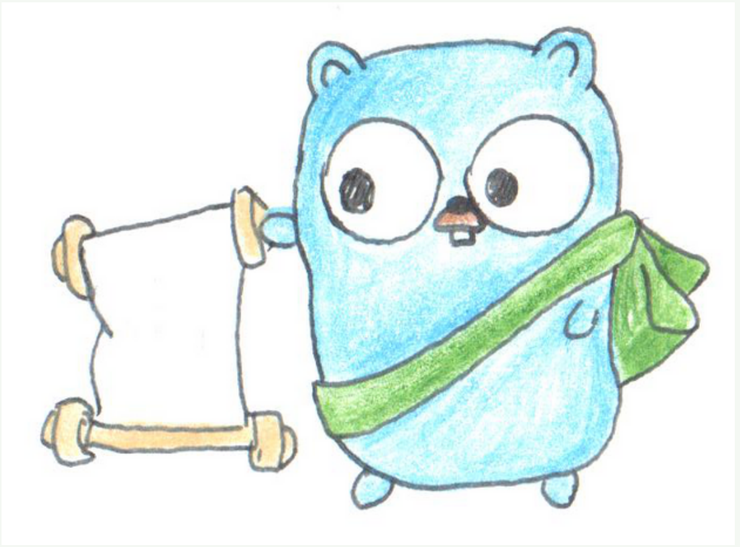
First block: Initialization vector (IV) .

Constant Encryption Key

Secret Encryption Key

Want: Message is a secret

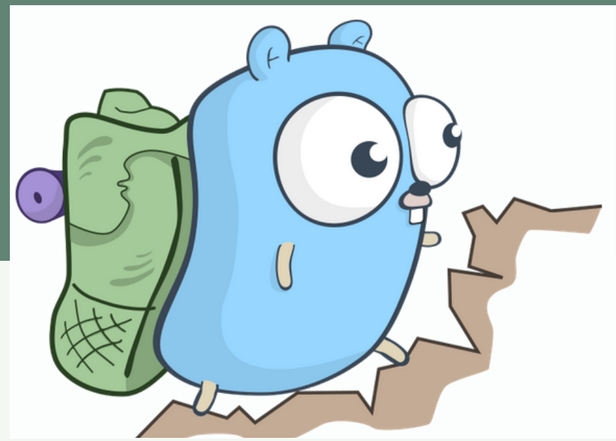
Got: Message is open



What About the Remaining?

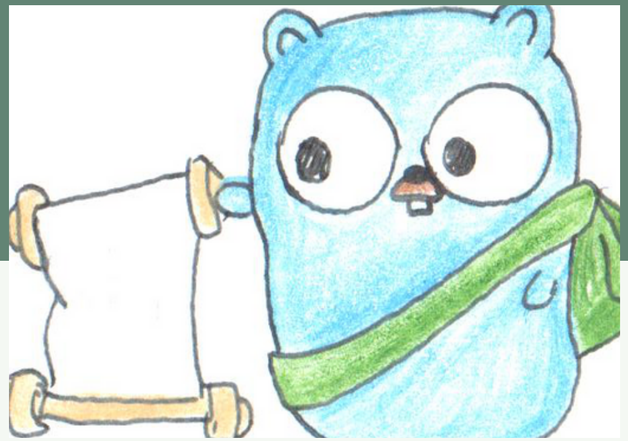
CONSTANT SALTS

Constant salt is known
Effect similar to no salt



>= 1000 ITERATIONS

Increases cost of key generation
Indirectly increase cost of attack



STATIC SEEDS SECURE RANDOM

Constant input generates constant "random"
Results constant key

Go is better than Java, isn't?

You still can repeat some misuses with the standard library

6 Crypto Misuses and the Go standard library

Do not use ECB mode for encryption

Do not use a non-random IV for CBC encryption

Do not use constant encryption keys

Do not use constant salts for PBE

Do not use fewer than 1000 iterations for PBE

Do not use static seeds to seed SecureRandom(.)

2 Code Examples

RANDOM NUMBERS AND AES ENCRYPTION

```
func NewRandom(length int) (random []byte, err error) {
    random = make([]byte, length)
    _, err = io.ReadFull(rand.Reader, random[:])
    if err != nil {
        return nil, err
    }
    return
}
```

Read from the cryptographic secure random number generator

AES Encryption with GCM
GCM = Cofidentiality + **Authentication**

```
func Encrypt(plaintext []byte, key *[32]byte) ([]byte, error) {
    block, err := aes.NewCipher(key[:])
    if err != nil {
        return nil, err
    }

    gcm, err := cipher.NewGCM(block)
    if err != nil {
        return nil, err
    }

    nonce := make([]byte, gcm.NonceSize())
    _, err = io.ReadFull(rand.Reader, nonce)
    if err != nil {
        return nil, err
    }

    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

Initialize block cipher

Choose GCM as block mode

Create a random nonce

Encrypt the plaintext

AES Encryption with GCM
GCM = Cofidentiality + **Authentication**

```
func Encrypt(plaintext []byte, key *[32]byte) ([]byte, error) {
    block, err := aes.NewCipher(key[:])
    if err != nil {
        return nil, err
    }

    gcm, err := cipher.NewGCM(block)
    if err != nil {
        return nil, err
    }

    nonce := make([]byte, gcm.NonceSize())
    _, err = io.ReadFull(rand.Reader, nonce)
    if err != nil {
        return nil, err
    }

    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

Initialize block cipher

Choose GCM as block mode

Create a random nonce

Encrypt the plaintext

AES Encryption with GCM
GCM = Cofidentiality + **Authentication**

```
func Encrypt(plaintext []byte, key *[32]byte) ([]byte, error) {
    block, err := aes.NewCipher(key[:])
    if err != nil {
        return nil, err
    }

    gcm, err := cipher.NewGCM(block)
    if err != nil {
        return nil, err
    }

    nonce := make([]byte, gcm.NonceSize())
    _, err = io.ReadFull(rand.Reader, nonce)
    if err != nil {
        return nil, err
    }

    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

Initialize block cipher

Choose GCM as block mode

Create a random nonce

Encrypt the plaintext

AES Encryption with GCM
GCM = Cofidentiality + **Authentication**

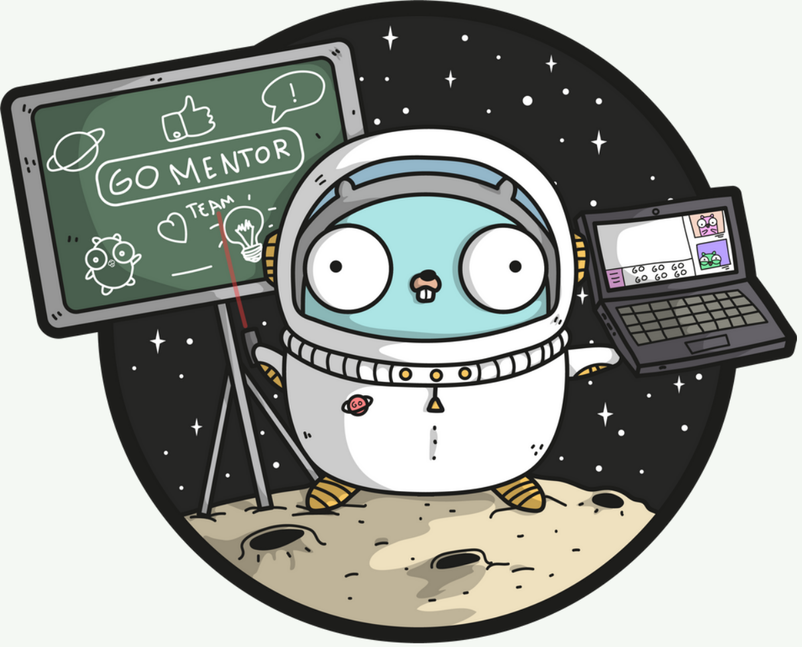
```
func Encrypt(plaintext []byte, key *[32]byte) ([]byte, error) {
    block, err := aes.NewCipher(key[:])
    if err != nil {
        return nil, err
    }

    gcm, err := cipher.NewGCM(block)
    if err != nil {
        return nil, err
    }

    nonce := make([]byte, gcm.NonceSize())
    _, err = io.ReadFull(rand.Reader, nonce)
    if err != nil {
        return nil, err
    }

    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

Initialize block cipher

Choose GCM as block mode

Create a random nonce

Encrypt the plaintext

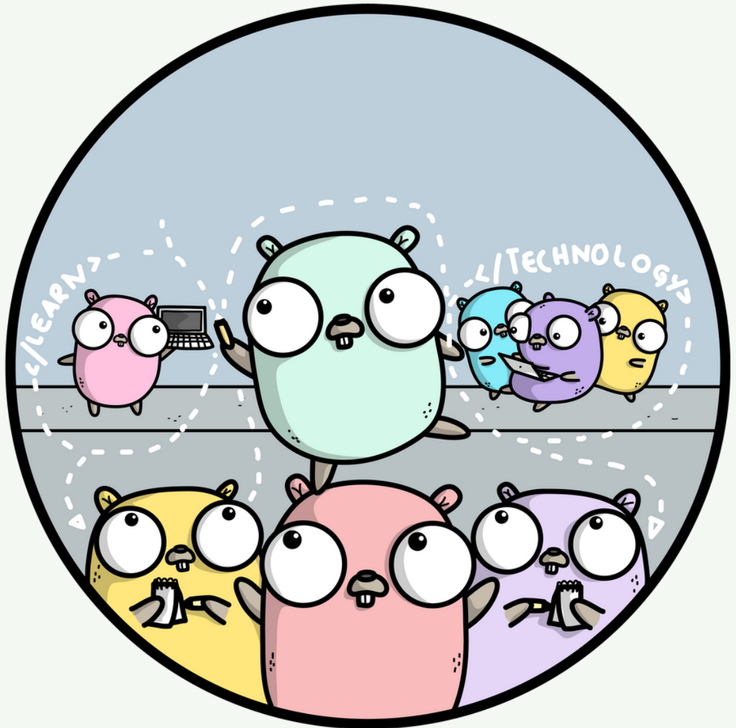


Existing Resources

https://github.com/gtank/cryptopasta

Gophers Slack - channel crypto

https://github.com/securego/gosec



GitHub Repo of this Talk

https://github.com/akwick/crypto-go

ECB Mode

Electronic Codebook

Splits the message into blocks
Encrypt each block separately

https://github.com/akwick/crypto-go
@akwickert

6 Crypto Misuses and the Go standard library

Do not use ECB mode for encryption

Do not use a non-random IV for CBC encryption

Do not use constant encryption keys

Do not use constant salts for PBE

Do not use fewer than 1000 iterations for PBE

Do not use static seeds to seed SecureRandom(.)

Random Numbers

```
func NewRandom(length int) (random []byte, err error) {
    random = make([]byte, length)
    _, err = io.ReadFull(rand.Reader, random[:])
    if err != nil {
        return nil, err
    }
    return
}
```

Read from the cryptographic secure random number generator

AES Encryption with GCM
GCM = Cofidentiality + **Authentication**

```
, key *[32]byte) ([]byte, error) {
(key[:])
    if err != nil {
        return nil, err
    }

    gcm, err := cipher.NewGCM(block)
    if err != nil {
        return nil, err
    }

    nonce := make([]byte, gcm.NonceSize())
    _, err = io.ReadFull(rand.Reader, nonce)
    if err != nil {
        return nil, err
    }

    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

Initialize block cipher

Choose GCM as block mode

Create a random nonce

Encrypt the plaintext

Acknowledgments - Images

- All gophers are based on the artwork by Renee French.
- 7TH_BIRTHDAY.png - Artwork by Ashley Willis licensed under CC BY-NC-SA 4.0 found on GitHub <https://github.com/ashleymcnamara/gophers>.
- Gophers message sketch - Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers>
- Gopher hiking - Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers>
- Gopher emoji - Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers>

Acknowledgments - Others

- Creating ECB-Gophers: The ECB Pinguin by Filippo Valsorda <https://blog.filippo.io/the-ecb-penguin/>
- Code examples are based on the available crypto pasta code snippets <https://github.com/gtank/cryptopasta>.