GCM Writeup

Author: Yaseen

Description:

```
An attacker managed to exfiltrate some sensitive data from an employee
personal laptop. You have been tasked to find out what that data is and
reassemble it.
The flag is the SHA-256 sum of the exfiltrated object.

Flag format: METACTF{SHA-256}
```

- We are handed a small pcap of total 90 packets

```
└─$ capinfos challenge.pcap


File name:           challenge.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Raw IPv4
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 65535 bytes
Number of packets:   90
File size:          9,065 bytes
Data size:          7,601 bytes
Capture duration:    89.000000 seconds
Earliest packet time: 2025-06-22 22:55:24.029197
Latest packet time:   2025-06-22 22:56:53.029197
Data byte rate:      85 bytes/s
Data bit rate:       683 bits/s
Average packet size: 84.46 bytes
Average packet rate: 1 packets/s
SHA256:
b28b5d17c82a2108aba36b57678796286d37c2b89ceb44e0eb93c4b3e4f53f3e
SHA1:                7c4ce0a306db895663e255e465a95227a2c98c0d
Strict time order:   True
Number of interfaces in file: 1
Interface #0 info:
                    Encapsulation = Raw IPv4 (129 - rawip4)
```

```
                      Capture length = 65535
                      Time precision = microseconds (6)
                      Time ticks per second = 1000000
                      Number of stat entries = 0
                      Number of packets = 90
```

Wireshark

- Follow HTTP Stream:

1. On stream `0`, we observe data in JSON format, including an entry named _key.

```
"_key":
"6433616462333366643361646233336664336164623333666433616462333366"
```

2. From Stream `#1` till the rest of the HTTP Stream, we notice a vectorsecret and what appears to be encrypted data.

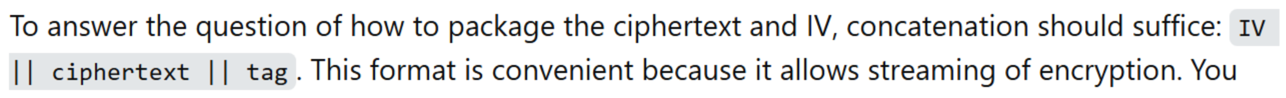
```
vectorsecret.........

....jY...E...1...g.....h.j.=..

.qK9&!.S.W.9^.k..[.Pfr.U.....DWY*a...
```

3. Going Back the challenge name it says **GCM** . A quick search on that encryption algorithm and the format to send encrypted data.

```
GCM (Galois/Counter Mode) is a widely used authenticated encryption mode
for block ciphers, like AES
```


To answer the question of how to package the ciphertext and IV, concatenation should suffice: `IV || ciphertext || tag`. This format is convenient because it allows streaming of encryption. You

- So each of the Data has this form IV || cipher || tag
- The IV is obvious and readable : vectorsecret (*12 bytes*)

- Taq size is 16 bytes https://iceberg.apache.org/gcm-stream-spec/#cipher-block-structure
- The rest in between is the cipher

sample:

```
└─$ tshark -r challenge.pcap -Y "http.request.method == POST" -T fields
-e data.data

766563746f72736563726574bfc4a6f9da8d8ea4bed10aab94a4b9315ff119d613acf147
77d9ebe832d8d1d8d41a6f5967f569160708b220136c2628f106d053ec6206f430eb9859
95536d9f8a7d71f7b17a642c3149314292a5c4ea
```

`IV: 766563746f72736563726574

`ct: bfc4a6f9da8d8ea4bed10aab94a4b9315ff119d613acf14777d9ebe832d8d1d8d41a6f5967f569160708b220136c2628f106d053ec6206f430eb985995536d9f

`Taq: 8a7d71f7b17a642c3149314292a5c4ea`

on `CyberChef` we get:

```
root:$5$ad3a540c20a34560$2a66d6298234107325cad208a549ba9bf15f915
```

- So the object seems to be the /etc/shadow from the user system
- Repeat these steps for each stream and assemble into one file then cacl it's hash

Check solve.py for automation