AEAD Praktika

Egileak: Oier Barrutiabengoa eta Gaizka Cerezo
Data: 27/10/2024

SARRERA

Praktika honetan UPV/EHUko Informatika Fakultateko CAU zerbitzuko langileak gara eta zifratutako 2 mezu susmagarri jaso ditugu:

- kaleratua izan zen lankide batek bidalitako barne-txateko mezua (CHAT_IV_CIPHER_HMAC.cipher).
- ventas@fnac.es kontuak bidalitako emaila (EMAIL_NONCE_CIPHER.cipher) (FACTURA_IV_CIPHER.cipher eta FACTURA_TAG.tag).

Mezu hauek UPV/EHUko barne exekutagarriak erabiliz deszifratzean ohartu gara exekutagarri hauek duela hilabete sortuak direla eta hauen kode zati batzuk hutsik ala gaizki funtzionatzen dute, beraz kodea rebisatzen hurrengoa ikusi dugu.

1. CHAT_DECRYPT_THEN_MAC.c

Hemen HMAC_SHA256 kalkulatzen duen funtzioa hutsa dago, eta beti “TAG IS VALID” pantailaratzen dela ikusi dugu. Hau konpontzeko lehenengo HMAC_SHA256 funtzioa bete egin dugu. Horretarako, lehenengo urratsa K0 xor ipad eta K0 xor opad egitea izan da, gero K0 xor ipad eta mezua jarraian uint berean gordetzen dugu eta sha256 funtzioari deituz hash berri bat lortzen dugu, gero k0 xor opad eta kalkulatutako hash-a jarraian gordeko ditugu uint berean eta guztia hasheatuko dugu.

```
void HMAC_SHA256(uint8_t* key, int nbytes_key, uint8_t* P, int nbytes_P, uint8_t*
HMAC) {
    uint8_t k0_ipad[nbytes_key];
    uint8_t k0_opad[nbytes_key];
    int total = nbytes_key+nbytes_P;
    uint8_t k0_ipad_m[total];

    for (int i = 0; i < nbytes_key; i++) {
            k0_ipad[i] = key[i] ^ IPAD;
            k0_opad[i] = key[i] ^ OPAD;
    }
    memcpy(k0_ipad_m, k0_ipad, nbytes_key);
    memcpy(k0_ipad_m + nbytes_key, P, nbytes_P);

    SHA256_CTX ctx;
    BYTE buf[SHA256_BLOCK_SIZE];
    sha256_init(&ctx);
```

```
    sha256_update(&ctx, k0_ipad_m, total);
    sha256_final(&ctx, buf);

    uint8_t k0_opad_hash[SHA256_BLOCK_SIZE+nbytes_key];
    memcpy(k0_opad_hash, k0_opad, nbytes_key);
    memcpy(k0_opad_hash + nbytes_key, buf, SHA256_BLOCK_SIZE);

    SHA256_CTX ctx2;

    sha256_init(&ctx2);
    sha256_update(&ctx2, k0_opad_hash, SHA256_BLOCK_SIZE+nbytes_key);
    sha256_final(&ctx2, HMAC);
}
```

Programa nagusian (main) erabiltzaileak emandako HMAK-a eta guk kalkulatutako emaitza konparatuko ditugu, biak berdinak baldin badira orduan TAG IS VALID pantailaratuko da eta bestela TAG IS INVALID.

```
if(memcmp(HMAC_rec, HMAC_calc, SHA256_BLOCK_SIZE)==0) {
        printf("TAG IS VALID!\n");
    } else {
        printf("TAG IS INVALID!\n");
    }
```

Programa exekutatu ondoren
`./CHAT_DECRYPT_THEN_HMAC CHAT_IV_CIPHER_HMAC.cipher ce44250a450433fe25a75f613ed7aa03 fe0431ed135846f0859143100e0bfe23`
eginez TAG IS INVALID pantailaratzen du, mezua aldatu egin dela egiaztatuz.

2. EMAIL_DECRYPT_CCM.c

Hemen nbytes ez zen egiaztatzen, eta beti “TAG IS VALID” bueltatzen zuen. Aes.c programan AES_CCM_DECRYPT funtzioa begiratu ondoren, tag-a ez zuela kalkulatzen ikusi dugu, hau konpontzeko calculate_tag_AES_CCM funtzioari deitu eta emaitza emandako tag-arekin konparatzen dugu, ez badira berdinak orduan nbytes_P=0 egiten dugu eta P ren posizio guztietan 0 bat jartzen dugu.

```
    uint8_t T_calc[nbytes_T];
    calculate_tag_AES_CCM(B, nbytes_B/AES_BLOCKLEN, key, T_calc, nbytes_T);
    print_hex(T_calc, nbytes_T);
    print_hex(T_rec, nbytes_T);

    if (memcmp(T_calc, T_rec, nbytes_T) != 0) {
        memset(P, 0, nbytes_P);
        nbytes_P = 0;
    }
    free(B); free(iv); free(T_rec);
    return(nbytes_P);
```

Programa nagusian bakarrik nbytes_P 0 den egiaztatzen dugu eta “TAG IS INVALID” EDO “TAG IS VALID” pantailaratuko du.

```
if(nbytes_P == 0) {
        printf("TAG IS INVALID!\n");
    } else {
        printf("TAG IS VALID!\n");
    }
```

Programa exekutatu ondoren `./EMAIL_DECRYPT_CCM EMAIL_NONCE_CIPHER.cipher EMAIL_ASSOCIATED_DATA.txt 1e2350aa546771f035478fdf30ee4a2e erabiliz` “TAG IS VALID” pantailaratuko du

3. ATTACH_DECRYPT_GCM.c

Hemen beti, “TAG IS VALID” pantailaratzen da eta AES_GCM_decrypt funtzioan beti valid 1 bueltatzen da, gainera tag-a ez da kalkulatzen. Programan J0 eta H kalkulatzen dira baina GHASH eta GCTR ez dira kalkulatzen, hau konpontzeko calculate_S funtzioari deituko diogu eta ondoren AES_CTR_xcrypt funtzioa erabiliz tag bat kalkulatuko dugu. Azkenik, emandako tag-a eta kalkulatutakoa konparatzen ditugu, berdinak ez badira orduan bufferra hustu eta valid=0 egingo dugu bestela falta den kalkulua egingo du P (mezua) kalkulatzeko AES_CTR_xcrypt eginez.

```
uint8_t S[AES_BLOCKLEN];
    calculate_S(buf, nbytes_buf, A, nbytes_A, H, S);
    AES_CTR_xcrypt(S, AES_BLOCKLEN, J0, key);
    uint8_t T_copy[AES_BLOCKLEN];
    memcpy(T_copy, S, AES_BLOCKLEN);

    int valid = 1;
    if (memcmp(T_copy, T, AES_BLOCKLEN) != 0) {
        memset(buf, 0, nbytes_buf);
        valid = 0;
    }
    J0[AES_BLOCKLEN-1] += 1;
    AES_CTR_xcrypt(buf, nbytes_buf, J0, key);
    free(H); free(J0);
    return valid;
```

Hemen programa exekutatzeko output file formatu ezberdinetan gorde ditugu, baina emaitza ona lortu ondoren pdf 1.5 dela egiaztatu dugu.
`./ATTACH_DECRYPT_GCM FACTURA_IV_CIPHER.cipher FACTURA_TAG.tag 1e2350aa546771f035478fdf30ee4a2e output_file.pdf` eginez TAG IS VALID eta fakturaren pdf-a bueltatzen digu

FACTURA SIMPLIFICADA

Nº: 22V000467

Nº documento: 22V000467
Referencia: 22_V000_467_2024
Fecha: 09/09/2024 13:51
IBAN: ES58 3035 0050 81 0500035746

REFERENCIA	PRODUCTO	PRECIO UNIDAD	IVA	UNIDADES	SUBTOTAL
023-5534	Apple iPad Pro 2024 13" M4 2TB Cellular Vidrio nanotexturizado Negro Tablet Microsoft Surface	3.139,00	IVA21	10	31.390,00
023-5522	Pro 9 16GB/1TB 13" i7 Plata	2.199,00	IVA21	15	32.985,00
express-48 – Express (48 horas)	Gastos de envio	7,00	IVA21	1	7,00

IMPUESTOS	%	IMPORTE	IMPORTE IVA	IMPORTE TOTAL
IVA21	21	1	13.520,22	77.902,22

TOTAL A PAGAR

77.902,22 EUR

ONDORIOA

UPV/EHUko barne exekutagarriak eguneratzen baditugu egindako zuzenketekin hau ez da berriro gertatuko. Beste aldetik, kaleratutako lankideak osotasuna egiaztatzen duten algoritmoak ez zituelako ezagutzen bere mezua faltsua dela egiaztatu dugu.