Implementasi Algoritma Kriptografi AES untuk Pengamanan Jalur Komunikasi Command and Control (C2) pada Simulasi RAT

Indraswara Galih Jayanegara and 13522119[1,2,3]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522119@std.stei.itb.ac.id, [2]indrswara@gmail.com, [3]hi@indraswara.me

***Abstrak*—*Malware* adalah ancaman yang semakin berkembang seiring dengan majunya teknologi saat ini. Para peretas juga semakin kreatif untuk menyembunyikan komunikasi dari *malware* seperti menggunakan kriptografi kunci-simetri untuk menyembunyikan komunikasi antara peretas dan *malware* yang ada pada korban. Makalah ini akan menunjukkan bagaimana cara kerja penyembunyian komunikasi antara *malware* dan server menggunakan AES.**

***Keywords*—*Malware*, *Trojan*, *Kunci-Publik*, *Command and Control*.**

I. Pendahuluan

Di era digital yang semakin maju saat ini, penggunaan teknologi tidak lepas dari keseharian sebagian besar manusia. Perubahan pesat ini mengubah manusia untuk beraktivitas sehari-hari karena hampir seluruh aktivitas yang ada pada manusia saat ini membutuhkan koneksi internet. Dengan adanya internet dan kemajuan yang sangat pesat membuat pekerjaan manusia menjadi lebih mudah. Hal ini, bisa dilihat banyaknya pekerjaan yang bisa dilakukan dari jarak jauh (*remote work*) [1]. Namun, hal ini juga meningkatkan ancaman keamanan siber yang semakin kompleks. Salah satu bentuk ancaman yang nyata adalah *malware* atau bisa disebut dengan *malicious software* yang dapat menyusup ke dalam perangkat pengguna. *Malware* yang dapat masuk ke dalam perangkat pengguna biasanya adalah *Trojan*. *Trojan* dapat masuk ke dalam sistem dan melakukan hal-hal yang tidak diinginkan oleh pengguna. Hal ini juga didukung dengan adanya COVID-19 yang membuat frekuensi kerja jarak jauh semakin meningkat. Namun, perlu diingat semakin tinggi frekuensi kerja jarak jauh semakin tinggi kemungkinan serangan siber yang terjadi akibat dari *phising, social engineering*, dan metode lainnya.

Semakin berkembangnya internet saat ini yang membuat manusia termudahkan dengan teknologinya ada hal lain yang juga berkembang yaitu, kreativitas para peretas dalam membuat *malware*. Hal-hal yang sudah diketahui seperti menyembunyikan komunikasi *malware* menggunakan enkripsi tambahan seperti AES agar *traffic*-nya sulit untuk dianalisis [2].

II. Dasar Teori

A. Kriptografi

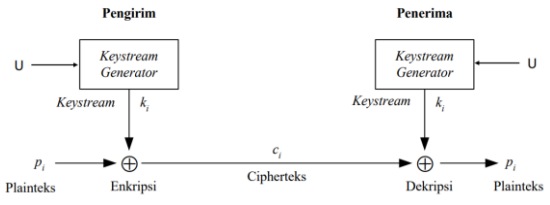
Secara etimologis, istilah kriptografi berakar dari bahasa Yunani, yakni *kryptós* (tersembunyi) dan *gráphein* (menulis), yang secara harfiah diartikan sebagai "tulisan rahasia". Definisi ini telah berevolusi dari sekadar teknik menyandikan pesan menjadi disiplin ilmu yang memadukan seni dan sains dalam mengamankan informasi. Secara teknis, kriptografi memanfaatkan algoritma matematis untuk mengubah pesan asli (*plaintext*) menjadi kode tersandi (*ciphertext*) melalui proses enkripsi dan dekripsi yang berbasis kunci. Tujuan penerapannya pun meluas; tidak hanya menjamin kerahasiaan (*confidentiality*), tetapi juga memastikan integritas data (*data integrity*) agar tidak dimanipulasi, memverifikasi keaslian pengirim (*authentication*), serta menyediakan fitur nirsangkal (*non-repudiation*) agar pengirim tidak dapat mengelak atas pesan yang dikirimnya [3].

B. Kriptografi Kunci-Simetri

Kriptografi memiliki banyak jenisnya mulai dari kriptografi klasik sampai yang modern. Salah satu kriptografi yang terbilang modern adalah kriptografi kunci-simetri. Terdapat dua tipe *cipher* alir dan *cipher* blok.

1. *Cipher* Alir

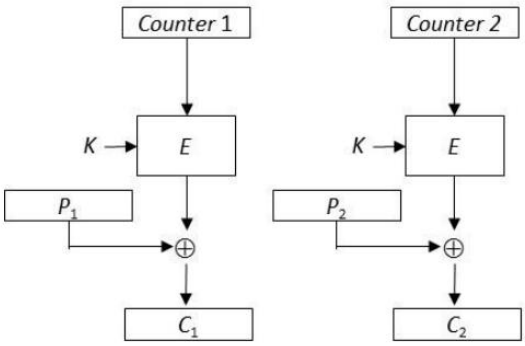
Cipher alir mengenkripsi data secara bit per bit atau *byte* per *byte*. Metode ini diperkenalkan oleh Vernam dan diadopsi dari konsep *One-Time Pad*. Enkripsi dan dekripsi dilakukan dengan operasi XOR antara data dan aliran kunci (*keystream*) [4].



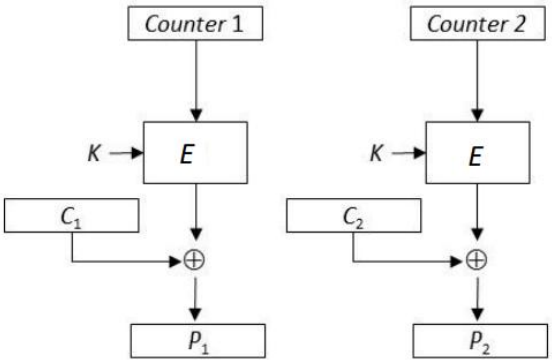
Gambar II.1 Cara kerja *stream cipher* [3]

2. *Cipher* Blok

Berbeda dengan *cipher* alir, cipher blok membagi plainteks menjadi blok-blok bit dengan panjang tetap (misalnya 64 bit atau 128 bit) sebelum dienkripsi. Karena data dibagi menjadi blok-blok, diperlukan mode operasi untuk menangani pesan yang panjangnya lebih dari satu blok. Terdapat 5 mode operasi utama yaitu, ECB (*Electronic Code Block*), CBC (*Cipher Block Chaining*), CFB (*Cipher Feedback*), OFB (*Output Feedback*), CTR (*Counter Mode*). Tidak semua dijelaskan karena akan diutamakan untuk CTR karena implementasi yang digunakan pada makalah ini adalah AES-CTR [5].



Gambar II.2 Enkripsi mode *counter*



Gambar II.3 Dekripsi mode *counter*

Seperti yang terlihat pada gambar II.2 dan II.3 setiap keluaran enkripsi/dekripsi yang ada akan dilakukan XOR terhadap *counter*.

C. Malware

Malware atau biasa disebut dengan *malicious software* merupakan *software* yang memiliki ancaman karena memiliki fungsionalitas yang berbahaya seperti *keylogger,* menginstall *malware* lainnya, memasang *spyware*. Peretas biasanya mengirim *malware* menggunakan beberapa metode *delivery* seperti *phising*, *social engineering*, atau melalui melalui website. *Malware* yang masuk tidak serta merta *malware* yang utama, tapi bisa *dropper* atau *injector*, *downloader*, dan teknik-teknik lainnya.

Dengan menggunakan *downloader* biasanya *malware* yang masuk akan mengunduh *payload* utama dari server lain. Akan tetapi, perlu diingat *payload* yang diundur biasanya dienkripsi lagi menggunakan enkripsi tambahan agar tidak terdeteksi oleh antivirus karena jika terdeksi *payload* yang diunduh adalah *malware*, maka unduhannya akan ditolak.

Teknik *dropper*, berbeda dari *downloader* yang mengunduh *payload* utama dari server eksternal. *Dropper* membawa *malware* utama bersama dirinya yang setelah berhasil di-*delivery dropper* akan menjalankan tugasnya untuk menginstall *malware* utama pada sistem.

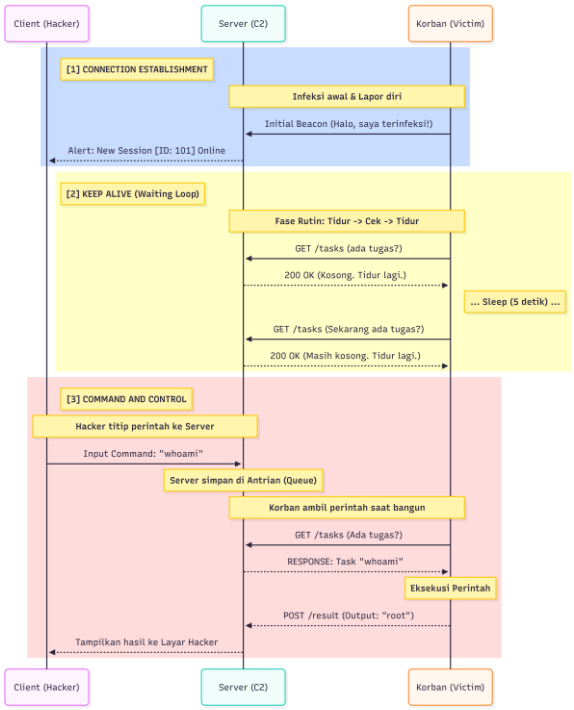
Selain menggunakan teknik untuk *delivery* ada pula teknik yang biasa digunakan oleh peretas seperti *bypass UAC* dan persistensi sehingga *malware* dapat aktif meskipun saat perangkat di-*restart* atau *reboot* karena sudah diatur untuk persisten.

D. Remote Access Trojan

Remote Access Trojan (*RAT*) merupakan bagian dari *Trojan*. RAT memungkinkan peretas untuk mengontrol sistem dan mencuri data dari korban dari jarak jauh. *Trojan* juga memungkinkan peretas untuk memata-matai korban menggunakan mekanisme *keylogger* untuk mencatat aktivitas *keyboard* dari korban dengan tujuan mencuri kredensial dan hal-hal sensitif lainnya. *Trojan* memiliki mekanisme koneksi seperti pada gambar II.1. Hal pertama yang dilakukan adalah melakukan *connection establishment* antara korban yang telah disisipi oleh *malware* dan server *Command and Control* (*C2*). Selanjutnya, bagian *keep alive* yang mana *malware* akan secara proaktif terhubung dengan server C2 untuk meminta tugas selanjutnya yang harus dieksekusi. Terakhir, *Command and Control*, Pada fase ini peretas akan mengirimkan *command* ke server C2 dan server akan meneruskan *command* tersebut pada *malware* yang sudah terinstal pada perangkat korban.

Fase pertama adalah fase yang terbilang sulit untuk dilalui karena pada fase ini *malware* harus tidak terdeteksi oleh antivirus seperti Windows Defender agar berjalan dengan normal, untuk mencapai hal ini biasanya dilakukan teknik seperti obfuskasi *malware* agar *malware* tidak terdeteksi dan bisa berjalan dengan normal.

Fase kedua dan ketiga bisa dilalui dengan normal jika fase pertama sudah dilalui. Biasanya pada dua fase ini dilakukan obfuskasi *payload* yang terkirim dan yang diterima. Baik itu dari sisi server C2, maupun dari sisi *malware* yang ada di korban. Enkripsi ini memungkinkan analisis *traffic* seperti *wireshark* sulit untuk dilakukan karena *traffic* yang lewat tidak dalam bentuk *plainteks*. Skema enkripsi yang dilakukan bisa menggunakan kunci-simetri, kunci publik, atau yang paling sederhana seperti *base64* [6].



Gambar II.1 Mekanisme koneksi *malware*

E. Command and Control

Command and Control (*C2*) merupakan server yang digunakan untuk mengontrol *malware* seperti *RAT* dari jarak jauh. Banyak tipe dari server *Command and Control* (C2) seperti di-*deploy* pada server privat yang bisa diakses selayaknya website pada umumnya, menggunakan CLI langsung, atau menggunakan *platform* publik seperti *twitter, youtube,* dan *platform* lainnya. Inti dari server C2 ini adalah mengontrol *malware* dari jarak jauh sesuai dengan kenginan peretas. Selain itu, *Command and Control* (*C2*) seharusnya mengikuti fungsionalitas yang ada pada *malware*. Umumnya, *Trojan* memiliki fungsionalitas seperti memiliki akses pada shell, bisa melakukan *download* file/folder dari perangkat korban ke perangkat lokal, *upload* file/folder dari perangkat local ke perangkat korban. Penggunaan *upload* ini yang dapat dimanfaatkan untuk eksploitasi lebih lanjut seperti menginstall *spyware* lain yang bisa mendapatkan kredensial dari korban, menyadap pembicaraan korban, dan bisa memantau korban menggunakan kamera perangkat.

F. Network Analyzer

Network Analyzer digunakan untuk menganalisa lalu lintas yang terjadi atau istilah yang umum adalah *sniffing*.

III. Desain dan Implementasi

A. Skema Kriptografi

Skema kriptografi yang akan digunakan adalah kriptografi kunci-publik untuk menyamarkan komunikasi antara server C2 dan *malware* yang terpasang pada korban. Pada paper ini percobaannya akan dilakukan pada perangkat-perangkat milik penulis sehingga tidak akan ada korban asli. Skema kunci-simetri yang dipilih adalah AES-CTR karena pada skema ini tidak perlu melakukan *padding*. Selain itu, AES-CTR dipilih karena implementasi kriptografi pada *malware*-nya sendiri dilakukan secara manual dikarenakan tidak adanya modul yang mirip dengan *pycryptodome* pada bahasa c++. Untuk kemudahan *Initialization Vector* (*IV*) dibuat statik pada *malware* dan server C2.

B. Implementasi

Bagian ini mendefinisikan parameter dasar algoritma AES-128 yang akan digunakan pada *malware*.

```
CONST AES_BLOCK_SIZE = 16
CONST AES_KEY_SIZE = 16
CONST AES_ROUND_KEY_SIZE = 176

// Substitution Box
CONST SBOX[256] = { 0x63, 0x7c, ... }

// Round Constants
CONST RCON[11] = { 0x8d, 0x01, 0x02, 0x04, 0x08,
0x10, 0x20, 0x40, 0x80, 0x1b, 0x36 }

STRUCT AESCTRStream:
    byte counter[16]    // IV + Counter (Big Endian)
    byte stream[16]     // Keystream buffer
    int offset          // Current byte position in stream
```

Pada bagian dibawah merupakan fungsi *KeyExpansion*. Bagian ini akan mengubah kunci yang awalnya 16-*byte* menjadi kunci yang panjangnya 176-*byte* yang akan dipakai pada setiap *round*. Menggabungkan semua operasi di atas untuk mengenkripsi satu blok data 16-byte.

```
FUNCTION KeyExpansion(output roundKey[], input
key[]):
    // Copy the first 16 bytes
    COPY key[0..15] TO roundKey[0..15]

    // Generate the rest
    FOR i = 16 TO 175 STEP 4:
        byte temp[4] = roundKey[i-4 .. i-1]

        IF (i MOD 16) == 0:
            // RotWord
            byte t = temp[0]
            temp[0] = temp[1]
            temp[1] = temp[2]
            temp[2] = temp[3]
            temp[3] = t

            // SubWord (SBOX)
            temp[0] = SBOX[temp[0]]
            temp[1] = SBOX[temp[1]]
            temp[2] = SBOX[temp[2]]
            temp[3] = SBOX[temp[3]]

            // XOR with Rcon
            temp[0] = temp[0] XOR RCON[i / 16]
```

```
        // XOR with the word 16 bytes back
        FOR j = 0 TO 3:
        roundKey[i + j] = roundKey[i - 16 + j] XOR temp[j]
```

Pada bagian dibawah merupakan fungsi utama dari AES yang dibuat. Fungsi-fungsi matematika untuk mengacak data dalam matriks 4x4.

```
   FUNCTION AddRoundKey(state[4][4], roundKey[], round):
      FOR i = 0 TO 3:
         FOR j = 0 TO 3:
            state[j][i] = state[j][i] XOR roundKey[(round * 16) + (i * 4) + j]

   FUNCTION SubBytes(state[4][4]):
      FOR i = 0 TO 3:
         FOR j = 0 TO 3:
            state[j][i] = SBOX[state[j][i]]

   FUNCTION ShiftRows(state[4][4]):
      // Row 0: No shift
      // Row 1: Shift left 1
      // Row 2: Shift left 2
      // Row 3: Shift left 3

   FUNCTION MixColumns(state[4][4]):
      // Matrix multiplication in Galois Field (GF(2^8))
      FOR i = 0 TO 3:
         t = state[i][0] XOR state[i][1] XOR state[i][2] XOR state[i][3]
      // ...

   FUNCTION Cipher(output[16], input[16], roundKey[]):
      byte state[4][4] = input  // Map linear input to 4x4 matrix

      // Initial Round
      AddRoundKey(state, roundKey, 0)

      // Main Rounds (1 to 9)
      FOR round = 1 TO 9:
         SubBytes(state)
         ShiftRows(state)
         MixColumns(state)
         AddRoundKey(state, roundKey, round)

      // Final Round (10)
      SubBytes(state)
      ShiftRows(state)
      AddRoundKey(state, roundKey, 10)

   output = state  // Map matrix back to linear output
```

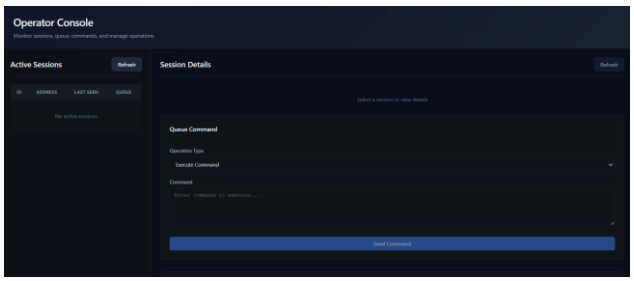
Pada bagian dibawah merupakan fungsi untuk *Counter Mode Logic* (CTR). Bagian ini mengubah Block Cipher menjadi Stream Cipher. Enkripsi dan dekripsinya melakukan XOR.

```
   FUNCTION increment_counter(counter[16]):
      // Loop backwards from the last byte
      FOR i = 15 DOWNTO 0:
         counter[i] = counter[i] + 1
         IF counter[i] != 0:
            BREAK // No overflow, done
         // If 0, loop continues to carry over to the next byte

   FUNCTION aes_ctr_apply(ctx, data[], length):
      WHILE length > 0:
         // If stream buffer is exhausted, generate a new block
         IF ctx.offset >= 16:
            byte block[16]

            // Encrypt the Counter to get Keystream
            Cipher(block, ctx.counter, GlobalRoundKey)

            // Update stream buffer
            COPY block TO ctx.stream

            // Reset offset and increment counter
            ctx.offset = 0
            increment_counter(ctx.counter)

         // Determine chunk size (remaining bytes in buffer vs remaining data)
         int chunk = 16 - ctx.offset
         IF chunk > length:
            chunk = length

         // XOR Data with Keystream
         FOR i = 0 TO chunk - 1:
            data[i] = data[i] XOR ctx.stream[ctx.offset + i]

         // Advance pointers
         data = data + chunk
         length = length - chunk
      ctx.offset = ctx.offset + chunk
```

Bagian di atas merupakan bagian penting untuk enkripsi yang dilakukan pada bahasa C yang nantinya akan digunakan oleh *malware*. Untuk server C2-nya

C. Tampilan Server C2

Tampilan server C2 yang akan mengontrol *malware* pada perangkat korban, Server C2 diimplementasikan menggunakan *python*.



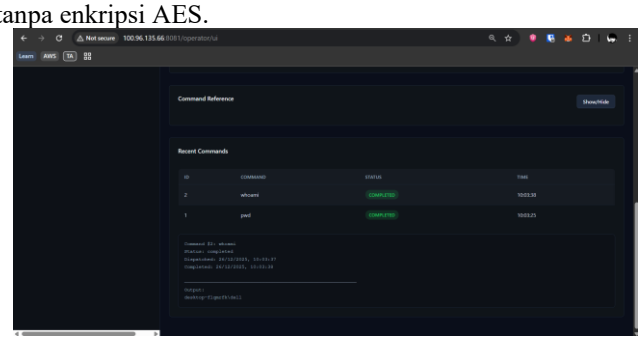
Gambar III.2 Tampilan GUI untuk server C2

IV. PENGUJIAN

Pengujian dilakukan dengan menggunakan *tools network sniffing* seperti *wireshark* untuk mengecek apakah berhasil terenkripsi atau tidak. Pengujian akan dilaukan dengan dua tipe. Satu yang menggunakan enkripsi AES dan satu lagi yang tidak menggunakan enkripsi. Pengujian juga dilakukan dengan mengabaikan antivirus untuk mencegah program tidak berjalan dengan normal ketika antivirus dinyalakan. Hal tersebut dilakukan karena makalah ini hanya menunjukkan pengamanan jalur komunikasi antara *malware Trojan* dan server *Command and Control* (*C2*) sehingga hal-hal yang berkaitan dengan *bypass* antivirus, *bypass UAC*, Persistensi tidak dibahas lebih detail. Selain itu, disini pada makalah ini koneksinya tidak memakai https, sehingga hanya memakai http selain karena lebih mudah tanpa memikirkan enkripsi tls, dengan menggunakan http

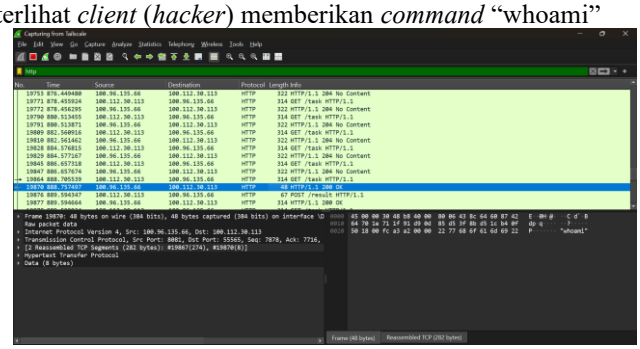
A. Pengujian Tanpa Enkripsi

Bagian ini adalah pengujian *malware* dan server C2 tanpa enkripsi AES.



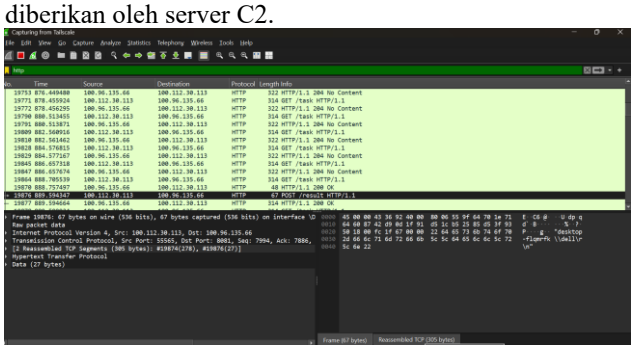
Gambar IV.1 *command* "whoami" pada server C2

Lalu *traffic*-nya bisa dilihat dari gambar IV.2 yang terlihat *client* (*hacker*) memberikan *command* "whoami"



Gambar IV.2 *packet* yang berhasil di-*sniff* melalui wireshark (*command* "whoami")

Malware yang terletak perangkat korban memberikan balasan dari hasil eksekusi command "whoami" yang diberikan oleh server C2.

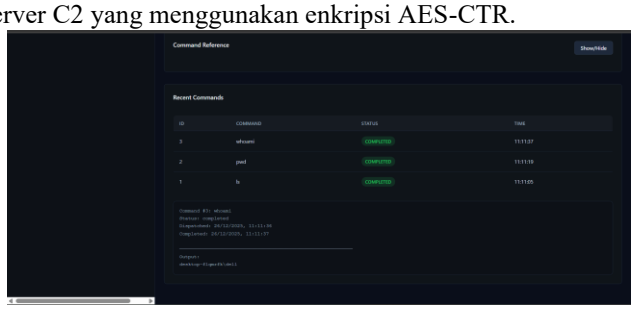


Gambar IV.3 *packet* dari *malware* ke *server*

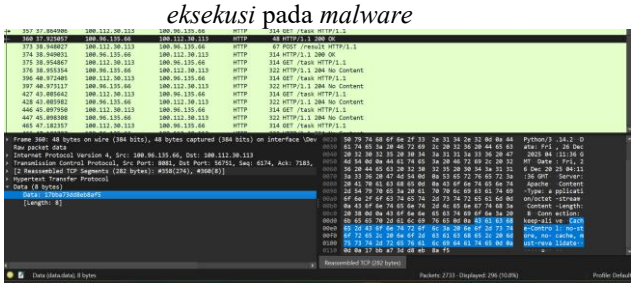
Setelah melihat *packet* yang dikirim dari server ke *malware* dan sebaliknya bisa diketahui ini tidak terenkripsi sama sekali.

B. Pengujian dengan Enkripsi

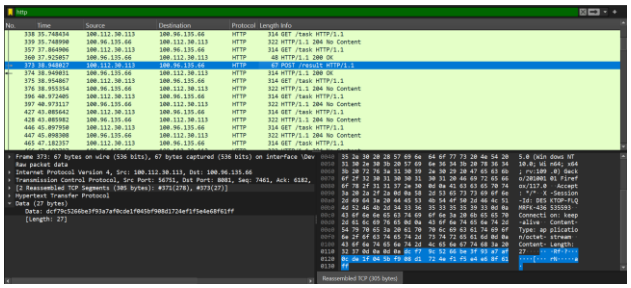
Bagian ini merupakan pengujian pada *malware* dan server C2 yang menggunakan enkripsi AES-CTR.



Gambar IV.4 server C2 yang berhasil menerima hasil *eksekusi* pada *malware*



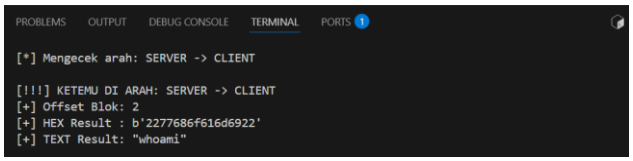
Gambar IV.5 *packet* yang berhasil di-*sniff* yang terdeteksi dan terenkripsi



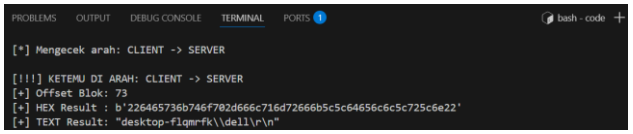
Gambar IV.6 *packet* dari *malware* ke server C2 yang terenkripsi

Bisa dilihat arah *packet* dari server C2 ke *malware* merupakan "17bba73dd8eb8af5" dan data dari *malware* ke server C2 adalah data hex berikut ini "dcf79c5266be3f93a7af0cde1f045bf908d1724ef1f5e4e68f61ff". Memang terlihat acak, tapi coba kita dekripsi ini

menggunakan *logic* yang sama dari AES yang sudah dibuat.



Gambar IV.7 hasil dekripsi data yang berhasil di-*sniff* dari server C2 ke *malware*.



Gambar IV.8 hasil dekripsi data yang berhasil di-*sniff* dari *malware* ke server C2.

Bisa dilihat ternyata hasilnya sesuai dengan apa yang dimasukkan di server C2 ke *malware* pada web C2. Hal ini menandakan enkripsinya AES-nya berhasil. *Command*-nya sendiri adalah "whoami" dan *result*-nya adalah "desktop-flqmrfk\\dell\r\n"

V. KESIMPULAN

Kesimpulannya yang bisa diambil adalah implementasi AES untuk mengenkripsi *traffic* antara server C2 dan *malware* yang terletak pada perangkat korban berhasil dilakukan. Ini bisa dilihat dari hasil pengujian sebelumnya yang mana terdapat dua versi yaitu, tanpa enkripsi dan dengan enkripsi.

VI. SARAN

Saran yang bisa dilakukan selanjutnya adalah dengan menggunakan HTTPS sebagai komunikasi antara *malware* dan server untuk menunjukkan semakin kreatifnya para penyerang dalam melakukan aksinya.

VII. UCAPAN TERIMA KASIH

Penulis berterima kasih kepada Allah SWT yang telah memberkahi rahmat-Nya sehingga penulis mampu menyelesaikan makalah IF4020 Kriptografi. Penulis mengucapkan terima kasih kepada Pak Rinaldi Munir yang membimbing penulis selama di kelas.

PRANALA GITHUB

https://github.com/Indraswara/aes-RAT

REFERENSI

[1] J. Kotak, E. Habler, O. Brodt, A. Shabtai, and Y. Elovici, "Information Security Threats and Working from Home Culture: Taxonomy, Risk Assessment and Solutions," Sensors (Basel), vol. 23, no. 8, p. 4018, Apr. 2023, doi: 10.3390/s23084018.

[2] D. Yadav, G. Kumar, D. Lakshmi Kameshwari, V. K. Gunjan, and S. Kumar, "Malware Techniques and Its Effect: A Survey," in ICCCE 2021, vol. 828, A. Kumar and S. Mozar, Eds., in Lecture Notes in Electrical Engineering, vol. 828. , Singapore: Springer Nature Singapore, 2022, pp. 1215–1225. doi: 10.1007/978-981-16-7985-8_127.

[3] Munir, R. (2019). Kriptografi (Edisi ke-2). Informatika

[4] R. Munir, "Stream Cipher," materi kuliah IF4020 Kriptografi, Program Studi Teknik Informatika, Institut Teknologi Bandung, Bandung, Indonesia, 2025. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2025-2026/13-Stream-Cipher-2025.pdf.

[5] R. Munir, "Block Cipher (Bagian 1)," materi kuliah IF4020 Kriptografi, Program Studi Teknik Informatika, Institut Teknologi Bandung, Bandung, Indonesia, 2025. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2025-2026/14-Block-Cipher-Bagian1-2025.pdf.

[6] C. Guo, Z. Song, Y. Ping, G. Shen, Y. Cui, and C. Jiang, "PRATD: A Phased Remote Access Trojan Detection Method with Double-Sided Features," Electronics, vol. 9, no. 11, p. 1894, Nov. 2020, doi: 10.3390/electronics9111894.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Desember 2025

Indraswara Galih Jayanegara
13522119