

[QuickNote] Decrypting the C2 configuration of Warzone RAT

 kienmanowar.wordpress.com/2023/03/25/quicknote-decrypting-the-c2-configuration-of-warzone-rat/

March 25, 2023

1. Introduction

Warzone RAT is a type of malware that is capable of infiltrating a victim's computer and giving attackers remote access and control over the system. The malware has gained notoriety for its advanced capabilities and ability to evade detection, making it a serious threat to computer security.

Warzone RAT is typically spread through phishing emails or other social engineering techniques, where attackers trick victims into downloading and installing the malware on their systems. Once the malware is installed, it can perform a variety of malicious actions, including stealing passwords, taking screenshots, and logging keystrokes. It can also download and execute additional malware, giving attackers even more control over the victim's system.

One of the key features of Warzone RAT is its ability to encrypt its configuration data, making it difficult for security experts to analyze and understand how the malware operates.

Currently, there are two variants of the malware in circulation, each using a different method to decode its configuration. The first variant uses standard RC4 encryption, while the second variant uses a modified version of RC4. This modification makes it even more challenging to decrypt and analyze the malware's configuration data.

2. Analysis

Sample1: 00930cccd81e184577b1ffeebf08ee6a32dd0ef416435f551c64d2bcb61d46cf (use standard RC4)

Malware Config (WarzoneRat)

Malware WarzoneRat

Samples [00930cccd81e184577b1fffebf08ee6a32dd0ef416435f551c64d2bcb61d46cf](#)

Settings

- Install Flag**
False
- Startup Flag**
False
- Proxy Port**
5000
- Builder Id**
HV9ZQDGSAAH

C2s onyem.duckdns.org:5353

Sample2: [61f8bf26e80b6d6a7126d6732b072223dfc94203bb7ae07f493aad93de5fa342](#) (use modified RC4)

61f8bf26e80b6d6a7126d6732b072223dfc94203bb7ae07f493aad93de5fa342 UnpacMe: Warzone Download

x32 exe 132 KB 21/07/2022 Malpedia: win_ave_maria_g0

File Hashes	
sha256	61f8bf26e80b6d6a7126d6732b072223dfc94203bb7ae07f493aad93de5fa342
md5	ed11094ac348124b4870f917cadcbcc1
sha1	2c0e8e750aff3d2c1eae3dec1b53f85869673f58

Metadata	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows
Compile Time	Thu Jul 21 07:34:06 2022 UTC
File Size	132 KB (135168 bytes)
Linker Version	14.31 - (1931 (Visual Studio 2019 version 17.1))
Characteristics	IMAGE_FILE_EXECUTABLE_IMAGE IMAGE_FILE_32BIT_MACHINE
Compressed	false
Entry Point	0x6da4
Image Base	0x400000
EP Bytes	558bec83ec4856ff159c9041008365e4
Sections	6
Checksum	0
Signature	17744
Subsystem	IMAGE_SUBSYSTEM_WINDOWS_GUI

Resources	
+ WM_DSP	

Rich Headers				
Prod Id	Product	Count	Build Id	Build
95	Utc1310_C	1	4035	7.1 2003
263	<Unknown>	1	27412	<Unknown>
93	Implib710	2	4035	7.1 2003
260	<Unknown>	3	27412	<Unknown>

In Warzone RAT, the configuration info is stored in the `.bss` PE section of the malware's code. The `.bss` section is typically used for storing uninitialized data. The format of the configuration is as follows: `[Key length] [RC4 key] [Encrypted data]`. Below is an illustration of the configuration stored in the `.bss` section in both samples.

Name	Start	End	R	W	X	D	L	Align	Base
.text	00401000	00414000	R	.	X	.	L	para	0001
.idata	00414000	00414370	R	.	.	.	L	para	0005
.rdata	00414370	00419000	R	.	.	.	L	para	0002
.data	00419000	0054F000	R	W	.	.	L	para	0003
.bss	00553000	00554000	R	.	.	.	L	para	0004

sample 1

```

0054EFE0  ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??  ??????????????????
0054EFF0  ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??  ??????????????????
00553000  32 00 00 00 45 62 F5 D6 BC DF 74 22 F2 EB 09 09 2...EböÖ%ßt"ðë..
00553010  AA B6 CD 3A 3D A6 E8 CB 96 95 19 9C 23 1B 44 94 ¼¶í:=|èË-•.æ#.D"
00553020  6C 2C B0 87 50 9A 87 99 DF 32 0C 76 0D DD 02 EC 1,°‡Pš†™B2.v.Ý.ì
00553030  49 7D 81 F0 0E 40 5D D0 5F 26 64 1A 4D A8 69 FE I}.ð.@]D_&d.M"ip
00553040  BB 5C 12 C0 BE 7F 1F 5F C3 CB CA E9 3C 7A 4E 46 »\.À%.._ÄËÉé<zNF
00553050  1D 91 37 6F A3 15 A5 C2 E8 07 63 69 C9 7D 1C 80 .‘7o£.¥Äè.ciÉ}.€
00553060  A4 38 57 23 82 91 6C 15 2A D0 70 14 2D 19 10 C2 µ8W#,‘1.*ðp.-..Â
00553070  73 31 F8 BC AF F4 36 84 34 CE 82 FC 5C FD 96 89 s1è%ˆô6,,4Ï,ü|i-‰
00553080  EB E2 CA F6 D1 6F 84 D3 1C 2C A4 DE EA EA F4 00 ëâËöŃo,,Ó.,µðëêô.
00553090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
005530A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
005530B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
005530C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Name	Start	End	R	W	X	D	L	Align	Base
.text	00401000	00419000	R	.	X	.	L	para	0001
.idata	00419000	00419380	R	.	.	.	L	para	0005
.rdata	00419380	0041E000	R	.	.	.	L	para	0002
.data	0041E000	00554000	R	W	.	.	L	para	0003
.bss	00559000	0055A000	R	.	.	.	L	para	0004

sample 2

```

00553FE0  ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??  ??????????????????
00553FF0  ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??  ??????????????????
00559000  32 00 00 00 4F 00 24 53 19 8A 90 48 48 8D CA 4E 2...O.$S.Š.HH.ËN
00559010  52 7C D8 06 9C 7D FA F1 FE 1F 0C 30 01 67 4A E4 R|ø.æ}úñþ..ø.gJä
00559020  28 CC C9 83 19 EB 2D 98 DD ED 4A 76 9A 22 8F 5C (ÏÉf.ë-~ÝíJvš".\
00559030  98 51 FE E1 FB DF 44 5C 0F B3 22 72 5B BB 14 46 ~QbáûB\..³"r[».F
00559040  CF FB 29 9F 26 11 F5 A5 AE 9B 3D 25 69 E7 C9 F8 Ìû)ÿ&.ð¥@)=%içÉø
00559050  A5 8B 53 EF 12 9A 1B 15 42 1C 85 D9 1A 4D 21 A8 ¥<Si.š..B...Û.M!""
00559060  81 C7 8B 29 6D 0D 5A BF 99 7E 15 4D 68 2B 87 EA .Ç<)m.Zç™~.Mh+‡ê
00559070  25 8F AB 96 3D 68 B2 17 ED D1 7F 17 43 BA 38 FD %.«-h².iŃ..Cº8ý
00559080  09 6A E7 DE BD 4F 58 41 99 F1 99 9A 8C F5 E7 00 .jçP%OXATMñ™šËEöç.
00559090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
005590A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
005590B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
005590C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

The steps to perform the process of retrieving information and copying data from the .bss section to memory are the same in both samples. The pseudo-code is shown below:

```

BOOL __usercall wzr_decrypt_config_stored_at_bss@<eax>(struct_this_20 *arg_struct_cfg@<ecx>, int arg_zero_value@<ebx>)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    Sleep(0x1F4u);
    wzr_init_struct_16(&struct_payload_info);
    wzr_payload_base_addr = wzr_get_base_addr_of_curr_pe_file();
    wzr_copy_pe_headers_and_sections_info(&struct_payload_info, wzr_payload_base_addr);
    var_str_bss = wzr_strcpy(&var_temp_data.cbData, arg_zero_value, ".bss");
    wzr_grab_info_of_bss_section(&struct_payload_info, &source_data, var_str_bss);
    wzr_VirtualFree(var_temp_data.cbData);
    wzr_clone_data(&var_temp_data, &source_data.section_address);
    wzr_clone_data2(&arg_struct_cfg->ptr_bss_content, &var_temp_data);
    wzr_free_heap_and_reset_pointer(&var_temp_data.pbData);
    wzr_decrypt_config(arg_struct_cfg, &v23);
}

```

Sample 1

```

BOOL __usercall wzr_decrypt_config_stored_at_bss@<eax>(wzr_struct_21 *arg_struct_cfg@<ecx>, int arg_zero_value@<ebx>)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    Sleep(0x1F4u);
    wzr_init_struct_16(&struct_payload_info);
    wzr_payload_base_addr = wzr_get_base_addr_of_curr_pe_file();
    wzr_copy_pe_headers_and_sections_info(&struct_payload_info, wzr_payload_base_addr);
    var_str_bss = wzr_strcpy(&var_temp_data.cbData, arg_zero_value, ".bss");
    wzr_grab_info_of_bss_section(&struct_payload_info, &foundSectionInfo, var_str_bss);
    wzr_VirtualFree(var_temp_data.cbData);
    wzr_clone_data(&var_temp_data, &foundSectionInfo.section_address);
    wzr_clone_data2(&arg_struct_cfg->ptr_bss_content, &var_temp_data);
    wzr_free_heap_and_reset_pointer(&var_temp_data.pbData);
    wzr_decrypt_config(arg_struct_cfg, &decrypted_config);
}

```

Sample 2

The pseudo code in function **wzr_decrypt_config** in both samples is the same, which involves extracting the RC4 Key and Encrypted data, and then using RC4 to decrypt the configuration. The difference lies in function **wzr_perform_rc4**.

```

int __thiscall wzr_decrypt_config(struct_this_20 *arg_struct_this_20, int decrypted_config)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    wzr_copy_from_offset(
        &arg_struct_this_20->ptr_bss_content.pbData,
        &var_encrypted_config,
        *arg_struct_this_20->ptr_bss_content.pbData + 4,
        arg_struct_this_20->ptr_bss_content.cbData - *arg_struct_this_20->ptr_bss_content.pbData - 4);
    var_encrypted_data.cbData = v3;
    var_encrypted_data.pbData = v3;
    wzr_clone_data(&var_encrypted_data, &var_encrypted_config);
    var_rc4_key.cbData = v4;
    var_rc4_key.pbData = v4;
    wzr_copy_data_0(arg_struct_this_20, &var_rc4_key);
    wzr_perform_rc4(decrypted_config, var_rc4_key.pbData, var_rc4_key.cbData, var_encrypted_data.pbData, var_encrypted_data.cbData);
    wzr_free_heap_and_reset_pointer(&var_encrypted_config.pbData);
    return decrypted_config;
}

```

Sample 1

```

wzr_data *__thiscall wzr_decrypt_config(wzr_struct_21 *this, wzr_data *decrypted_config)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    wzr_clone_data3(
        &this->ptr_bss_content.pbData,
        &var_encrypted_config,
        *this->ptr_bss_content.pbData + 4,
        this->ptr_bss_content.cbData - *this->ptr_bss_content.pbData - 4);
    var_encrypted_config_cp.cbData = v3;
    var_encrypted_config_cp.pbData = v3;
    wzr_clone_data(&var_encrypted_config_cp, &var_encrypted_config);
    var_rc4_key.cbData = v4;
    var_rc4_key.pbData = v4;
    wzr_clone_data4(this, &var_rc4_key);
    wzr_perform_rc4(decrypted_config, var_rc4_key.pbData, var_rc4_key.cbData, var_encrypted_config_cp.pbData, var_encrypted_config_cp.cbData);
    wzr_free_heap_and_reset_pointer(&var_encrypted_config.pbData);
    return decrypted_config;
}

```

Sample 2

The function **wzr_perform_rc4** in sample 1 uses standard RC4 to decrypt the configuration. Its pseudocode is shown below:

```

struct_data *_thiscall w3r_perform_rc4(struct_data *decrypted_config, _BYTE *rc4_key, int rc4_key_len, _BYTE *data, int data_len)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"" TO EXPAND]

    w3r_clone_data(&dest_data, &data);
    for ( i = 0; i < 256; ++i )
    {
        rc4_sbox[i] = i;
    }
    w3r_rc4_ksa(rc4_key, rc4_sbox, rc4_key_len);
    w3r_rc4_prng(data, rc4_sbox, data_len);
    w3r_clone_data(decrypted_config, &data);
    w3r_free_heap_and_reset_pointer(&dest_data.pbdata);
    w3r_free_heap_and_reset_pointer(&rc4_key);
    w3r_free_heap_and_reset_pointer(&data);
    return decrypted_config;
}

```

sample 1

```

char __usercall w3r_rc4_ksa@cal>(_BYTE *rc4_key@edx, _BYTE *rc4_sbox@ecx, int rc4_key_len)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"" TO EXPAND]

    j = 0;
    rc4_sbox_ = rc4_sbox;
    for ( i = 0; i < 256; ++i )
    {
        sbox_value = rc4_sbox[i];
        j = (sbox_value + rc4_key[i % rc4_key_len] + j) % 0x100;
        rc4_sbox_ = rc4_sbox_;
        result = rc4_sbox_[j];
        rc4_sbox_[i] = result;
        rc4_sbox_[j] = sbox_value;
    }
    return result;
}

```

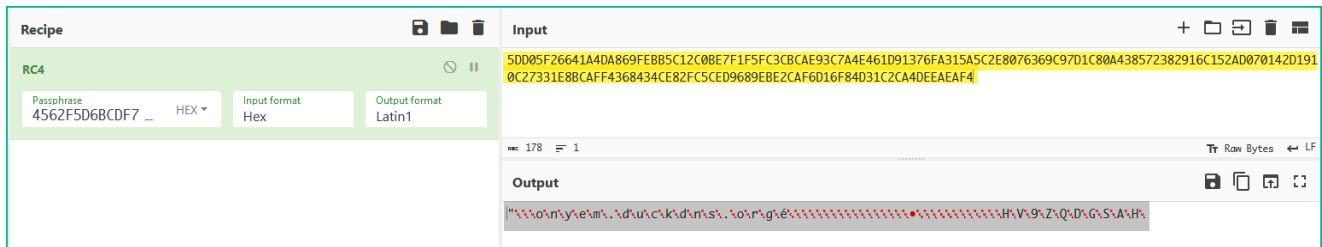
```

int __usercall w3r_rc4_prng@eax>(_BYTE *data@edx, _BYTE *rc4_sbox@ecx, int data_len)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"" TO EXPAND]

    i = 0;
    j = 0;
    while ( data_len > 0 )
    {
        i = (i + 1) % 0x100;
        sbox_value = rc4_sbox[i];
        j = (sbox_value + j) % 0x100;
        rc4_sbox[i] = rc4_sbox[j];
        rc4_sbox[j] = sbox_value;
        *data ^= rc4_sbox[(sbox_value + rc4_sbox[i])];
        --data_len;
        ++data;
    }
    return j;
}

```

Thus, we can easily use CyberChef to perform configuration decoding or write a Python script to automate for similar samples.



The pseudocode for function `w3r_perform_rc4` in sample 2 as shown below. Prior to decryption, it allocates an array of 250 bytes, filled with zero values. Then, it copies the extracted `rc4_key` into this array. Finally, it calls the `w3r_rc4_crypt` function, which uses the **modified RC4** algorithm to decrypt the configuration.

```

w3r_data *_thiscall w3r_perform_rc4(w3r_data *decrypted_config, _BYTE *rc4_key, int rc4_key_len, _BYTE *encrypted_data, int encrypted_data_len)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"" TO EXPAND]

    w3r_clone_data(&encrypted_data_cp, &encrypted_data);
    w3r_rc4_crypt_wrap1(rc4_key_len, rc4_key, encrypted_data, encrypted_data_len);
    w3r_clone_data(decrypted_config, &encrypted_data);
    w3r_free_heap_and_reset_pointer(&encrypted_data_cp.pbdata);
    w3r_free_heap_and_reset_pointer(&rc4_key);
    w3r_free_heap_and_reset_pointer(&encrypted_data);
    return decrypted_config;
}

```

Sample 2

```

_BYTE __usercall w3r_rc4_crypt_wrap1@eax>(int rc4_key_len@edx, _BYTE *rc4_key@ecx, _BYTE *encrypted_data, unsigned int encrypted_data_len)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"" TO EXPAND]

    rc4Sbox = LocalAlloc(PAGE_EXECUTE_READWRITE, 256u);
    w3r_memset_wrap(rc4_key_buffer_250bytes, 0, 250u);
    // Copy rc4 key to new buffer (buffer size = 250 bytes)
    w3r_memcpy(rc4_key_buffer_250bytes, rc4_key, rc4_key_len);
    rc4_data.rc4Sbox = rc4Sbox;
    rc4_data.rc4_key_250b = rc4_key_buffer_250bytes;
    rc4_data.data_length = encrypted_data_len;
    w3r_rc4_crypt(&rc4_data, encrypted_data);
    LocalFree(rc4Sbox);
    return encrypted_data;
}

```

The complete pseudocode of the `w3r_rc4_crypt` function is as follows:

```

void __thiscall wzr_rc4_crypt(wzr_rc4_data *rc4_info, _BYTE *data)
{
    idx = 0;
    if ( rc4_info->rc4Sbox )
    {
        if ( rc4_info->rc4_key_250b )
        {
            rc4_info->counter2 = 0;
            LOBYTE(i) = 0;
            rc4_info->counter1 = 0;

            do
            {
                rc4_info->rc4Sbox[i] = rc4_info->counter1;
                i = rc4_info->counter1 + 1;
                rc4_info->counter1 = i;
            }
            while ( i < 256 );

            rc4_info->counter1 = 0;
            for ( i = 0; i < 256; rc4_info->counter1 = i )
            {
                rc4Sbox = rc4_info->rc4Sbox;
                rc4_info->counter2 += rc4Sbox[i] + rc4_info->rc4_key_250b[i % 250];
                rc4Sbox[i] ^= rc4Sbox[rc4_info->counter2];
                // swap values
                rc4_info->rc4Sbox[LOBYTE(rc4_info->counter2)] ^= rc4_info-
>rc4Sbox[LOBYTE(rc4_info->counter1)];
                rc4_info->rc4Sbox[LOBYTE(rc4_info->counter1)] ^= rc4_info-
>rc4Sbox[LOBYTE(rc4_info->counter2)];
                i = rc4_info->counter1 + 1;
            }
            rc4_info->counter1 = 0;
            rc4_info->counter2 = 0;
            // Decrypt data
            if ( rc4_info->data_length )
            {
                j = 0;
                do
                {
                    rc4_info->counter1 = j + 1;
                    rc4Sbox = rc4_info->rc4Sbox;
                    k = (j + 1);
                    rc4Sbox_value1 = rc4Sbox[k];
                    rc4_info->counter2 += rc4Sbox_value1;
                    rc4Sbox_value1_ = rc4Sbox_value1;
                    rc4Sbox_value2 = rc4Sbox[rc4_info->counter2];
                    rc4Sbox[k] = rc4Sbox_value2;
                    rc4_info->rc4Sbox[LOBYTE(rc4_info->counter2)] = rc4Sbox_value1;
                    rc4Sbox_ = rc4_info->rc4Sbox;
                    data[idx] ^= rc4Sbox_[(rc4_info->counter2 + rc4Sbox_value2)] ^
(rc4Sbox_[(rc4Sbox_value2 + rc4Sbox_value1_)])

```

```

rc4Sbox_[(rc4Sbox_[((0x20 * rc4_info->counter2) ^ (rc4_info->counter1 >> 3))]
+
rc4Sbox_[((0x20 * rc4_info->counter1) ^ (rc4_info->counter2 >> 3))] ^ 0xAA)];
    j = ++rc4_info->counter1;
    ++idx;
}
while ( idx < rc4_info->data_length );
}
}
}
}
}

```

With the pseudocode above, we can rewrite the decoding code in Python as follows. This is the code I wrote, and you can write it in your own way as long as it performs the task correctly.

```

# Refs: https://stackoverflow.com/questions/9433541/movsx-in-python
def SIGNEXT(x, b):
    m = (1 << (b - 1))
    x = x & ((1 << b) - 1)
    return ((x ^ m) - m)

# This routine is responsible for decrypting the stored C2.
def rc4_customized_decryptor(data, key):
    idx = 0
    counter1 = 0
    counter2 = 0

    # Initialize RC4 S-box
    rc4Sbox = list(range(256))

    # Modify RC4 S-box
    for i in range(256):
        counter2 += (rc4Sbox[i] + key[i%250])
        counter2 = counter2 & 0x000000FF
        rc4Sbox[i] ^= rc4Sbox[counter2]
        rc4Sbox[counter2 & 0xFF] ^= rc4Sbox[counter1 & 0xFF]
        rc4Sbox[counter1 & 0xFF] ^= rc4Sbox[counter2 & 0xFF]
        counter1 = i+1

    # Decrypt data
    counter1 = 0
    counter2 = 0
    j = 0
    decrypted = []
    while(idx < len(data)):
        counter1 = j + 1
        k = (j+1)
        rc4Sbox_value1 = rc4Sbox[k]
        counter2 += (SIGNEXT(rc4Sbox_value1, 8) & 0xFFFFFFFF)
        rc4Sbox_value1_ = (SIGNEXT(rc4Sbox_value1, 8) & 0xFFFFFFFF)
        rc4Sbox_value2 = rc4Sbox[counter2 & 0x000000FF]
        rc4Sbox[k] = rc4Sbox_value2
        rc4Sbox[(counter2 & 0x000000FF)] = rc4Sbox_value1
        tmp1 = rc4Sbox[((0x20 * counter1) ^ (counter2 >> 3)) & 0x000000FF]
        tmp2 = rc4Sbox[((0x20 * counter2) ^ (counter1 >> 3)) & 0x000000FF]
        tmp3 = rc4Sbox[((tmp1 + tmp2) & 0x000000FF) ^ 0xAA]
        tmp4 = rc4Sbox[(rc4Sbox_value2 + rc4Sbox_value1_) & 0x000000FF]
        tmp5 = (tmp3 + tmp4) & 0x000000FF
        tmp6 = rc4Sbox[(counter2 + rc4Sbox_value2) & 0x000000FF]
        decrypted.append(data[idx] ^ (tmp5 ^ tmp6))

        counter1 += 1
        j = counter1
        idx += 1

    return bytes(decrypted)

```


Below are the results of using a Python script to extract the configuration of Warzone RAT from the samples used in the article.

```
λ python warzone_rat_decrypt_config_use_ordinal_rc4.py -i wzr_oalab1.bin
Extracted C2: onyem.duckdns.org:5353
Builder ID or Warzone Key: HV9ZQDGSAAH Sample 1
```

```
λ python warzone_rat_decrypt_config_use_custom_rc4.py -i wzr_oalab2.bin
Extracted C2: 81.161.229.75:5200
Builder ID or Warzone Key: OWUZ370WDG Sample 2
```

3. End

The article would like to conclude here. I hope that it provides useful information for you during the process of analyzing the Warzone RAT malware. To protect against Warzone RAT and other types of malware, users should take precautions such as being cautious when opening email attachments, using strong passwords, and keeping their software up to date. It is also important to use antivirus software and to keep it updated regularly. By taking these steps, users can help to protect themselves against the threat of Warzone RAT and other types of malware.

4. Refs

https://research.openanalysis.net/warzone/malware/config/2021/05/31/warzone_rat_config.html

https://exploitreversing.files.wordpress.com/2022/11/mas_6-1.pdf