QakBot CCs prioritization and new record types

securityhomework.net/articles/qakbot_ccs_prioritization_and_new_record_types/qakbot_ccs_prioritization_and_new_record_types.php

Tweet Follow @sdkddk

October 30, 2022

This short article is about the QakBot new ability to prioritize its Command and Control servers and the new, for the moment unused, Command and Control server record types.

Summary

QakBot CC record structures

Legacy and modern CC record structures

There was a « legacy » QakBot text CC (Command and Control server) record structure, but for a while now QakBot CC record structure stored in the ciphered embedded configuration used to be the following :

```
typedef struct
  {
    BYTE record_type;            // Always binary '01'
    BYTE ip_v4_address[4];       // IP v4 address
    WORD port_in_big_endian;     // Port in big endian
  } QAKBOT_CC_ENTRY;

// Note : structure and members names are mines.
```

Here is for example the beginning of a deciphered QakBot `Obama217` campaign cc list (built on the October 26, 2022) :

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  01 C5 CC 35 F2 01 BB 01 69 6A 3C 95 01 BB 01 66  .ÅÌ5ò.».ij<•.».f
00000010  9F 6E 4F 03 E3 01 40 CF ED 76 01 BB 01 9C D8 86  ŸnO.ã.@Ïív.».œØ†
00000020  46 03 E3 01 B4 97 74 43 01 BB 01 BE C7 61 6C 03  F.ã.´—tC.».¾Çal.
00000030  E1 01 CE 01 CB 00 01 BB 01 BA BC 60 C5 01 BB 01  á.Î.Ë..».º¼`Å.».
00000040  CE 01 80 CB 01 BB 01 C9 F9 64 D0 03 E3 01 BE 4B  Î.€Ë.».ÉùdÐ.ã.¾K
00000050  97 42 08 AE 01 C6 02 33 F2 03 E1 01 5A A5 6D 04  —B.®.Æ.3ò.á.Z¥m.
00000060  08 AE 01 47 C7 A8 B9 01 BB 01 B5 38 AB 03 03 E3  .®.GÇ¨¹.».µ8«..ã
00000070  01 2B F1 9F 94 01 BB 01 29 67 01 10 01 BB 01 18  .+ñŸ”.».)g...»..
00000080  CF 61 75 01 BB 01 69 9D 56 76 01 BB 01 C9 DF A9  Ïau.».i.Vv.».Éß©
00000090  EE 7D 64 01 2F 0E E5 04 01 BB 01 46 3C 8E D6 08  î}d./.å..».F<ŽÖ.
000000A0  AE 01 29 2F F9 B9 01 BB 01 8E B5 B7 2A 08 AE 01  ®.)/ù¹.».Žµ·*.®.
000000B0  29 3E A5 98 01 BB 01 29 61 CD 60 01 BB 01 29 61  )>¥˜.».)aÍ`.».)a
000000C0  0E 3C 01 BB 01 97 D5 B7 8D 03 E3 01 4B 54 EA 44  .<.».—Õ·..ã.KTêD
000000D0  01 BB 01 BA 12 D2 10 01 BB 01 29 60 CC C4 01 BB  .».º.Ò..».)`ÌÄ.»
```

This is from sample `81d5fd23e26eca131eaf4748c44a6485f84827f7448e9833b3c8b8fd975fe5db` which can be downloaded on tria.ge.

The first cc structure `01 C5 CC 35 F2 01 BB` can be decoded to :

- `01` always 1;
- `C5 CC 35 F2` ip = 197.204.53.242 ;
- `01 BB` port = 443

New QakBot CC record structure

Starting with `Obama218` campaign, the CC list seems to have evolved to a different structure.

Looking to a deciphered `Obama218` cc list (built on the October 27, 2022), we can see that the cc struct is now 8 bytes long (instead of 7). The extra byte seems to be some sort of boolean which values can be `0` or `1` :

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  01 1B 6E 86 CA 03 E3 01 01 9C DC 2F 43 03 E1 00  ..n†Ê.ã..œÜ/C.á.
00000010  01 8E 73 54 58 08 AE 00 01 9C D8 86 46 03 E3 01  .ŽsTX.®..œØ†F.ã.
00000020  01 3A F7 73 7E 03 E3 01 01 18 09 DC A7 01 BB 01  .:÷s~.ã....Ü§.».
00000030  01 18 74 2D 79 01 BB 01 01 BA BC 50 86 01 BB 01  ..t-y.»..º¼P†.».
00000040  01 BE C7 65 25 08 AE 00 01 18 CE 1B 27 01 BB 01  .¾Çe%.®...Î.'.».
00000050  01 B5 A4 C2 E4 01 BB 01 01 69 60 C6 58 01 BB 00  .µ¤Âä.»..i`ÆX.».
00000060  01 70 8D B8 F6 03 E3 00 01 40 CF ED 76 01 BB 01  .p.¸ö.ã..@Ïív.».
00000070  01 76 C8 53 E2 01 BB 00 01 95 7E 9F E0 01 BB 01  .vÈSâ.»..•~Ÿà.».
00000080  01 B5 76 B7 7C 01 BB 00 01 90 CA 0F 3A 01 BB 01  .µv·|.»...Ê.:.».
00000090  01 AC 75 8B 8E 03 E3 01 01 C8 E9 6C 99 03 E3 01  .¬u‹Ž.ã..Èél™.ã.
000000A0  01 6D 88 AE C8 03 E3 00 01 C1 03 13 89 01 BB 01  .mˆ®È.ã..Á..‰.».
000000B0  01 C9 44 D1 2F 7D 65 01 01 2D 30 24 E2 08 27 00  .ÉDÑ/}e..-0$â.'.
000000C0  01 2D 23 61 2D 01 BB 00 01 A7 3A FE 55 01 BB 01  .-#a-.»..§:þU.».
000000D0  01 29 60 66 72 01 BB 00 01 29 C8 75 52 01 BB 00  .)`fr.»..)ÈuR.».
```

This is from sample `0dcd86692d92c058625ab343e472eef571514cc62d676288044ad26caa263fad` which packed samples like `9bea9743ed86d925f88d75077ef37b3a4a6a652bbdd2f0e516efdfbb94fb5e06` or `cb5b8365be065ab9870b15a524decf7474575b0b14e796ee77d6f482dfb6d53c` can be found on https://tria.ge/221027-ttm52acgb4 or https://tria.ge/221031-jg24baadb3.

`BB04` fresh samples show the same new CC record structure. See for example `b6e629128e9316820cfd5bdfe4d621d5a7435717879d554567df31352fb8558e` on tria.ge.

Beside the functionnal evolution, another objective pursued by the QakBot authors is achieved : at the time these lines are written, the configurations of these new samples are not or incorrectly extracted.

The new QakBot CC type 1 record structure looks like this :

```
typedef struct
  {
    BYTE record_type;              // Always binary '01'
    BYTE ip_v4_address[4];         // IP v4 address
    WORD port_in_big_endian;       // Port in big endian
    BYTE field_7;                  // 0 = priority CC ?
  } QAKBOT_NEW_TYPE1_CC_ENTRY;

// Note : structure and members names are mines.
```

As we will see below, the new field `field_7` seems to be a priority level assigned to each CC. And there are also two new record types, just skipped for now.

Looking under the hood

QakBot « explode_cc_list » function

The function which « explodes » cc list has evolved in order to interpret the extra byte `field_7` for type 1 records and to be able to manage three different record types. Here is what the function does :

1. it parses all the records in order to count type 1 ones. It counts the type 1 with the new field with a 0 value and the type 1 with the new field with a different value. The new types 2 and 3 are just skipped, respectively for 0x18 and 0x14 bytes long ;
2. it allocates two buffers to hold the type 1 with new field=0 and type1 with new field != 0 ;
3. it fills these two tables with memory cc structures of according CC ;
4. it randomizes the two tables to change the order of the CC in each of the two tables ;
5. it returns pointers to the two tables and the number of CC in each table.

The memory structure of CC records stored in the two CC tables is as follows :

```
typedef struct
  {
    DWORD type;                                     // 1 for IP v4 CC
    BYTE  ip_v4_address[4];                         // ip v4 address
    DWORD port;                                     // port in little endian
    DWORD unknown1;
    DWORD unknown2;
    DWORD unknown3;
    DWORD unknown4;
    DWORD unknown5;
    BOOL  field_7;                                  // priority ?
  } QAKBOT_MEMORY_CC_ENTRY;

// Note : structure and members names are mines.
```

QakBot « build_cc_list » function

The function in charge of building CC list has also evolved. Here is what it does now :

1. allocates a structure in order to store the CC lists which are about to be built ;
2. tries to get the registry saved « supernodes » list from configuration variable `CONFVAR_SUPERNODES` (id=0x39) ;
3. if supernodes have been retrieved, call the explode_cc_list() function to build two lists of CC ;
4. load and decipher the configuration resource from the PE file which contains the embedded static CC list, then call the explode_cc_list() function to build two other lists of CC ;
5. removes duplicates CC from static configuration tables ;
6. return a pointer to the structure holding the four CC lists.

The structure returned is this one :

```
typedef struct
  {
    QAKBOT_MEMORY_CC_ENTRY *lp_cc_table_field7_0_from_registry;        // Table of priority CC from supernodes list saved in
registry
    DWORD                   nb_cc_field7_0_from_registry;              // Number of CC in the table
    QAKBOT_MEMORY_CC_ENTRY *lp_cc_table_field7_not_0_from_registry;    // Table of non priority CC from supernodes list saved
in registry
    DWORD                   nb_cc_field7_not_0_from_registry;          // Number of CC in the table
    QAKBOT_MEMORY_CC_ENTRY *lp_cc_table_field7_0_from_configuration;   // Table of priority CC from embedded configuration
stored in resource
    DWORD                   nb_cc_field7_0_from_registry;              // Number of CC in the table
    QAKBOT_MEMORY_CC_ENTRY *lp_cc_table_field7_not_0_from_registry;    // Table of non priority CC from embedded configuration
stored in resource
    DWORD                   nb_cc_field7_not_0_from_registry;          // Number of CC in the table
  } QAKBOT_CC_LISTS;

// Note : structure and members names are mines.
```

QakBot CC prioritization

When looking for a CC, QakBot will now look in the four tables in this order :

1. table of priority CC from supernodes ;
2. table of priority CC from configuration ;
3. table of non priority CC from supernodes ;
4. table of non priority CC from configuration.

So, QakBot now manages to prioritize his CC.

New QakBot CC record types

Aside from prioritization, another evolution is the introduction of two new record types in the CC configuration list. The old type still has id=1 and there are now types 2 and 3.

The three QakBot cc record types

QakBot CC record types are now :

ID	Length	Content
1	8	IP v4, port and priority cc informations

ID	Length	Content
2	24	?
3	20	IP v6 record ?

And the new QakBot CC record structures are :

```
  typedef struct
    {
      BYTE ip_v4_address[4];        // IP v4 address
      WORD port_in_big_endian;      // Port in big endian
      BYTE priority;                // 0 = priority CC
    } QAKBOT_NEW_TYPE1_CC_RECORD;

  typedef struct
    {
      BYTE unknown[0x18];
    } QAKBOT_TYPE2_CC_RECORD;

  typedef struct
    {
      BYTE unknown[0x14];
    } QAKBOT_TYPE3_CC_RECORD;

  typedef union
    {
      QAKBOT_NEW_TYPE1_CC_RECORD ipv4_record;
      QAKBOT_TYPE2_CC_RECORD     type2_record;
      QAKBOT_TYPE3_CC_RECORD     type3_record;
    } QAKBOT_CC_RECORDS;

  typedef struct
    {
      BYTE              record_type;     // 1 = IPv4 CC, 2 = ?, 3 = IPv6 CC ?
      QAKBOT_CC_RECORDS record;
    } QAKBOT_NEW_CC_ENTRY;

// Note : structure and members names are mines.
```

Is QakBot on the way to IPv6 ?

When the configuration CC records are parsed in the explode_cc_list function, the records of types 2 and 3 are skipped. So, what can be the purpose of these new record types ?

The introduction of a prioritization capability for CC and these (for now) skipped new CC record types raises questions about wether QakBot author is preparing IPv6 support ?

The type 3 has a length of 20 bytes. It could be a future IPv6 record like the one below (warning, this is a conjecture) :

```
typedef struct
  {
    BYTE record_type;          //  1 byte (value = 3)
    WORD ip_v6_address[8];     // 16 bytes for IP v6 address
    WORD port_in_big_endian;   //  2 bytes
    BYTE priority;             //  1 byte
  } QAKBOT_CC_IPV6_ENTRY;      // 20 bytes total !

// Note : structure and members names are mines.
```

We have to wait for future QakBot samples to know what those new types really are.

In the past, when the CC record type was not the expected `01` value, the CC configuration parser skipped 17 bytes. So maybe we will never know what these new record types were for !

Some QakBot old and new configurations

Unpacked sample	QakBot version	Campaign ID	Timestamp	Details	Format

Unpacked sample	QakBot version	Campaign ID	Timestamp	Details	Format
1ee5c674b64eee2406d7dc3620874459eaae6403cb05960bc6c7f05d7a33a067	404.30	obama223	2022-11-18 07:42:25	json	new
322bf52085516358bd06bf6dcefef7cf93b281ac69b6c6c703477082c563f7f7	404.30	BB06	2022-11-18 06:25:05	json	new
83f217438eb62ae5617a51bfc73f7e078ae30c9d500950fd6ae544605597ebeb	404.30	BB06	2022-11-17 07:35:10	json	new
978d6ef0a1e6ec741405207fb53acf184490d26d7fd44b7f18473dbfaf8c0a3e	404.30	BB06	2022-11-16 14:57:52	json	new
ed175ae13525a1c4150ae0be57faebaea5b8d82e047991ec75907b2a278d9513	404.27	BB06	2022-11-15 06:05:08	json	new
6440ddf20cd76372a19f2eb71148a0548607ab41677a0a9b3dcee4bc8b7629d3	404.27	BB06	2022-11-14 09:41:56	json	new
83f5ae849f900a34a4d9e62c7c570728d934f8ae6f0b7887580359a3950e5caf	404.26	notset	2022-11-08 13:00:28	json	new
00ccb81a87edee4c73b6f6984ddc4ae72d3372858bd1ae1cf4f8824746efe888	404.20	BB05	2022-11-04 06:32:02	json	new
6b727c747e3b4fd742eeab3bf5b96e66b04168debcb8d01fd004f442b2c214f2	404.14	obama220	2022-11-02 07:21:10	json	new
6f155626d54b775502453131796e739a894c3e40d073568f07a78f95b5a39fa0	404.20	gld03	2022-11-01 16:51:54	json	new
a9e975631d60cbff28ae17b12cce3d02bb39d2133dde693037f45352702b521b	404.14	gld02	2022-10-31 15:10:39	json	new
b6343a93d7a5d1b1cf2bcc94bd096c3bc29c3bb9634f220f23f024545a9bd847	404.14	notset	2022-10-31 13:14:08	json	new
daa3557a9a632d9f897a8d7c1ef0e40a5715f0badc424f57f5ea50525fdd7122	404.14	BB05	2022-10-31 09:29:17	json	new
e0d35c06970ef8979600e362d6d619b6ef27217a130d4120f38abe1f66f67f12	404.14	obama219	2022-10-31 06:46:32	json	new
0dcd86692d92c058625ab343e472eef571514cc62d676288044ad26caa263fad	404.2	obama218	2022-10-27 11:41:26	json	new
ee1a401be2134b757ffabff69f7951cddc08c9e572d84271a6d8102e69b01b67	404.2	BB04	2022-10-27 09:46:15	json	new
fc64f51c1e1ff1c4ccd717ed7eb8298c70640864e3abba9f2ee1836fd8b1aa53	403.1051	BB04	2022-10-26 09:28:17	json	old