Tracking 15 Years of Qakbot Development | ThreatLabz

By Javier Vicente Vallejo

Published: 2024-01-31 · Archived: 2026-04-06 00:10:49 UTC

Network Communication

Qakbot has leveraged HTTP for C2 communication from the beginning. However, the network protocol on top of HTTP has changed significantly over the years with encryption, RSA signature verification, and the addition of a JSON-based message format.

Network protocol and encryption

Qakbot has continuously updated its message protocol with version 19 being the latest. The protocol specifies the format of the message. In version 3, Qakbot sent requests in a format similar to the following:

```
protoversion=9&r=1&n=kvtjmq970452&os=6.1.1.7601.1.0.0100&bg=b&it=3&qv=0300.288&ec=1453922906&av=0&salt=qrTMyfvp
```

However, this protocol format was later replaced with a JSON-based protocol with integer key values that denote specific fields as shown below:

```
{
  "8":1,
  "5":1035,
  "1":19, // protocol version
  "59":0,
  "3":"obama259",
  "4":1028,
  "10":1683022694,
  "2":"kqsvfc505763",
  "6":59661,
  "14":"Xgd1KxZQKTHGB6IxwtIy2e0RAq4iFNE6w6",
  "7":16759,
  "101":1,
  "26":"WORKGROUP",
  "73":0
}
```

This encoding adds a layer of obfuscation for each of the message fields.

Qakbot's network encryption has used RC4 with the key consisting of 16 random bytes concatenated with a hardcoded salt and hashed using SHA1. The most recent version of Qakbot now uses AES encryption with the key

consisting of 16 random bytes concatenated with a hardcoded salt and hashed using SHA256. After encryption, the data is Base64 encoded and prepended to a variable in the body of an HTTP POST request.

Domain generation algorithm

The first versions of Qakbot only used hardcoded C2s as shown in Figure 7.

Address	Length	Type	String
.data:0040F...	00000010	C	http://nt16.in/1
.data:0040F...	00000025	C	http://redserver.com.ua/cgi-bin/ss.pl
.data:0040F...	00000033	C	http://swallowthewhistle.com/cgi-bin/clientinfo3.pl
.data:0040F...	00000023	C	http://www.cdcdcdcdc2121cdsfdfd.com
.data:00410...	00000028	C	http://nt16.in/cgi-bin/jl/jloader.pl?r=q
.data:00410...	0000000B	C	http://%s%s
.data:00410...	00000025	C	http://nt16.in/cgi-bin/jl/jloader.pl?
.data:00411...	0000002A	C	http://nt002.cn/cgi-bin/jl/jloader.pl?r=3d
.data:00411...	0000002D	C	http://redserver.com.ua/cgi-bin/exhandler4.pl

Figure 7. Example of hardcoded Qakbot C2s

However, in version 2.0.1 a DGA was added as a backup C2 channel in the event that the hardcoded C2s were unreachable. Qakbot used a time-based DGA to generate up to 5,000 C2 domains for a specific date interval as shown in Figure 8.

Figure 8. Qakbot DGA code

Interestingly, some versions of Qakbot would generate fake domains if an analysis environment was detected in an effort to mislead researchers, as shown in Figure 9.

```
if ( DGA_get_date_from_legit_domains_http_response(date, 0x64) )
{
  date_seed = DGA_make_final_date_and_crc32(date);
  if ( ANTI_netmonitor_tools() > 0 )
    ++date_seed;
  DGA_make_seed(date_seed, dga_seed);
  v12 = 0;
  while ( 2 )
  {
    v11 = 5;
    v13 = DGA_gen_n_domains((int)dga_seed, 5u, 1);
```

```
int ANTI_netmonitor_tools()
{
  int v0; // esi
  int splitted_tools; // [esp+4h] [ebp-10h] BYREF
  int v3; // [esp+8h] [ebp-Ch] BYREF
  char *wpcap; // [esp+Ch] [ebp-8h] BYREF
  char *tools; // [esp+10h] [ebp-4h] BYREF

  tools = CORE_strings_decryptor(0x436u);       // .tcpdump.exe;windump.exe;ethereal.exe;
                                                // wireshark.exe;ettercap.exe;rtsniff.exe;
                                                // packetcapture.exe;capturenet.exe;
                                                // wireshark.exe
  splitted_tools = split(tools, ';', 0, &v3);
  wpcap = CORE_strings_decryptor(0x5CFu);       // wpcap.dll
  v0 = ((int (__cdecl *)(_DWORD, int *))UTIL_procfind_with_callback)(pcallback, &splitted_tools);
  UTIL_free_decstr(&wpcap);
  UTIL_free_list_of_ptrs(&splitted_tools, &v3);
  UTIL_free_decstr(&tools);
  return v0;
}
```

zscaler | ThreatLabz

Figure 9. Example of Qakbot generating fake domains if network monitoring tools were detected

Data exfiltration to compromised FTP servers

Qakbot versions 3.0.0 and earlier used compromised FTP servers to exfiltrate data rather than sending the data directly to their C2 server. The FTP credentials were stored in Qakbot's configuration files as shown below:

```
 22=:xxx@credsuser1.com::
23=:xxx@credsuser2.com::
24=:xxx@credsuser3.com::
25=:xxx@credsuser4.com::
26=
3=1581496845
```

This design had an inherent weakness since anyone with the FTP credentials could potentially have accessed and recovered the stolen information. To address this weakness, Qakbot was later updated to send the stolen data directly to Qakbot's C2 infrastructure.

Using compromised systems as relays

After version 3.2.4.8, Qakbot ceased using the DGA. Instead, Qakbot started using compromised systems themselves as C2 servers, and embedded a list of IP addresses and port numbers in the malware configuration.

Before version 4.0.3.2, the configuration file (stored as an encrypted resource) contained the list of IP addresses in a text-based format:

```
 45.45.105.94;0;443
86.107.20.14;0;443
99.228.5.106;0;443
184.191.62.24;0;995
47.153.115.154;0;995
206.169.163.147;0;995
96.35.170.82;0;2222
73.210.114.187;0;443
75.70.218.193;0;443
...
```

However, after version 4.0.3.2, the Qakbot C2 list evolved into a binary format as shown in Figure 10.

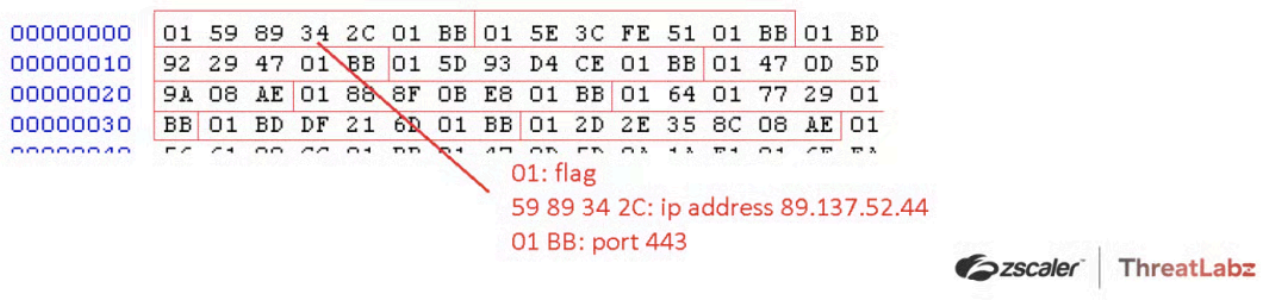

Figure 10. Qakbot C2 list binary format

Commands

In the first versions of Qakbot, the server sent commands in a descriptive text-based format. The following commands were supported in Qakbot versions 1.0 and 2.0:

- certssave
- ckkill
- cksave
- clearvars
- cron
- cronload
- cronsave
- forceexec
- ftpwork
- getip
- install3
- instwd
- kill

- killall
- loadconf
- nbscan
- psdump
- reload
- rm
- saveconf
- sleep
- socks
- sxordec
- sxorenc
- sysinfo
- thkill
- thkillall
- uninstall
- update
- update_finish
- uploaddata
- var
- wget

In order to obfuscate these commands, the Qakbot author replaced these string commands with integer values starting in the later builds of version 3.

Addition of RSA signature verification

Qakbot version 3.0.0.443 introduced RSA digital signatures (initially using the MatrixSSL library) to prevent tampering. This was especially important when the DGA and compromised systems were used as C2 servers.
