WINNTI GROUP: Insights From the Past

By Allison Ebel

Published: 2020-04-20 · Archived: 2026-04-05 19:37:15 UTC

Newly uncovered DNS tunnelling technique, and new campaign against South Korean gaming company

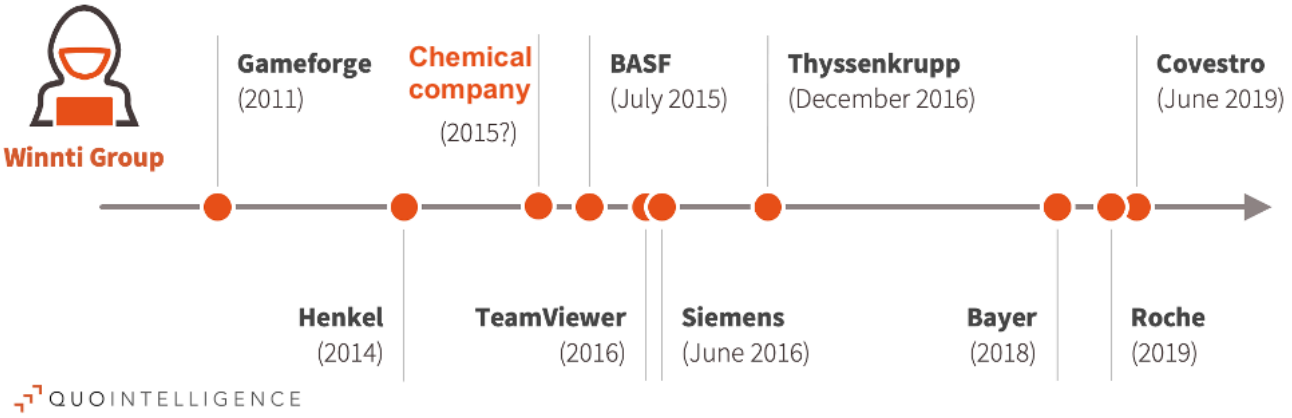

Figure 1: Timeline of attacks located in Germany and attributed to Winnti

HASHES	**Compilation Timestamp**	**Filename**
MD5: c893a12ff72698f09f89f778e4c9cd2b **SHA1**: 06256946a69409cd18859bfa429184a282374d76 **SHA256:** df6af36626d375c5e8aff45c64bfc1975d753b109e126a6cb30ee0523550329c	2015-08-06 01:52:51	TmPfwRVS.dll

Table 1 – Winnti sample

HASHES	**Compilation Timestamp**	**Filename**
MD5: c893a12ff72698f09f89f778e4c9cd2b **SHA1**: 06256946a69409cd18859bfa429184a282374d76	2015-08-06 01:52:51	TmPfwRVS.dll

SHA256: df6af36626d375c5e8aff45c64bfc1975d753b109e126a6cb30ee0523550329c		
MD5: cf140dc4cad9e8216545593a4c08c7be **SHA1**: 2b319b44451abb0596b9187e06f1fb7b4ace969d **SHA256**: bfa8948f72061eded548ef683830de068e438a6eaf2da44e0398a37ac3e26860	2015-08-05 12:08:00	driver1.sys
MD5: cc95391d75ce2443740f60114fe30ae9 **SHA1**: 30d1dd1dd4f0ace7a4f2c24e31fb6a0ee33e8a3a **SHA256**: 8ddc6dd9fc3640cd786dfbc72212cd001d9369817aa69e0a2fa25e29560badcf	2015-05-05 11:31:13	driver2.sys
MD5: b4e66b445b39d0368bbe4b91a3cd98ff **SHA1**: 2bc358ddc72f59ba0373b8635ab08ad747c12180 **SHA256**: 1865013aaca0f12679e35f06c4dad4e00d6372415ee8390b17b4f910fee1f7a2	2014-12-15 16:06:41	dsefix.exe
MD5: eaea9ccb40c82af8f3867cd0f4dd5e9d **SHA1**: 7c1b25518dee1e30b5a6eaa1ea8e4a3780c24d0c **SHA256**: cf3a7d4285d65bf8688215407bce1b51d7c6b22497f09021f0fce31cbeb78986	2008-May-31 02:18:53 UTC	vboxdrv.sys

Table 2 – Additional artifacts contained by bfa8948f72061eded548ef683830de068e438a6eaf2da44e0398a37ac3e26860

Analysis of dsefix.exe

This is essentially Windows x64 Driver Signature Enforcement Overrider (DSEFix), used to temporarily disable the driver signature enforcement on Windows systems by using an included old, legit VirtualBox driver, both signed and exploitable. By running dsefix.exe, the malware can bypass driver verification and install its own drivers. We identified the following two drivers which were embedded in the earlier described main artifact. To

note that this technique does not work on moden Windows (e.g. Windows 10) – yet another piece of evidence that this malware was designed and used multiple years ago.

Analysis of vboxdriver

This is the vulnerable, correctly signed with a digital certificate, VirtualBox driver that is used for exploitation. It is used regularly by various threat actors, and by the previously highlighted dsefix.exe. The driver can also be used to perform the Turla Driver Loader (TDL) exploitation technique, a similar technique as DSEFix.

Analysis of driver1.sys

In late 2019, ExaTrack released their analysis of a signed Winnti rootkit previously observed in the wild, which we confirm is essentially the same rootkit driver. The sample is capable of injecting raw packets into the network and receiving special formatted packets. In comparison, our variant has the same exact number of bytes, and there are large parts exactly matching.

Analysis of driver2.sys

This rootkit driver seems to be largely the same as driver1.sys with the same characteristics including structure, I/O control, and device strings. However, this driver supports different versions of Windows. It checks for ranges of Windows New Technology (NT) build numbers and returns early.

C2 DNS Tunneling

By analyzing the malware, it is possible to find two network indicators within its code:

```
dick.mooo.com
208.67.222.222
```

Figure 3 – observed network IoCs in Winnti sample

The hardcoded 208.67.222.222 is a legitimate OpenDNS DNS server (resolver1.opendns.com). This IP is pushed into a list that is generated by the malware at runtime. Likely, the initiation routine also populates the list with the system's DNS, and the OpenDNS server is only used as fallback case to ensure the C2 domain gets resolved.

The dick[.]mooo[.]com FQDN name is offered by FreeDNS, which is a free dynamic DNS service. Notably, in the last years, multiple researchers have reported Winnti/PlugX C2 hostnames hosted in the mooo[.]com zone. In the code we observed that a dot (.) is enforced before the FQDN. Additional analysis revealed that the malware generates subdomains with base128 encoding, and append them to FQDN.

```
dst_domain_offset = (char *)dst + strlen((const char *)dst) - 1;
if ( *dst_domain_offset != '.' )
  *++dst_domain_offset = '.';        // add dot if previous part does not end with it
strncpy(dst_domain_offset + 1, domain, strlen(domain) + 1);
```

Code 1 – Hostname length restriction

Further, dots are added into it every 57 characters potentially as a hostname length restriction, suggesting the expectation of long hostnames. We confirm the buffer can support FQDN's up to 2000 characters.

Upon further investigation we found out that the malware includes the open source iodine source code – software that enables the tunneling of IPv4 data through a DNS server. Interestingly, we are not aware of any earlier documentation highlighting Winnti specifically leveraging iodine for DNS tunneling. However, researchers at the Ruhr University Bochum, while hunting for DNS tunnels, observed APT32 and Wekby APT groups using NULL and TXT records as a C2 communication channel, as well as mentioning the mooo[.]com top level domain in their findings.

The implementation of iodine used in the Winnti sample is integrated and uses some custom wrapper, as evidence by the matching functions we discuss in further detail in the following section.

Use of Iodine for C2 DNS Tunneling

The iodine DNS tunneling solution is embedded in the DLL that is initially loaded and executed in memory, and includes at least the following 15 matching functions:

build_hostname	base32_handles_dots	base128_decode
inline_dotify	base64_decode	base128_encode
base32_decode	base64_encode	base128_reverse_init
base32_encode	base64_reverse_init	base128_blksize_enc
base32_reverse_init	base64_blksize_enc	base128_blksize_raw

Table 3 – Malware DLL Functions

For instance, the 64bit executable contains the build_hostname function (Code 2), which corresponds with the older 32bit version (compiled with debug symbols) of iodine 0.6.0 (Code 3):

```
__int64 __fastcall build_hostname(int *buf_500_0, unsigned __int64 a2_2000, __int64 a3, __int64 a4, const char
{
  unsigned __int64 v8; // r10
  __int64 v10; // rdi
  unsigned __int64 v12; // rsi
  _BYTE *v13; // r9
  unsigned __int64 v15; // [rsp+40h] [rbp+8h]

  v8 = a2_2000;
  if ( a7_2000 < a2_2000 )
    v8 = a7_2000;
  v10 = a6_unknown;
  v12 = a2_2000;
  v15 = v8 - strlen(dickmoo) - 8;
  if ( !(*(unsigned int (**)(void))(a6_unknown + 0x18))() )
    v15 -= v15 / 0x39;
  memset(buf_500_0, 0i64, v12);
  (*(void (__fastcall **)(int *, unsigned __int64 *, __int64, __int64))(v10 + 8))(buf_500_0, &v15, a3, a4);
  if ( !(*(unsigned int (**)(void))(v10 + 24))() )
    inline_dotify((const char *)buf_500_0, v12);
  v13 = (char *)buf_500_0 + strlen((const char *)buf_500_0) - 1;
  if ( *v13 != '.' )
    *++v13 = '.';
  c_strcpy((__int64)(v13 + 1), dickmoo, strlen(dickmoo) + 1);
  return (unsigned int)v15;
}
```

Code 2- iodine_0-6-0_build_hostname

```
int __cdecl build_hostname(char *buf, size_t buflen, const char *data,
{
  size_t v7; // ebx
  size_t v8; // eax
  size_t v9; // eax
  size_t space; // [esp+28h] [ebp-10h]
  int encsize; // [esp+2Ch] [ebp-Ch]
  char *b; // [esp+30h] [ebp-8h]

  v7 = buflen;
  if ( maxlen <= buflen )
    v7 = maxlen;
  space = v7 - strlen(topdomain) - 8;
  if ( !encoder->places_dots() )
    space -= space / 0x39;
  memset(buf, 0, buflen);
  encsize = encoder->encode(buf, &space, data, datalen);
  if ( !encoder->places_dots() )
    inline_dotify(buf, buflen);
  b = buf;
  v8 = strlen(buf);
  b += v8;
  if ( *--b != 46 )
    *++b = 46;
  ++b;
  v9 = strlen(topdomain);
  strncpy(b, topdomain, v9 + 1);
  return space;
}
```

Code 3- iodine_0-6-0_build_hostname

Based on the presence of the functions base128_blksize_enc and base128_blksize_raw, we determined the version used, while not exactly known, is from before May 2017 when a patch removed those functions. Further, comparative analysis indicates that for the implementation of iodine in this Winnti attack operation, there is no perfect match for the two versions having 64bit pre-compiled binaries. This indicates iodine was compiled from source, and it is reasonable it is being used as a library, and not in its normal distribution format of a standalone executable.

The DNS Tunneling technique adopted by the malware through the use of iodine is detailed in the figure below.

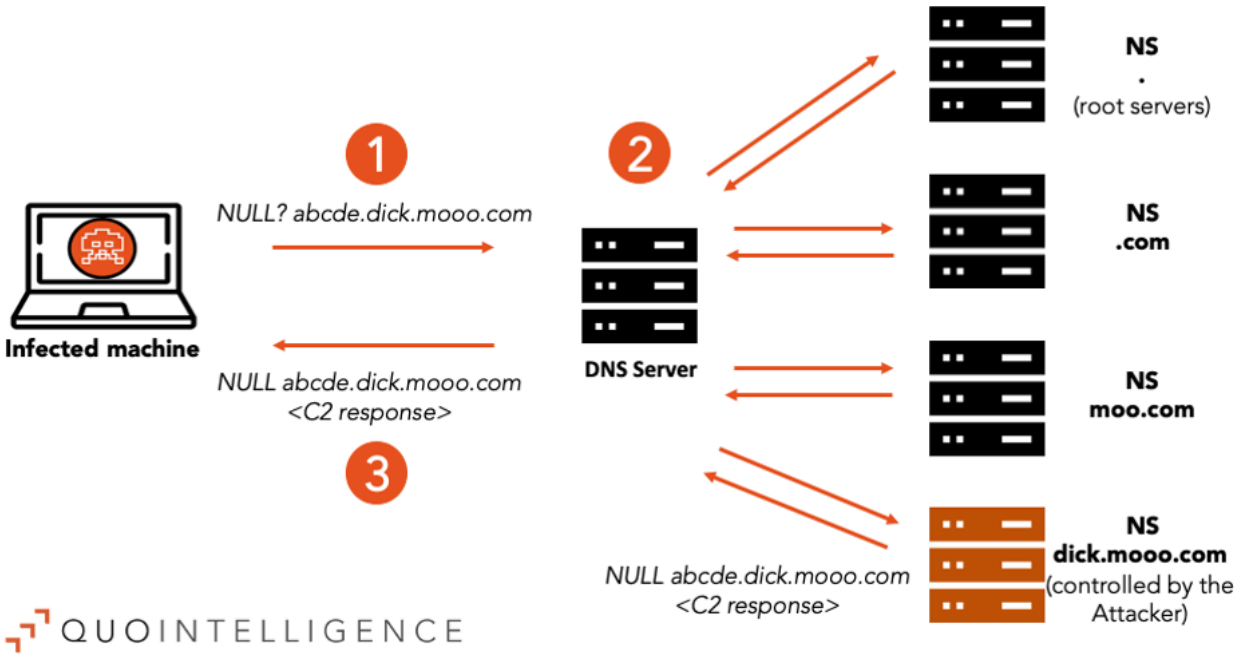

Figure 4 – C2 channel over DNS

The NULL DNS record type

The implementation of NULL type tunneling can be observed in the following excerpt taken while reversing the malware:

```
LOWORD(query[128]) = 0xA;
result = dns_encode((int *)a1, (unsigned int)v8, (__int64)query, 0, (const char *)v13, strlen((const char *)v13));
```

The third argument when calling dns_encode shall be of the type “struct query”
from Iodine’s [dns.c]

```
/* Only used when iodined gets an NS type query */
/* Mostly same as dns_encode_a_response() below */
int dns_encode_ns_response(char *buf, size_t buflen, struct query *q,
                           char *topdomain)
{
    HEADER *header;
    int len;
    short name;
    short topname;
    short nsname;
    char *ipp;
    int domain_len;
    char *p;

    if (buflen < sizeof(HEADER))
        return 0;

    memset(buf, 0, buflen);

    header = (HEADER*)buf;
```

According to common.h, "struct query" is defined as:

```
struct query {
    char name[QUERY_NAME_SIZE];
    unsigned short type;
    unsigned short rcode;
    unsigned short id;
    struct sockaddr_storage destination;
    socklen_t dest_len;
    struct sockaddr_storage from;
    socklen_t fromlen;
    unsigned short id2;
    struct sockaddr_storage from2;
    socklen_t fromlen2;
};
```

Since QUERY_NAME_SIZE equals to 512 (4 *128 – integers are 4 bytes) the query[128] call obtained from the reversing activity is indeed the DNS query type.

As noted, reversing activity detailed the query[128] value to be 0xa. From Iodine's windows.h T_NULL is DNS_TYPE_NULL

```
#define T_A DNS_TYPE_A
#define T_NS DNS_TYPE_NS
#define T_NULL DNS_TYPE_NULL
#define T_CNAME DNS_TYPE_CNAME
#define T_MX DNS_TYPE_MX
#define T_TXT DNS_TYPE_TXT
#define T_SRV DNS_TYPE_SRV
```

Finally, from Windows's documentation, DNS_TYPE_NULL is actually a Windows constant equal to 0xa

DNS Constants

5 minutes to read •

The following constants are defined for DNS in host byte order.

DNS Record Types

Constant	Value
DNS_TYPE_A	0x0001
DNS_TYPE_NS	0x0002
DNS_TYPE_MD	0x0003
DNS_TYPE_MF	0x0004
DNS_TYPE_CNAME	0x0005
DNS_TYPE_SOA	0x0006
DNS_TYPE_MB	0x0007
DNS_TYPE_MG	0x0008
DNS_TYPE_MR	0x0009
DNS_TYPE_NULL	0x000a
DNS_TYPE_WKS	0x000b

Winnti Signed Code With Digital Certificate from IQ Technology

During our analysis of the Winnti sample configured to target the German chemical company, our comparative analysis of other Winnti related drivers revealed a digital certificate issued to *IQ Technology*, a Taiwanese company producing Natural language processing (NLP) and Artificial Intelligence (AI) software. The rootkit driver aligns with the already known driver1.sys. While it is a known TTP that Winnti attributed attacks have involved stolen digital certificates for code signing its malware components, the use of this certificate is not publicly discussed, except for a brief report from a security researcher apparently associated with a Vietnamese security company. Although the report is no longer online, the discussed sample contains a compilation timestamp of August 2015, which is the earliest one observed in the wild that we identified using this digital certificate. At the time of analysis, the digital certificate was already revoked.

The sample's structure, debug symbols, and explanatory debug messages included within suggests it is highly likely be a development version. Additionally, the compilation timestamp indicates the sample was created 20 minutes prior to the *driver1.sys.* Both samples are highly related, and its contents combined with the date of analysis and reporting essentially solidify that it existed in 2015; however, this does not necessarily corroborate to an attack timeframe.

Sample Targeting South Korean Gaming Company

HASHES	**Compilation Timestamp**
MD5: 00961922e22e6a5d30b1d6fbd667d3c4 **SHA1**: 0fd54c26b593bd9e9218492d50d8873521c0ec0d **SHA256:** 4209b457f3b42dd2e1e119f2c9dd5b5fb1d063a77b49c7acbae89bbe4e284fb9	2016-03-07 09:44:01

Table 3 – observed network IoCs in Winnti sample

On 21 February, we detected the new submission of a 64-bit Winnti executable to a public online malware scanning service. As multiple researchers have reported, Winnti operators embed the name of their target directly into the malware, but in an obfuscated manner.

Binary Analysis

The sample resembles the Winnti Dropper Install.exe described by ESET, since it is a command line executable used to drop and load additional encrypted payload:

```
rundll32.exe "%s", Install %s
```

Unfortunately we were not able to find the payload meant to be decrypted by this dropper.

However, we were able to extract the malware's configuration file and identify the intended target. In this case, the following string was included within the extracted configuration:

```
0x1A0:  "GRAVITY"
```

Based on previous knowledge and targeting of the Winnti Group, we assess that this sample was likely used to target Gravity Co., Ltd., a South Korean video game company. The company is known for its Massive Multiplayer Online Role Playing Game (MMORPG) *Ragnarok Online*, which is also offered as a mobile application. As we have also reported in the past, the video game industry is one of the preferred targets of the Winnti Group, especially for those companies operating in South Korea and Taiwan. Interestingly, ESET researchers, while reporting on multiple Winnti Group campaigns targeting the video game industry, listed in their report a C2 server having a Campaign ID ***GRA*** *KR 0629*.

At this time, we do not have any further evidence supporting a potential link between the sample we analysed and the C2 detailed by ESET, but the coincidence between the C2 Campaign ID/Location with the Campaign ID we extracted from the Winnti dropper is worth noting.

Conclusion

The Winnti Group has exhibited their ability to breach different organizations and conduct sophisticated attack operations, typically motivated by espionage and financial gain, with various TTPs and malware toolkits. While attribution is not concrete due to the complexity of the group, there are links that can be drawn between operations which suggest the threat actors purporting the attacks are likely operating within the Winnti Group, or at least sharing resources.

The detection of this unreported Winnti variant uploaded to VirusTotal and targeting a German chemical company aligns with our prior observations and research from previous intelligence reporting highlighting Winnti Group's interest in German DAX companies. As a result, organizations of all sizes, but especially small to medium sized companies, including Germany's *hidden champions*, should prepare against such threats as they are vital to the economic ecosystem and continuous development of niche markets. Government oversight (local, regional, and across the EU) should ensure susceptible organizations such as these are following regulation and implementing security best practices to protect against future attacks.

Appendix

Indicators of Compromise

4209b457f3b42dd2e1e119f2c9dd5b5fb1d063a77b49c7acbae89bbe4e284fb9

cf3a7d4285d65bf8688215407bce1b51d7c6b22497f09021f0fce31cbeb78986

1865013aaca0f12679e35f06c4dad4e00d6372415ee8390b17b4f910fee1f7a2

8ddc6dd9fc3640cd786dfbc72212cd001d9369817aa69e0a2fa25e29560badcf

bfa8948f72061eded548ef683830de068e438a6eaf2da44e0398a37ac3e26860

df6af36626d375c5e8aff45c64bfc1975d753b109e126a6cb30ee0523550329c

4209b457f3b42dd2e1e119f2c9dd5b5fb1d063a77b49c7acbae89bbe4e284fb9

cf3a7d4285d65bf8688215407bce1b51d7c6b22497f09021f0fce31cbeb78986

1865013aaca0f12679e35f06c4dad4e00d6372415ee8390b17b4f910fee1f7a2

8ddc6dd9fc3640cd786dfbc72212cd001d9369817aa69e0a2fa25e29560badcf

bfa8948f72061eded548ef683830de068e438a6eaf2da44e0398a37ac3e26860

df6af36626d375c5e8aff45c64bfc1975d753b109e126a6cb30ee0523550329c

*.dick[.]mooo[.]com

208[.]67[.]222[.]222

45[.]248[.]85[.]200

MITRE ATT&CK

Tactic	Technique	Recommended Course of Action
Persistence / Privilege Escalation	T1215 Kernel Modules and Extensions T1068 Exploitation for Privilege Escalation	Anti-Virus software and Advanced End-Point solution can drastically reduce the risk of both techniques.
Defense Evasion	T1009 Binary Padding T1014 Rootkit T1116 Code Signing	Many of the samples analysed were signed with expired / revoked certificate. Enforce signature validation via Group Policies for executables
Exfiltration	T1022 Data Encrypted T1048 Exfiltration Over Alternative	– For DNS Tunnel, ensure that DNS logs are collected and reviewed. – DNS telemetic data should be also collected in order to spot frequent and heavy loaded DNS communication that are

	Protocol	definitely not related to an usual DNS query. – FreeDNS and DynDNS servers should be blacklisted/synkholed if not strictly required – DNS NULL type queries should not be avoided. Only the required record types should be allowed.
