Winnti: More than just Windows and Gates

Chronicle Follow
May 15 · 7 min read

The Winnti malware family was first reported in 2013 by Kaspersky Lab[1]. Since then, threat actors leveraging Winnti malware have victimized a diverse set of targets for varied motivations. While the name 'Winnti' in public reporting was previously used to signify a single actor, pronounced divergence in targeting and tradecraft between campaigns has led industry consensus to break up the tracking of the continued use of the Winnti malware under different actor clusters. The underlying hypothesis[2] is that the malware itself may be shared (or sold) across a small group of actors.

In April 2019, reports[3] emerged of an intrusion involving Winnti[4] malware at a German Pharmaceutical company. Following these reports, Chronicle researchers doubled down on efforts to try to unravel the various campaigns where Winnti was leveraged. Analysis of these larger convoluted clusters is ongoing. While reviewing a 2015 report[5] of a Winnti intrusion at a Vietnamese gaming company, we identified a small cluster of Winnti[6] samples designed specifically for Linux[7]. The following is a technical analysis of this variant.

Technical Analysis

The Linux version of Winnti is comprised of two files: a main backdoor (libxselinux) and a library (libxselinux.so) used to hide it's activity on an infected system.

As with other versions of Winnti, the core component of the malware doesn't natively provide the operators with distinct functionality[8]. This component is primarily designed to handle communications and the deployment of modules directly from the command-and-control servers. During our analysis, we were unable to recover any active plugins. However, prior reporting[9] suggests that the operators commonly deploy plugins for remote command execution, file exfiltration, and socks5 proxying on the infected host. We expect similar functionality to be leveraged via additional modules for Linux.

'libxselinux.so'—the userland rootkit

MD5	11a9f798227be8a53b06d7e8943f8d68
SHA1	906dc86cb466c1a22cf847dda27a434d04adf065
SHA256	4741c2884d1ca3a40dadd3f3f61cb95a59b11f99a0f980dbadc663b85eb77a2a
Size	23.94 KB (24512 bytes)
Filetype	ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, stripped
Filename	libxselinux.so.old
Config	**Control Servers** ip.1ds.me:53 ip.1ds.me:80

The library used to hide Winnti's system activity is a copy of the open-source userland rootkit Azazel[10], with minor changes. When executed, it will register symbols for multiple commonly used functions, including: open(), rmdir(), and unlink(), and modify their returns to hide the malware's operations. Below is a side-by-side comparison of the Azazel source code and the relevant function decompilation from 'libxselinux.so'.

```
FILE *fopen64 (const char *filename, const char *mode) {
        DEBUG("fopen hooked %s.\n", filename);
        if (is_owner())
                return syscall_list[SYS_FOPEN64].syscall_func(filename, mode);

        if (is_procnet(filename))
                return hide_ports(filename);

        if (is_invisible(filename)) {
                errno = ENOENT;
                return NULL;
        }

        return syscall_list[SYS_FOPEN64].syscall_func(filename, mode);
}
```

```
FILE *__fastcall fopen64(const char *filename, __int64 mode)
{
  if ( is_owner() )
    return (syscall_list.SYS_FOPEN64)(filename, mode);
  if ( is_procnet(filename) )
    return hide_ports(filename);
  if ( !is_invisible(filename) )
    return (syscall_list.SYS_FOPEN64)(filename, mode);
  *__errno_location() = ENOENT;
  return 0LL;
}
```

Figure 1: identidical fopen64() functions, Azazel shown first, libselinux.so shown second

Distinct changes to Azazel by the Winnti developers include the addition of a function named 'Decrypt2', which is used to decode an embedded configuration similar to the core implant. Unlike standard Azazel which is configured to hide network activity based on port

ranges, the Winnti-modified version keeps a list of process identifiers and network connections associated with the malware's activity. This modification likely serves to simplify the operator's sample configuration process by not having to denote specific ports to hide.

Strings within this sample associated with the malware's operations are encoded using a single-byte XOR encoding. The following is an example Python function to decode these strings.

```
def ConfigDecode(indata)
    for index, byte in enumerate(instr):
        decstr += chr( ord(byte) ^ 0xFE )
    return decstr
```

Figure 2: IDA Script to decode single-byte XOR strings

libxselinux

MD5	7f4764c6e6dabd262341fd23a9b105a3
SHA1	dc96d0f02151e702ef764bbc234d1e73d2811416
SHA256	ae9d6848f33644795a0cc3928a76ea194b99da3c10f802db22034d9f695a0c23
Size	407.12 KB (416888 bytes)
Filetype	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, from '8@%rdi 8@%rsi', stripped
Filename	libxselinux.old
Config	**Control Servers**: ip.1ds.me:53 ip.1ds.me:80 ip.1ds.me:8443 **Tags**: idea, 20161207

Winnti Linux variant's core functionality is within 'libxselinux'. Upon execution, an embedded configuration is decoded from the data section using a simple XOR cipher. An example Python function to decode this configuration is shown below:

```
def ConfigDecode(indata)
    key = instr[39]
    for index, byte in enumerate(instr):
        decstr += chr( ord(byte) ^ (ord(key) + index) & 0xFF)
    return decstr
```

Figure 3: Example python c

The decoded configuration is similar in structure to the version Kaspersky classifies as Winnti 2.0[11], as well as samples in the 2015

Novetta report[12]. Embedded in this sample's configuration three command-and-control server addresses and two additional strings we believe to be campaign designators. Winnti ver. 1, these values were designated as '*tag*' and '*group*'. A sample decoded configuration is shown below:

```
00000000: 6970 2e31 6473 2e6d 653a 3830 0000 0000  ip.1ds.me:80....
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000080: 6970 2e31 6473 2e6d 653a 3533 0000 0000  ip.1ds.me:53....
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 6970 2e31 6473 2e6d 653a 3834 3433 0000  ip.1ds.me:8443..
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000170: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000180: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000190: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000200: 3230 3136 3132 3037 0000 0000 0000 0000  20161207........
00000210: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000220: 6964 6561 0000 0000                      idea....
```

Figure 4: Decoded configuration

Winnti Linux samples identified so far fall under three distinct campaign designators:

```
20161207
idea
b08
```

Figure 5: Campaign designators Winnti Linux

For context, embedded Winnti campaign designators have ranged from target names, geographic areas, industry, and profanity.

0414	40329	c&c media	IDCA	OMG
0419	717	Default	jp	Test
0511w	722_64	DF	Kog	Tibet
0523_wh	apphelp_0707	eyaap80	lp80wi	TRIONWORLD
0713WM	apphelp_x86	Frogster	lpwi80	vn80
1	ARCHITEW32	Fucker	myth	VTC
2	ARCHITEW64	GameNet	mywi80	wasabii
3	asiasoft	Group32	nd	Wix64ap

Figure 6: Subset of designators from all Winnti versions

Interactions with control servers

Winnti malware handles outbound communications using multiple protocols including: ICMP, HTTP, as well as custom TCP and UDP protocols. Use of these protocols is thoroughly documented in the Novetta and Kaspersky reports. While the outbound communication mechanisms are well documented, less attention has been paid to a feature of recent versions of Winnti we came across in the Linux variant (as well as Windows) that allows the operators to initiate a connection directly to an infected host, without requiring a connection to a control server.

This secondary communication channel may be used by operators when access to the hard-coded control servers is disrupted. Additionally, the operators could leverage this feature when infecting internet-facing devices in a targeted organization to allow them to reenter a network if evicted from internal hosts. This passive implant approach to network persistence has been previously observed with threat actors like Project Sauron and the Lamberts.

Initial technical information about this feature was shared by the Thyssenkrupp CERT in the form of an Nmap[13] script[14] that could be used to identify Winnti infections through network scanning. This script identifies infected hosts by first sending a custom *hello* packet, immediately followed by an encoded request for host information, and then parsing the response. The workflow of the script is diagrammed below:

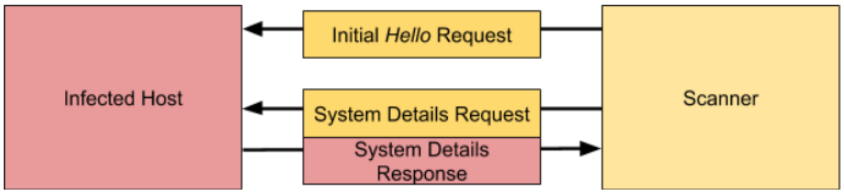

Figure 7: Network traffic breakdown

The initial request, referred to as the *helo/hello* request in the Nmap script, is comprised of four DWORDs. The first three are generated by rand() and the fourth is computed based on the first and third. When received by a Winnti-infected host, it will validate the received packet and listen for a second inbound request containing tasking. A breakdown of this traffic is shown below.

```
Initial Hello packet
00000000:  f00a aff1 2c0a bc8d 9833 5de8 f219 6839  ....,....3]...h9

Encoded Get System Information Request
00000010:  b01c 03d4 9038 41d4 2ab4 807f 9c61 32dd  .....8A.*....a2.
00000020:  913c 03d4 9038 42dc d8db 9e36 f30e cca4  .<...8B....6....
00000030:  3a38 41d4 9038 6aff 9038 41d4 b838 41d4  :8A..8j..8A..8A.
00000040:  9038 41d4 9038 41d4 9038 41d4 9139 41d4  .8A..8A..8A..9A.
00000050:  9038 57d4 9038 41d4 9038 41d4 9038 41d4  .8W..8A..8A..8A.
00000060:  9038 41d4 9038 45d4 9038 41d4 9038 41d4  .8A..8E..8A..8A.
00000070:  0400                                     ..
----

Decoded Get System Information Request
00000000: 2024 4200 0000 0000 ba8c c1ab 0c59 7309    $B.........Ys.
00000010: 0104 4200 0000 0308 48e3 dfe2 6336 8d70   ..B.....H...c6.p
00000020: aa00 0000 0000 2b2b 0000 0000 2800 0000   ......++....(...
00000030: 0000 0000 0000 0000 0000 0000 0101 0000   ................
00000040: 0000 1600 0000 0000 0000 0000 0000 0000   ................
00000050: 0000 0000 0000 0400 0000 0000 0000 0000   ................
00000060: 0400                                      .8
```

Figure 8: Network traffic breakdown

This second request (*Encoded Get System Information Request*) is encoded using the same method as the custom TCP protocol used for communication with command-and-control servers, which uses a four-byte XOR encoding. Before acting on the request, Winnti will validate the third DWORD contains the magic value 0xABC18CBA before executing tasking.

While it may be possible to conduct broad scanning to identify infected systems, the results would likely only be the subset that are directly Internet accessible.

Conclusion

Clusters of Winnti-related activity have become a complex topic in threat intelligence circles, with activity vaguely attributed to different codenamed threat actors. The threat actors utilizing this toolset have repeatedly demonstrated their expertise in compromising Windows-based environments. An expansion into Linux tooling indicates iteration outside of their *traditional* comfort zone. This may indicate the OS requirements of their intended targets but it may also be an attempt to take advantage of a security telemitry blindspot in many enterprises, as is with Penquin Turla and APT28's Linux XAgent variant. Utilizing a passive listener as a communications channel is characteristic of the Winnti developers' foresight in needing a failsafe secondary command-and-control mechanisms. Chronicle researchers maintain an active interest in clusters of Winnti activity and our research is ongoing.

Additional Indicators

MD5	b45f5a1548e213699a2802f8b99da08b
SHA1	0ee2e1ef7bb5092d9f28e4a6ea2fa8fa9b7d39bc
SHA256	da6ad48a2b680d6c3764f450380693d69cdc303025339c057b58c1edfd4dc548
Size	407.18 KB (416952 bytes)
Filetype	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, from '8@%rdi 8@%rsi', stripped
Filename	libxselinux
Config	**Control Servers**: 103.224.81.48:9747 **Tags**: b08

MD5	2a9f5d3fb47838937d282c552865863f
SHA1	b19c557986850e840961eb2d6d984ed64c9a60d4
SHA256	b80d57acd405d2ff58b1637b4e5dea412414297bfb4cde4b050413a77ffd6901
Size	407.12 KB (416888 bytes)
Filetype	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, from '8@%rdi 8@%rsi', stripped
Filename	libxselinux
Config	**Control Servers**: bbcnews.deepseaengine.com:53 bbcnews.deepseaengine.com:8040 **Tags**: b08

YARA

(Click here for source rule text and additional IoCs)

```
rule WinntiLinux_Dropper : azazel_fork
{
    meta:
        desc = "Detection of Linux variant of Winnti"
        author = "Silas Cutler (havex [@] chronicle.security), Chronicle
Security"
        version = "1.0"
        date = "2019-05-15"
        TLP = "White"
        sha256 =
"4741c2884d1ca3a40dadd3f3f61cb95a59b11f99a0f980dbadc663b85eb77a2a"
    strings:
        $config_decr = { 48 89 45 F0 C7 45 EC 08 01 00 00 C7 45 FC 28 00 00
00 EB 31 8B 45 FC 48 63 D0 48 8B 45 F0 48 01 C2 8B 45 FC 48 63 C8 48 8B 45
F0 48 01 C8 0F B6 00 89 C1 8B 45 F8 89 C6 8B 45 FC 01 F0 31 C8 88 02 83 45
FC 01 }
        $export1 = "our_sockets"
        $export2 = "get_our_pids"
    condition:
        uint16(0) == 0x457f and all of them
}

rule WinntiLinux_Main
{
    meta:
        desc = "Detection of Linux variant of Winnti"
        author = "Silas Cutler (havex [@] chronicle.security), Chronicle
Security"
        version = "1.0"
        date = "2019-05-15"
        TLP = "White"
        sha256 =
"ae9d6848f33644795a0cc3928a76ea194b99da3c10f802db22034d9f695a0c23"
    strings:
        $uuid_lookup = "/usr/sbin/dmidecode  | grep -i 'UUID' |cut -d' ' -f2
2>/dev/null"
        $dbg_msg = "[advNetSrv] can not create a PF_INET socket"
        $rtti_name1 = "CNetBase"
        $rtti_name2 = "CMyEngineNetEvent"
        $rtti_name3 = "CBufferCache"
        $rtti_name4 = "CSocks5Base"
        $rtti_name5 = "CDataEngine"
        $rtti_name6 = "CSocks5Mgr"
        $rtti_name7 = "CRemoteMsg"

    condition:
        uint16(0) == 0x457f and ( ($dbg_msg and 1 of ($rtti*)) or (5 of
($rtti*)) or ($uuid_lookup and 2 of ($rtti*)) )
}
```

. . .

[1] However, Kaspersky researchers credited HBGary for prior analysis in 2010

[2]https://401trg.com/burning-umbrella/

[3]https://www.scmagazine.com/home/security-news/malware/pharma-firm-bayer-hit-with-winnti-malware/

[4]https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/20134508/winnti-more-than-just-a-game-130410.pdf

[5]https://blog.vsec.com.vn/apt/initial-winnti-analysis-against-vietnam-game-company.html

[6]As in the malware, not a reference to an actor group

[7]PwC researchers were aware of the Winnti Linux variant, as revealed in Kris McConkey's SAS 2019 talk, "Skeletons in the supply chain".

[8]https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/20134508/winnti-more-than-just-a-game-130410.pdf page 21

[9]https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/20134508/winnti-more-than-just-a-game-130410.pdf page 21

[10]https://github.com/chokepoint/azazel

[11] In the report *Winnti: More than just a game*

[12] https://www.novetta.com/wp-content/uploads/2015/04/novetta_winntianalysis.pdf

[13] https://nmap.org/

[14] https://github.com/TKCERT/winnti-nmap-script