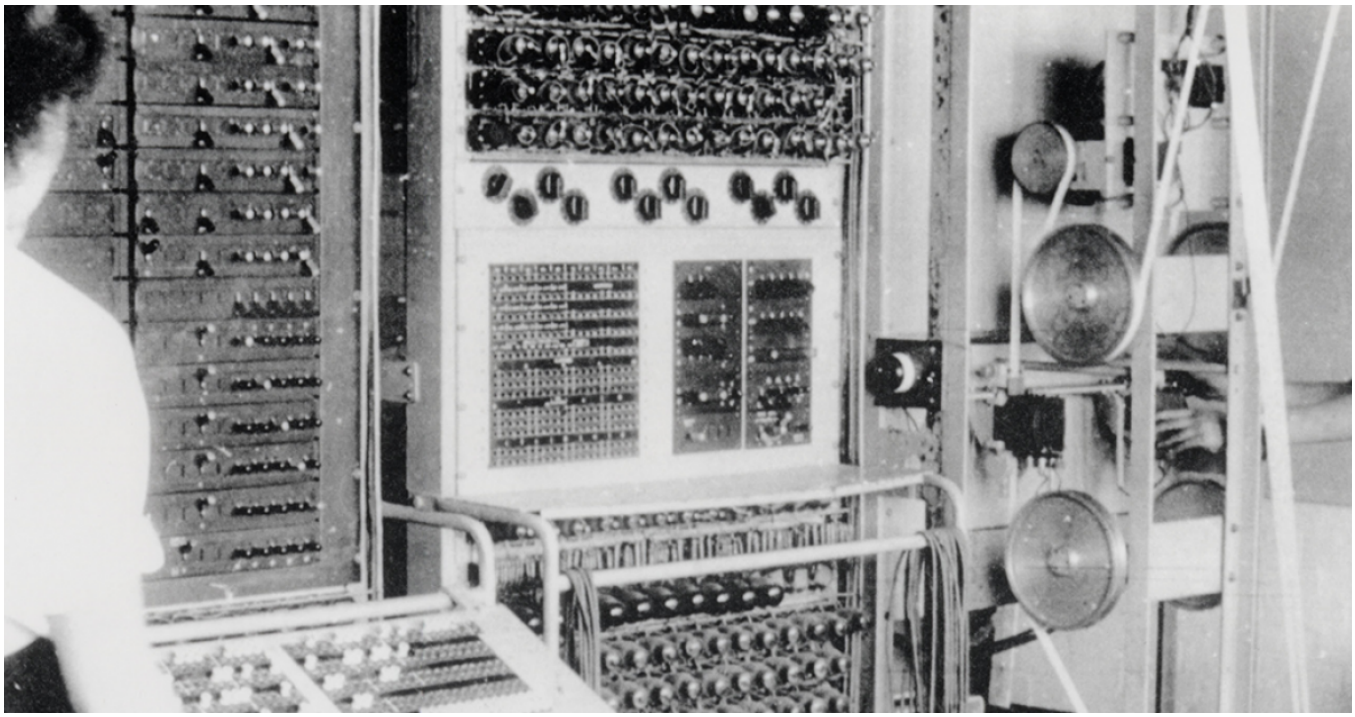
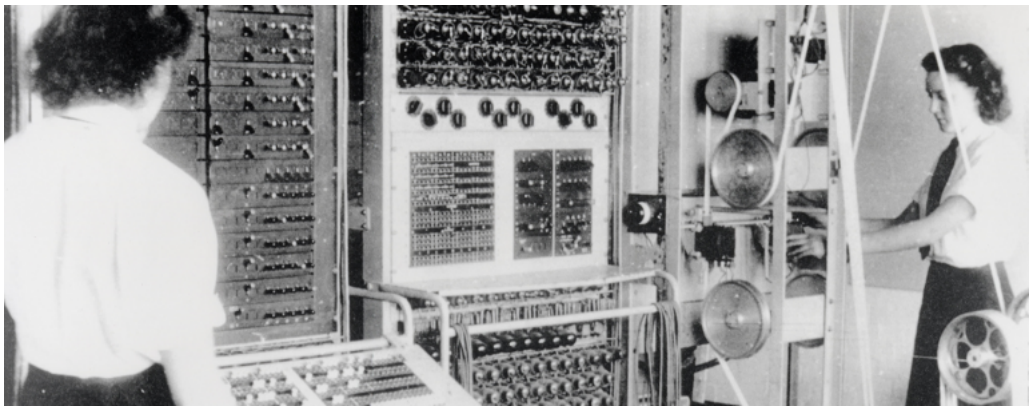
Nymaim config decoded

proofpoint.com/us/threat-insight/post/nymaim-config-decoded

March 12, 2019



Blog
Threat Insight
Nymaim config decoded



March 12, 2019 Georgi Mladenov

Overview

First documented in 2013 [1], Nymaim was originally identified as both a first-stage downloader and second-stage locking malware. Primarily distributed via the Blackhole exploit kit, most users found out they were infected because of the screen lock that demanded varying ransoms. In 2016, we documented distribution of the Ursnif banking Trojan via email campaigns and the presence of webinjects within Nymaim itself [2]. More recently, Nymaim has evolved into an even more robust downloader that includes a range of information stealing and system profiling capabilities. This incarnation of Nymaim has appeared in both global campaigns as well as attacks targeting North America, Germany, Italy, and Poland. In this respect, Nymaim is following global malware trends, with a focus on persistent, non-destructive infection to collect information long-term and flexibly download additional malware of the threat actor's choosing.

Despite its long history and increasing incidence of spreading via email, many aspects of Nymaim are not well understood, including its ownership and availability to groups of threat actors. Moreover, the configuration file format outlined in this blog and the config's interaction with a virtual machine running within the malware itself appears to be unique. While CERT.pl described the configuration encryption algorithms previously [3], recent samples now employ a bytecode language with its own logic that is interpreted by the malware, running in a virtual machine managed by Nymaim. Technical details of the config file and its interaction with the Nymaim interpreter are outlined below.

Analysis

Config parsing

As other researchers have noted for various components of Nymaim [4], the configuration data is stored in an encrypted and, in some cases, aPLib-compressed format.

The configuration data consists of a binary structure composed of multiple config components or chunks. Each data component has a recognizable pattern that is structured in the following format:

```
struct CONFIG_LINE {
        DWORD       opcode;
        DWORD       params_length;
        byte    params[params_length];
}
```

Upon further analysis, we found that the configuration is compiled bytecode-like data that runs in a custom virtual machine environment inside Nymaim. The config has its own CODE and DATA sections, stack, local variables, registers, conditional cases, procedures, and API calls.

Additionally, the config parser includes a built-in integrity check of the params data, such that on initialization, the structure outlined above is expanded in memory to four parameters:

```
struct CONFIG_LINE_PARSED {
        DWORD       opcode;
        DWORD       crc;
        DWORD       params_length;
        byte    params[params_length];
}
```

The integrity check uses a checksum algorithm that is widely used in the malware and this is its implementation:

```
def crc(data, data_len):
    delta = 0x9AF598DC
    crc = 0
    for i in range(0, data_len // 4):
            crc += struct.unpack_from("<L", data, i * 4)[0]
            crc   += data_len
            crc   += delta
    return crc & 0xFFFFFFFF
```

The first entries from the config contain the compile timestamp, version, and -- if present -- the expiration date. Although the virtual machine skips these entries, they allow us to precisely date each sample from the day it was first distributed to the day on which the campaign ended.

The virtual machine parses the rest of the config file line by line from top to bottom, first reading the code section and then reading the data section at the bottom of the file.

Config execution

As part of the virtual machine, the config interpreter can use its own stack and local variables, along with six general purpose registers. Additionally there is designated space for instruction pointers and flags used by the IF-THEN-GOTO code and API/procedure results.

The config interpreter running on the virtual machine can communicate with other parts of the Nymaim code using a structure holding initial data, which includes:

- IsAdmin flag;
- System version from a OSVERSIONINFOEXW structure;
- SubAuthID;
- Locale obtained by GetLocaleInfoA;
- Pointer to the PEB;
- Event handles;
- Many additional flags;

In the samples we analyzed, the interpreter did not necessarily use all of these parameters. However they are all accessible from the config's code.

The interpreter itself uses a limited number of basic code logic instructions, including flexible variable and register assignments represented in pseudocode below:

```
// &ADDR:**** is the addressing, line by line of the config code

// SP_00 to SP_** are represented as stack pointers

// R0 to R5 are general purpose registers

// Additionally local variables decompiled as LOC_** can be used

&ADDR:0003   SP_00 = 0xFFFFFFFF; // immediate value to stack pointer assignment

&ADDR:0004   SP_04 = 0x00000000;

&ADDR:0005   SP_08 = 0x00000002;

&ADDR:0006   SP_0C = 0x00000004;

&ADDR:0007   SP_10 = 0x00000008;

&ADDR:0008   SP_14 = 0x0000000C;

&ADDR:0009   SP_18 = 0x00000010;

...

&ADDR:0017   R0 = &CPU; // pointer to the CPU data to stack pointer assignment

&ADDR:0018   R1 = 0x0000003C;  // immediate value to register assignment

&ADDR:0019   R2 = 0x00000004;

&ADDR:001A   R3 = &SP_04;  // stack pointer to register assignment

&ADDR:001B   R4 = 0x00000000;

&ADDR:001C   R5 = 0x00000004;
```

Data initialization for the next stage, using labels as delimiters, is shown below:

```
&ADDR:006D   InitData(start=&ADDR:007A, end=&ADDR:0081);

...

&ADDR:0079   LABEL_79:
```

...

&ADDR:0082 LABEL_82:

TEST and IF-THEN-GOTO instructions for conditional branching:

&ADDR:0021 TEST &SP_34 == &SP_44;

&ADDR:0022 IF True GOTO LABEL_4B;

...

&ADDR:002D IF &R0[R1] == &R3[R4] GOTO LABEL_4B;

Procedure calls using entry point LABEL and RET instruction:

&ADDR:001F CALL PROC_69;

...

&ADDR:0069 PROC_69:

...

&ADDR:006E RET;

API-like functions can call into other Nymaim code for more complicated jobs as outlined below:

- Determine presence of a Environment strings (IsEnvStringSet())
- Determine presence of certain processes running (IsProcessRunning())
- Terminate config code execution (Exit())
- Signaling events created by Nymaim (SignalEvent())
- Sending debug messages to Nymaim (DebugMessage())
- Detecting sandboxing and debugging environment (IsDebugged())

Upon further analysis, we were able to decode the checksums for the following processes:

- updatesrv.exe
- vsserv.exe
- pchooklaunch32.exe
- bdagent.exe
- seccenter.exe
- aswidsagenta.exe
- avastui.exe
- avastsvc.exe

All of these point to executables associated with antivirus applications, suggesting that IsProcessRunning() is used to detect installed AV utilities.

The virtual machine uses IsDebugged() for anti-debugging checks, looking for blacklisted items associated with research environments:

- MAC addresses associated with virtual machine platform vendors VmWare, Dell, PCS Computer Systems GmbH, Microsoft Corporation, Parallels, and Xensource.
- Loaded libraries "dbghelp.dll" and "SbieDll.dll" (parts of Debugging Tools For Windows and Sandboxie).
- User names "currentuser" and "sandbox" along with computer names including "sandbox"

Nymaim does not appear to currently use the DebugMessage() function, but passes the argument to the two occurrences of this API call in plaintext:

- "own inside started" // on stage 1 initialization
- "no known av detected" // self explanatory

Typical decompiled configuration

With this information, we were able to decompile Nymaim's config bytecode to a more human-readable pseudocode script. For reference, a complete implementation of the decompiler that generates the following output is available here [6].

// VM &CPU @ B2564545

// VM &TIME @ 06C742A3

```
// VM &FLAG @ 5878305F
&ADDR:0000   // compile timestamp: 2018-11-20T16:36:01.263625
&ADDR:0001   // version: 2.1.20.21
&ADDR:0002   // expiration date: 23 November 2018
&ADDR:0003   SP_00 = 0xFFFFFFFF;
&ADDR:0004   SP_04 = 0x00000000;
&ADDR:0005   SP_08 = 0x00000002;
&ADDR:0006   SP_0C = 0x00000004;
&ADDR:0007   SP_10 = 0x00000008;
&ADDR:0008   SP_14 = 0x0000000C;
&ADDR:0009   SP_18 = 0x00000010;
&ADDR:000A   SP_1C = &SP_00;
&ADDR:000B   SP_20 = &SP_04;
&ADDR:000C   SP_24 = &SP_0C;
&ADDR:000D   SP_28 = &SP_10;
&ADDR:000E   SP_2C = &SP_14;
&ADDR:000F   SP_30 = &SP_18;
&ADDR:0010   SP_34 = &FLAG;
&ADDR:0011   SP_38 = 0x00000000;
&ADDR:0012   SP_3C = 0x0DDD766C;
&ADDR:0013   SP_40 = 0x0CD874EC;
&ADDR:0014   SP_44 = &SP_3C;
&ADDR:0015   SP_48 = &SP_40;
&ADDR:0016   InitData(start=&ADDR:0084, end=&ADDR:0091);
&ADDR:0017   R0 = &CPU;
&ADDR:0018   R1 = 0x0000003C;
&ADDR:0019   R2 = 0x00000004;
&ADDR:001A   R3 = &SP_04;
&ADDR:001B   R4 = 0x00000000;
&ADDR:001C   R5 = 0x00000004;
&ADDR:001D   IF &R0[R1] == &R3[R4] GOTO LABEL_5F;
&ADDR:001E   DebugMessage("own inside started"); // not processed in any way
&ADDR:001F   CALL PROC_69;
&ADDR:0020   IsProcessRunning(0x9BC217C0); // Enabled
&ADDR:0021   TEST &SP_34 == &SP_44;
&ADDR:0022   IF True GOTO LABEL_4B;
&ADDR:0023   IsEnvStringSet(0xDD076E3D);
```

```
&ADDR:0024   TEST &SP_34 == &SP_44;
&ADDR:0025   IF True GOTO LABEL_4B;
&ADDR:0026   IsDebugged(TODO flags); // 00000000 00000001 0000000F 00000000
&ADDR:0027   R0 = &FLAG;
&ADDR:0028   R1 = 0x00000010;
&ADDR:0029   R2 = 0x00000004;
&ADDR:002A   R3 = &SP_04;
&ADDR:002B   R4 = 0x00000000;
&ADDR:002C   R5 = 0x00000004;
&ADDR:002D   IF &R0[R1] == &R3[R4] GOTO LABEL_4B;
&ADDR:002E   R0 = &FLAG;
&ADDR:002F   R1 = 0x00000004;
&ADDR:0030   R2 = 0x00000004;
&ADDR:0031   R3 = &SP_04;
&ADDR:0032   R4 = 0x00000000;
&ADDR:0033   R5 = 0x00000004;
&ADDR:0034   IF &R0[R1] == &R3[R4] GOTO LABEL_66;
&ADDR:0035   R0 = &FLAG;
&ADDR:0036   R1 = 0x00000008;
&ADDR:0037   R2 = 0x00000004;
&ADDR:0038   R3 = &SP_04;
&ADDR:0039   R4 = 0x00000000;
&ADDR:003A   R5 = 0x00000004;
&ADDR:003B   IF &R0[R1] == &R3[R4] GOTO LABEL_66;
&ADDR:003C   SP_4C = 0xFFFFFFF7;
&ADDR:003D   R0 = &FLAG;
&ADDR:003E   R1 = 0x0000000C;
&ADDR:003F   R2 = 0x00000004;
&ADDR:0040   R3 = &SP_4C;
&ADDR:0041   R4 = 0x00000000;
&ADDR:0042   R5 = 0x00000004;
&ADDR:0043   LOC_00 = &R0[R1] & &R3[R4];
&ADDR:0044   R0 = &LOC_00;
&ADDR:0045   R1 = 0x00000000;
&ADDR:0046   R2 = 0x00000004;
&ADDR:0047   R3 = &SP_04;
&ADDR:0048   R4 = 0x00000000;
```

```
&ADDR:0049   R5 = 0x00000004;
&ADDR:004A   IF &R0[R1] == &R3[R4] GOTO LABEL_66;
&ADDR:004B   LABEL_4B:
&ADDR:004C   IsProcessRunning("updatesrv.exe", "vsserv.exe", "pchooklaunch32.exe", "bdagent.exe", "seccenter.exe"); // Enabled
&ADDR:004D   TEST &SP_34 == &SP_44;
&ADDR:004E   IF True GOTO LABEL_5D;
&ADDR:004F   R0 = 0x00000100;
&ADDR:0050   R1 = 0x00000400;
&ADDR:0051   R2 = 0x00000000;
&ADDR:0052   R3 = 0x00000003;
&ADDR:0053   // TODO ID:7DD14382 DATA:b'00000000'
&ADDR:0054   TEST &SP_34 == &SP_44;
&ADDR:0055   IF True GOTO LABEL_5D;
&ADDR:0056   R0 = &FLAG;
&ADDR:0057   R1 = 0x00000004;
&ADDR:0058   R2 = 0x00000004;
&ADDR:0059   R3 = &SP_04;
&ADDR:005A   R4 = 0x00000000;
&ADDR:005B   R5 = 0x00000004;
&ADDR:005C   IF &R0[R1] == &R3[R4] GOTO LABEL_63;
&ADDR:005D   LABEL_5D:
&ADDR:005E   GOTO LABEL_61;
&ADDR:005F   LABEL_5F:
&ADDR:0060   CALL PROC_69;
&ADDR:0061   LABEL_61:
&ADDR:0062   Exit(0); // Exit process
&ADDR:0063   LABEL_63:
&ADDR:0064   SignalEvent(); // pre-process termination
&ADDR:0065   Exit(0); // Exit process
&ADDR:0066   LABEL_66:
&ADDR:0067   SignalEvent(); // pre-process termination
&ADDR:0068   Exit(0); // Exit process
&ADDR:0069   PROC_69:
&ADDR:006A   IsProcessRunning("aswidsagenta.exe", "avastui.exe", "avastsvc.exe"); // Enabled
&ADDR:006B   TEST &SP_34 == &SP_44;
&ADDR:006C   IF True GOTO LABEL_6F;
&ADDR:006D   InitData(start=&ADDR:007A, end=&ADDR:0081);
```

&ADDR:006E   RET;

&ADDR:006F   LABEL_6F:

&ADDR:0070   DebugMessage("no known av detected"); // not processed in any way

&ADDR:0071   InitData(start=&ADDR:007A, end=&ADDR:0081);

&ADDR:0072   RET;

&ADDR:0073   LABEL_73:

&ADDR:0074   DATA_DFE8715B = [0x00001F40, 0x00000001, 0x00000004, 0x00000001, 0x00001770, 0x00002000, 0x00000000, 0x000000C8, 0x00000064];

&ADDR:0075   DATA_97AC42CB = [0x00000000];

&ADDR:0076   DATA_PAYLOAD_TARGET_ENUMERATOR = [0x00000000, 0x00000000, "*'%ProgramFiles(x86)%;#!#*.exe';@'#!#*avast#*;#!#*defender#*;#!#*uninstall#*;#!#*instal#*;#!#*setup#*;#!#*config#*;#!#*iexplore.exe;#!#*chrome %!rndl_0_0_2_1_3%';$'#*';!'0;0;-%!rndl_0_0_2_1_3%';*'%ProgramW6432%;#!#*.exe';@'#!#*avast#*;#!#*defender#*;#!#*uninstall#*;#!#*install#*;#!#*setup#*;#!#*config#*';$'#!#*win %!rndl_0_0_2_1_3%';$'#*';!'0;0;-%!rndl_0_0_2_1_3%'"];

&ADDR:0077   DATA_D3955141 = [0x00000001, 0x00000000, "%!system32W6432%\rundll32.exe; -%!rndl_0_0_2_1_3% %!rndl_0_0_2_3_8%.dll"];

&ADDR:0078   LABEL_78:

&ADDR:0079   LABEL_79:

&ADDR:007A   DATA_DFE8715B = [0x00001F40, 0x00000001, 0x00000004, 0x00000001, 0x00001770, 0x00002000, 0x00000000, 0x000000C8, 0x00000064];

&ADDR:007B   DATA_97AC42CB = [0x00000001];

&ADDR:007C   DATA_BB98FAB8 = [0x00000400];

&ADDR:007D   DATA_3FEF1B94 = [0x00000004];

&ADDR:007E   DATA_PAYLOAD_DROPPATH = "%SystemRoot%#*";

&ADDR:007F   DATA_PAYLOAD_DROPFILE = "#!#*flashutil32#*;#!#*flashplayerpl#*;#!#*windowslivemail#*";

&ADDR:0080   DATA_PROCLIST_EXCLUDE_EXT = ["iexplore.exe", "firefox.exe", "chrome.exe", "winword.exe", "outlook.exe", "excel.exe", "powerpnt.exe", "iexplore.exe", "firefox.exe", "chrome.exe", "winword.exe", "outlook.exe", "excel.exe", "powerpnt.exe", "acrord32.exe", "java.exe", "javaw.exe", "regsvr32.exe", "wscript.exe", "cscript.exe", "powershell.exe", "mshta.exe", "certutil.exe", "sqlservr.exe", "opera.exe", "msaccess.exe", "sysprep.exe", "setupsqm.exe", "cliconfg.exe", "winsat.exe", "mmc.exe", "oobe.exe", "inetmgr.exe", "taskhost.exe", "inetmgr.exe", "dism.exe", "dismhost.exe", "taskeng.exe", "cmd.exe", "wscript.exe", "cscript.exe", "java.exe", "powershell.exe", "mshta.exe", "winlogon.exe", "services.exe", "svchost.exe", "spoolsv.exe", "explorer.exe", "explorer.exe", "cmd.exe", "rundll32.exe", "msiexec.exe", "mspaint.exe", "notepad.exe", "calc.exe", "taskhost.exe", "dwm.exe", "taskmgr.exe", "msnmsgr.exe", 0xF6FE5321, "dllhost.exe", "runouce.exe", "vds.exe", "mstsc.exe", "mysqld.exe", "onenotem.exe", "dropbox.exe", "conhost.exe", "jusched.exe", "sihost.exe", "splwow64.exe", "msdt.exe", "onedrive.exe", "skype.exe", "solitaire.exe", "steam.exe"];

&ADDR:0081   DATA_PAYLOAD_TARGET_ENUMERATOR = [0x00000000, 0x00000000, "*'%ProgramFiles(x86)%;#!#*.exe';@'#!#*avast#*;#!#*defender#*;#!#*uninstall#*;#!#*instal#*;#!#*setup#*;#!#*config#*;#!#*iexplore.exe;#!#*chrome %!rndl_0_0_2_1_3%';$'#*';!'0;0;-%!rndl_0_0_2_1_3%';*'%ProgramW6432%;#!#*.exe';@'#!#*avast#*;#!#*defender#*;#!#*uninstall#*;#!#*install#*;#!#*setup#*;#!#*config#*';$'#!#*win %!rndl_0_0_2_1_3%';$'#*';!'0;0;-%!rndl_0_0_2_1_3%'"];

&ADDR:0082   LABEL_82:

&ADDR:0083   LABEL_83:

&ADDR:0084   DATA_8568A01D = [0x00000001, 0x00000001];

&ADDR:0085   DATA_14A8D56E = [0x00, 0x02, 0x00, 0x00, 0xD4, 0x30, 0x37, 0xBF, 0x80, 0x4B, 0x8F, 0xE0, 0xE5, 0xCC, 0x27, 0xE7, 0x23, 0xCF, 0x53, 0x37, 0xB1, 0x46, 0xD8, 0x03, 0xCA, 0xEE, 0x29, 0x4B, 0x66, 0x8A, 0x9A, 0xCA, 0x51, 0xA8, 0x95, 0x81, 0x28, 0x93, 0x01, 0xF9, 0x60, 0x7D, 0x9A, 0x6C, 0xB2, 0x93, 0x15, 0x2B, 0x44, 0xC0, 0x6F, 0x6F, 0xB7, 0x1A, 0xC8, 0x12, 0x52, 0x1A, 0xC7, 0x28, 0x0A, 0x46, 0x0D, 0x2E, 0x84, 0x50, 0x9B, 0xFD, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x01, 0x00, 0x01];

&ADDR:0086   DATA_56B437D3 = ["system", "lsass.exe", "smss.exe", "csrss.exe", "wininit.exe", "services.exe", "lsm.exe", "audiodg.exe", "svchost.exe", "winlogon.exe", "spoolsv.exe", "searchindexer.exe", "msdtc.exe", "searchprotocolhost.exe", "dllhost.exe"];

&ADDR:0087   DATA_DBED7DFB = ["system", "lsass.exe", "smss.exe", "csrss.exe", "wininit.exe", "services.exe", "lsm.exe", "audiodg.exe", "svchost.exe", "winlogon.exe", "spoolsv.exe", "searchindexer.exe", "msdtc.exe", "explorer.exe", "dwm.exe"];

&ADDR:0088   DATA_PROCLIST_EXCLUDE = ["wininit.exe", "csrss.exe", "smss.exe", "avp.exe", "avpui.exe", "avgnt.exe", "avguard.exe", "sched.exe", "aswidsagenta.exe", "avastsvc.exe", "avastui.exe", "ccsvchst.exe", "avgcsrvx.exe", "avgnsx.exe", "avgrsx.exe", "avgtray.exe", 0x6EEA1E2D, "avgwdsvc.exe", "ekrn.exe", "egui.exe", "msmpeng.exe", "msseces.exe", "mpcmdrun.exe", "psimsvc.exe", "apvxdwin.exe", "avengine.exe", "pavfnsvr.exe", "pavprsrv.exe", "pavsrvx86.exe", "psctrls.exe", "psksvc.exe", "fpavserver.exe", 0xBAA21E23, "fpwin.exe", "cmdagent.exe", "cfp.exe", "gdscan.exe", "gdsc.exe", "avktray.exe", "avkservice.exe", "avkproxy.exe", "pctsgui.exe", "pctssvc.exe", "pctsauxs.exe", "update.exe", "updatesrv.exe", "vsserv.exe", "pchooklaunch32.exe", "bdagent.exe", "seccenter.exe", "mrt.exe", "mcupdate.exe", "mcagent.exe", "mcmscsvc.exe", "mcnasvc.exe", "mcproxy.exe", "mcsacore.exe", "mcshell.exe", "mcshield.exe", "mcsysmon.exe", "mpfsrv.exe", "msksrver.exe", "udaterui.exe", "engineserver.exe", "frameworkservice.exe", "engineserver.exe", "frameworkservice.exe", "mctray.exe", "naprdmgr.exe", "shstat.exe", "vstskmgr.exe", "almon.exe", "alsvc.exe", "savadminservice.exe", 0x12FCC4EB, "savservice.exe", "swi_service.exe", 0xBC6D3196, "protoolbarupdate.exe", "sfctlcom.exe", "tmbmsrv.exe", "tmpfw.exe", "tmproxy.exe", "tscfplatformcomsvr.exe", 0xAEF84E17, "ufseagnt.exe", 0x3DC44746, 0x13F2ED0D, 0x4B4EA20C, "umxagent.exe", "umxcfg.exe", "umxpol.exe", "caav.exe", "casc.exe", "cavrid.exe", "ccprovsp.exe", "vetmsg.exe", "vmwaretray.exe", "vmwareuser.exe", 0xBD157242, "vmtoolsd.exe", "wrsa.exe"];

&ADDR:0089   DATA_5DD00BF4 = ["winlogon.exe"];

&ADDR:008A   DATA_87045172 = ["explorer.exe", "dwm.exe", "taskhost.exe", "conhost.exe"];

&ADDR:008B   DATA_0B7EEE53 = [0x00000000];

&ADDR:008C   DATA_FFF28C72 = "zepter.com;carfax.com;";

&ADDR:008D   DATA_GOOGLE_DNS = [0x00000003, timeout_write=8000, timeout_read=8000, 0x00000001, 0x00000002, "8.8.8.8:53;8.8.4.4:53"];

&ADDR:008E   DATA_992CC894 = [0x00000000, 0x00000001, 0x00000001, 0x00000002, 0x00000001, 0x00000001, 0x00000000, 0x00000000, 0x00000000, 0x00000000, "gx3Gd93kdXdjd]dGdg573"];

&ADDR:008F   DATA_8DB1E244 = [0x00000001, 0x18482642, 0x78643587, 0x87568289, 0x00000010];

&ADDR:0090   DATA_C9393B40 = "~[duewosgems.com];fiosbewos.com;";

&ADDR:0091   DATA_F50DF89A = [0x00000019, 0x000493E0, 0x00000001, 0x00000001, "~[duewosgems.com]/pkbn74is/index.php;fiosbewos.com/pkbn74is/index.php;"];

&ADDR:0092   LABEL_92:

&ADDR:0093   END;

Conclusion

Nymaim is one of a number of downloaders appearing regularly in the wild, reflecting the trend of installing persistent malware on victim devices. Although Nymaim has a long history as a downloader and earlier incarnations as ransomware and a desktop locker, little is known about its ownership or availability to other threat actors. We were, however, able to decode the unique config, providing additional insight for defenders and potential victims of increasingly frequent Nymaim campaigns.

References

[1] https://www.welivesecurity.com/2013/08/26/nymaim-obfuscation-chronicles/

[2] https://www.proofpoint.com/us/what-old-new-again-nymaim-moves-past-its-ransomware-roots-0

[3] https://www.welivesecurity.com/2013/10/23/nymaim-browsing-for-trouble/

[4] http://www.eset.com/int/about/press/articles/article/nymaim-ransomware-still-active-finding-new-infection-vector-to-spread-black-hat-seo/

[5] https://www.microsoft.com/security/portal/threat/Encyclopedia/Entry.aspx?Name=Ransom:Win32/Nymaim.F#tab=2

[6] https://github.com/EmergingThreats/threatresearch/blob/master/Nymaim/config_decompiler.py

Indicators of Compromise (IOCs)

IOC	IOC Type	Description
76b855f4822c0b26e098d7395723b31ad73c1606aebdb972380ef6c9f0bb4936	SHA256	Nymaim sample (2019)
8cb27fca6cf68888126a82c304083ebd78bba2b9f6fb241d2a177a3a80f12e8a	SHA256	Nymaim sample (2019)
0f115ff9d7ecbe2b4872a18c14e97d6071a61435690729c9aa741cecc8982383	SHA256	Nymaim sample (2019)
7541c32d82b17e9d3a993f6721a1b84221dfbee6cbe7f060413a118c48ae64ee	SHA256	Nymaim sample (2019)
43c19be78773a14196abb4ecb6436b54729373eacf84da7a9a2c3592ad960cae	SHA256	Nymaim sample (2019)