MoonPeak malware from North Korean actors unveils new details on attacker infrastructure

Asheer Malhotra ⋮⋮ 8/21/2024

By Asheer Malhotra, Guilherme Venere, Vitor Ventura

Wednesday, August 21, 2024 06:00
APT DPRK North Korea malware

- Cisco Talos is exposing infrastructure we assess with high confidence is being used by a state-sponsored North Korean nexus of threat actors we track as “UAT-5394," including for staging, command and control (C2) servers, and test machines the threat actors use to test their implants.
- Our analysis of the threat actor’s infrastructure indicates they pivoted across C2s and staging servers to set up new infrastructure and modify existing servers.
- This campaign consists of distributing a variant of the open-source XenoRAT malware we're calling “MoonPeak,” a remote access trojan (RAT) being actively developed by the threat actor.
- Analysis of XenoRAT against MoonPeak malware samples we’ve discovered so far illustrates the evolution of the malware family after it was forked by the threat actors.

Cisco Talos has uncovered a new remote access trojan (RAT) family we are calling “MoonPeak.” This a XenoRAT-based malware, which is under active development by a North Korean nexus cluster we are calling “UAT-5394.” Our analysis of infrastructure used in the campaign reveals additional links to the UAT-5394 infrastructure and new tactics, techniques and procedures (TTPs) of the threat actor.

In a recent report, AhnLab disclosed a spear-phishing campaign employing the use of an early variant of XenoRAT, an open-source RAT family, which evolved into what we track as “MoonPeak.”

This cluster of activity has some overlaps in TTPs and infrastructure patterns with the North Korean state-sponsored group “Kimsuky,” however, we do not have substantial technical evidence to link this campaign with the APT.

Since Kimsuky has been rapidly evolving and upgrading their infrastructure and tooling since 2024, the development and usage of a new RAT in this specific campaign raises two possibilities we must consider:

- Either UAT-5394 is actually Kimsuky (or a sub-group within Kimsuky) and they are replacing QuasarRAT with MoonPeak. (We have observed UAT-5394 actively setting up and operating QuasarRAT C2 servers before they eventually adopted the use of XenoRAT and MoonPeak.)

- Or, UAT-5394 is another group within the North Korean APT machinery that borrows their TTPs and infrastructure patterns from Kimsuky.

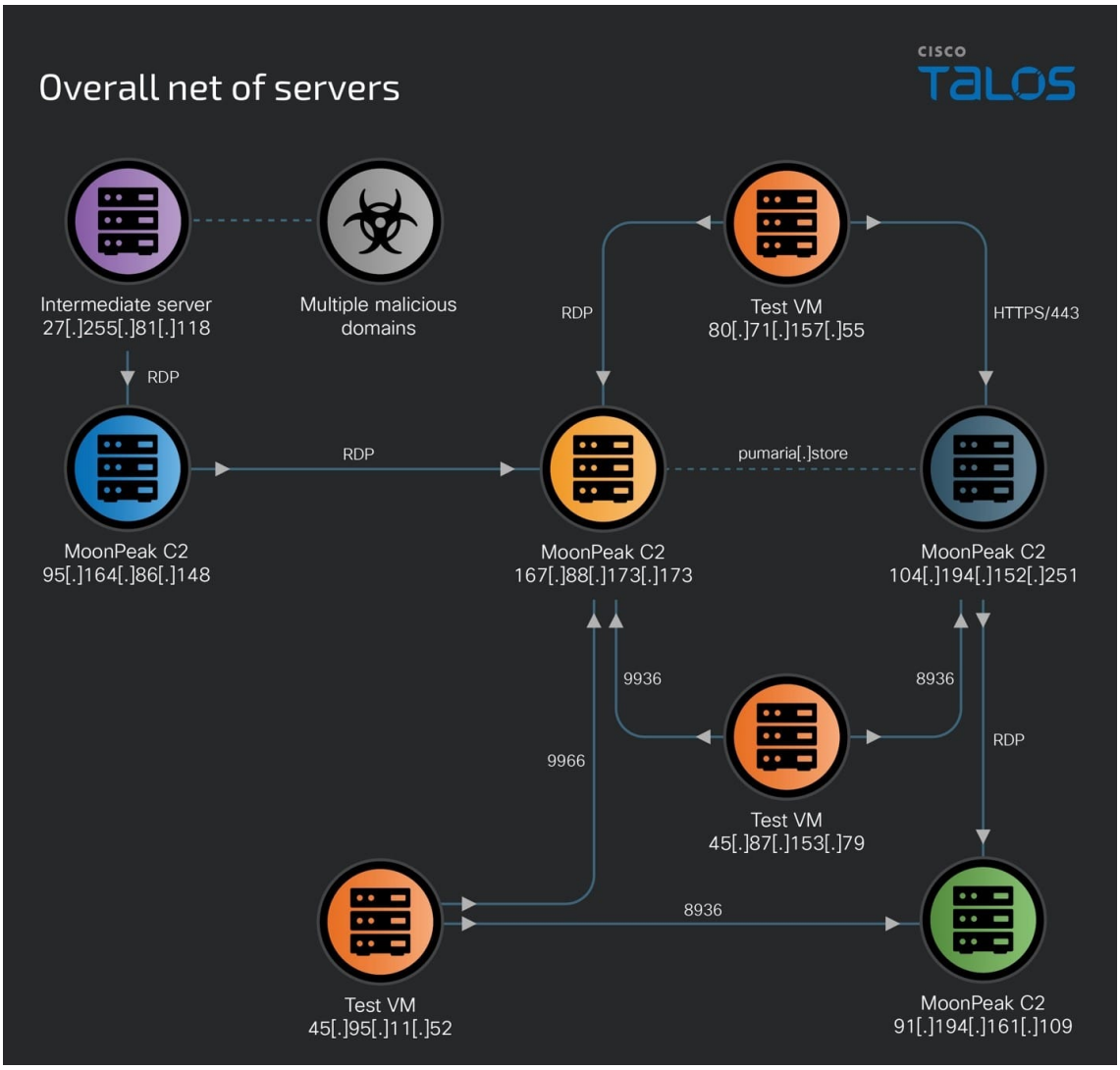
We will, for the time being, consider this cluster of activity an independent campaign owned and operated under the UAT-5394 moniker until we have more intelligence to merge this campaign into Kimsuky's attacks or determine that UAT-5394 is, in fact, a disparate/unique group operating within DPRK's APT machinery.

Talos' research has uncovered the testing and staging infrastructure used to create new iterations of MoonPeak. The C2 server hosts malicious artifacts for download, which is then used to access and set up new infrastructure to support this campaign. In multiple instances, we also observed the threat actor access existing servers to update their payloads and retrieve logs and information collected from MoonPeak infections. Apart from accessing servers from other servers, the threat actors also accessed their infrastructure from VPN nodes.

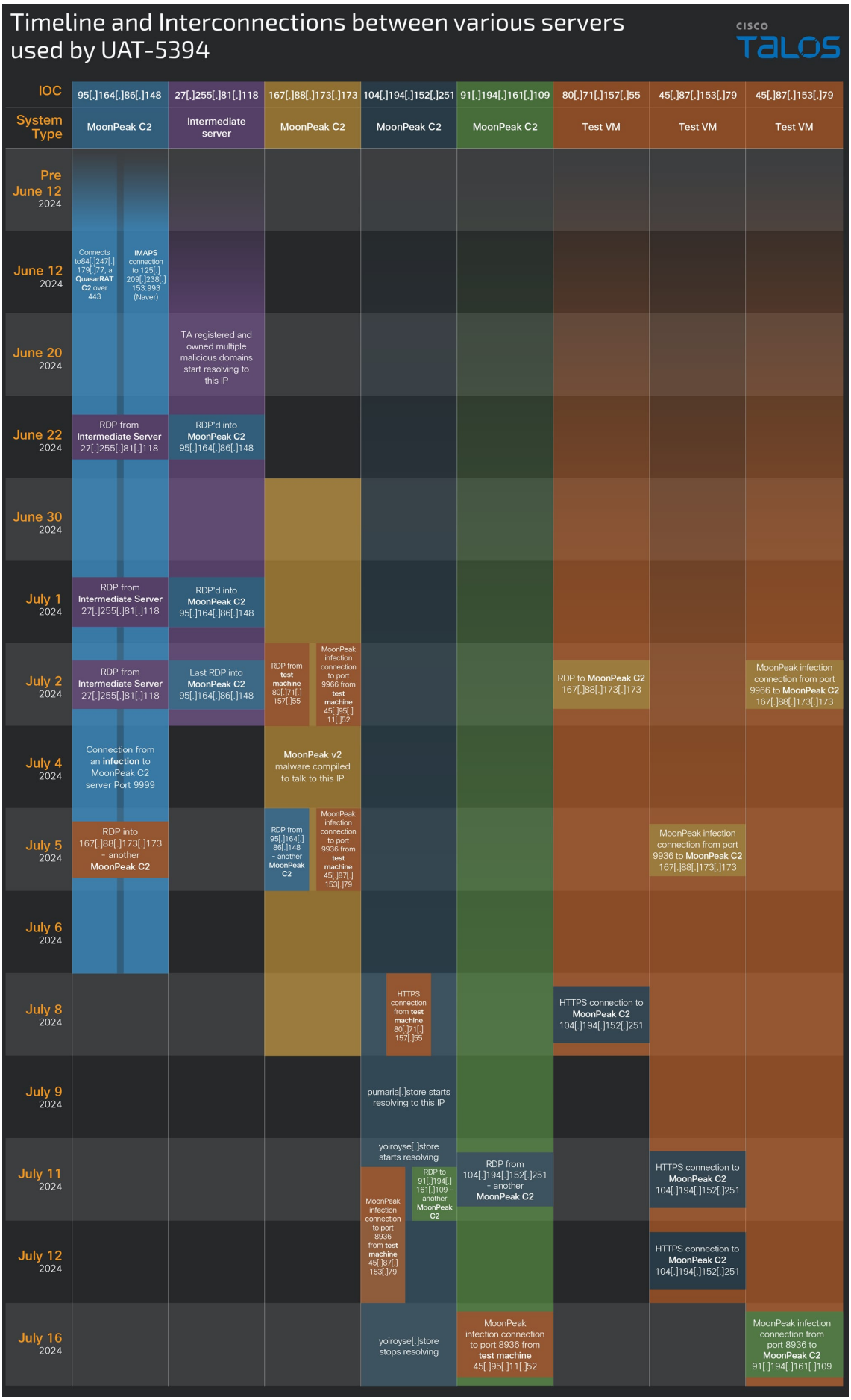
Mapping out UAT-5394 infrastructure

Talos' analysis mapping out the infrastructure involved in this campaign led to a plethora of new servers owned, operated and administered by the threat actor. This infrastructure includes remote access and C2 servers, payload-hosting sites and even test virtual machines the attackers used to test their MoonPeak implants before distributing them to potential targets.

The net of the infrastructure can be mapped as:



The UAT-5394 activity over the past two months shows the interconnections between the several servers used by this threat actor.



View a larger version of this image

UAT-5394 servers.jpg

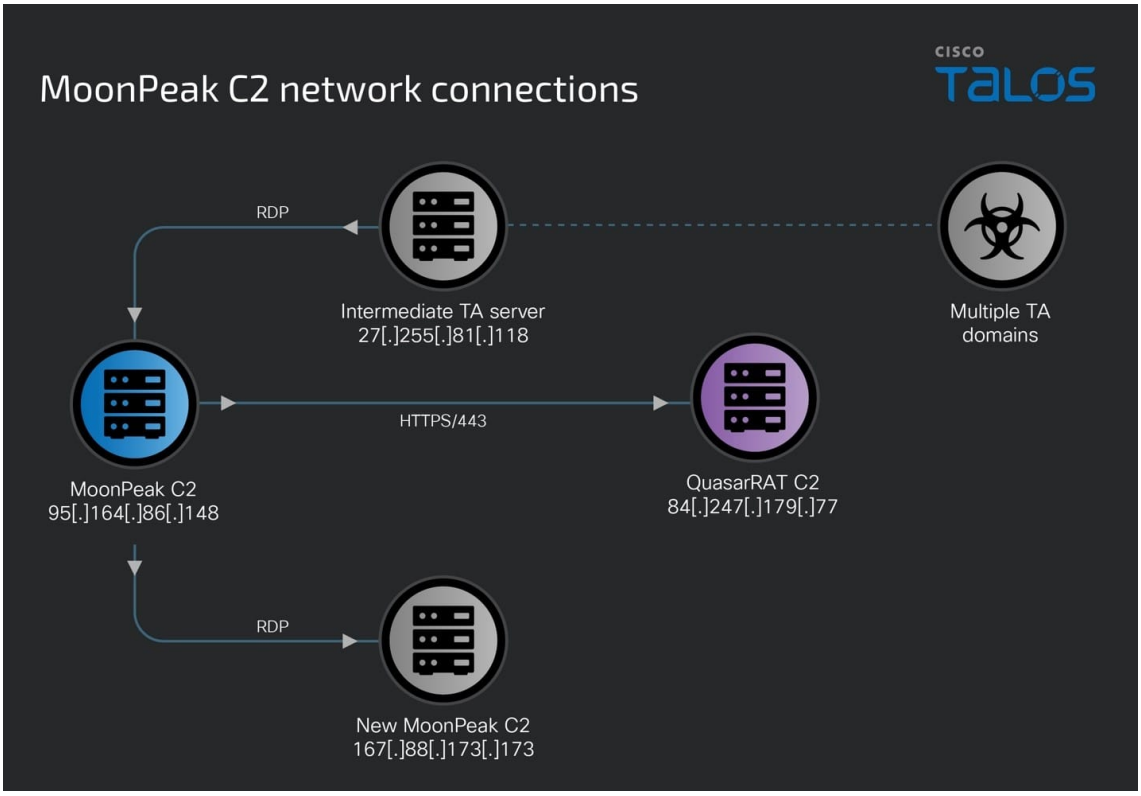
107 KB

The move from cloud services to attacker-owned infrastructure: 95[.]164[.]86[.]148

Since June 11, 2024, we saw a distinct shift in the actor's tactics with respect to setting up supporting infrastructure. After AhnLab's disclosure, UAT-5394 moved from hosting their malicious payloads on legitimate cloud storage providers to systems and servers they now owned and controlled. It is likely that this move was made to preserve their infections from potential shutdown of cloud locations by the service providers.

95[.]164[.]86[.]148 is one of the earliest servers set up and actively used by UAT-5394 since at least June 12, 2024, to host malicious artifacts (described in AhnLab's disclosure) and served as a MoonPeak C2 server on Port 9999 until at least July 4, 2024. This C2 server was accessed between this time frame, over RDP by 27[.]255[.]81[.]118, another UAT-5394 IOC resolving multiple malicious domains registered by the threat actors.

On July 5, 2024, the threat actors now used 95[.]164[.]86[.]148 to RDP into a second malicious server, 167[.]88[.]173[.]173 which was already serving as a MoonPeak C2 on Port 9966. This RDP access to 167[.]88[.]173[.]173 resulted in a second deployment of MoonPeak's C2 on Port 9936.



It is also worth noting that we observed the MoonPeak C2 server (95[.]164[.]86[.]148) connect to 84[.]247 [.]179 [.]77:443 between June 22 and July 2, 2024. 84[.]247 [.]179 [.]77 is a known C2 server for another open-sourced RAT family, QuasarRAT, and hosts an SSL cert with serial number "8cf5fb326e1e6d3015c3846f09c93b" and the CN is listed as "Quasar Server CA." Talos assesses with low confidence that the MoonPeak server may have been infected with QuasarRAT so that the threat actors had a parallel means of maintaining access to the C2 from the QuasarRAT C2.

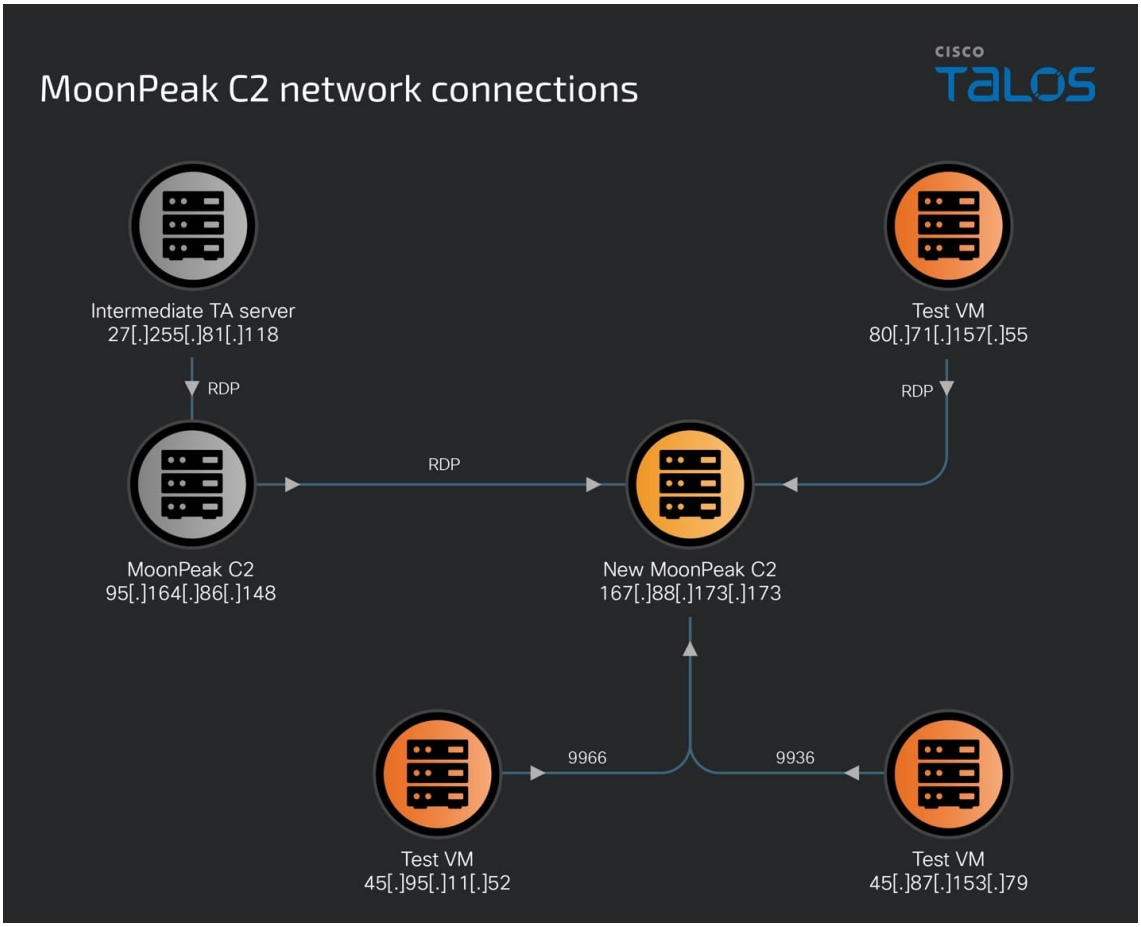
A pivotal server - 167[.]88[.]173[.]173

167[.]88[.]173[.]173 is a high-flux server that has changed operating systems and web servers four times in less than two months. Looking at passive DNS data, this would seem like a Linux server owned by the Gamaredon APT, a threat group allegedly linked to the Russian FSB by the Security Service of Ukraine (SSU). However, our analysis has found a window of time in which we assess with high confidence that the IP was under UAT-5394 ownership and control.

Between June 30 and July 8, 2024, this server had the Windows Server 2022 operating system and was under ownership of UAT-5394. During this time,specifically on July 2, 2024, UAT-5394 compiled MoonPeak v2 malware samples pointing to this C2's Port 9966.

This IP was also accessed by two other IP addresses, 45[.]87[.]153[.]79 and 45[.]95[.]11[.]52 over ports 9936 and 9966 on the same day, which are the C2 ports used by MoonPeak malware created by the threat actor. Both IP addresses from AS 44477 (Stark Industries Solutions Ltd.) are important since we will later demonstrate that these are test machines/VM used by the threat actors to test their implants.

The new MoonPeak C2 server was also accessed by the attackers using RDP (port 3389) using two other remote IP addresses – 80[.]71[.]157[.]55 and 95[.]164[.]86[.]148 during the time it was under UAT-5394's control.

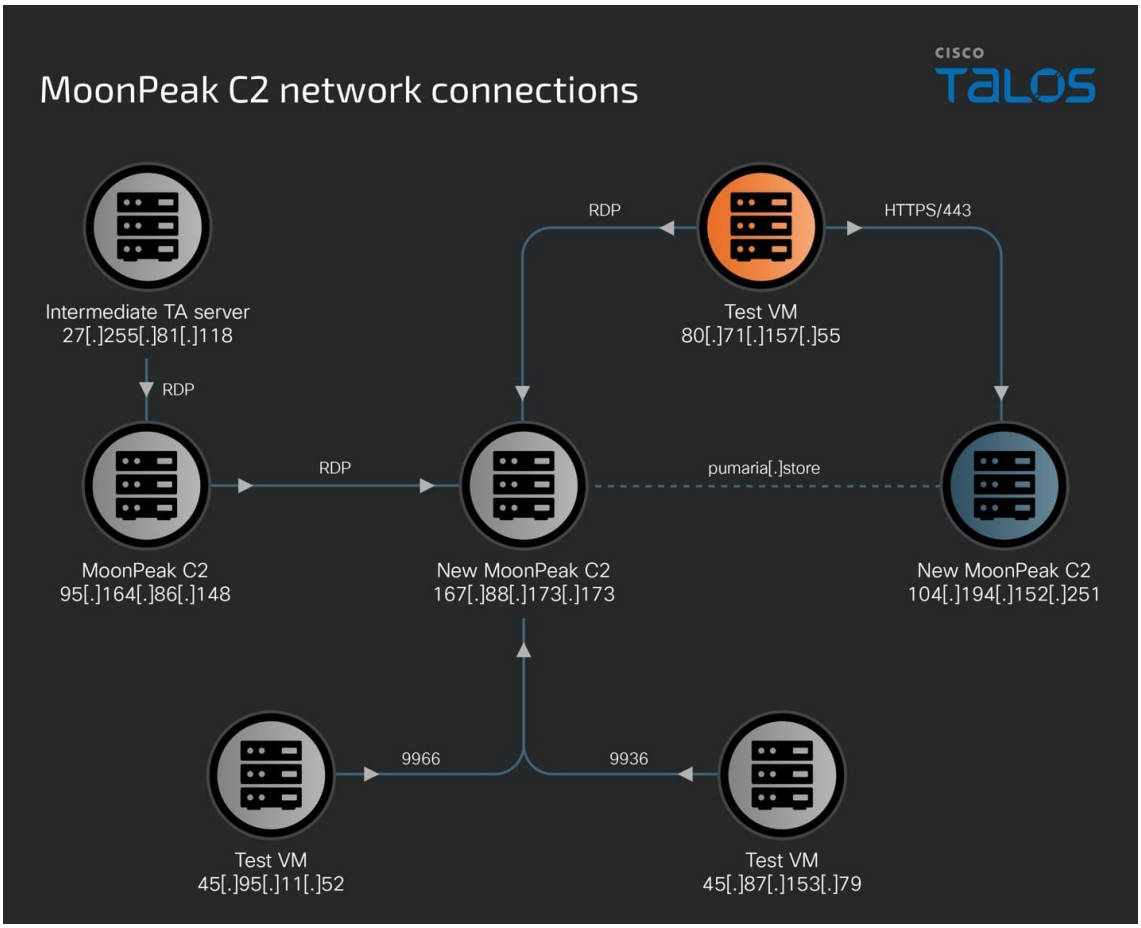


Intermediate system, 80[.]71[.]157[.]55, establishes links between two C2 servers

We discovered that 167[.]88[.]173[.]173 resolves to and hosts an SSL certificate for pumaria[.]store, a malicious domain owned by UAT-5394. This domain later resolved to 104[.]194[.]152[.]251 on July –11 2024, the same day that the threat actors tested another implant on one of their test VMs that connected back to this IP on Port 8936 — another XenoRAT infection — indicating that 104[.]194[.]152[.]251 was a new MoonPeak C2 being set up by the threat actors.

Furthermore, we also found evidence that another one of UAT-5394's test machine (80[.]71[.]157[.]55) also reached out to and communicated with the IP over Port 443 on July 8, 2024. Based on infection logs and network analysis, we assess with medium confidence that, in addition to remotely accessing other systems, this system (80[.]71[.]157[.]55) has also been used to test MoonPeak infections since January 2024 and the HTTPS (443) communication with 104[.]194[.]152[.]251 is one of these tests.

Our analysis deems 104[.]194[.]152[.]251 a malicious C2 server for communicating with the MoonPeak malware and the threat actors were preparing to use the system in their malicious operations and had it ready for use by July 8, 2024.

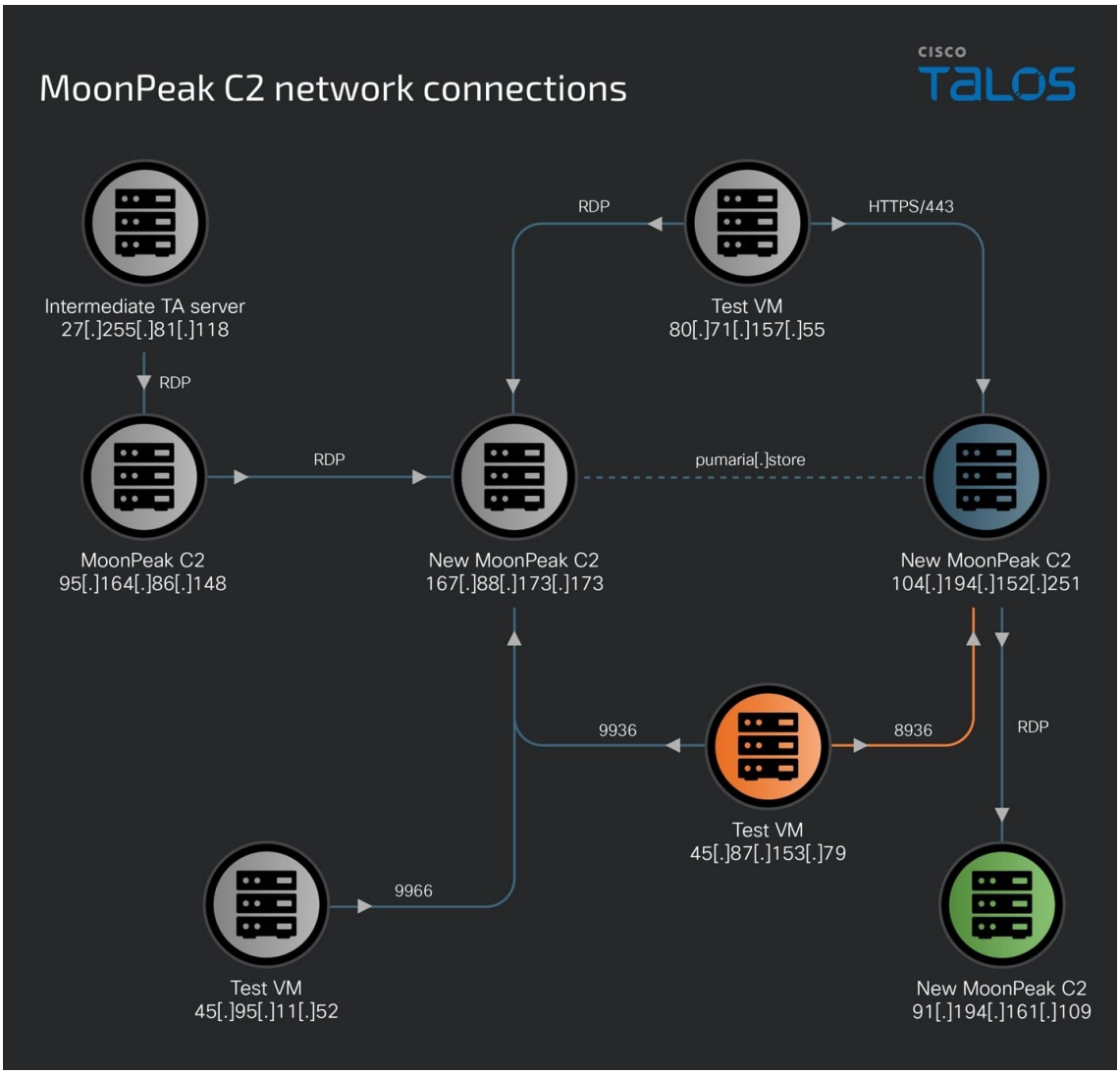


From C2 to C2: Finding a newly minted MoonPeak C2

Talos' analysis of 104[.]194[.]152[.]251 between June and July 2024 revealed that it resolved to the pumaria[.]store and yoiroyse[.]store domains that we attribute to UAT-5394.

We also observed one of UAT-5394's infection testing machines (45[.]87 [.]153 [.]79) communicate with this server over port 8936 (the same port used in MoonPeak v2) on July 11, 2024, reinforcing our assessment that 104[.]194[.]152[.]251 hosted a MoonPeak C2.

We also discovered that the threat actors used this system to access another IP address, 91[.]194[.]161[.]109 via RDP (port 3389) on July 11, 2024, to set up the IP address as the newest host for the malware and MoonPeak C2.



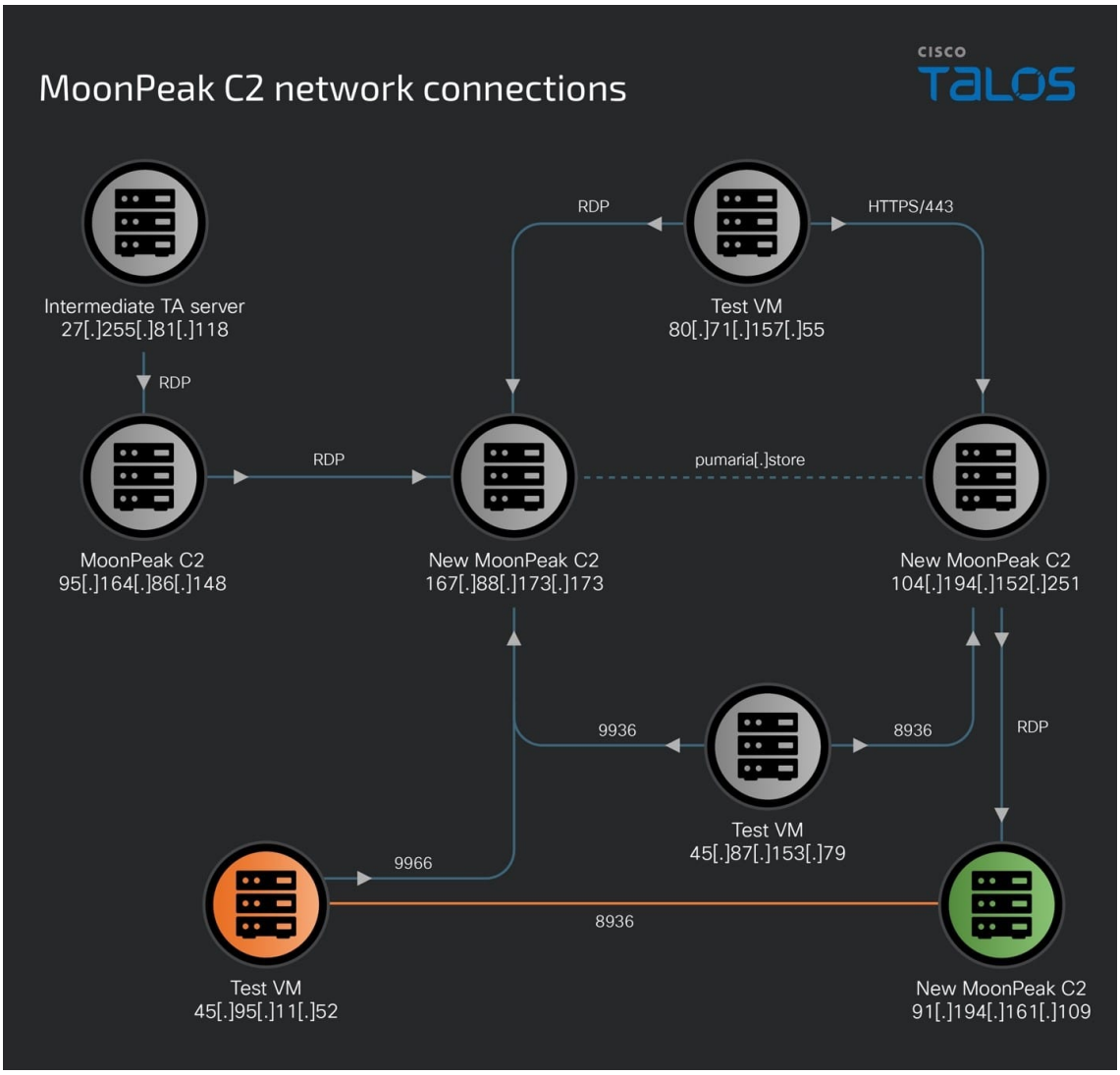
Enter MoonPeak's newest C2 server — 91[.]194[.]161[.]109

91[.]194[.]161[.]109 hosts multiple malicious artifacts including scripts to carry out the infection chain ultimately serving MoonPeak to targets.

The latest version of MoonPeak discovered on this server is built as recently as July 16, 2024, and connects to the server over Port 8936.

On the same day, we observed one of the threat actor's test machines (45[.]95[.]11[.]52) began communicating with this C2 server.

Based on this timeline, it is likely that the threat actors built the MoonPeak implant, set up the C2 component and then executed MoonPeak on their test VM to test the correct functioning of the variant.



This MoonPeak server hosts other artifacts from the infection chain leading up to MoonPeak:

- A PHP file that serves malicious artifacts based on the “id” value provided.
- A PowerShell script (calc[.]txt) was meant to pull down an RTF file from the remote host. The RTF is downloaded and then the first six bytes are replaced with the GZIP header. The resulting GZIP contains the MoonPeak malware that is executed on the compromised endpoint.
- Another PowerShell script to replace the GZIP header bytes with those of an RTF. This script converts the GZIP to RTF and stores it on the server.

```powershell
function GzCompress {

    [CmdletBinding()]
    Param (
        [Parameter(Position = 0, Mandatory = $True)]
        [string] $SrcPath = $(Throw("-SrcPath is required")),

        [Parameter(Position = 1, Mandatory = $True)]
        [string] $DstPath = $(Throw("-DstPath is required"))
    )

    Process {
        $byteArray = [System.IO.File]::ReadAllBytes($SrcPath)
        if( $byteArray.Length -le 0 ) {
            Throw("file read error.")
        }

        [System.IO.MemoryStream] $output = New-Object System.IO.MemoryStream
        $gzipStream = New-Object System.IO.Compression.GzipStream $output,
([IO.Compression.CompressionMode]::Compress)
        $gzipStream.Write( $byteArray, 0, $byteArray.Length )
        $gzipStream.Close()
        $output.Close()
        $tmp = $output.ToArray()
        # to rtf file
        $tmp[0]  = 0x7B;
        $tmp[1]  = 0x5C;
        $tmp[2] = 0x72;
        $tmp[3] = 0x74;
        $tmp[4] = 0x66;
        $tmp[5] = 0x31;
        $tmp[6] = 0x7D;
        $tmp[7] = 0x00;
        [System.IO.File]::WriteAllBytes($DstPath, $tmp)
    }
}

GzCompress -SrcPath <source_path> -DstPath <resulting_RTF_path>
```

Virtual machines for testing implants and infection chains

We discovered several instances of two virtual machines hosted on public IPs that reached out to various MoonPeak C2 servers over ports configured in the malware.

The timing of the communications over specific C2 ports corresponds to the compilation times of various MoonPeak samples we've discovered in this campaign so far.

These VMs, 45[.]87[.]153[.]79 and 45[.]95[.]11[.]52, have been used to test MoonPeak infections over Ports 9966, 9936, 8936 and 9999 since at least July 2, 2024.

A third test machine, 80[.]71[.]157[.]55, was also used by UAT-5394, but it is a dual-purpose machine – testing infections and used to RDP into C2 servers.

MoonPeak Malware – Slowly evolving toward evasion

An analysis of MoonPeak samples reveals an evolution in the malware and its corresponding C2 components that warranted the threat actors deploy their implant variants several times on their test machines. The constant evolution of MoonPeak runs hand-in-hand with new infrastructure set up by the threat actors. Each new increment of MoonPeak differs from the previous one in two aspects:

- Just enough to introduce more obfuscation to make detection and identification more cumbersome.
- Just enough tweaks in communication and peripheral characteristics of the malware and the corresponding XenoRAT server code to prevent unauthorized connections and instrumentation of MoonPeak malware and C2

servers. Simply put, the threat actors ensured that specific variants of MoonPeak only work with specific variants of the C2 server.

Evolution of MoonPeak over time

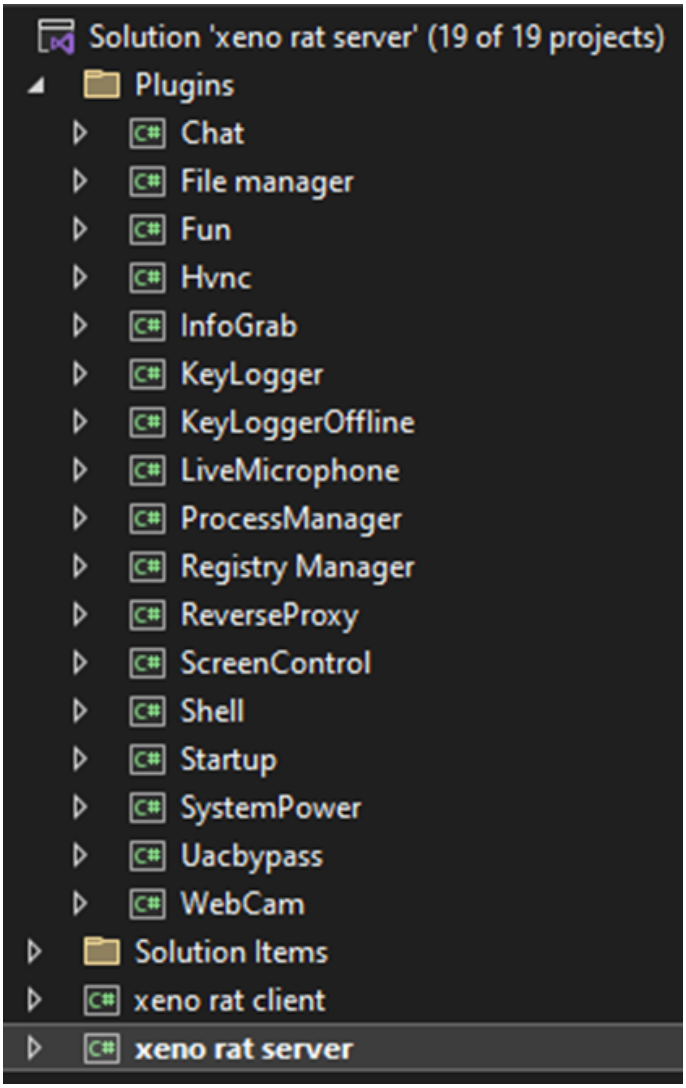
MoonPeak is based on a publicly available source code for XenoRAT made available on GitHub around October 2023. While MoonPeak contains most of the functionalities of the original XenoRAT, our analysis observed consistent changes throughout the variants that shows the threat actors are modifying and evolving the code independently from the open-source version.

The samples used in this analysis were found based on characteristics matching the samples reported by Ahnlab, which we are calling “MoonPeak v1,” as well as samples Talos independently found we are calling “Moonpeak v2.” We compared these samples with each other and with samples compiled by us from the original source code to understand the effect of each change.

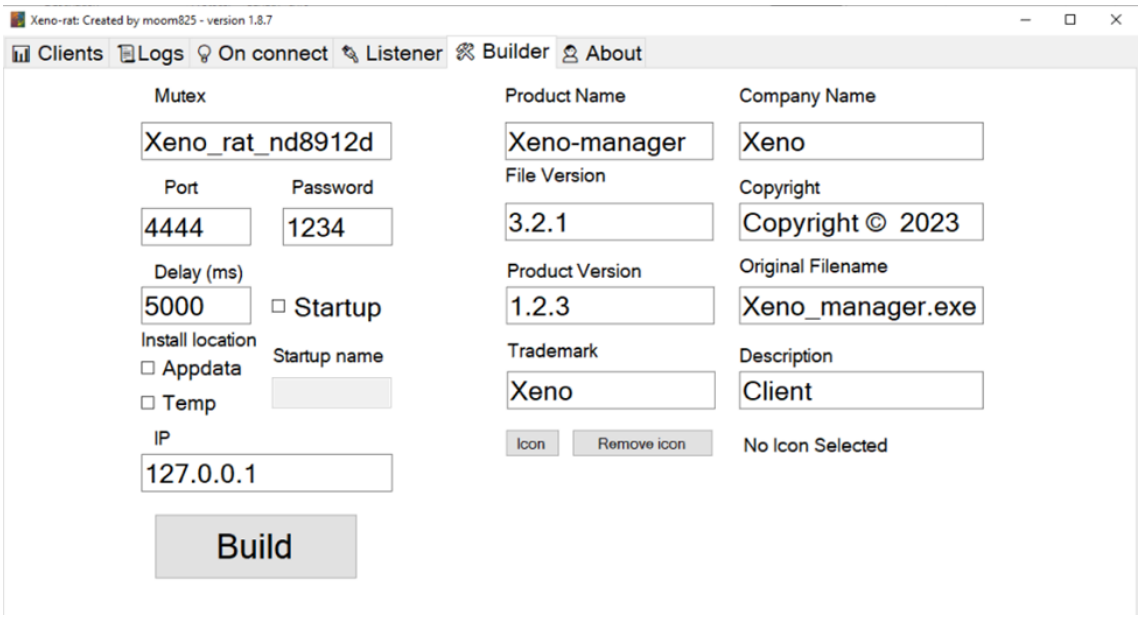
Talos also found a version of the MoonPeak source code which matches the samples built and used by UAT-5394 during the initial stages of their campaign, also reported by Ahnlab, as we will demonstrate later.

Understanding XenoRAT architecture

Before diving into the changes implemented in MoonPeak, a quick explanation about how XenoRAT works is necessary to understand the reason for some changes adopted by the threat actors. The original source code released by the developer contains a Visual Studio project including the code for a RAT client stub, the main server component and additional plugins the C2 server can deploy to the RAT.



The server can build a new RAT client based on the stub, modifying features like the C2 server address and port, mutex name, startup settings and the generated malware sample PE's version properties:



For the implant to communicate back to this server, it needs to provide a proper encryption password and authentication string, and use the right compression protocol, which is defined in the server and client source code and need to match each other. A mismatch in settings will prevent a client/RAT/implant from effectively communicating with the C2 server. Therefore, all XenoRAT samples may not be compatible with all C2 servers if there is a settings mismatch.

The server can also deliver additional plugins to be executed by the client, but these plugins depend on code present in the client to properly execute. This can be seen in a plugin source code which includes the client namespace as necessary before executing.

```
using Chat;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using xeno_rat_client;


namespace Plugin
{
    0 references
    public class Main
    {
        0 references
        public async Task Run(Node node)
        {
            await node.SendAsync(new byte[] { 3 });//indicate that it has connected
            Application.Run(new ChatForm(node));
        }
    }
}
```

If the implant uses a different namespace than the one used while compiling the server and plugins, an error will occur every time the server attempts to send a plugin to the client, even if the client provided the correct password and authentication string.

```
09:57:23 AM  Exception has been thrown by the target of an invocation.
09:57:23 AM  Starting Fun dll failed !
09:57:23 AM  Loading Fun dll...
```

MoonPeak changes over XenoRAT source code

The first change observed in MoonPeak samples which is consistent across all version 1 and version 2 samples is to change the client namespace to “cmdline” instead of “xeno rat client.” With this simple change, any attempt to connect the MoonPeak implant to the original server code will fail, and vice versa, other variants of Xeno RAT will not work when connected to a MoonPeak server.

This is a typical example of updates made to the RAT by the threat actors to enable tactical evasion. The namespace change prevents rogue implants from connecting to their infrastructure and furthermore prevents their own implants from connecting to out-of-box XenoRAT C2 servers.

Another change consistent among all variants of MoonPeak is the forced use of compression before the encryption in the communication protocol. While the original source code uses a variable to define if compression should be used before or after encryption. The modified code always assumes compression will happen before encryption and the alternative code was removed from the source project.

Original code

```
public async Task<bool> SendAsync(byte[] data)
{
    if (data == null)
    {
        throw new ArgumentNullException(nameof(data), "data can not be null!");
    }

    try
    {
        if (doProtocolUpgrade)
        {
            byte[] compressedData = Compression.Compress(data);
            byte didCompress = 0;
            int orgLen = data.Length;
```

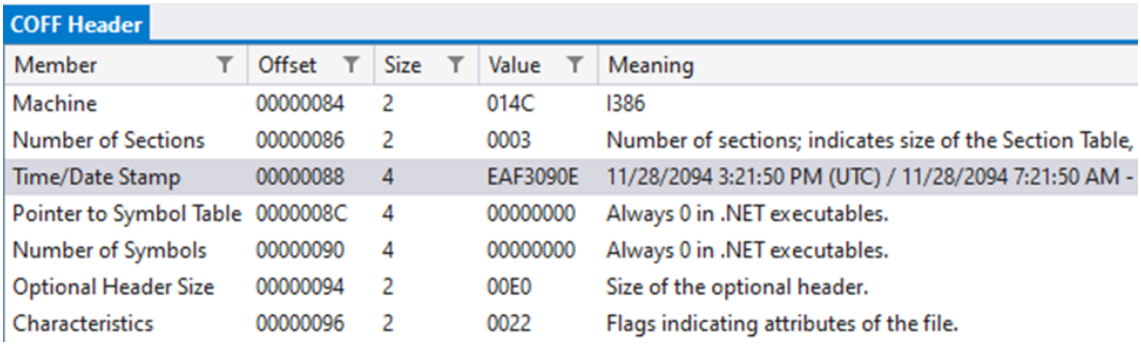
Modified code

```
public async Task<bool> SendAsync(byte[] data)
{
    if (data == null)
    {
        throw new ArgumentNullException(nameof(data), "data can not be null!");
    }

    try
    {
        data = Encryption.Encrypt(data, ekkkk);
        byte[] compressedData = Compression.Compress(data);
        byte didCompress = 0;
        int orgLen = data.Length;
```

These changes were introduced in MoonPeak source code early in the development, sometime around January - February 2024, since all versions of MoonPeak contain this modification.

While we attempted to create a timeline of when these samples were created, we noticed the samples for MoonPeak v1 contained invalid timestamps in their “compiled date” field.



COFF Header

Member	Offset	Size	Value	Meaning
Machine	00000084	2	014C	I386
Number of Sections	00000086	2	0003	Number of sections; indicates size of the Section Table,
Time/Date Stamp	00000088	4	EAF3090E	11/28/2094 3:21:50 PM (UTC) / 11/28/2094 7:21:50 AM -
Pointer to Symbol Table	0000008C	4	00000000	Always 0 in .NET executables.
Number of Symbols	00000090	4	00000000	Always 0 in .NET executables.
Optional Header Size	00000094	2	00E0	Size of the optional header.
Characteristics	00000096	2	0022	Flags indicating attributes of the file.

According to Microsoft, this artifact is caused by the samples being compiled with the "/deterministic" parameter, included by default in the XenoRAT source code, which uses the timestamp field to store a hash of all the options used during compilation. That means we had to use the dates in which these samples were submitted to VT to have an idea of when they were compiled. All the samples for version 1 were compiled with this parameter which is the default setting in Visual Studio. The samples for version 2, however, had valid timestamps which indicate they were compiled without the option, a change made in the development environment by UAT-5394. The table below shows a summary of these dates.

Hash	Version	Creation Date
facf3b40a2b99cc15eee7b7aee3b36a57f0951cda45931fcde311c0cc21cdc71	MoonPeak_V1	2/28/24 5:42 AM
b8233fe9e903ca08b9b1836fe6197e7d3e98e36b13815d8662de09832367a98a	MoonPeak_V1	3/1/24 5:20 AM
44e492d5b9c48c1df7ef5e0fe9a732f271234219d8377cf909a431a386759555	MoonPeak_V1	3/1/24 5:35 AM
97ba8d30cf8393c39f61f7e63266914ecafd07bd49911370afb866399446f37d	MoonPeak_V1	3/2/24 5:31 AM
0b8897103135d92b89a83093f00d1da845a1eae63da7b57f638bab48a779808e	MoonPeak_V1	5/17/24 9:30 PM
148c69a7a1e06dc06e52db5c3f5895de6adc3d79498bc3ccc2cbd8fdf28b2070	MoonPeak_V2	7/2/2024 3:55:41 AM
1ad43ddfce147c1ec71b37011d522c11999a974811fead11fee6761ceb920b10	MoonPeak_V2	7/2/2024 2:49:59 AM
458641936e2b41c425161a9b892d2aa08d1de2bc0db446f214b5f87a6a506432	MoonPeak_V2	7/2/2024 3:39:03 AM
8a4fbcdec5c08e6324e3142f8b8c41da5b8e714b9398c425c47189f17a51d07b	MoonPeak_V2	7/2/2024 6:06:17 AM
293b1a7e923be0f554ec44c87c0981c9b5cf0f20c3ad89d767f366afb0c1f24a	MoonPeak_V2	7/16/2024 2:23:22 AM

We can see the group of samples were created close to each other, except 0b8897103135d92b89a83093f00d1da845a1eae63da7b57f638bab48a779808e, compiled around May 2024. This sample shares the same C2 server (159[.]100[.]29[.]122) with MoonPeak v1, albeit on a different port (8811). This is an interesting bridge between the two sample sets, as it starts to show changes to the code base present in future variants indicating the threat actor actively updated the RAT code.

The first characteristic of this sample is that it does not contain code to authenticate to the C2 server. This code was intentionally removed from the client/RAT, leaving only the code to connect to the server port and immediately quit. Since this change would prevent the RAT from working as expected, we assess this version was intended to be a test sample used to test code changes before deploying actual malicious samples.

```
public static async Task Main()
{
    bool createdNew;
    using (new Mutex(initiallyOwned: true, "swolf-0410", out createdNew))
    {
        if (!createdNew)
        {
            return;
        }
        while (true)
        {
            try
            {
                await new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp).ConnectAsync
                (ssssiiiii, ssppp);
            }
            catch (Exception e)
            {
                await Task.Delay(10000);
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

This version also introduces a change only observed in MoonPeak v2, which is the change to the class name used to store utility functions used by the RAT. This class is called “Utils” in the original source code and all other MoonPeak v1 samples, and changed to “Tools” in this sample and all subsequent MoonPeak v2 samples.

From this point on, other changes were introduced in MoonPeak v2 samples to improve the obfuscation and thwart analysis.

Hash	Version	Uses StrObf	Uses ClassObf	Cre D
1ad43ddfce147c1ec71b37011d522c11999a974811fead11fee6761ceb920b10	MoonPeak_V2	no	yes	7/2/2 2:49 AM
458641936e2b41c425161a9b892d2aa08d1de2bc0db446f214b5f87a6a506432	MoonPeak_V2	no	yes	7/2/2 3:39 AM
148c69a7a1e06dc06e52db5c3f5895de6adc3d79498bc3ccc2cbd8fdf28b2070	MoonPeak_V2	yes	yes	7/2/2 3:55 AM
8a4fbcdec5c08e6324e3142f8b8c41da5b8e714b9398c425c47189f17a51d07b	MoonPeak_V2	yes	yes	7/2/2 6:06 AM
293b1a7e923be0f554ec44c87c0981c9b5cf0f20c3ad89d767f366afb0c1f24a	MoonPeak_V2	yes	yes	7/16 2:23 AM

These samples introduced the use of State Machines in MoonPeak. State Machines can be used to asynchronously perform tasks, and the original function now delegates the actual implementation to a member method of the State Machine. .NET analysis tools such as dnSpy will recognize the State Machine but hide the actual implementation of the function being delegated to the State Machine, making reversing of the malware a more cumbersome task. It is likely that the use of State Machines is a tactic used by UAT-5394 to prevent or thwart analysis attempts of the malware.

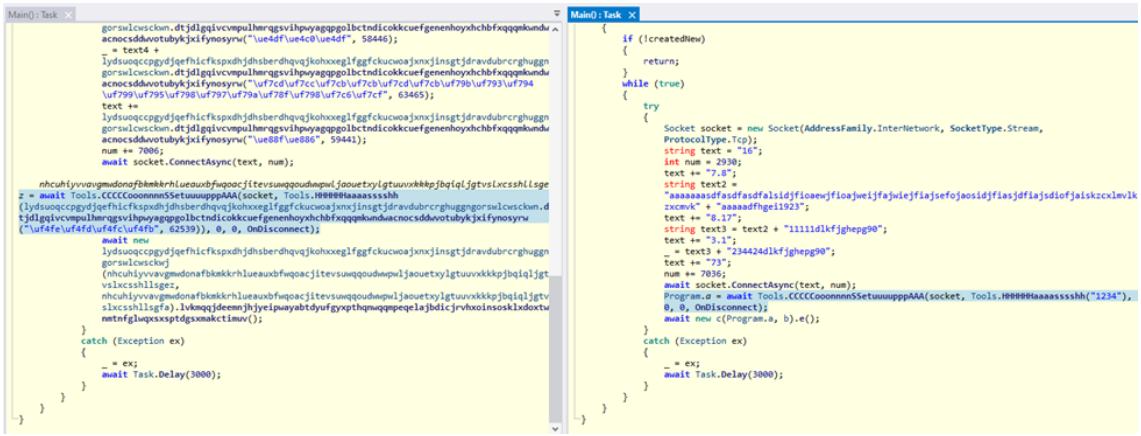
ILSpy, however, allows an analyst to view the State Machine and the actual implementation of code.

```
[AsyncStateMachine(typeof(dwgqu))]
public Task<Node> consubso(int type, int retid, Action<Node> OnDisconnect = null)
{
    dwgqu stateMachine = default(dwgqu);
    stateMachine.xcgoi = AsyncTaskMethodBuilder<Node>.Create();
    stateMachine.xcgoj = this;
    stateMachine.xcgok = type;
    stateMachine.xcgom = retid;
    stateMachine.xcgol = OnDisconnect;
    stateMachine.xcgoh = -1;
    stateMachine.xcgoi.Start(ref stateMachine);
    return stateMachine.xcgoi.Task;
}
```

The actual implementation in the State Machine above executed the original code present in the called class.

```
switch (num)
{
default:
    xcgon = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    awaiter3 = xcgon.ConnectAsync(node.sock.sock.RemoteEndPoint).GetAwaiter();
    if (!awaiter3.IsCompleted)
    {
        num = (xcgoh = 0);
        xcgop = awaiter3;
        xcgoi.AwaitUnsafeOnCompleted(ref awaiter3, ref this);
        return;
    }
    goto IL_00b0;
```

The number of functions changed and complexity of the class names also increased with each variant above. We can observe these changes were introduced in a short span of time for the samples we found. An additional change added later was the obfuscation of strings used in the code. The obfuscation is a simple AES encryption with the key present in a DotNet resource object and disguised as a Unicode string.



This change makes the analysis more demanding, as now it is necessary to decrypt the strings before extracting information like the C2 IP and port.

MoonPeak source code

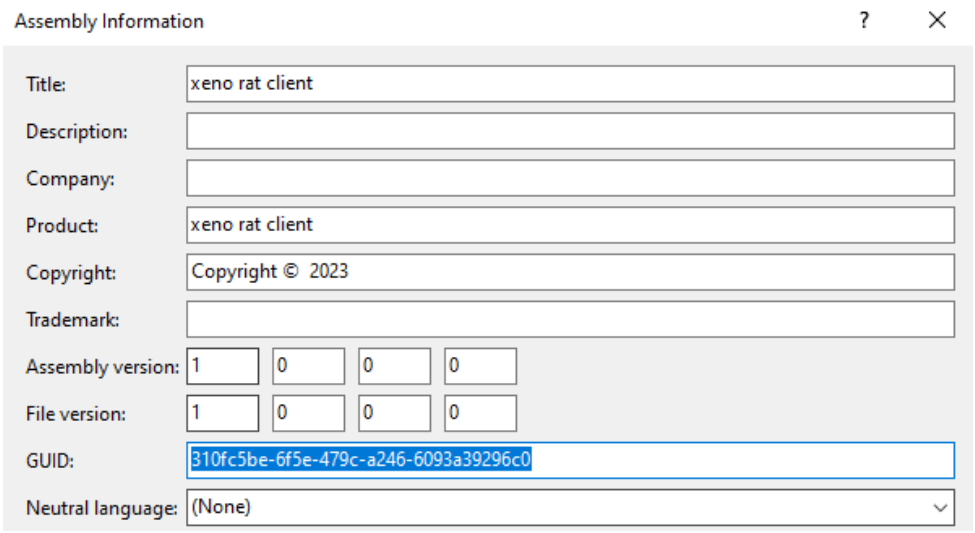
During our investigation, we came across a ZIP file with a copy of the source code for XenoRAT, but upon closer inspection, we noticed the source code had many characteristics like the MoonPeak v1 samples we found before.

The first thing we noticed was the change of the client namespace to “**cmdline**” which we explained before was done to prevent unwanted clients from communicating with their server, and vice-versa.

The project folder also contained several pre-compiled binaries in both their debug and release versions. Even though the source code itself had the C2 hardcoded to the localhost IP 127[.]0[.]0[.]1, the compiled binaries all had the C2 configured to the IP used by MoonPeak v1:

- private static string ssssiiii = "**159[.]100[.]29[.]122**";

Another similarity was present in the assembly information page inside the project properties. This page contains details about the project name, description, company, copyright information and assembly GUID. As an example, this is what the original XenoRAT source code contains:



On the other hand, all the MoonPeak v1 samples as well as the project inside the forked source code use a different GUID.

```
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyTitle("cmdline")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("cmdline")]
[assembly: AssemblyCopyright("Copyright ©  2023")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: Guid("581aa3bd-054b-4e6a-bfc0-e7dc10cfa452")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: TargetFramework(".NETFramework,Version=v4.8", FrameworkDisplayName = ".NET Framework 4.8")]
[assembly: AssemblyVersion("1.0.0.0")]
```

We can see the change in the Assembly Title also reflects the change made to the client namespace. The MoonPeak v2 samples, however, use different GUID and assembly details than previous samples.

```
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyTitle("Kinder")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Kinder")]
[assembly: AssemblyCopyright("Copyright ©  2024")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: Guid("00000000-1111-2222-3333-555555555555")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: TargetFramework(".NETFramework,Version=v4.8", FrameworkDisplayName = ".NET Framework 4.8")]
[assembly: AssemblyVersion("1.0.0.0")]
```

The timelines of the consistent adoption of new malware and its evolution such as in the case of MoonPeak highlights that UAT-5394 continues to add and enhance more tooling into their arsenal. The rapid pace of establishing new supporting infrastructure by UAT-5394 indicates that the group is aiming to rapidly proliferate this campaign and set up more drop points and C2 servers.

IOCs

IOCs for this research can also be found at our GitHub repository here.

MoonPeak v1

0b8897103135d92b89a83093f00d1da845a1eae63da7b57f638bab48a779808e

2b35ef3080dcc13e2d907f681443f3fc3eda832ae66b0458ca5c97050f849306

4108c5096a62c0a6664eed781c39bb042eb0adf166fcc5d64d7c89139d525d4f

44e492d5b9c48c1df7ef5e0fe9a732f271234219d8377cf909a431a386759555

4599a9421e83fb0e2c005e5d9ac171305192beabe965f3385accaf2647be3e8e

58fdc1b6ce4744d6331f8e2efc4652d754e803cae4cc16101fc78438184995e6

97ba8d30cf8393c39f61f7e63266914ecafd07bd49911370afb866399446f37d

a80a35649f638049244a06dd4fb6eca4de0757ef566bfbe1affe1c8bf1d96b04

b8233fe9e903ca08b9b1836fe6197e7d3e98e36b13815d8662de09832367a98a

f4aa4c6942a87087530494cba770a1dcbc263514d874f12ba93a64b1edbae21c

facf3b40a2b99cc15eee7b7aee3b36a57f0951cda45931fcde311c0cc21cdc71

0ed643a30a82daacecfec946031143b962f693104bcb7087ec6bda09ade0f3cb

41d4f7734fbf14ebcdf63f51093718fd5a22ec38a297c0dc3d7704a3fb48b3f9

6a3839788c0dafe591718a3fb6316d12ccd8e82dbcb41ce40e66b743f2dd344d

MoonPeak v2

148c69a7a1e06dc06e52db5c3f5895de6adc3d79498bc3ccc2cbd8fdf28b2070

1ad43ddfce147c1ec71b37011d522c11999a974811fead11fee6761ceb920b10

458641936e2b41c425161a9b892d2aa08d1de2bc0db446f214b5f87a6a506432

8a4fbcdec5c08e6324e3142f8b8c41da5b8e714b9398c425c47189f17a51d07b

293b1a7e923be0f554ec44c87c0981c9b5cf0f20c3ad89d767f366afb0c1f24a

PowerShell scripts

6bf8a19deb443bde013678f3ff83ab9db4ddc47838cd9d00935888e00b30cee6

72a25d959d12e3efe9604aee4b1e7e4db1ef590848d207007419838ddbad5e3f

15eee641978ac318dabc397d9c39fb4cb8e1a854883d8c2401f6f04845a79b4b

3e39fc595db9db1706828b0791161440dc1571eaa07b523df9b721ad65e2369b

f928a0887cf3319a74c90c0bdf63b5f79710e9f9e2f769038ec9969fcc8ee329

27202534cc03a398308475146f6710b790aa31361931d4fe1b495c31c3ed54f7

Network IOCs

167[.]88[.]173[.]173

95[.]164[.]86[.]148

80[.]71[.]157[.]55

84[.]247[.]179[.]77

45[.]87[.]153[.]79

45[.]95[.]11[.]52

104[.]194[.]152[.]251

yoiroyse[.]store

pumaria[.]store

27[.]255[.]81[.]118

212[.]224[.]107[.]244

27[.]255[.]80[.]162

nmailhostserver[.]store

210[.]92[.]18[.]169

91[.]194[.]161[.]109

nsonlines[.]store