Unpacking .NET Malware With Process Hacker and Dnspy

embee-research.ghost.io/unpacking-net-malware-with-process-hacker/

Matthew

October 30, 2023

Last updated on Oct 30, 2023

Process Hacker [DESKTOP-TLPNL Hacker View Tools Users Help	Process Hacker [DESKTOP-TLPNUG\Lenny]+ (Administrator) – C × Hacker View Tools Users Help							
Unpacking .NET Malware With Process Hacker and Dnspy								
svchost.exe	6588			1.2 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser		
svchost.exe	5896			1.59 MB	NT AU\LOCAL SERVICE	Host Process for Windows Ser		
svchost.exe	6904			5.78 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser		
svchost.exe	6740			1.43 MB	NT AU\LOCAL SERVICE	Host Process for Windows Ser		
svchost.exe	5992			7.68 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser		
Isass.exe	668			5.29 MB	NT AUTHORITY\SYSTEM	Local Security Authority Process		
fontdrvhost.exe	792		500.01	1.29 MB	Font Driver Host\UMFD-(Usermode Font Driver Host		
Csrss.exe	528	0.30	528 B/s	2.19 MB	NT AUTHORITY\SYSTEM	Client Server Runtime Process		
V winlogon.exe	616			2.48 MB	NT AUTHORITY/SYSTEM	Windows Logon Application		
fontdrvnost.exe	300	4 20		10.38 MB	Font Driver Host\UMFD-	Usermode Font Driver Host		
MicrosoftEdgeLindate.eve	4506	4.20		1 0 MP	NT ALITHOPITASYSTEM	Misrosoft Edge Llodate		
Coogle Indate eve	1080			2 24 MR	NT AUTHORITY/SYSTEM	Google Installer		
	4428	0.40		128.59 MB	DESKTOP-TLPNUG\Lenny	Windows Explorer		
vm vmtoolsd.exe	5292	0.15	684 B/s	39.06 MB	DESKTOP-TLPNUG\Lenny	VMware Tools Core Service		
PE-bear.exe	6328	0.03		64.91 MB	DESKTOP-TLPNIJG\Lenny			
✓ 🔤 cmd.exe	7344			3.68 MB	DESKTOP-TLPNIJG\Lenny	Windows Command Processor		
conhost.exe	8568			2.38 MB	DESKTOP-TLPNIJG\Lenny	Console Window Host		
ProcessHacker.exe	4260	1.72		22.32 MB	DESKTOP-TLPNIJG\Lenny	Process Hacker		
Die die.exe	5688			29.87 MB	DESKTOP-TLPNIJG\Lenny			
dnSpy.exe	7140			432.85 MB	DESKTOP-TLPNIJG\Lenny	dnSpy		
doSpy eve	8680	8 37		433.69 MR	DESKTOP-TLPNUG Lenny	dnSpy		
aspnet_compiler.exe	1608			16.56 MB	DESKTOP-TLPNIJG\Lenny	aspnet_compiler.exe	v	
CPU Usage: 21.27% Physical memory	y: 3.28 GB (3	3.56%)	Processes: 12	8				

Unpacking malware can be a tedious task. Often involving intensive static analysis and indepth knowledge of debugging.

In this post, I'll demonstrate an easy method that can be used to unpack files that ultimately load a .NET based malware.

This method primarily involves running the file and monitoring for process executions using Process Hacker. Upon execution, Process Hacker can be used to observe any .NET files loaded into memory. If a file is identified, it can then be obtained using Dnspy.

Link to the File

Sha256: 05c2195aa671d62b3b47ff42630db25f39453375de9cffa92fc4a67fa5b6493b

Malware Bazaar

Analysis

I will begin analysis by saving the file into my virtual machine and unzipping it with the password infected.

After unzipping, I will also create a copy of the file with a shorter filename.

Name	Date modified	Туре	Size
∞ ∎ 05c.exe	28/10/2023 3:03 AM	Application	574 KB
05c2195aa671d62b3b47ff42630db25f394533	28/10/2023 3:03 AM	Application	574 KB

I will also perform a basic initial assessment using Detect-it-easy.

Initial Assessment with Detect-it-easy

My primary goal here is to review the entropy graph. Here I can determine if there are any high-entropy areas large enough to store a file.

In this case, there is such an area (as seen in below screenshot). This area suggests that the file could be a loader (as it contains a possible encrypted payload).



Initial Assessment With DnSpy

Before attempting to unpack the file, I will also open it within DnSpy.

This is to make sure that the file is not already unpacked. In my initial assessment, I didn't see any functionality that suggested the file was already unpacked.



Observing Unpacked Content With Process Hacker

At this point, I want to run the file and attempt to let it unpack itself.

This can be achieved by running the file for a few seconds, and observing the process as well as any new processes that are spawned.

After a few seconds have passed, we can go ahead and view the process to see if any new .NET modules have been loaded.

Running the file for a few seconds, we can see that it spawns <u>aspnet_compiler.exe</u>. This is suspicious and something we can hone in on.

■ [®] dnSpy.exe	/140			332.14 MB	DESKIOP-ILPNIJG\Lenny	dnSpy
✓ 🗾 05c.exe	6784			12.82 MB	DESKTOP-TLPNIJG\Lenny	utilityyy
aspnet_compiler.exe	1608			16.53 MB	DESKTOP-TLPNIJG\Lenny	aspnet_compiler.exe
൙ ExpressVPNNotificationServi	5632	0.08	68 B/s	41.3 MB	DESKTOP-TLPNIJG\Lenny	ExpressVPN
🗸 🛃 javaw.exe	8972	1.21		1.39 GB	DESKTOP-TLPNIJG\Lenny	OpenJDK Platform binary

We can also observe that after the new process is spawned, the original process 05c.exe exits a few seconds later.

This is an indicator that any suspicious or unpacked content is likely contained within aspnet_compiler.exe.

		0.74	30.76 MB DESKTOP-TLPNUG\Lenny
🗊 dnSpy.exe	7140		328.91 MB DESKTOP-TLPNIJG\Lenny dnSpy
ExpressVPNNotificationSe	ervi 5632	0.02	41.43 MB DESKTOP-TLPNIJG\Lenny ExpressVPN
aspnet_compiler.exe	1608		16.53 MB DESKTOP-TLPNIJG\Lenny aspnet_compiler.exe

Identifying Unpacked .NET Files Using Process Hacker

With the suspicious <u>aspnet_compiler.exe</u> identified, we can go ahead and inspect it using Process Hacker.

We can do this by double clicking on the process name, or right-clicking and selecting "Properties".

This will open a window like the following. There are two main points here.

- .NET Assemblies tab This shows us that some kind of .NET module is loaded into the process.
- Image Type 32bit The process is 32-bit, this tells us that any future debugging will require a 32-bit debugger (eg Dnspy x86)
- (Verified) Microsoft Corporation This is likely a legitimate process that has been hijacked.

aspnet_compiler.exe (1608) Properties General Statistics Performance Threads Token Modules Memory Environment Handle INET assemblies INET performance SPU Disk and Network Comment File aspnet_compiler.exe (Verified) Microsoft Corporation Version: 4.8.4084.0	X	
Image file name: C:\Windows\Wicrosoft NET\Framework\v4.0.30319\aspnet_compiler.exe		
ev fundous hier executes a name unit/Langeort's frabilite "combiner reve		
Process		
Command line: C. (windows/wild usoricive) (rranework/v4.0.50519/applie_complier.exe		
Started: 3 minutes and 44 seconds ago (11:35:34 PM 29/10/2023)		
PEB address: 0x248000 (32-bit: 0x249000)	Image type: 32-bit	
Parent: Non-existent process (6784)		í.
Mitigation policies: None	Details	
Protection: None	Permissions Terminate	
	Close	ń

Inspecting Loaded .NET Modules With Process Hacker

We can go ahead an inspect any loaded modules with the .NET assemblies tab.

This will list any loaded .NET modules within the current process. As well as information for each module. We want to look for loaded modules that look out of place, or different to the others.

In this case, there is a loaded module named vik that doesn't look right. It has a completely different style of name to the other modules, and doesn't have a corresponding native image path (like all the other modules)



If we look closer, we can also see that the "regular" path is that of aspnet_compiler.exe. This is suspicious, why would aspnet_compiler be named vik?

aspnet_compiler.exe (1608) Properties			X
General Statistics Performance Threads To	oken Modules Memory Environme	t Handles .NET assemblies .NET performance GPU Disk and Network Comment	
Structure V CLR v4.0.30319.0	ID Flags 7 CONCURRENT_GC, 4877006 Default Executable	Path "C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_compiler.exe"	Native image path
AppDomain: apine_compile.exe System.Configuration System.Core Optem.Mini Vik Vik Victoreation System.Core System.Core System.Core System.Core System.Core System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Configuration System.Core System.System.Core System.System.Core System.Core	48/7096 Default, Executable 51502120 Native 5150384 Native 5211128 Native 5092192 19479 Shared 5061960 DomainNeutral, Native	C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System\v4.0_4.0.0_b77a5c561934e089\Sys. C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Cor(H0.4.0.0_b77a5c561934e08. C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Cor(H0.4.0.0_b77a5c561934e08. C:\Windows\Microsoft.NetT\systembly\GAC_MSIL\System.Cor(H0.1.0_b77a5c561934e08. C:\Windows\Microsoft.NetT\systembly\GAC_32\mscorlib\v4.0_4.0.0_b77a5c561934e08. C:\Windows\Microsoft.Net\assembly\GAC_32\mscorlib\v4.0_4.0.0_b77a5c561934e08.	 C:\Windows\assembly\WativeImages_v4.0.30319_32\System\0438d3cafb68664b5 C:\Windows\assembly\WativeImages_v4.0.30319_32\System.Configuration\e810b C:\Windows\assembly\WativeImages_v4.0.30319_32\System.Configuration\e810b C:\Windows\assembly\WativeImages_v4.0.30319_32\System.Xml\229b90c088ea1 C:\Windows\assembly\WativeImages_v4.0.30319_32\System.Xml\229b90c088ea1 C:\Windows\assembly\WativeImages_v4.0.30319_32\system.Xml\239b73a8c4c5dd
	Why would aspne	t_compiler be named "vik"?	

Verifying Suspicious .NET Modules Using DnSpy

Now that we have identified a suspicious module, we can go ahead and obtain it using DnSpy.

To obtain the file, we can open up Dnspy (32-bit) and attach to the aspnet_compiler.exe process.

This will allow us to inspect the loaded modules and view their corresponding source code.

Attaching Dnspy To a .NET Process

We can attach to aspnet_compiler.exe using Debug -> Attach To Process -> Aspnet compiler.exe

dnSpy v6.1.8 (32-bit, .NET)		
File Edit View Debug Window	Help 🛛 🕥 💿 🚰 🚰 🛛 C#	ŧ
Assembly Explorer Windows	▶ ı.Ru	intime
▶ 🗗 System.Private 🕨 Start Debuggir	ng F5 1	//
🕨 🗗 System.Private 🕨 Start Without 🛛	Debugging Ctrl+F5 2	//
System.Ling (6 Attach to Proce	ess Ctrl+Alt+P	11
System.Private System Xaml (Attach to Proce	ess (Unity) 5	
▶ ☐ WindowsBase Toggle Breakpo	oint F9 7	us
🕨 🗇 PresentationCo 🎦 Delete All Brea	kpoints Ctrl+Shift+F9 8	us
PresentationFr	akpoints 9	us
$\mathbf{P} \square^{r} dnlib (3.6.0.0) \qquad \qquad$	10	us
System.Runtime (6.0.0.0)	11	
▶	13	us
▶ 🗇 mscorlib (4.0.0.0)	14	us
Attach to Process		×
Search		0
Process ID Title Type Architect aspnet_compiler.exe 1608 CLR v4.0.30319 x86	ture Filename C:\Windows\Microsoft.NET\Framework\v4.0.30319\asş	pnet_co
Refresh Use 64-bit dnSpy to attach to 64-bit processes	Cartach Ca	► ancel

With the process attached, we now want to inspect any loaded modules.

We can do this by opening a "Modules" tab, using Debug -> Windows -> Modules.

o d	nSpy v6.	1.8 (32-l	bit, .N	IET, Debugging)					
File	Edit	View	Deb	oug Window Help	o o 츧 🗳	C#			11
Asse	mbly Exp	olorer		Windows		•]	Breakpoints	Ctrl+Alt+B	
Þ⊡	¹ System	n.Private		Start Debugging		-	Module Breakpoints		1
	System	n.Private				¢.	Exception Settings	Ctrl+Alt+E	e
	l System	n.Linq (6 Privato				₽	Output		5
	System	n.Xaml (н	Break All	Ctrl+Alt+Break	⇔	Watch	•	
≬⊡	Windo	wsBase	•	Stop Debugging	Shift+F5	,	Autos	Ctrl+Alt+V, A	
	Present	tationCo	×	Detach All		[4;]	Locals	Alt+4	;
	^r Present ¹ dnlib (3	tationFr 3.6.0.0)		Terminate All		⋶	Call Stack	Ctrl+Alt+C	
▶₫	dnSpy	(6.4.0.0	Ó		Ctrl+Shift+F5	7 1.	Threads	Ctrl+Alt+H	, c
	System	Runtin	¢ [‡]	Attach to Process	Ctrl+Alt+P		Modules	Ctrl+Alt+U	
	ⁿ Micros	oft.Visu ib (4.0.0	\otimes	Attach to Process (Unity)	l	-ci st	Processes	Ctrl+Alt+Z	s
						₿	Memory	•	A
						8	using System.IO.	Enumeration;	
						9	using System.Net	;	
			ו	Delete All Breakpoints	Ctrl+Shift+F9		using System.Num	merics;	
			6	Disable All Breakpoints		2	using System.Res	sources;	
			₽	Options			using System.Run	ntime;	
						∠4 25	using System.Run using System.Run	itime.Compile itime.Constra	rS ir

With the new Modules tab, we can list the same loaded modules that were observed with Process Hacker.

Interestingly, there is no vik module, but there is an aspnet_compiler.exe module that we know was associated with vik.

100 %	•										► I
Modules											····· ×
Process	All	- ¥	Search								•
Nam	ie	Optimized	Dynamic	InMemory	Order	Version	Timestamp	Address	Process	AppDomain	Path
😐 msco						4.8.4645.0 built by: NET48REL1LAST_B	6/18/2023 1:54:24 PM	724A0000-738F8000	[0x648] aspnet_compiler.exe		C:\Windows\Microsoft.Net\ass
🗗 aspn		Yes				4.8.4084.0 built by: NET48REL1	11/24/2019 12:24:33 AM	00400000-00412000	[0x648] aspnet_compiler.exe		C:\Windows\Microsoft.NET\Fr
💾 Syste		Yes				4.8.4650.0 built by: NET48REL1LAST_C	6/21/2023 3:22:50 PM		[0x648] aspnet_compiler.exe		C:\Windows\Microsoft.Net\as:
😐 Syste						4.8.4644.0 built by: NET48REL1LAST_B	5/3/2023 7:35:14 PM	71190000-719A8000	[0x648] aspnet_compiler.exe		C:\Windows\Microsoft.Net\as:
😬 Syste		Yes				4.8.4190.0 built by: NET48REL1LAST_B	6/4/2020 8:49:48 PM		[0x648] aspnet_compiler.exe		C:\Windows\Microsoft.Net\as:
😬 Syste						4.8.4084.0 built by: NET48REL1	11/24/2019 12:24:14 AM		[0x648] aspnet_compiler.exe		C:\Windows\Microsoft.Net\ass
•											

By clicking on aspnet_compiler.exe, and selecting "Go To Module", we can view the module contents and corresponding decompiled code.



However, this will open the original aspnet_compiler.exe file from disk and not from within memory.

Hence, the "real" file will be loaded and we won't see anything suspicious.



Instead, we can go back and re-open the file from memory.



With the file opened "from memory", we can obtain the real suspicious content. Which has likely been used to overwrite the original file in memory.

Here we can see the vik module loaded into DnSpy.



Jumping to the Entry Point of .NET Malware

To inspect the vik file more closely, we can righ-click on vik and select "Go To Entry Point".

This will take us to the beginning of the code. Which very closely resembles that of Asyncrat.



Clicking on the Settings.InitializeSettings() method, we can see where the configuration values are decrypted and loaded into the file.



Identifying the Malware With Google

If you haven't seen Asyncrat before, you could instead take some of the values in the "unpacked" sample and google them.

If the malware is known and there are existing reports, you will likely encounter reports that will suggest which family the malware belongs to.

You may have to experiment with which values to google, some return better results than others. Below we can see Asyncrat comes up straight away when googling Settings.InitializeSettings Malware

Settings.InitializeSettings malware	× 🔱 🙃 🤇	
Images Videos News Shopping Books Maps	s Flights Finance	
About 1,360 results (0.40 seconds)		
Did you mean: Settings. <i>Initialize Settings</i> malware		
Qualys Security Blog https://blog.qualys.com > 2022/08/16 > asyncrat-c2-fr		
AsyncRAT C2 Framework: Overview, Technical A	Analysis	
	function enables all	
16 Aug 2022 — Initialize Settings Function. The Initialize Settings f hardcoded malware [.] com. Version, 1.5. Install, False. MTX (Mute	·x)	
16 Aug 2022 — Initialize Settings Function. The Initialize Settings f hardcoded malware[.] com. Version, 1.5. Install, False. MTX (Mute People also ask	**)	•
16 Aug 2022 — Initialize Settings Function. The Initialize Settings f hardcoded malware[.] com. Version, 1.5. Install, False. MTX (Mute People also ask : What is AsyncRAT malware?	·X) 	•
16 Aug 2022 — Initialize Settings Function. The Initialize Settings function. The Initialize Settings function and the setting of the setting	·*>) ··································	►.
16 Aug 2022 — Initialize Settings Function. The Initial Settings function. The Initialize Settings function.	•**) • • • Feedback	►.
16 Aug 2022 — Initialize Settings Function. The Initialize Settings f hardcoded malware[.] com. Version, 1.5. Install, False. MTX (Mute People also ask : What is AsyncRAT malware? What is the use of AsyncRAT?	•**) • • • Feedback	►
16 Aug 2022 — Initialize Settings Function. The Initialize Settings for hardcoded malware[.] com. Version, 1.5. Install, False. MTX (Muter People also ask : What is AsyncRAT malware? What is the use of AsyncRAT?	•**) •** •** •** •** •** •** •** •** •**	×
 16 Aug 2022 — Initialize Settings Function. The Initial Settings function. The Initial Settings	Feedback	k
 16 Aug 2022 — Initialize Settings Function. The Initial Settings function. The Initial Settings	Feedback	•
 16 Aug 2022 — Initialize Settings Function. The Initialize Settings function function function. The Initialize Settings function. The	Feedback	F

Verifying With a Sandbox

With an unpacked module now obtained, you can use DnSpy to save the file for additional analysis.

From here, you can submit the unpacked file to a sandbox or scan it against a set of <u>Yara</u> <u>rules</u>. This is useful if the strings/functions within the file are obfuscated or you aren't able to obtain a good result from google.

This will save the file from memory, so you don't have to worry about saving the "wrong" file



Submitting the File To Hatching Triage

After saving, you can submit the file to an online sandbox like <u>Hatching Triage</u>.

Hatching Triage is correctly able to identify the file as Asyncrat and extract the associated configuration values.

Subill					
Target				Score	
aspnet_asy	/nc.zip		Ĉ	00016	
Filesize				10/10	
22.1kB					
Submitted 30-10-202	3 07:13				
Password infected					
ASY		DEEALUT	DAT		
	NCRAI	DEFAULI			
Malwar	e Config	DEFAULI			^
Malwar tracted	e Config asyncrat	DEFAULI			^
Malwar tracted mily rsion	e Config asyncrat 0.5.7B	DEFAULI			^
Malwar ttracted mily ersion otnet	e Config asyncrat 0.5.7B Default	DEFAULI			
Malwar atracted mily arsion atnet	e Config asyncrat 0.5.7B Default 84.54.50.31:8877	DEFAULI			
Malwar atracted mily rsion tnet autex	e Config asyncrat 0.5.7B Default 84.54.50.31:8877 AsyncMutex_6SI8O	kPnk			
Malwar tracted mily rsion tnet tributes	e Config asyncrat 0.5.7B Default 84.54.50.31:8877 AsyncMutex_6SI800 delay 3	kPnk			
Malwar tracted mily rsion tnet tret	e Config asyncrat 0.5.7B Default 84.54.50.31:8877 AsyncMutex_6SI8O delay 3 install false	kPnk			
Malwar ttracted mily ersion otnet 2 utex tributes	e Config asyncrat 0.5.7B Default 84.54.50.31:8877 AsyncMutex_6SI8O delay 3 install false install_folder %AppData%	kPnk			

Submitting the File to Unpacme

Another option which is effective and significantly cheaper for researchers, is <u>Unpacme</u>.

Unpacme is correctly able to identify the file as Asyncrat and extract all configuration values.

Results

Hunt Q O

Submitted	Sample	Status
30/10/2023 16:14:34	e64cd2a7ce815e4bd86d1348e21dc7823404ce1e4965e3c8032f5c3aa0c59018 Q. aspnet_async.bin 	complete Not Packed
🕀 Insights		-
Classification Packer Threat Type Malware Yara Matches References	Malicious Not Packed RAT C AsyncRAT C UnpacMe: AsyncRAT C UnpacMe: AsyncRAT C Qualys 2 AsyncRAT C2 Framework: Overview, Technical Analysis & Detection Hack Sydney, Michael Elford 2 AsyncRAT: Analysing the Three Stages of Execution	
🏶 Malware C	Malpedia 🖸 Asyncrat	_

Malware Configuration: AsyncRAT Config - Null Extension Param			
Extractor Metadata			Config Data
Extractor	static_asyncrat (9)	Ports	8877
Rule Match	AsyncRAT (4)	Hosts	84.54.50.31
Run Date	30/10/2023 16:14:40 UTC	Version	0.5.7B
		Version	false
Command and Control (C2)		Key	356e69304c494f4862746b514664784e7350564561394e416731556157546342
in	84 54 59 31:8877	Mutex	AsyncMutex_6SI80kPnk
		Certificate	<pre>MIIEBjCCAtqgAwIBAgTQAMuJT3x+opgDgdtz4XxeR2ANBgkqhkiG9w0BAQ0#FADAMHgwFgVDVQQDDA9Bc3luV 1JBVCBTZXJ2ZXIwIBcNMjMwNjESMDUIDTI4WhgPOTkSOTEyMzEyMzUSNTlaMBoxGDAWBgNVBAMMD0FzeWSjUk FUIFNIcnZLcjCcAliwDQV1KoZIhvcNAQEB8QADggIPADCAQcSQcggIBAJX38qxwSkEba8obqYG8WXFFKN18Lvm k0mmfTtRfa9h1ZZumz+X2rcZ8rLvPVDx0VUu0ZS6/1xf3CNe9YBUz79As0xgcmjKFAKamiTunaBh0Z78z0GAp sjT71b7/GyJb1N0qTvcPv7zgXIDtJf0P91D8TDXTdi0xdhU9XUFiG1lv7zqeW4Qj0kTE3K3yzbtToTBvA0EA1 qu8w06AI/CKt4X3q6JjcgSeaVes8murVLC1ahP5E3aTgdqempsBSac2VMA8X11VUS1TJ51kg8J4UUZh3rHSv vmjYd2zpGq+SPLfZhv93MbS59Q/Wra4QJxcgn1/Dgxv/ZUaS7y16Z75tD0T3AUzaYX146BrBWUDifkbJRStRb y/0EP/01MyXmWfFSVC/WD1ygCmpJHLeTLVIYGL17BqvbB0yERX6n1V82s0zGxHpJH+oF5/HCtPoeST3daAncK QWtRVhxZ4nZE12fkWir281chost1xcnVnTBkV/Gv3uo96K/kbAjrh9EQXzuf+htjtbnqHtcRgfYQ5xaLTEt8x SoTbtHHfnV3mZRxxRz3WN0rXT45V+AJ1E/ctU1bS1+DAiv/1d8D01oPGxpzK6ghWMkWAUJ6CoUeY1yXHVqXx8 W8z1v2Y2Hp83c+3d1naEtwnkbSx1XpaG7100hieqZA7qAnETV7TBSW5QNpZDAgWBAAGjMjAwH80GA1UdDgQWBBT qmZ1niC6t3HtQv7msIvV+f3x5gDAPBgNVHRMBAf8BETADAQH/MA0GC5qGS1b3DQEBDQUAAHCAQAG1czK7R9k BF1r5VFHWzyyYt5ZJLFSeWWZ3y76M0MqMGNux5MaZRmvHQU9V1bA3/8k78kE4hcQ1407UT0/GsThBmWAUzuG 3yZNBUJW3X/x1jJayInzNA6AMsD1bGyNzDGt+r8JMGvxTgXQfmKKQstJSyLVNT+ezcsX5fXZi2FJoyyVd3cjp HUjPWDRL1Rd3J6PRNUTUrJEZ04a0GTHuYLB1+a10szyKA51WVM3R+fBHX9c083pkqD+FLNLkg/iPi42rSkoV HpVdG3087j5LiADBTjZNEPHo328fN1EkemRXx1+uH/UEVmhULh73rCP0j4bEj1027RkvBnudXL2B0n+jj/h oSyn5niVTUkLBUP1WMKWs+zqrru9QIHW/pJ2A2JTjFVNjmI1SPP+uoAw00R1WLma3BHiKODwtG0JFkg2PsB ejM12TFIrKn/A76+LjrTPoZkt51S1LXuAyEHgzTPEBQffhPt1Wsw6W91nJ/f2SP7wJ2LkT1ird116t5g8/t 7qiNuuKnH0nH00a3hAfk10T8GpgExVV0CWF6S/3hS5j5D9CWAior1AGLJXPvvQoSF454wTXFQ2NIGIroOL1 U1v5pijkYWI6fgvML3Q5NS+avxyD0K1JombkZXHnz1ao1ap4XPa69F5WbWJx9DCddEQ5hV8jGJ211w==</pre>

Conclusion

In this post, we performed some basic analysis of an Asyncrat loader, and utilised Process Hacker to identify an unpacked payload. We then used Dnspy to obtain the unpacked malware and identify it as Asyncrat.