MetaStealer Part 2, Google Cookie Refresher Madness and Stealer Drama

russianpanda.com/2023/12/28/MetaStealer-Part-2/

Stealer's World of Drama

Previously, I wrote a blog going through some of MetaStealer's functionalities and did a brief comparison with Redline since they are both very similar but, at the same time, different. You might say that all stealers are the same because they have one purpose - to steal. However, each of them is somewhat different from the others, even if they borrowed the code from their predecessors.

Every stealer tries to be better than the other one despite having similar code and functionality. What is considered a good stealer? The stealer has a low detection rate and a high rate of successful infection, or what we call "отстук" in Russian. Stealers such as Redline, Metastealer, Raccoon Stealer, Lumma, RisePro, and Vidar have earned their names in the stealer market. Below is the list of top stealers' whose logs are being sold on RussianMarket.



The popularity of mentioned stealers among users, mainly those developed by native Russian speakers, could be attributed to the ease of communication and support in their native language. As you might have noticed, stealers are notably prevalent among Russianspeaking communities. The ability to interact in one's native language - whether it is to request new features, report issues, or inquire about the functionality of the stealer significantly simplifies the process compared to the effort required for translation into English. This linguistic accessibility potentially broadens the client base, offering the stealer more opportunities to attract additional users.

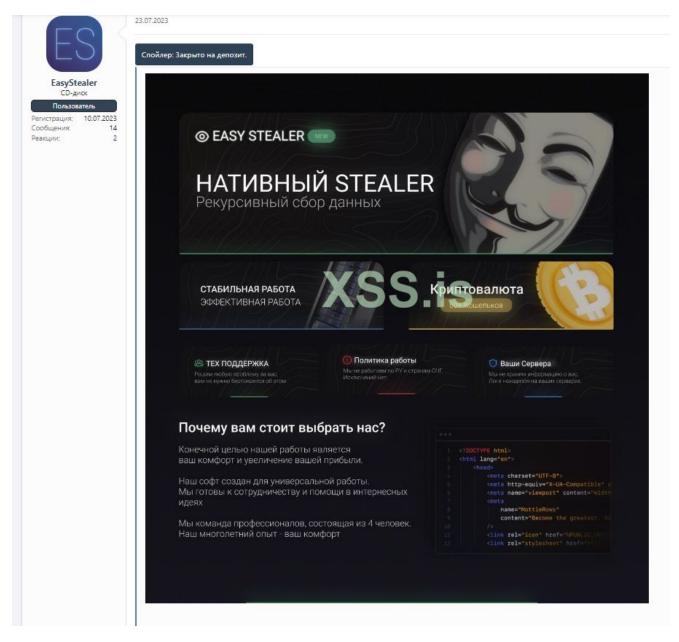
The world of stealers is rife with drama, much like any other corner of the cybercriminal ecosystem. I was recently informed about an incident related to the Santa Barbara topic on XSS forums. This topic was created by one of Lumma's former coders, coinciding with

Lumma's one-year anniversary. To put it briefly, Lumma's founder did not adequately recognize or compensate the coder's contributions, leading to dissatisfaction and underpayment.

	111111 < 1 a
Coder	Hy viro, Lumma, repute to separa mologarame tells or mologarame tells or mologarame tells or mologarame tells or
	В дого различана, сом и повышать общатенного, кат и кательски на цаков дала, кито расти ний А потов коточних, терей цака с неб ука дала, трети бурат лицини. Та ука не дилам по ука пеба бость постоба да на босскато сополний. А потов коточних, терей цака с неб ука дала,
dark_dream H00-drive	This codes become anythin weight-hole mail - topes until calibration above. In the manual part of the mail unservice and we can be mail to the calibration and the calibration calibration and the calibration calibration and the calibration calibration and the calibration
Пользоватиль Регистрация: 26.02.2023	Для тек, кто не понимает, о чем селька кает рень, в - бывший системный каето проекта lumma. Я пришел в проект примерно через 2-3 искоцая после его опоралия, по прообе lumma, и работая там до надавного тремени.
Coodupeers 20 Peacure 47	Мы со вторыи кодерон (он же Lumma, он же теб-кодер), он же основатем) работали на протяжении принерно в месяцея но через «эсования» месяцея после ничала совичстной работы стали происходите событии, которые и прияван к моему укоду из провета. Об этих собитали к рассквар нике,
Meacyor 4/	Что бы несоночно полноги исказить действительность лемной, в сразу прилатаео часть доазательств к номи соваль. Все остальные прубы буду социальть онер запросод, лемь было тратить иного тремяни на этого товиться. Ты почистих диалоги в темете [Limma, но на этор сокрытия в доазательств к номи соваль. Все остальные прубы буду социальть по нара запросод, лемь было тратить иного тремяни на этого товиться. Ты почистих диалоги в темете [Limma, но на этор сокрытия] в соват расстать на поражении и диалого товиться. Ты почистих диалоги в темете [Limma, но на этор сокрытия]
	илизть до последнего дня.
	На экот раз викрупных на получится, но я живо, ты буреаны плиталься, как обычно.
	Неину с предестории. Когда в прицев в проект, Цитта был в сложой ситуации по билдан, на ник висело 12-15 детектов по ческлике и он никак не мог их вынистить. Эта ситуация заставиля его обратиться ко ине. До этого он работал один.
	Export register call papendia distances al logge, d'arrange ey reput aux aux al contencial photos a cautor follo dista, agant resource este contracte distances ou repuesto aux este ou repuesto aux e
	А тенун клончени улад, ок сего А
	1) Очень закняя сообенность, которую следует заполнить и дерхать в толове, читая этот пункт - на протяжении всего времени я не инка доступа к наший рабочей телете, с которой шли продаки, и не энал колько реально у нас кленнов.
	E ague interest ague interpreta quade in discrete/in concepts interpreta paraluma in a foreira concerta foreira give interpreta quade c
	Я наблодал за этих несколько недель, может мескц, и после задал вопроси, на которые получил ответ "бро, у меня другие техни на удоких. П в поднял эту тему повторно, но уже в более агрессивной манере.
	На этот раз оправдением люмим быто у меня кощильки на телефоне радои стоят, пузаю иногда дарека". И так, талько за один месяц мож наблодений он "напута" примерно на 4-5 с язлени.
	Capacity on onecompanies approx in the to exclusive spacing particul approach approach particul approx appr
	in specy and into a priory science in priory science and an and an an and an and an an and and

Another drama story: some of you might know how Aurora Stealer left the market before their infamous botnet release; some users deposited money for the botnet and never got it back, of course. Now, Aurora has become a meme within the stealer's community.

In July 2023, an advertisement was posted on XSS forums for a new stealer written in Golang, known as "EasyStealer", then the rumors started spreading among the stealer's community that this was the work of an Aurora developer, now the stealer is nowhere to be found.



Does all of this impact the sales of stealers? Not at all. People continue to purchase stealers as long as their functionality meets their requirements.

Google Cookie Refresher "feature" or a "0day"

So, you've likely heard about the <u>ongoing</u> Google "0day" vulnerability, which allows attackers to obtain fresh cookies, granting them "indefinite" access to Google accounts. It is a rather convenient "feature," isn't it? However, it is also quite dangerous because an attacker would be able to get fresh cookies to Google accounts each time the old ones expire.



As @g0njxa <u>mentioned</u>, the feature is abused by many stealers, including RisePro, MetaStealer, Whitesnake, StealC, Lumma, Rhadamanthys, and Meduza. Additionally, as of December 29th, Vidar Stealer has implemented this feature.

The question of how long it will take Google to respond to this issue remains unanswered. However, this situation presents even more opportunities for stealers to take advantage of the vulnerability.

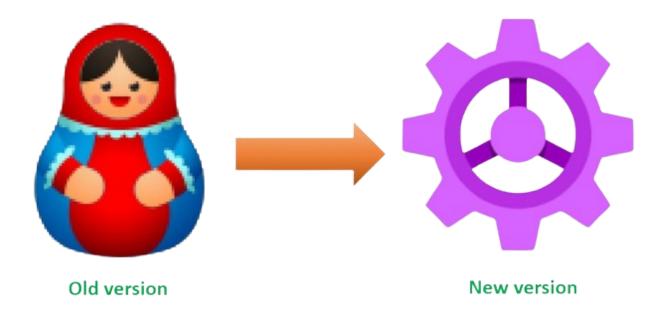
The reason why I brought this up is how easily it can be exploited with just a few lines of Python code that includes the decrypted token value, account ID, and the proper request to the server if some people are curious enough to find out. Although, certain parameters need to be slightly tweaked from the server's response to make it work. Here is my video with proof-of-concept on how it works on a high level. I have created a video demonstrating the proof-of-concept at a high level. For ethical reasons, I will not delve into the technical details of the POC.

MetaStealer Part 2: Technical Analysis

In November 2023, I released the <u>writeup</u> on MetaStealer. However, soon after its release, the malware developer made another update that changed the class names, string encryption algorithm, binary description, and file icon.

MetaStealer new version is approximately 368KB in size with the binary description **Cavils Corp. 2010** (the previous one was **METRO 2022 Dev**).

The logo change:



If previously, MetaStealer used "Entity" for class names; now it's using "Schema" and "TreeObject" to store data and configurations instead of **MSValue**.

GClass1 @02000022	▶
Handler @02000031	OBJECT_NAME_INFORMATION @02000048
Pv4Helper @0200009F	ParametersWithIV @0200002F
IRemoteEndpoint @020000B8	▶ 🖧 PBE @02000075
▶ 🖓 Json @02000038	PBKDF2 @02000076
A ListExtensions @02000046	Proc @02000062
MainFrm @02000A5	▶
Mapping @020000A1	▶ 📰 ProcessFileHandle @02000045
♦ 43 MSObject1 @020000B5	Program @02000081
♦ Ag MSObject10 @020000A8	▶ 📻 RcHdrFd @0200007E
MSObject11 @020000A9	▶ 🍫 Schema1 @02000078
MSObject12 @020000AA	▶ 🥳 Schema10 @02000066
MSObject13 @02000032	▶ 🤹 Schema11 @02000067
MSObject14 @020000AF	Schema12 @02000068
▶ ිද MSObject16 @020000B1	Schema13 @0200001B
◊ Ag MSObject17 @020000B3	▶ 🗗 Schema14 @0200006D
MSObject18 @020000B4	Schema16 @0200006F
MSObject19 @0200001D	Schema17 @02000071
MSObject20 @0200001E	Schema18 @02000077
MSObject21 @02000020	Schema19 @0200006
MSObject23 @02000028	Schema20 @02000007
MSObject24 @02000029	Schema21 @0200000A
MSObject25 @0200028 Old version	Schema23 @02000011 New version
MSObject26 @0200002C	🕨 😪 Schema24 @02000012
MSObject27 @0200002D	🕨 😪 Schema25 @02000014
MSObject28 @0200002E	🕨 쓚 Schema26 @02000015
MSObject29 @0200002F	🕨 😪 Schema27 @02000016
MSObject3 @02000086	Schema28 @02000017
MSObject31 @02000026	Schema29 @02000018
MSObject32 @02000027	Schema3 @02000079
MSObject4 @020000B7	Schema31 @0200000F
MSObject5 @020000B9	Schema32 @02000010
MSObject7 @020000BA	Schema4 @0200007A
MSObject8 @020000A6	Schema5 @0200007C
♦ 93 MSObject9 @020000A7	Schema7 @0200007D
MSObjectReaderSql @02000035	Schema8 @02000063
MSObjectRoot @02000024	Schema9 @02000064
NativeMethods @0200000C	SchemaReaderSql @02000021
NullableExtensions @02000047	SchemaRoot @0200000D
Proc @020000A4	SME @0200007F
Program @020000BE	StringDecrypt @0200003E
RcHdrFd @020000BB	StringExt @02000026
▶ 📰 SME @020000BC	Strings @0200005F
StringDecrypt @02000098	SystemInfoHelper @02000060

Instead of string replacement operations, it now accesses a decrypted string from an array based on the given index. For example, below, where it uses **ManagementObjectSearcher** class to query system management information. The constructor of

ManagementObjectSearcher takes two parameters: a WMI query path and a query string, for example "**ROOT\SecurityCenter: SELECT * FROM AntivirusProduct**".

	237 // Token: 0x060001A0 RID: 416 RVA: 0x000132E4 File Offset: 0x000114E4 238 public static List <string> GetVs()</string>											
		236 public static List(string) GetVs() 239 {										
		240 List <string> list = new List<string>();</string></string>										
		241 try										
		242 { 243 foreach (string text in Strings. <mark>Get(42).Split</mark> (new char[] { ' ' }))										
	24											
		245 try										
		47 using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher(Strings.Get(43), Strings.Get(44) + text)) 48 {										
-		using (#anagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())										
	25	51 foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)										
		253 try										
		<pre>55 string text2 = managementBaseObject[Strings.Get(45)] as string;</pre>										
		56 57 { 19 {										
		1ist.Add(text2);										
		60) 61 catch										
		ol Catton 62 f										
		67 } 68 catch										
10	0 %											
	cals											
4	lame											
-		Strings.Get returned @"ROOT\SecurityCenter" string										
		Strings.Get returned "SELECT * FROM " string										
		string.Concat returned "SELECT * FROM AntivirusProduct" string										
	0											
P		array (string[0x000003]) string[]										
	0											
	0											
		managementObjectSearcher {System.Management.ManagementObjectSearcher} System.Management.Managemet.Managemet.Mangement.Managemet.Managemet.Managemet.Manage										
		management/OpertCollection null System.Management										
P	Ø managementBaseObject null System.Management.Managementa											

The new string decryption algorithm works the following way:

First, the base64-encoded string gets base64-decoded and XOR'ed with the hardcoded key (in our example, it is **Crayfish**); the XOR'ed string then gets base64-decoded again.

9 public sta 10 { 11 // Tok 12 static 13 { 14 St 15 16 St	<pre>Strings() rings.Keys[0] = rings.Keys[1] = rings.Keys[2] =</pre>	gs RID: 405 RVA: 0x00011868 / "Crayfish"; .Bend("GTcM	DgU/MQYRHgU+AywhIBU	INES	uoBaHlGicvTCkxJSAOJAkgYC0pGSEfCg4/WT0uFjYjADcTFFogCFFE", Strings.Keys[0]); gelgRZDCorFjwTQ1E+", Strings.Keys[0]);
Recipe			8 🖬		Input
From Base64		0	11	GTcMDgU/MQVRHgU+AywhIBUFHEso8AMIGGcvTCkx3SA03AkgMC8pGSEFCg4/hff8uFjYj4DcTFFogCFFE	
Alphabet A-Za-z0-9+/=	- 1	Remove non-alphabet chars	Strict mode		
XOR			0	п	
Kev Crayfish	LATIN1 *	Scheme Standard	Null preserving		
From Base64			0	11	
Alphabet A-Za-z0-9+/=	1	Remove non-alphabet chars	Strict mode		Output 2
					jdL0qPgFNFxDGNov6jLaCy9uG10CT6jx10cCEP0rC86s=

- Each XOR'ed and base64-decoded string is assigned as an AES key and IV (Keys[1] and Keys[2]).
- The encrypted byte arrays are then reversed and decrypted using the keys and IV mentioned above

{							
11, 119, 137, 182, 243, 227, 132, 10, 61, 133,							
43, 194, 91, 4, 103, 135, 5, 246, 219, 176,	Recipe					8 🖿	Input Input
165, 16							08778986F3E3840A3D858F3A395C129624124F812BC25884678705F6DB80A51d
<pre>Strings.Array.Add(Strings.Decrypt(new byte[]</pre>	From Hex					0 1	00//0300/323040430030/34535(123024124/01202230040/0/05/000004310
f							
138, 109, 210, 67, 80, 19, 32, 168, 55, 190,	Delimiter Auto						
212, 80, 118, 0, 223, 3	Plato						
	<u> </u>						
<pre>Strings.Array.Add(Strings.Decrypt(new byte[]</pre>	Reverse					0 1	
	By						
206, 24, 20, 184, 152, 31, 12, 81, 176, 169,	Character						
101, 5, 101, 104, 185, 152 }));							
<pre>Strings.Array.Add(Strings.Decrypt(new byte[]</pre>	AES Decrypt					0 1	
{	AES Decrypt					0	nc 64 gr 1
163, 170, 191, 40, 175, 112, 182, 112, 227, 176,	Key		IV		Mode	Input	No. 1. 1
192, 108, 0, 176, 37, 146	+qiw0kCGxX	BASE64 *	ilK+nrSl4T	BASE64 *	CBC/NoPadd	Raw	Output
<pre>Strings.Array.Add(Strings.Decrypt(new byte[]</pre>	Output Raw						5.42.65.34:25538bulbububububububububububububububububu
	naw						
27, 56, 65, 171, 203, 169, 42, 50, 39, 112,							
85, 124, 80, 179, 112, 6 }));							

To save us some time, we can use the dynamic approach to decrypt the strings using **dnlib**. The wonderful approach was detailed by @n1ghtw0lf in this <u>blog</u>. Also, I want to thank <u>@cod3nym</u> for amazing tips when it comes to dealing with .NET shenanigans!

Here are the steps to decrypt the strings:

We will use **dnlib**, a library for reading and writing .NET assemblies to load a .NET module and assembly from a given file path.

```
def load_net_module(file_path):
    return ModuleDefMD.Load(file_path)

def load_net_assembly(file_path):
    return Assembly.LoadFile(file_path)
# Main script
module = load_net_module(file_path)
assembly = load_net_assembly(file_path)
```

We will define the decryption signature (**decryption_signature**) to identify methods that are likely used for decryption. This signature includes the expected parameters and return type of the decryption methods.

```
decryption_signature = [
    {"Parameters": ["System.Int32"], "ReturnType": "System.String"}
]
```

We will search the loaded assembly for methods that match the defined decryption signature.

```
def find_decryption_methods(assembly):
    suspected_methods = []
    flags = BindingFlags.Static | BindingFlags.Public | BindingFlags.NonPublic
    for module_type in assembly.GetTypes():
        for method in module_type.GetMethods(flags):
            for sig in decryption_signature:
                if method_matches_signature(method, sig):
                    suspected_methods.append(method)
    return suspected_methods
```

Finally, we will invoke the suspected decryption methods by scanning the assembly's methods for calls to the suspected decryption methods, extracting the parameters passed to these methods, and invoking the decryption methods with the extracted parameters.

```
def invoke_methods(module, suspected_methods):
    results = {}
    for method in suspected_methods:
        for module_type in module.Types:
            if not module_type.HasMethods:
                continue
            for m in module_type.Methods:
                if m.HasBodv:
                    for insnIdx, insn in enumerate(m.Body.Instructions):
                        if insn.OpCode == OpCodes.Call:
                            called_method_name = str(insn.Operand)
                            if method.Name in called_method_name:
                                params = extract_parameters(m.Body.Instructions,
insnIdx, method)
                                if len(params) == len(method.GetParameters()):
                                    try:
                                        result = invoke_method_safely(method, params)
                                        if result is not None:
                                             location = f"{module_type.FullName}.
{m.Name}"
                                             results[location] = result
                                    except Exception as e:
                                        None
```

return results

We will also include the logic to handle different types of parameters, such as integers and strings. It uses **get_operand_value** to extract values from method instructions based on their type.

```
def get_operand_value(insn, param_type):
    if "Int32" in param_type and insn.IsLdcI4():
        return Int32(insn.GetLdcI4Value())
    elif "String" in param_type and insn.OpCode == OpCodes.Ldstr:
        return insn.Operand
    return None
```

You can access the full script here.

Note: Please run the script strictly in a sandbox environment.

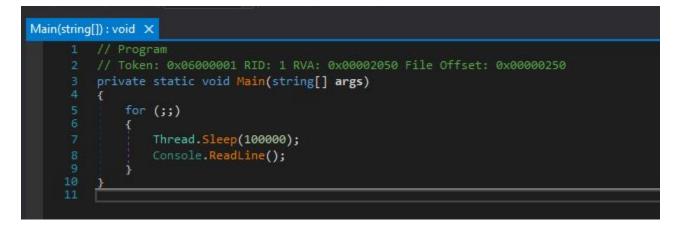
The output of the script (tested on the deobfuscated sample *MD5: e6db93b513085fe253753cff76054a2a*):

Decryption Location: API.CreateFileMapping, Decrypted String: CreateFileMappingA
Decryption Location: APIcctor, Decrypted String: gdi32.dll
Decryption Location: SystemInfoHelper.Add, Decrypted String: TVqQAAMAAAAAAAAA//8AALgAAAAAAAAAAAAAAAAAAAAAAAA
RUANT REDAIL 12 U JANANANANANANANANANANANANANANANANANANAN
алаалаалаалаалаалаалаалаалаалаалаалаала
CAABEAQAAIIN0cmTuZĴMAAĂĂAaAQAAAQAAAAjVVMAbAQAABĂAAAAjRIVĴRAAAAHwEAADQAAAAIOJsbZIAAAĂAAAQAAAUCVAAAJAAAAPOBMwAWAAABAAAAEAAAAAAAAAAAAAAAAA
BgBMAbÖBBgAYAb0BBgAXAb0BBgBrAL0BBgBAA0gBBgCFAL0BBgCGAI8BBgA7ArEBBgAbALEBAAAAAAEAAAAAAAAAAAAQQBAAD0AAQABAFAgAAAAA JEAuAEnAAEAWCA
ABhAM8BFQBpAM8BEABxAM8BEACBACMAGgB5AM8BBgAuAAsALQAuABMANgAuABsAVQAuACMAXgAuACsAawAuADMAawAuADsAawAuAEMAXgAuAEsAcQAuAFMAawAuAFsAawAu
9ybGliAENvbnNvbGUAUmVhZExpbmUARGvidWdnYWJsZUF0dHJpYnV0ZQBDb21WaXNpYmxlQxR0cmlidXRlAEFzc2VtYmx5VGl0bGVBdHRyaWJ1dGUAQXNzZW1ibHlUcmFk2
pb25mawd1cmF0aw9uQXR0cmlidXR1AEFzc2vtYmx5RGvzY3jpcHRpb25BdHRyaWJ1dGUAQ29tcG1sYXRpb25SZWxheGF0aW9uc0F0dHjpYnv0ZQBBc3N1bWjsevByb2R1Y
Ym1saXR5QXR0cm1idXR1AHF1bXUtZ2EuZXh1AFN5c3R1b555dW50aW11L1z1cnnpb25pbmcAUHJvz31hb0BTeXN0ZW0ATWFpbgBTeXN0ZW0uUwVmbGvjdG1vbgAuY3RvcgE
mljzXMARGVidWdnaw5nTw9kZXMAYXJncwBPYmplY3QAAAAAAAAAAAS9VoDs6MQECMfXX7M7EqUwAEIAEBCAMgAAEFIAEBEREEIAEBDgQgAQECAwAADgi3elxwGTTgiQUAAQEc lyaWdodCDCgSAgMjAyMwAADAEABzEuMC4wLjAAAEcBABouTkVURnJhbWV3b3jrLFZlcnNpb249djQuMAEAVA4URnJhbWV3b3jrRGlzcGxheU5hbWUQLk5FVCBGcmFtZXdvc
Tyawaaacaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
ΑQΑΑΛΑΙΑΛΑgBgAAABQAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BAAAAAQAAAĂAĂAAAABAAAAAAA/AAAAAAAAAAAAAA
AAEAAQBDAG8AbQBtAGUAbgB0AHMAAAAAAAAIgABAAEAQwBvAG0AcABhAG4AeQBOAGEAbQBTAAAAAAAAAAAAAAAAAAAAAAAAZAABABCWBJAHIAaQBwAHQAaQBvAG4
G4AdAB]AHIAbgBhAGwATgBhAG0AZQAAAHEAZQBtAHUALQBNAGEALgB]AHgAZQAAAEgAEgABAEwAZQBNAGEAbABDAG8AcAB5AHIAaQBNAGgAdAAAAEMAbwBwAHkAcgBpAGcA
kAZwBpAG4AYQBSAEYAAQBSAGUAbgBhAG0AZQAAAHEAZQBtAHUALQBnAGEALgBlAHgAZQAAADAACAABAFAAcgBvAGQAdQBjAHQATgBhAG0AZQAAAAAAAQBlAG0AdQAtAGCA AIABWAGUAcgBZAGKAbwBuAAAAMQAuADAALgAwAC4AMAAAAKxDAADgAQAAAAAAAAAAAAAAADvu788P3htbCB2ZXJzaw9uPSIxLjAiIGVuY29kaw5nPSJVVEYt0CIgc3RhbmRhbC
ALABWAGGGZ BZZASZAWBUJAAAAMUJAJALGJWWAL+AMAAAAXXXXAAUJAQAAAAAAAAAAAAAVAADUU 6067311052ZXJZZW9U751XLJATIGU129KZWJIF5JVVETUC1953KIIIIMIINIU c21vb101M54Wi14Nc1Ap6Fcz2VtYms55WF1DmRDHkgdMvVc21vb101M54Wi140mFtZT01TX1BeHBszANHAG1Vb15hcHATLZ4Nc1AgPHRydXN05W5mbyB4bWxucz(
2VzIHtbb5zP5J1cm46c2NozW1hcv1taWNvb3NvzNotV2D0tV2D0Fzb552WvI+D0ogICAgICAgICAgICXzXIbC95XZXIbC95XZXIbC95XZXIbC95XXZ1bC95XZXIC92X
5mbz4nC jwvYXnzZwlibHk+ΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛΑΛ
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
~^^^^^^
^^^^^
Decryption Location: SystemInfoHelper.GetProcessors, Decrypted String: NumberOfCores Decryption Location: SystemInfoHelper.GetGraphicCards, Decrypted String: Name
Decryption Location: SystemInfohelper.GetBrowsers, Decrypted String: Unknown
Decryption Location: SystemInforeIper.actionsers, Decrypted String: SerialNumber
Decryption Location: SystemInfoHelper.ListOfProcesses, Decrypted String: CommandLine
Decryption Location: SystemInfoHelper.GetVs, Decrypted String: displayName
Decryption Location: SystemInfoHelper.FindProc, Decrypted String: Name
Decryption Location: SystemInfoHe]per.ListOfPrograms, Decrypted_String: [^\u0020-\u007F]
Decryption Location: SystemInfoHelper.collectMemory, Decrypted String: Mb or
Decryption Location: SystemInfoHelper.GetWindowsVersion, Decrypted String: CSDVersion Decryption Location: FileHelper.Search, Decrypted String: \Program Data\
Decryption Location: FileHelper.ChromederName, Decrypted String: Unknown
Decryption Location: FileHelper.ChrRm, Decrypted String: AppData\Roaming\
Decryption Location: FileHelper.ChrLm, Decrypted String: AppData\Local\
Decryption Location: Asn1DerObject.ToString, Decrypted String: OBJECTIDENTIFIER
Decryption Location: Program.ReadLine, Decrypted String: 5.42.65.34:25530
c2: 5.42.65.34:25530
Build ID: 211220231235

You might have noticed an interesting base64-encoded string in the output above.

Upon decoding, we receive a .NET executable **qemu-ga.exe** (MD5: e6db93b513085fe253753cff76054a2a).

Now, an interesting moment: MetaStealer writes that executable to the Startup after successfully receiving the configuration from the C2 server and collecting user information. The executable does not do anything but enters the indefinite loop that alternates between sleeping for 100 seconds and waiting for user input without doing anything with that input.



Another addition to the new version of MetaStealer is the username and computer name check to avoid sandbox environments; if any of the usernames/computer names are found in the list, the stealer process will exit.

List of computer names:

```
{
                "bee7370c-8c0c-4", "desktop-nakffmt", "win-5e07cos9alr", "b30f0242-
1c6a-4", "desktop-vrsqlag", "q9iatrkprh", "xc64zb", "desktop-d019qdm", "desktop-
wi8clet", "server1",
                "lisa-pc", "john-pc", "desktop-b0t93d6", "desktop-1pykp29", "desktop-
1y2433r", "wileypc", "work", "6c4e733f-c2d9-4", "ralphs-pc", "desktop-wg3myjs",
                "desktop-7xc6gez", "desktop-5ov9s0o", "qarzhrdbpj", "oreleepc",
"archibaldpc", "julia-pc", "d1bnjkfvlh", "compname_5076", "desktop-vkeons4", "NTT-
EFF-2W11WSS"
        };
List of usernames:
        {
                "wdagutilityaccount", "abby", "peter wilson", "hmarc", "patex",
"john-pc", "rdhj0cnfevzx", "keecfmwgj", "frank", "8nl0colnq5bq",
                "lisa", "john", "george", "pxmduopvyx", "8vizsm", "w0fjuovmccp5a",
"lmvwjj9b", "pqonjhvwexss", "3u2v9m8", "julia",
```

```
"heuerzl", "harry johnson", "j.seance", "a.monaldo", "tvm"
};
```

Detection Rules

You can access Yara rules here.

You can access Sigma rules here.

Indicators of Compromise

Name Indicator

Name	Indicator
MetaStealer	e6db93b513085fe253753cff76054a2a
MetaStealer	a8d6e729b4911e1a0e3e9053eab2392b
MetaStealer	b3cca536bf466f360b7d38bb3c9fc9bc
C2	5.42.65[.]34:25530

For more samples, please refer to the result of my Yara scan on <u>UnpacMe</u>.

Yara Rule D									
Rule Validation: Passed									
Matches: 66 In 12 week lookback window			+	Scan Coverage: 93 %		+			
Observed Lifespan First Seen Last Seen	4 Weeks 28/11/2023 28/12/2023								
26 64 2		<50KB <100KB <250KB <500KB <1MB <5MB <10MB <25MB <50MB <100MB	0 46 18 0 0 0 0 0	redine RedLine_Campaign_June2021 MALWARE_Win_zgRAT INDICATOR_EXE_Packed_DotNe	57				

Reference

https://x.com/g0njxa/status/1739689195403100336?s=20

https://russianpanda.com/2023/11/20/MetaStealer-Redline's-Doppelganger

https://n1ght-w0lf.github.io/tutorials/dotnet-string-decryptor

https://twitter.com/cod3nym

https://www.unpac.me/yara/results/f87b8452-ba6d-4c8b-8adb-1ba3986eb4d9#

https://github.com/RussianPanda95/Configuration_extractors/blob/main/metastealer_string_ decryptor.py

https://github.com/RussianPanda95/Yara-Rules/tree/main/MetaStealer

https://github.com/RussianPanda95/Sigma-Rules/blob/main/MetaStealer/suspicious_qemu_file_creation.yaml



Previous Post

Pure Logs Stealer Fails to Impress



Next Post