

Nimbus Manticore Deploys New Malware Targeting Europe - Check Point Research

By samanthar@checkpoint.com

Published: 2025-09-22 · Archived: 2026-04-05 13:06:19 UTC

Nimbus Manticore Deploys New Malware Targeting Europe

Key Findings

- Check Point Research is tracking a long-running campaign by the Iranian threat actor **Nimbus Manticore**, which overlaps with UNC1549, Smoke Sandstorm, and the “Iranian Dream Job” operations. The ongoing campaign targets defense manufacturing, telecommunications, and aviation that are aligned with IRGC strategic priorities.
- Nimbus Manticore’s recent activity indicates a heightened focus on **Western Europe**, specifically Denmark, Sweden, and Portugal. The threat actor impersonates local and global aerospace, defense manufacturing, and telecommunications organizations.
- The threat actor uses tailored spear-phishing from alleged HR recruiters directing victims to fake career portals. Each target receives a unique URL and credentials, enabling tracking and controlled access of each victim. This approach demonstrates strong OPSEC and credible pretexting.
- The attacker uses previously undocumented low-level APIs to establish a multi-stage DLL side-loading chain. This causes a legitimate process to sideload a malicious DLL from a different location and override the normal DLL search order.
- The Nimbus Manticore toolset includes the **MiniJunk** backdoor and the **MiniBrowse** stealer. The tools continuously evolve to remain covert, leveraging valid digital signatures, inflate binary sizes, and use multi-stage sideloading and heavy, compiler-level obfuscation that renders samples be “irreversible” for regular advanced static analysis.
- Overall, the campaign reflects a mature, well-resourced actor prioritizing stealth, resiliency, and operational security across delivery, infrastructure, and payload layers, an approach consistent with nation-state tradecraft.

Introduction

Since early 2025, Check Point Research (CPR) has tracked waves of Nimbus Manticore activity. Known as **UNC1549** or **Smoke Sandstorm**, Nimbus Manticore is a mature Iran-nexus APT group that primarily targets aerospace and defense organizations in the Middle East and Europe. Some of its operations were also previously described as the [Iranian DreamJob](#) campaign.

Nimbus Manticore’s activity is characterized by highly targeted phishing campaigns leading to the deployment of custom implants, including *Minibike*. First [reported](#) by Mandiant in **June 2022**, *Minibike*, also known

as *SlugResin*, has evolved steadily since its creation. Sample analysis over the years shows its progress, including the addition of obfuscation techniques to evade detection and static analysis, a modular architecture, and the introduction of redundant command-and-control (C2) servers.

The most recent *Minibike* variants suggest a significant increase in the actor's abilities, including using a novel (and previously undocumented) technique to load DLLs from alternate paths by modifying process execution parameters. This variant has new TTPs such as size inflation, junk code, obfuscation, and code signing to lower detection rates.

In this article, we highlight the evolution of *Minibike* into a new variant dubbed **MiniJunk**. We also examine a distinct cluster within the Nimbus Manticore umbrella that targets different sectors and employs unique domain naming conventions, while continuing to use similar spear-phishing techniques and share malware resources.

While we were finalizing this publication, PRODAFT has released a comprehensive [report](#) on Subtle Snail, an espionage group with connections to Iran. In this publication, we address Subtle Snail in the chapter entitled 'Separate Cluster of Activity'. While this cluster employs tactics, techniques, and procedures (TTPs) that broadly align with those observed in Nimbus Manticore operations, it is differentiated by its unique malware capabilities, command-and-control (C2) infrastructure, and targeting preferences.

Malware Delivery websites

The attack starts with a phishing link that directs the victim to a fake job-related login page:

Figure 1 - Websites used to deliver malicious archives after successful login.

Figure 1 – Websites used to deliver malicious archives after successful login.

The infrastructure used by the attacker to lure job seekers is based on the React template, which varies depending on the impersonated brand, such as Boeing, Airbus, Rheinmetall and flydubai.

The domain naming convention is usually “**career**” themed and registered behind **Cloudflare**, most likely to keep the real server IP confidential.

The credentials for these login panels are pre-shared with the victim together with a link to the login page. After entering credentials and clicking the login button, a post request is sent to `/login-user` api. If the credentials are not correct, a `401 Unauthorized` response is returned. Otherwise, the user downloads a malicious archive with the malware.

Infection Chain

A malicious archive downloaded by the victim often masquerades as legitimate hiring process-related software. In the following example, a ZIP archive named `Survey.zip` starts an elaborated infection chain. The execution chain leverages a unique technique which we call **multi-stage sideloading**:

Figure 2 - The infection chain.

Figure 2 – The infection chain. The infection chain includes the following stages:

- **User Execution:** The victim runs `Setup.exe` from the archive. This is a legitimate Windows executable, which sideloads `userenv.dll` from the same archive.
- **Malware setup:** `Setup.exe` starts another benign binary, a Windows Defender component called `SenseSampleUploader.exe`. This executable in turn sideloads the malware loader, `xmllite.dll`, from the archive's directory.
- **Persistence:** The loader copies `Setup.exe` under its original name, `MigAutoPlay.exe`, and the malicious `userenv.dll` that it sideloads, to the malware working directory `%AppData%\Local\Microsoft\MigAutoPlay\`. It creates a scheduled task to run the executable.

Figure 3 - The contents of malicious ZIP archive downloaded from the fake recruiting website.

Figure 3 – The contents of malicious ZIP archive downloaded from the fake recruiting website.

Userenv.dll in the malware setup stage

Once the initial Setup executable runs, it sideloads the `userenv.dll` from the same folder. The DLL first checks the name of the executing PE module to determine the current stage of the infection chain. This way, if the DLL does not run from `MigAutoPlay.exe` (meaning the setup of the backdoor did not occur yet), it will load the Loader DLL in a special way, exploiting undocumented low-level API to hijack the DLL loading path.

`userenv.dll` uses low-level `ntdll` API calls to execute a Windows Defender binary located at `C:\Program Files\Windows Defender Advanced Threat Protection\SenseSampleUploader.exe`. The Windows Defender executable is vulnerable to DLL hijacking due to using the relative path to `xmllite.dll`. This flaw is abused to sideload the `xmllite.dll`. However, the actor manages to sideload it from the **same folder** as the malicious archive as part of a unique multi-stage sideloading attack chain.

Normally, a legitimate Windows Defender executable does not load random DLLs from folders outside the Windows DLL search order path. So, what's happening here?

When using low-level NT API calls to create a process, a call to `RtlCreateProcessParameters` is mandatory to build a process parameter struct `RTL_USER_PROCESS_PARAMETERS` which is then handed to `RtlCreateUserProcess`. A key field in this structure is the `DllPath` parameter, which defines the search path that the process loader uses to locate and resolve imported modules. If set, it specifies the location where the loader should search for a DLL if it is not found in the application directory.

The malware abuses this undocumented feature by using `GetModuleHandle` to get a path to `Setup.exe` and then provides it as a `DllPath` parameter. As `Setup.exe` and `xmllite.dll` are next to each other in the malicious archive, when the dll is not found next to `SenseSampleUploader.exe`, it will be loaded from the archive directory:

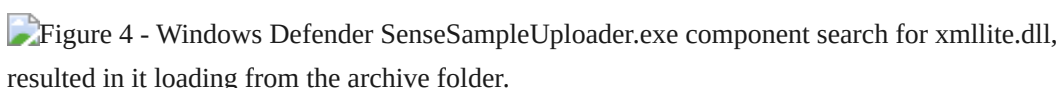
Figure 4 - Windows Defender SenseSampleUploader.exe component search for xmllite.dll, resulted in it loading from the archive folder.

Figure 4 – Windows Defender SenseSampleUploader.exe component search for xmllite.dll, resulted in it loading from the archive folder.

Once the `xmllite.dll` is loaded, its actions are pretty straightforward. It creates a working folder under the path `AppData\Local\Microsoft\MigAutoPlay\`. It copies the backdoor `userenv.dll` to it, also places the

legitimate executable there as `MigAutoPlay.exe` , and then adds an auto-run registry key to execute the benign executable.

 Figure 5 - Sideloaded DLL.

Figure 5 – Sideloaded DLL.

After persistence is completed, the malware is launched through the `MigAutoPlay.exe` , which sideloads `userenv.dll` and shows the victim a fake error pop-up about network issues blocking the lure program from running.

 Figure 6 - Fake error at the end of the malware setup process.

Figure 6 – Fake error at the end of the malware setup process.

The Backdoor: MiniJunk

In the last year, the actor introduced a lot of changes to the backdoor, first documented by Mandiant as “Minibike.” We chose to track this sample as “MiniJunk.”

The `userenv.dll` backdoor core logic starts from the `DLLMain` function. The backdoor first resolves many imports needed for it to function, but oddly enough, when it wants to use a function that was already resolved, it resolves it again. This behavior is unusual, but it might have been leveraged in the development cycle to identify API resolution issues. The backdoor then collects two identifiers from the infected system: the **computer name** and the **domain name with the username**.

Although the sample employs a substantial amount of obfuscation (which is discussed in detail in the next section), it does not encrypt the network data. Instead, it encodes it. We saw similar samples in the past that used a simple encryption on the network data, such as XOR with a few bytes. In this case, however, it uses a simple encoding algorithm: data is collected in a wide string, then converted to bytes, and the bytes are reversed. Finally, the entire string is reversed.

When the main logic starts, the backdoor checks if the running executable is called `MigAutoPlay.exe` (meaning the backdoor is running after the setup from its permanent working directory) and hooks the `ExitProcess` function to a function that sleeps, probably preventing fatal exits or allowing other threads to run in case of a program crash:

 Figure 7 - `ExitProcess` function hook.

Figure 7 – `ExitProcess` function hook.

After initialization, the backdoor starts a main thread that handles networking and the remaining logic. Analyzing the sample in this part is quite tricky: the logic is heavily branched through functions that utilize various states, for example, a large number of classes for network requests, and obfuscations in combination with library functions. However, all of this ultimately masks simple backdoor functionality.

Command and Control

The backdoor variants typically utilize multiple command and control (C2) servers in rotation for redundancy. There are several (between three to five) hardcoded C2 servers, so if one C2 goes down, the next one in the list will be used. The backdoor uses regular HTTPS requests using the Windows API. When the C2 responds, a thread is created to parse the C2 request. The C2 responds using encoded strings, similar to the initial network data. The response structure consists of a string separated by `##`. It is parsed by the backdoor and split into a vector of strings. Most C2 commands need 3 values:

- Command settings
- Command ID
- Command argument

For example, a “read file” command looks like this:

```
##[chunks size]##[read file command id]##[file path]
```

After parsing the command, in this case, the backdoor sends the file from the specified path via several network requests, based on the chunk size provided as an argument. The backdoor supports the following commands:

Command Id	Description	Arguments
0	Collect computer name, domain name with the username	None
1	Get computer name	None
2	Read a file and send it back	File path / chunks
3	Create file	File path, URL to the file on the C2
4	List hard drives / List files in a folder	String to list all hard drives or a directory path to list all files in
5	Delete file	File path
6	Create a process and use a named pipe for its output	Process path
7	Load DLL	DLL path
8 / 9	Do nothing / Placeholder	None
10	Move / Rename file	File target, File destination
11	Not implemented	None

As can be seen by the functions, these are pretty standard for a backdoor. The real complexity in the samples comes from their obfuscations which make the samples harder to analyze.

MiniBrowse – Stealer component

MiniBrowse is a lightweight stealer used by Nimbus Manticore. We observed two variants, one to steal Chrome credentials and another which targets Edge. Both versions are DLL designed to be injected into browsers to steal the stored passwords.

After it's executed, MiniBrowse first collects two identifiers from the system, username and domain name, and then connects to a predefined endpoint on the C2 server sending data in JSON payload.

 Figure 8 - MiniBrowse sends victim data.

Figure 8 – MiniBrowse sends victim data.

We identified a unique network communication behavior, as the C2 needs to respond with any HTTP response except for 200. If it does, the backdoor continues its execution looking for several files related to Edge login data.

Each of those files is then exfiltrated to the C2, using a simple POST request:

 Figure 9 - MiniBrowse exfiltrating data stolen from Edge browser.

Figure 9 – MiniBrowse exfiltrating data stolen from Edge browser.

Another method of sending those files is through connecting and sending the JSONs to a named pipe. We identified multiple MiniBrowse versions with support for this functionality.

Obfuscation

The MiniJunk and MiniBrowse samples that we investigated exhibit heavy compiler-level code obfuscation, possibly implemented via custom LLVM passes. We had to address several obfuscation techniques to facilitate analysis, including junk code insertion, control-flow obfuscation, opaque predicates, obfuscated function calls, and encrypted strings. The attacker invested significant effort in developing these LLVM passes and continues to refine them; each “generation” of samples shows improvements over the previous one, typically introduced between campaigns. The actor appears to be targeting a substantial number of victims, and these obfuscations help the malware remain undetected while at the same time slowing down researchers trying to determine the samples' behavior. As with most obfuscation, no single tool addresses all cases: off-the-shelf tools often fail unless the scheme matches a generic framework such as OLLVM – which is not the case here. This underscores the attacker's willingness to invest in their toolset and, conversely, benefits researchers by exposing new techniques. We invested considerable effort to make the samples sufficiently “reversible” for analysis.

Function call obfuscations

The backdoors contain compiler-level obfuscation. As a result, almost all function calls are obfuscated. The decision on what function to call is based on several arithmetic operations, which are then stored in the RAX register. Here is an example of a DLL's primary function:

 Figure 10 - DLL's primary function with obfuscated function calls.

Figure 10 – DLL's primary function with obfuscated function calls.

Obfuscated Control Flow

Not only are function calls obfuscated, but there are also obfuscated branches inside functions.

In this next example, there is a JMP RAX instruction, but it's not a single JMP. Depending on various conditions that are met when the code is running, the JMP can lead to two different places, just like a conditional JMP, but masked as a single JMP.

 Figure 11 - Obfuscated branch.

Figure 11 – Obfuscated branch.

String encryption

Each string is individually encrypted with its own key. The encrypted bytes are stored in memory with the key placed at the end of each string. Each string gets its own decryption function, adding another layer of complexity. To top it off, the decryption routines are each overloaded with opaque predicates:

 Figure 12 - String encryption.

Figure 12 – String encryption.

We used LLM to simplify the function mentioned above. Eventually the encryption algorithm is just `string[i] ^ key[i % key_length]`. Once we established that, we were able to automate and decrypt all strings.

Junk code

The samples contain a bit of unused junk code:

 Figure 13 - Functions with junk code.

Figure 13 – Functions with junk code.

Distinct patterns helped us deduce that a “block” of instructions can be classified as junk code, highly repetitive in the code. Then we can exclude it in the decompile view, and continue with static analysis:


 Figure 14 - The same function without junk code.

Figure 14 – The same function without junk code.

The evolution of MiniJunk

Over the past year, MiniJunk has undergone many changes and incorporated a variety of techniques. In this section, we describe the most significant.

Signing

In May, Nimbus Manticore started to use the service [SSL.com](https://www.ssl.com/) to sign their code. This led to a drastic decrease in detections, with many samples remaining undetectable by multiple malware engines.

Based on the signing dates and our analysis of samples signed by this certificate, we determined that they were generated by the threat actor, masquerading as existing IT organizations in Europe.

Command and control

In June, the actor re-architected C2 to combine Cloudflare and Azure App Service. This improved the resiliency so execution could continue even if a provider or domain was suspended.

File Size and detections

Large malware files often have lower endpoint detection, as many Antivirus engines enforce time, size, and resource limits that truncate deep unpacking, emulation, and heuristic layers on oversized inputs. Nimbus Manticore exploits this by inflating binaries with inert junk code blocks. Feature extraction and ML models frequently cap analysis to the first portion of a file, so padding pushes discriminative byte patterns past those limits, while some engines simply skip or downgrade scanning of large files to avoid false positives and performance hits. The combination of obfuscations, size, and codesigning result in lower endpoint detection. As you can see, some of the largest samples remained with zero detections on VirusTotal:

Figure 15 - MiniJunk with zero detection in VirusTotal.

Figure 15 – MiniJunk with zero detection in VirusTotal.

Separate Cluster of Activity

In addition to the operations involving the MiniJunk backdoor we described earlier in this blog, we observed a separate but closely related activity cluster. This cluster, first [reported](#) by PRODAFT, employs TTPs that broadly align with those documented above, but is distinguished by much smaller payloads and a lack of sophisticated obfuscation.

Spear phishing emails

Check Point Harmony Email & Collaboration platform identified and blocked a spear-phishing attack against a telecommunication provider in Israel.

As documented in past intrusions, the attacker uses professional social media such as LinkedIn, masquerades as an HR specialist, then asks the target to move to another platform such as email.

A malicious email sent from an Outlook account with a job application invitation:

Figure 16 – Malicious email sent by Nimbus Manticore.

Figure 16 – Malicious email sent by Nimbus Manticore.

As previously observed in other Nimbus Manticore campaigns, the link leads to a React-based fake recruiting login page:

Figure 17 - Fake page delivering malware after login.

Figure 17 – Fake page delivering malware after login.

Payload

The malware used in this operation is delivered through DLL hijacking of `dxgi.dll` :

 Figure 18 - Contents of the malware folder.

Figure 18 – Contents of the malware folder.

The malware strings were obfuscated by using simple one-byte XOR with `0x55` .

The execution started by decrypting 5 predefined C&C servers:

`services-update-check.azurewebsites[.]net`

`send-feedback.azurewebsites[.]net`

`send-feedback-413.azurewebsites[.]net`

`send-feedback-838.azurewebsites[.]net`

`send-feedback-296.azurewebsites[.]net`

`services-update-check.azurewebsites[.]net send-feedback.azurewebsites[.]net send-feedback-413.azurewebsites[.]net send-feedback-838.azurewebsites[.]net send-feedback-296.azurewebsites[.]net`

```
services-update-check.azurewebsites[.]net
send-feedback.azurewebsites[.]net
send-feedback-413.azurewebsites[.]net
send-feedback-838.azurewebsites[.]net
send-feedback-296.azurewebsites[.]net
```


 Figure 19 - C2 domain encryption.

Figure 19 – C2 domain encryption.

Despite overlapping infection chain steps and infrastructure, `dxgi` and `MiniJunk` implement different command sets. At the same time, `dxgi` does not exhibit evasion or obfuscation techniques. All this indicates parallel activity that could be conducted by more than one actor.

Command ID	Behavior (high-level)
0	Do nothing
1	Get computer name
2	Get username
3	List files and folders in a directory
4	Delete a file

Command ID	Behavior (high-level)
5	Move / rename a file
6	Enumerate hard drives
7	Upload a file
8	Get a list of running processes
9	Kill a process
10	Execute a bat/exe/cmd command/load dll
11	Create a process and use a named pipe for its output
12	Load a DLL

Comparison to MiniJunk

`dxgi.dll` and MiniJunk samples overlap in multiple details: they hook the exit process in a very similar way, and they both collect the username and desktop name (but the new sample also collects adapter information).

In terms of C2 communication, the key similarities lie in two areas: the parsing of network responses from C2 server, and the set of C2 commands.

The responses from C2 to MiniJunk use various separators for the data, such as `###` or `---`.

The `dxgi.dll` backdoor includes an additional verification of the request by hashing one of the parameters with FNV and comparing the result with a generated value. Overall, the C2 communication between these backdoors is not identical but is still quite similar.

The C2 commands in both versions closely resemble each other, with very similar logic and an identical order of operations within the functions themselves. While the command ID varies, the underlying code base appears to be the same.

Regarding significant differences, for network communications, `dxgi.dll` adds a layer of encryption. In addition, the backdoor utilizes the WinHTTP API, but unlike MiniJunk which employs classes and branching on network requests, this current backdoor handles all network logic within a single function. Finally, it appears the backdoor developer didn't bother changing the user agent, instead keeping it as is `WinHTTP Example:`

 Figure 20 - Network communications using WinHTTP API and a sample user agent.

Figure 20 – Network communications using WinHTTP API and a sample user agent.

The findings above suggest `dxgi.dll` shares a common code base with MiniJunk versions. Both of the activity clusters may have access to the code base, and can modify the code as needed, adding compiler passes, and altering the logic slightly. At the same time, the **programming paradigm remains similar**. This is hard to notice at first, due to MiniJunk obfuscations, the different layout of the HTTP request method (classes vs non-classes), and other variations. But once the obfuscations are addressed, it becomes clear that they share the same code base.

Infrastructure

MiniJunk campaigns use long, concatenated health-themed subdomains of azurewebsites[.]net. Notably, the domain naming convention in this campaign is different: the unique domain pattern is `[a-z]-[a-z]+-[a-z]+-[0-9]{3}.azurewebsites.net` combining multiple words joined by hyphen separators.

While hunting for these domain naming conventions, we observed a distinct set of domains used to target Europe which featured the following sequence of malicious domains:

1. `telespazio-careers[.]com` – Lure website
2. `update-health-service[.]azurewebsites[.]net` – First observed Azure app service domain (mentioned by PRODAFT)

We were able to capture the following domain block, which we believe is unique for each sample:

`check-backup-service.azurewebsites[.]net`

`check-backup-service-288.azurewebsites[.]net`

`check-backup-service-179.azurewebsites[.]net`

`check-backup-service-736.azurewebsites[.]net`

`check-backup-service.azurewebsites[.]net check-backup-service-288.azurewebsites[.]net check-backup-service-179.azurewebsites[.]net check-backup-service-736.azurewebsites[.]net`

```
check-backup-service.azurewebsites[.]net
check-backup-service-288.azurewebsites[.]net
check-backup-service-179.azurewebsites[.]net
check-backup-service-736.azurewebsites[.]net
```

C2 Infrastructure based on azurewebsites allows the attacker flexibility and redundancy; if one C2 goes down, they can easily set up a new one.

Victimology

While Nimbus Manticore consistently targets the Middle East, especially Israel and the UAE, recent operations show increased interest in Western Europe. We found a correlation between the malware delivery websites and the targeted sectors. For example, a fake hiring portal of a telecommunication company will target an employee and organizations in this sector. Our findings point to similar targets in several key sectors: telecommunications, especially **satellite providers**, defense contractors, aerospace and airlines. These sectors align with the IRGC's strategic intelligence collection efforts.



Figure 21 – Geographic distribution of targeted organizations.

The deployment of Minibike samples in June suggests “business as usual”, occurring as it did against the backdrop of the **twelve-day conflict between Israel and Iran**. The identified samples indicate that Israel was the primary focus at that time.

Conclusion

In our research, we uncovered the elusive operations of the Iranian threat actor known as **Nimbus Manticore**. Over the last year, this threat actor adopted a new set of techniques that allowed them to remain under the radar and continue operating even during the twelve-day Israeli-Iranian conflict.

Nimbus Manticore also expanded its interest in European targets, particularly in the **telecommunications, defense, aerospace, satellite and airline** sectors. We analyzed the evolution of the Minibike implant, which has incorporated multi-layered obfuscation and increasingly relies on legitimate cloud services to remain stealthy and difficult to detect.

IOCs:

Hashes:

```
23c0b4f1733284934c071df2bf953a1a894bb77c84cff71d9bfcf80ce3dc4c16- malicious zip
0b2c137ef9087cb4635e110f8e12bb0ed43b6d6e30c62d1f880db20778b73c9a - malicious zip
6780116ec3eb7d26cf721607e14f352957a495d97d74234aade67adbdc3ed339 - malicious zip
41d60b7090607e0d4048a3317b45ec7af637d27e5c3e6e89ea8bdcad62c15bf9 - malicious zip
4260328c81e13a65a081be30958d94b945fea6f2a483d051c52537798b100c69 -malicious zip
a37d36ade863966fb8520ea819b1fd580bc13314fac6e73cb62f74192021dab9- malicious zip
5d832f1da0c7e07927dcf72d6a6f011bfc7737dc34f39c561d1457af83e04e70- malicious zip
fffeacef025ef32ad092eea4761e4eec3c96d4ac46682a0ae15c9303b5c654e3e
c22b12d8b1e21468ed5d163efbf7fee306e357053d454e1683ddc3fe14d25db5
4da158293f93db27906e364a33e5adf8de07a97edaba052d4a9c1c3c3a7f234d
061c28a9cf06c9f338655a520d13d9b0373ba9826a2759f989985713b5a4ba2b
bc9f2abce42141329b2ecd0bf5d63e329a657a0d7f33ccdf78b87cf4e172fbd1
```

e69c7ea1301e8d723f775ee911900fbf7caf8dcd9c85728f178f0703c4e6c5c0
e77b7ec4ace252d37956d6a68663692e6bde90cdbbb07c1b8990bfaa311ecfb2
b43487153219d960b585c5e3ea5bb38f6ea04ec9830cca183eb39ccc95d15793
1b629042b5f08b7460975b5ecabc5b195fcbdf76ea50416f512a3ae7a677614a
f8a1c69c0300222980963a5d50ab9257bc4a1f2f486c3e7912d75558432be88
954de96c7fcc84fb062ca1e68831ae5745cf091ef5fb2cb2622edf2358e749e0
afe679de1a84301048ce1313a057af456e7ee055519b3693654bbb7312083876
9ec7899729aac48481272d4b305cefffa7799dcdad88d02278ee14315a0a8cc1
3b4667af3a3e6ed905ae73683ee78d2c608a00e566ae446003da47947320097f
a4f5251c81f080d80d1f75ad4cc8f5bc751e7c6df5addcfa268d59107737bd0
cf0c50670102e7fc6499e8d912ce1f5bd389fad5358d5cae53884593c337ac2e
3b58fd0cef8a42226be4d26a64235da059986ec7f5990d5c50d47b7a6cfadcd
7c77865f27b8f749b7df805ee76cf6e4575cbe0c4d9c29b75f8260210a802fce
d2db5b9b554470f5e9ad26f37b6b3f4f3dae336b3deea3f189933d007c17e3d8
b9b3ba39dbb6f4da3ed492140ffc167bde5dee005a35228ce156bed413af622d
53ff76014f650b3180bc87a23d40dc861a005f47a6977cb2fba8907259c3cf7a
b405ae67c4ad4704c2ae33b2cf60f5b0ccdaf65c2ec44f5913664805d446c9b
5985bf904c546c2474cbf94d6d6b2a18a4c82a1407c23a5a5eca3cd828f03826
0e4ff052250ade1edaab87de194e87a9afef903695799bc3571918b131100
8e7771ed1126b79c9a6a1093b2598282221cad8524c061943185272fbe58142d
f54fccb26a6f65de0d0e09324c84e8d85e7549d4d04e0aa81e4c7b1ae2f3c0f8
054483046c9f593114bc3ddc3613f71af6b30d2e4b7e7faec1f26e72ae6d7669
95d246e4956ad5e6b167a3d9d939542d6d80ec7301f337e00bb109cc220432cf - Minibrowse
9b186530f291f0e6ebc981399c956e1de3ba26b0315b945a263250c06831f281 - Minibrowse

Domains:

asylimed[.]azurewebsites[.]net
clinichaven[.]azurewebsites[.]net
healsanctum[.]azurewebsites[.]net
mediasylum[.]azurewebsites[.]net
therashelter[.]azurewebsites[.]net
arabiccountriestalent[.]com
arabiccountriestalenthr[.]azurewebsites[.]net
arabiccountriestalents[.]azurewebsites[.]net
arabiccountriestalentshr[.]azurewebsites[.]net
talenthumanresourcetalent[.]com
carebytesolutions[.]azurewebsites[.]net
medicoreit[.]azurewebsites[.]net
smartmediq[.]azurewebsites[.]net
vitatechlink[.]azurewebsites[.]net
biolinksystems[.]azurewebsites[.]net
digicura[.]azurewebsites[.]net
healthcarefluent[.]com
hivemedtech[.]azurewebsites[.]net
neurocloudhq[.]azurewebsites[.]net
marsoxygen[.]azurewebsites[.]net
nanobreathe[.]azurewebsites[.]net

turbulencemd[.]azurewebsites[.]net
zerogmed[.]azurewebsites[.]net
virgomarketingsolutions[.]com
virgomarketingsolutions[.]comtions[.]com
airtravellog[.]com
masterflexiblecloud[.]azurewebsites[.]net
storagewiz[.]co[.]azurewebsites[.]net
thecloudappbox[.]azurewebsites[.]net
arabiccountriestalent[.]azurewebsites[.]net
focusfusion[.]eastus[.]cloudapp[.]azure[.]com
frameforward[.]azurewebsites[.]net
tacticalsnap[.]eastus[.]cloudapp[.]azure[.]com
thetacticstore[.]com
lensvisionary[.]azurewebsites[.]net
wellnessglowluth[.]azurewebsites[.]net
activehealthlab[.]azurewebsites[.]net
ehealthpsuluth[.]com
grownehealth[.]eastus[.]cloudapp[.]azure[.]com
activespiritluth[.]eastus[.]cloudapp[.]azure[.]com
createformquestionshelper[.]com[.]net
collaboromarketing[.]com
cloudaskquestioning[.]eastus[.]cloudapp[.]azure[.]com[.]net
cloudaskquestionanswers[.]com[.]net
cloudaskquestionanswers[.]azurewebsites[.]net[.]net
cloudaskingquestions[.]eastus[.]cloudapp[.]azure[.]com[.]net
cloudaskingquestions[.]azurewebsites[.]net[.]net
cloudaskingquestioning[.]azurewebsites[.]net[.]net
vitatechlinks[.]azurewebsites[.]net
mojavemassageandwellness[.]com
airmdsolutions[.]azurewebsites[.]net
ventilateainest[.]azurewebsites[.]net
aeroclinicit[.]azurewebsites[.]net
exchtestcheckingapijson[.]azurewebsites[.]net
exchtestcheckingapihealth[.]com
exchtestchecking[.]azurewebsites[.]net
maydaymed[.]azurewebsites[.]net
traveltipspage[.]com
smartapptools[.]azurewebsites[.]net
createformquestionshelper[.]com
cloudaskquestioning[.]eastus[.]cloudapp[.]azure[.]com
cloudaskquestionanswers[.]com
cloudaskquestionanswers[.]azurewebsites[.]net
cloudaskingquestions[.]eastus[.]cloudapp[.]azure[.]com
cloudaskingquestioning[.]azurewebsites[.]net
healthbodymonitoring[.]azurewebsites[.]net
healthcare-azureapi[.]azurewebsites[.]net
healthdataanalyticsrecord[.]azurewebsites[.]net

medical-deepresearch[.]azurewebsites[.]net
medicalit-imaging[.]azurewebsites[.]net
mentalhealth-support-portal[.]azurewebsites[.]net
patient-azureportal[.]azurewebsites[.]net
pharmainfo[.]azurewebsites[.]net
symptom-recordchecker[.]azurewebsites[.]net
systemmedicaleducation[.]azurewebsites[.]net
acupuncturebentonville[.]com
cardiomedspecialists[.]azurewebsites[.]net
digithealthplatform[.]azurewebsites[.]net
medicpathsolutions[.]azurewebsites[.]net
nextgenhealthtrack[.]azurewebsites[.]net
sulumorbusinessservices[.]com
telehealthconnectpro[.]azurewebsites[.]net
totalcaremedcenter[.]azurewebsites[.]net
trustedcarehub360[.]azurewebsites[.]net
virtualcliniczone[.]azurewebsites[.]net
wellnessfirstgroup[.]azurewebsites[.]net
yourfamilymdclinic[.]azurewebsites[.]net
doctorconsult-app.azurewebsites[.]net
managetools-platform.azurewebsites[.]net
msnotetask-insights.azurewebsites[.]net
mstrakcer-tools.azurewebsites[.]net
olemanage-dashboard.azurewebsites[.]net
oletask-tracker.azurewebsites[.]net
patientcare-portal.azurewebsites[.]net

Similar activity cluster:

rpcconnection.azurewebsites[.]net
backsrv66.azurewebsites[.]net
backsrv74.azurewebsites[.]net
datasheet96.azurewebsites[.]net
mainrepo10.azurewebsites[.]net
services-update-check[.]azurewebsites[.]net
send-feedback[.]azurewebsites[.]net
send-feedback-413[.]azurewebsites[.]net
send-feedback-838[.]azurewebsites[.]net
send-feedback-296[.]azurewebsites[.]net
check-backup-service[.]azurewebsites[.]net
check-backup-service-288[.]azurewebsites[.]net
check-backup-service-179[.]azurewebsites[.]net
check-backup-service-736[.]azurewebsites[.]net
boeing-careers[.]com
rheinmetallcareer[.]org
rheinmetallcareer[.]com
airbus[.]global-careers[.]com
airbus[.]careersworld[.]org

```
airbus[.]usa-careers[.]com
airbus[.]germanywork[.]org
airbus[.]careers-portal[.]org
rheinmetall[.]careersworld[.]org
rheinmetall[.]careers-hub[.]org
rheinmetall[.]theworldcareers[.]com
rheinmetall[.]gocareers[.]org
flydubaicareers[.]ae[.]org
global-careers[.]com
careers-hub[.]org
careersworld[.]org
usa-careers[.]com
germanywork[.]org
careers-portal[.]org
theworldcareers[.]com
gocareers[.]org
```

Source: <https://research.checkpoint.com/2025/nimbus-manticore-deploys-new-malware-targeting-europe>