

Fast and Furious – Nimbus Manticore Operations During the Iranian Conflict

By stcpresearch

Published: 2026-05-22 · Archived: 2026-06-03 02:01:10 UTC

Key Findings

- The Iranian, IRGC affiliated, threat actor **Nimbus Manticore** resurfaced during Operation Epic Fury, the US military campaign against Iran launched on February 28, 2026, demonstrating newly adopted techniques and enhanced capabilities.
- The campaign leveraged malicious lures impersonating organizations in the aviation and software sectors across the United States, Europe and the Middle East.
- For the first time, we observed the use of SEO poisoning as an additional malware delivery method.
- The operation introduced a previously undocumented backdoor, named **MiniFast**, which appears to incorporate AI-assisted development practices, enabling the threat actor to rapidly develop and adapt tooling while maintaining high operational availability during the war.
- The actor also used a Zoom installer's execution flow and abused it to stage a time-sensitive infection chain for malware deployment while blending into legitimate system activity.

Introduction

During the recent geopolitical tensions in the Middle East, we [reported](#) on multiple Iran-nexus threat actors advancing Iran's strategic objectives through cyber operations. These activities included [targeting internet-connected cameras](#), conducting [destructive attacks](#) against US and Israeli entities, and [exfiltrating data](#) from cloud environments to support broader kinetic and intelligence-gathering efforts.

Nimbus Manticore (also tracked as [UNC1549](#)) is an IRGC-affiliated threat actor who primarily targets the defense, aviation and telecommunication sectors through career-themed phishing campaigns. Nimbus Manticore stands out compared to other Iranian-linked groups due to its complex malware toolset.

In 2025, we documented the [MiniJunk](#) malware framework used by Nimbus Manticore to target high-profile organizations across Western Europe and the Middle East.

In the recent campaign, the actor adopted several new techniques, including AppDomain (application domain) hijacking, AI-assisted malware development, and SEO poisoning.

In this article, we focus on three waves of the threat actor's activity in the last few months, as well as discuss their latest techniques.



Figure 1 – 2026 campaign timeline during the ongoing military campaign.

Campaign 1: Rising Tension

In February 2026, amid rising tensions between the US, Israel and Iran and weeks of military buildup, we monitored new Nimbus Manticore phishing activity worldwide. In this campaign, the threat actor introduced a modified infection chain by abusing AppDomain Hijacking for execution instead of relying on the usual DLL sideloading techniques.

AppDomain Hijacking is a technique that abuses legitimate .NET applications to load a malicious DLL at launch time. This is achieved by placing a Trojanized XML `.config` file in the same directory as the target application. The configuration file, named after the abused binary with the `.config` suffix, specifies an attacker-controlled `AppDomainManager` class that points to a malicious DLL. When the application starts, the .NET runtime loads the DLL, enabling malicious code execution within the context of the trusted process.



Figure 2 – Config file pointing the `appDomainManager` class to the attacker-controlled DLL.

The phishing lure is consistent with previous Nimbus Manticore campaigns, targeting employees in selected organizations (primarily software and aviation sectors) with fake career opportunities. Targeted organizations in Saudi Arabia and Australia were directed to download a compressed ZIP archive stored on the OnlyOffice platform.



Figure 3 – ZIP file hosted on Onlyoffice.

The downloaded ZIP file contains these files:

- Setup.exe – Benign Microsoft-signed binary.
- Setup.exe.config – AppDomain Hijacking configuration file pointing to uevmonitor.dll.
- uevmonitor.dll – A first stage Dropper.
- Interop.TaskScheduler.dll – a benign DLL.



Figure 4 – Zip file masquerading as an Accenture job opportunity.

After the setup.exe binary is executed, the first-stage loader (uevmonitor.dll) is loaded. This component is responsible for extracting and deploying the next-stage payload, which is stored in encrypted form within the loader itself.

The extracted files are written into C:\Users\<USER>\AppData\Local\Packages\ and include a legitimate executable used for DLL sideloading alongside a malicious DLL identified as a new version of the **MiniJunk** backdoor.

The first-stage loader uevmonitor.dll shares multiple behaviors similar to older MiniJunk loader variants. These include validating that it is loaded specifically by the Setup.exe process and displaying a fake error message stating "Couldn't connect to survey server" to appear as a legitimate application failure and reduce user suspicion.

Campaign 2: During Operation Epic Fury



Figure 5 – Campaign 2: During Operation Epic Fury – Attack Chain.

During Operation Epic Fury, we continued to observe activity from the threat actor. Despite the challenging environment, Nimbus Manticore demonstrated a strong ability to rapidly adapt, maintain infrastructure, and develop new tooling. We assess that this capability was likely supported, at least in part, by LLM-based tools and AI-assisted development techniques.

In addition to career-themed phishing lures masquerading as a US-based airline, the threat actor also used a Trojanized Zoom installer, which we assess was part of a phishing campaign using fake meeting invitations. In addition, the Trojanized Zoom installer demonstrated in-depth research into the original application's installation and execution flow, enabling it to be seamlessly integrated into the infection chain.

Similar to previous campaigns, the threat actor continued leveraging AppDomain Hijacking, not just for the initial execution stage but also during the deployment and execution of the final backdoor. For the final payload, the threat actor introduced a new backdoor that we named **MiniFast**, replacing the previously used **MiniJunk** malware family.

Many of the files used throughout the campaign had valid digital signatures via [SSL.com](https://www.ssl.com), continuing the abuse of trusted signing infrastructure we previously documented in [our 2025 report](#). We identified the use of at least two certificates during the current activity, including:

- Gray Matter Software S.R.L.
- Kirubel Kerie Negeya

Infection Chain

The infection chain begins with the victim downloading a compressed archive named `Zoominstall64.zip`, which contains the following files:

- `Setup.exe` – Benign Microsoft-signed binary (`ServiceHub.VSDetouredHost.exe`).
- `Setup.exe.config` – AppDomain Hijacking configuration file pointing to `InitInstall.dll`.
- `InitInstall.dll` – First-stage loader.
- `Zoom_cm.exe` – Original Zoom installer.
- `UpdateConfig.xml` – AppDomain Hijacking configuration file pointing to `Updater.dll`.
- `Updater.dll` – Second-stage loader.
- `UpdateChecker.dll` – Final backdoor payload (`MiniFast`).

First-Stage Deployment

After `Setup.exe` is launched by the user, the first-stage loader (`InitInstall.dll`) is executed through AppDomain Hijacking using the accompanying `.config` file.

The loader itself is lightly obfuscated. Most readable strings are decrypted at runtime using a simple combination of ROT13 encoding and reversed-string transformations. Aside from the string obfuscation layer, the codebase contains meaningful function names and relatively well-structured logic. Execution begins with the malware displaying a fake installation progress window intended to mimic legitimate software installation activity. At the same time, the loader launches the legitimate Zoom installer (`Zoom_cm.exe`) to make the execution flow appear to the victim as a normal software installation.

Persistence through Task hijacking

After launching the installer, the malware enters a loop that lasts approximately one minute, continuously monitoring the system for the creation of a scheduled task matching this format:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

ZoomUpdateTaskUser-<current user SID>

ZoomUpdateTaskUser-<current user SID>

```
ZoomUpdateTaskUser-<current user SID>
```

This scheduled task is usually created by the legitimate Zoom installer during installation.

When the task is created, the malware hijacks and modifies it to execute the second-stage component instead. By abusing an existing Zoom scheduled task rather than creating a new suspicious persistence mechanism, the malware attempts to blend into legitimate system activity and reduce detection opportunities.

Second-Stage Deployment

The next-stage files are copied into `C:\Users\<USER>\AppData\Local\Zoom\bin\update`. This directory contains four files copied from the original archive, including the benign Microsoft-signed binary from the first stage, now renamed to `Update.exe`. The malware again abuses AppDomain Hijacking to load the second-stage loader (`Updater.dll`) through the trusted `Update.exe` process.

Similar to the first stage, the second-stage loader uses the same runtime string decryption routine based on ROT13 and reversed strings.

At the beginning of its execution, the loader performs a simple anti-analysis validation intended to evade sandbox environments and automated dynamic analysis systems. The malware only continues execution if:

- The hosting process name is `update.exe`
- The parent process is `svchost.exe`

This execution-chain validation ensures that the DLL is loaded by the malware's intended loader component and that execution originates from the scheduled-task persistence mechanism instead of launched directly through `explorer.exe` etc.

The primary purpose of the second-stage loader is to dynamically load the final MiniFast payload (`UpdateChecker.dll`), locate its exported function named `CheckForUpdates`, and execute it.

Adoption of AI

This campaign also provides multiple indications that the threat actor leveraged AI-assisted development during the malware creation. We see evidence for this in both the initial access loaders and within the MiniFast backdoor itself.

Several coding patterns and implementation details strongly suggest the use of AI-generated or AI-assisted code during development, including:

- Excessive error handling and defensive programming logic, even around simple API calls such as `GetUserName`.
- Repetitive function and method naming patterns containing descriptive or verbose identifiers.
- Multiple detailed error-reporting strings and debug-style status messages embedded throughout the codebase.
- Modular code organization despite the malware's overall simplicity.

These characteristics are increasingly prevalent in malware development as threat actors leverage AI-assisted tools to accelerate development, improve code structure, and rapidly utilize new capabilities.

Campaign 3: Post Ceasfire – “SQL developer” Campaign

In April, we observed a new infection method, a fake website impersonating a download page for **SQL Developer**, a graphical tool used for working with databases. Users who attempted to download the software from the fake site instead received a weaponized installer that delivered the **MiniFast** backdoor.



Figure 6 – Screenshot of the getsqldeveloper[.]com site.

This malware delivery method differs from Nimbus Manticore’s usual infection chains which typically rely on career-themed phishing lures. In this campaign, the actor abuses search engine optimization techniques by registering dozens of domains that link to the bogus domain, `getsqldeveloper[.]com`. This is likely an attempt to increase the site’s visibility through link-based reputation signals.

At the time of our analysis, the malicious domain ranked high in the results returned by multiple search engines, such as Bing and DuckDuckGo, for the query “sql developer.” This increased the likelihood that users searching for legitimate SQL Developer downloads would encounter the site.

The pages also rely on keyword stuffing, repeatedly using search-oriented phrases such as “Download SQL Developer” and “SQL Developer Free,” likely to improve ranking for users searching for SQL Developer-related downloads.

MiniFast Technical Analysis

MiniFast is a 64-bit Windows PE DLL that exposes a single export named `CheckForUpdates` which acts as the main entry point. The DLL operates as a fully featured backdoor designed for long-term persistence and remote command execution. Analysis of multiple samples indicates the malware is undergoing active development, with the threat actor continuously modifying and improving the implant across versions.



Figure 7 – Export function *CheckForUpdates* structure.

Similar to the previous stage, the backdoor again appears to be executing under the expected process chain by verifying that the hosting process is named `update.exe` and that its parent process is `svchost.exe`

The implant communicates with its C2 (command and control) infrastructure using an API-style architecture with JSON-formatted data exchanges. To blend into legitimate network traffic, the malware impersonates a Chrome browser using the following hardcoded User-Agent string: `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari/537.36`

The backdoor implements several structured HTTP endpoints throughout the infection lifecycle:

URI	Method	Purpose
<code>/rg</code>	POST	Initial handshake
<code>/agent/init</code>	POST	Initial victim registration
<code>/agent/poll?token=</code>	GET	Task retrieval
<code>/agent/result</code>	POST	Command execution result upload
<code>/upload/</code>	PUT	File exfiltration
<code>/files/</code>	GET	File download from the C2

Before entering its tasking loop, the malware performs basic host reconnaissance by collecting information such as the username, hostname, and domain info, and then submits the collected data as a unique `clientId` to the `/rg` endpoint using a POST request.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
{  
  
"clientId":"<ComputerName>:<USERDOMAIN>\<UserName>",  
  
"type":"poll"  
  
}  
  
{ "clientId":"<ComputerName>:<USERDOMAIN>\<UserName>", "type":"poll" }
```

```
{  
  "clientId":"<ComputerName>:<USERDOMAIN>\<UserName>",  
  "type":"poll"  
}
```

If the server responds with HTTP status code `200`, the backdoor skips parsing the response body and continues executing normally. However, when the server responds with status code `400`, the malware parses the returned JSON object and extracts a `socketId`, which acts as the session identifier for all future communications.

In addition, the server response may include updated values for `pollInterval` and `jitterTime`, allowing the operator to dynamically adjust the timing between subsequent communications with the C2 infrastructure.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
{  
  
"socketId":"<string>",  
  
"pollInterval":120000,  
  
"jitterTime":5000
```

```
}

{ "socketId":"<string>", "pollInterval":120000, "jitterTime":5000 }
```

```
{
  "socketId":"<string>",
  "pollInterval":120000,
  "jitterTime":5000
}
```

Next, the backdoor continues to register the infected host by again sending the machine information, this time to the `/agent/init` in the following format:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
{

"token": "<socketId>",

"pcName": "<computer_name>",

"userName": "<user_name>",

"domainName": "<USERDOMAIN>",

"isElevated": true_or_false

}

{ "token": "<socketId>", "pcName": "<computer_name>", "userName": "<user_name>", "domainName": "<USERDOMAIN>", "isElevated": true_or_false }
```

```
{
  "token": "<socketId>",
  "pcName": "<computer_name>",
  "userName": "<user_name>",
  "domainName": "<USERDOMAIN>",
  "isElevated": true_or_false
}
```

Only after it receives an HTTP status code `200` from the C2 server does the backdoor proceed to fetch commands for execution using a GET request to `/agent/poll?token=<socketId>` .

Here, the communication between the implant and the C2 server is not in a JSON format and is performed using Base64-encoded serialized task structures, where each response contains one or more encoded tasks that are later decoded and processed by the backdoor.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
struct PollEnvelope {  
  
    uint32_t task_count;  
  
    struct TaskDescriptor {  
  
        uint32_t len_base64;  
  
        char base64_task[len_base64]; // ASCII, no null terminator  
  
    } tasks[task_count];  
  
};  
  
struct PollEnvelope { uint32_t task_count; struct TaskDescriptor { uint32_t len_base64; char  
base64_task[len_base64]; // ASCII, no null terminator } tasks[task_count]; };
```

```
struct PollEnvelope {  
    uint32_t task_count;  
    struct TaskDescriptor {  
        uint32_t len_base64;  
        char    base64_task[len_base64]; // ASCII, no null terminator  
    } tasks[task_count];  
};
```

Each task is then Base64-decoded into a secondary structure, containing the opcode and associated arguments:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
struct TaskRecord {
```

```
uint8_t opcode;

uint8_t pad[7]; // alignment

custom_str_struct arg_main; // at offset +0x08: main command argument

custom_str_struct arg_aux; // at offset +0x28: secondary arg (if needed)

custom_str_struct taskId; // at offset +0x48: unique task identifier

}

struct TaskRecord { uint8_t opcode; uint8_t pad[7]; // alignment custom_str_struct arg_main; // at offset +0x08:
main command argument custom_str_struct arg_aux; // at offset +0x28: secondary arg (if needed)
custom_str_struct taskId; // at offset +0x48: unique task identifier }
```

```
struct TaskRecord {
    uint8_t opcode;
    uint8_t pad[7];           // alignment
    custom_str_struct arg_main; // at offset +0x08: main command argument
    custom_str_struct arg_aux;  // at offset +0x28: secondary arg (if needed)
    custom_str_struct taskId;   // at offset +0x48: unique task identifier
}
```

The opcode determines which capability is executed, while the remaining fields contain command arguments and task tracking identifiers. The malware implements a structured opcode-based command handler that provides operators with extensive control over infected systems.



Figure 8 – MiniFast Command switch.

The supported command set:

Opcode	Capability	Arguments	Description
0x02	List Directory	path	Lists files and folders inside a specified directory.
0x03	Move / Rename	source , destination	Moves or renames files and directories on the victim machine.
0x04	Execute Command	command	Executes shell commands using <code>cmd.exe /c</code> and returns captured output.
0x05	Enumerate Processes	None	Enumerates running processes and returns process names alongside their PIDs.
0x06	Delete File / Directory	path	Deletes files or directories depending on the target type.
0x07	Download File	fileUuid , destinationPath	Downloads a file from the C2 server to the local machine.

Opcode	Capability	Arguments	Description
0x08	Upload File	path	Uploads local files from the infected machine to the C2 server.
0x09	Enumerate Drives	None	Lists available logical drives on the infected machine.
0x0A	Kill Process	pid	Terminates a process using its PID.
0x0B	Load DLL	dllPath , exportName	Dynamically loads a DLL and invokes a specified exported function.
0x0C	Create Directory	path	Creates a new directory on the victim machine.
0x0D	Create ZIP Archive	sourcePath , zipPath	Creates a ZIP archive from files or directories.
0xB0	Request UAC Elevation	pathOrCommand	Attempts to relaunch a process with elevated privileges using runas .
0xB1	Install Persistence	binaryPath	Creates or updates a scheduled task named WindowsSecurityUpdate .
0xF0	Set Poll Interval	milliseconds	Updates the beacon polling interval.
0xF1	Idle Command Acknowledge	None	Acknowledges an idle-time command without modifying behavior.
0xF2	Set Jitter	milliseconds	Updates the jitter value applied to beacon intervals.
Default	Unknown Opcode	Any	Returns an error for unsupported commands.

After executing a task, the implant serializes the execution result into a dedicated response structure which is Base64-encoded and submitted back to the C2 server through the `/agent/result` endpoint. The encoded result object contains the task identifier, execution status, and command output:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
struct ResultEntry {  
  
    uint32_t taskIdLen;  
  
    char taskId[taskIdLen]; // unique task identifier  
  
    uint32_t status; // 0 = success, 1 = error  
  
    uint8_t resultText[resultLen]; // command output  
  
};  
  
struct ResultEntry { uint32_t taskIdLen; char taskId[taskIdLen]; // unique task identifier uint32_t status; // 0 =  
success, 1 = error uint8_t resultText[resultLen]; // command output };
```

```
struct ResultEntry {  
    uint32_t taskIdLen;  
    char    taskId[taskIdLen];    // unique task identifier  
    uint32_t status;              // 0 = success, 1 = error  
    uint8_t  resultText[resultLen]; // command output  
};
```

Victimology

Nimbus Manticore consistently focuses on Europe, the Middle East and Africa, particularly Israel and the United Arab Emirates. However, in contrast to our previous research, the actor's recent operations demonstrate an expansion toward aviation-sector targets in the United States.

As observed in prior campaigns, there appears to be a strong correlation between the phishing lure and the targeted sector. For example, fraudulent hiring portals impersonating aviation companies were used to target employees and organizations operating within that industry. In the current campaign, impersonate US domestic airlines suggest a deliberate focus on US-based targets.

Our findings indicate targeting extends across several strategic sectors, including aviation and software development. These sectors align with the IRGC's broader intelligence collection priorities.



Figure 9 – Geographic Distribution of victims around the world.

Conclusion

Nimbus Manticore is one of the most sophisticated Iranian-aligned threat actors with a long-standing focus on the defense, telecommunications, and aviation sectors. The ongoing conflict in the Middle East, combined with the operational demands of wartime activity, appears to have significantly accelerated their malware evolution.

As an IRGC-affiliated entity operating under heightened geopolitical conditions, Nimbus Manticore demonstrated a rapid adoption cycle for new techniques, tooling, and operational methodologies. The actor's activity during Operation Epic Fury highlights their increasing adaptability, particularly through the integration of AI-assisted malware development, novel infection vectors, and advanced stealth mechanisms.

IOCs

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

SHA256

```
10fd541674adadfbb99b54280f7e59732746faf2b10ce68521866f737f1e46d
eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71
781605ce9d4a9869e846f6c9657d71437cb6240ab27ffbc4cd550c0e06996690
2c214494fd0bad31473ca8adce78a4f50847876584571e66aadeae70827ec2dc
f08b17856616d66492a24dced27f788e235f35f42fa7cd10f315000d3a2f4c03
```

a57ffb819fe8d98ff925c5d7b239598fe302acf5a13193d7a535040a71298fdf
63d0d3c4a7f71bdbca720903d6a99b832089cc093c64d2938e7e001e56c17ab4
74882085db2088356ed7f72f01e0404a0a98cda88ef56fb15ce74c1f36b26d27
bc3b44154518c5794ce639108e7b9c5fecb0c189607a26de1aaed518d890c7ad
ecaf493c320d201d285ef5f61d75744216e47cf1115b4af528f9a78883cc446e
44f4f7aca7f1d9bfdaf7b3736934cbe19f851a707662f8f0b0c49b383e054250
0db36a04d304ad96f9e6f97b531934594cd95a5cea9ff2c9af249201089dc864
485f182f7b74ea4013b2539275a95d21e3a9bf0082c331937af9353a324b36f3
64530d7e6ee30e4a66d9eed6b8595c33fd72f5f73409133ca40539e5695df4c
332ba2f0297dfb1599adecc3e9067893e7cf243aa23aedce4906a4c480574c17
9e4a658e6d831c9e9bdfe11884a75b7c64812ed0a80e8495ddf6b316505acac1
43dc62cef52ebdd69e79f10015b3e13890f26c058325c0ff139c70f8d8eadcfa
8808c794c24367438f183e4be941876f1d3ecd0c8d2eb43b10d2380841d2283b
5c3362d20229597d11380f56d1f2eb39647fb6afad7be8392a7abcd18dff12f8
0291ef318576953f7f3fe287e7775ed1d7c3206119dc7b9cd6d85c02779e6e40
d4a7e9f107fe40c1a5d0139c6c6e25bf6bf57f61feff090bee28f476bb3cc3c2
38bd137c672bd58d08c4f0502f993a6561e2c3411773d1ae57ee0151a0a9d11d
f54cd38632ac9da3af3533ae93e92625cbcb04df521dbf1b6acfaa81218f9e8c
b19e06da580cf91691eda066ac9ee4b09c6e5dc26c367af12660fe1f9306eec4
9cf029daca89523d917dafed0568d11d00e45ec96b5b90b4a1f7fd4018c7da84
a13ba3c5aff46e9daf2d23df4b3e3d49dc7236c207c56f0a1433051f3450d441
dfa1e3137a032ee8561a1cd5e1a0f71a10bebb36aef7c336c878638a9c1239ee

Domains

business-startup[.]org

business-startup.azurewebsites[.]net

businessstartup.azurewebsites[.]net

buisness-central.azurewebsites[.]net

buisness-central-transportation.azurewebsites[.]net

buisness-central-transportation[.]com

licencemanagers.azurewebsites[.]net

licencesupporting.azurewebsites[.]net

peerdistsvcmangers.azurewebsites[.]net

nanomatrix.azurewebsites[.]net

PremierHealthAdvisory[.]com

PremierHealthAdvisory[.]azurewebsites.net

Premier-HealthAdvisory[.]azurewebsites.net

ramiltonsfinance[.]com

ramiltonsfinance.azurewebsites[.]net

ramiltons-finance.azurewebsites[.]net

globalitconsultants.azurewebsites[.]net

globalit-consultants.azurewebsites[.]net

global-it-consultants.azurewebsites[.]net

global-it-checkers.azurewebsites[.]net

global-it-checkbusiness.azurewebsites[.]net

global-check-itbusiness.azurewebsites[.]net

global-check-business-it.azurewebsites[.]net

globalbusiness-checkers-it.azurewebsites[.]net

getsqldeveloper[.]com

SHA256 10fd541674adadfbba99b54280f7e59732746faf2b10ce68521866f737f1e46d
eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71
781605ce9d4a9869e846f6c9657d71437cb6240ab27ffbc4cd550c0e06996690
2c214494fd0bad31473ca8adce78a4f50847876584571e66aadeae70827ec2dc
f08b17856616d66492a24dced27f788e235f35f42fa7cd10f315000d3a2f4c03
a57ffb819fe8d98ff925c5d7b239598fe302acf5a13193d7a535040a71298fdf

63d0d3c4a7f71bdbca720903d6a99b832089cc093c64d2938e7e001e56c17ab4
74882085db2088356ed7f72f01e0404a0a98cda88ef56fb15ce74c1f36b26d27
bc3b44154518c5794ce639108e7b9c5fecb0c189607a26de1aaed518d890c7ad
ecaf493c320d201d285ef5f61d75744216e47cf1115b4af528f9a78883cc446e
44f4f7aca7f1d9bfdaf7b3736934cbe19f851a707662f8f0b0c49b383e054250
0db36a04d304ad96f9e6f97b531934594cd95a5cea9ff2c9af249201089dc864
485f182f7b74ea4013b2539275a95d21e3a9bf0082c331937af9353a324b36f3
64530d7e6ee30e4a66d9eed6b8595c33fd72f5f73409133ca40539e5695df4c
332ba2f0297dfb1599adecc3e9067893e7cf243aa23aedce4906a4c480574c17
9e4a658e6d831c9e9bdfe11884a75b7c64812ed0a80e8495ddf6b316505acac1
43dc62cef52ebdd69e79f10015b3e13890f26c058325c0ff139c70f8d8eadcfa
8808c794c24367438f183e4be941876f1d3ecd0c8d2eb43b10d2380841d2283b
5c3362d20229597d11380f56d1f2eb39647fb6afad7be8392a7abcd18dff12f8
0291ef318576953f7f3fe287e7775ed1d7c3206119dc7b9cd6d85c02779e6e40
d4a7e9f107fe40c1a5d0139c6c6e25bf6bf57f61feff090bee28f476bb3cc3c2
38bd137c672bd58d08c4f0502f993a6561e2c3411773d1ae57ee0151a0a9d11d
f54cd38632ac9da3af3533ae93e92625cbcb04df521dbf1b6acfaa81218f9e8c
b19e06da580cf91691eda066ac9ee4b09c6e5dc26c367af12660fe1f9306eec4
9cf029daca89523d917dafed0568d11d00e45ec96b5b90b4a1f7fd4018c7da84
a13ba3c5aff46e9daf2d23df4b3e3d49dc7236c207c56f0a1433051f3450d441
dfa1e3137a032ee8561a1cd5e1a0f71a10bebb36aef7c336c878638a9c1239ee Domains business-startup[.]org
business-startup.azurewebsites[.]net businessstartup.azurewebsites[.]net buisness-central.azurewebsites[.]net
buisness-central-transportation.azurewebsites[.]net buisness-central-transportation[.]com
licencemanagers.azurewebsites[.]net licencesupporting.azurewebsites[.]net
peerdistsvcmangers.azurewebsites[.]net nanomatrix.azurewebsites[.]net PremierHealthAdvisory[.]com
PremierHealthAdvisory[.]azurewebsites.net Premier-HealthAdvisory[.]azurewebsites.net ramiltonsfinance[.]com
ramiltonsfinance.azurewebsites[.]net ramiltons-finance.azurewebsites[.]net globalitconsultants.azurewebsites[.]net
globalit-consultants.azurewebsites[.]net global-it-consultants.azurewebsites[.]net global-it-
checkers.azurewebsites[.]net global-it-checkbusiness.azurewebsites[.]net global-check-
itbusiness.azurewebsites[.]net global-check-business-it.azurewebsites[.]net globalbusiness-checkers-
it.azurewebsites[.]net getsqldveloper[.]com

SHA256

10fd541674adadfbbba99b54280f7e59732746faf2b10ce68521866f737f1e46d
eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71
781605ce9d4a9869e846f6c9657d71437cb6240ab27ffbc4cd550c0e06996690
2c214494fd0bad31473ca8adce78a4f50847876584571e66aadeae70827ec2dc
f08b17856616d66492a24dced27f788e235f35f42fa7cd10f315000d3a2f4c03
a57fffb819fe8d98ff925c5d7b239598fe302acf5a13193d7a535040a71298fdf
63d0d3c4a7f71bdbca720903d6a99b832089cc093c64d2938e7e001e56c17ab4
74882085db2088356ed7f72f01e0404a0a98cda88ef56fb15ce74c1f36b26d27
bc3b44154518c5794ce639108e7b9c5fecb0c189607a26de1aaed518d890c7ad
ecaf493c320d201d285ef5f61d75744216e47cf1115b4af528f9a78883cc446e

44f4f7aca7f1d9bfdaf7b3736934cbe19f851a707662f8f0b0c49b383e054250
0db36a04d304ad96f9e6f97b531934594cd95a5cea9ff2c9af249201089dc864
485f182f7b74ea4013b2539275a95d21e3a9bf0082c331937af9353a324b36f3
64530d7e6ee30e4a66d9eed6b8595c33fd72f5f73409133ca40539e5695df4c
332ba2f0297dfb1599adec3e9067893e7cf243aa23aedce4906a4c480574c17
9e4a658e6d831c9e9bdfe11884a75b7c64812ed0a80e8495ddf6b316505acac1
43dc62cef52ebdd69e79f10015b3e13890f26c058325c0ff139c70f8d8eadcfa
8808c794c24367438f183e4be941876f1d3ecd0c8d2eb43b10d2380841d2283b
5c3362d20229597d11380f56d1f2eb39647fb6afad7be8392a7abcd18dff12f8
0291ef318576953f7f3fe287e7775ed1d7c3206119dc7b9cd6d85c02779e6e40
d4a7e9f107fe40c1a5d0139c6c6e25bf6bf57f61feff090bee28f476bb3cc3c2
38bd137c672bd58d08c4f0502f993a6561e2c3411773d1ae57ee0151a0a9d11d
f54cd38632ac9da3af3533ae93e92625cbcb04df521dbf1b6acfaa81218f9e8c
b19e06da580cf91691eda066ac9ee4b09c6e5dc26c367af12660fe1f9306eec4
9cf029daca89523d917dafed0568d11d00e45ec96b5b90b4a1f7fd4018c7da84
a13ba3c5aff46e9daf2d23df4b3e3d49dc7236c207c56f0a1433051f3450d441
dfa1e3137a032ee8561a1cd5e1a0f71a10bebb36aef7c336c878638a9c1239ee

Domains

business-startup[.]org
business-startup.azurewebsites[.]net
businessstartup.azurewebsites[.]net
buisness-central.azurewebsites[.]net
buisness-central-transportation.azurewebsites[.]net
buisness-central-transportation[.]com
licencemanagers.azurewebsites[.]net
licencesupporting.azurewebsites[.]net
peerdistsvcmangers.azurewebsites[.]net
nanomatrix.azurewebsites[.]net
PremierHealthAdvisory[.]com
PremierHealthAdvisory[.]azurewebsites.net
Premier-HealthAdvisory[.]azurewebsites.net
ramiltonsfinance[.]com
ramiltonsfinance.azurewebsites[.]net
ramiltons-finance.azurewebsites[.]net
globalitconsultants.azurewebsites[.]net
globalit-consultants.azurewebsites[.]net
global-it-consultants.azurewebsites[.]net
global-it-checkers.azurewebsites[.]net
global-it-checkbusiness.azurewebsites[.]net
global-check-itbusiness.azurewebsites[.]net
global-check-business-it.azurewebsites[.]net
globalbusiness-checkers-it.azurewebsites[.]net
getsqldeveloper[.]com

Source: <https://research.checkpoint.com/2026/fast-and-furious-nimbus-manticore-operations-during-the-iranian-conflict/>