

Detecting Nimbus Manticore and their sideloading infection chains

By Jonathan Peters

Published: 2026-06-02 · Archived: 2026-06-03 02:01:41 UTC

During a recent incident, we identified a sophisticated sideloading infection chain dropping a custom implant for data exfiltration. Further analysis allowed us to attribute the activity to the Iran-nexus APT group [Nimbus Manticore](#), also tracked as UNC1549 and Smoke Sandstorm. The group primarily targets aerospace and defense organizations in the Middle East and Europe.

Nimbus Manticore is known for targeted phishing operations and the use of custom implants. While individual payloads and tooling vary in between campaigns, the group's core tradecraft remains relatively consistent. In this article, we compare implants previously reported by Mandiant and Check Point Research with the activity we observed, highlighting both the changes in implementation and the recurring behavioral patterns.

A recurring element in Nimbus Manticore operations is the abuse of legitimate applications and trusted cloud infrastructure, including Microsoft Azure, to blend in with normal activity. Previous reporting described DLL search order hijacking involving legitimate applications from VMware, NVIDIA, Citrix, and Microsoft. The campaign we observed follows the same general sideloading pattern, but introduces [AppDomain hijacking](#) through legitimate .NET applications by using crafted executable configuration files to manipulate dependency resolution and execute the attacker-controlled payload.

The Lure

The lures are tailored to the industries and organizations targeted by the operators. In the incident we handled, the threat actor approached an employee via LinkedIn using a well-prepared, verified profile that impersonated a headhunter. The persona claimed to be recruiting talent for Ebix and used a generous salary offer of \$200,000 USD to lure potential victims.

The fake headhunter even provided a fully branded PDF that gave a seemingly legitimate job description, including key responsibilities, qualifications required as well as salary and benefits. Hunting for markers present in this PDF we identified additional PDF lures branded as Airbus, a target consistent with previous campaigns of Nimbus Manticore

Fake job description used by the threat actor

The interaction stayed professional and directed the victim to a fake hiring portal branded as Ebix Recruitment at `hxps://ebix[.]recruitment-flow[.]com/`. Ebix is a legitimate company offering solutions for insurance, banking, and business process outsourcing. The threat actor exploited their reputation for this campaign; Ebix is not involved. The portal's branding matches the recruitment pretext, lowering suspicion.

Fake hiring portal impersonating Ebix

The portal uses a legitimate appearing generic recruitment themed domain registered via Namecheap and hosted behind Cloudflare, to obscure the underlying infrastructure. The victim received temporary credentials to log in and begin the supposed application process. Without valid credentials the side does not provide any malware.

After authentication, the portal prompted the victim to download a two-factor authentication application for “additional security”. The advertised 2FA application was delivered as a ZIP archive and contained the malware payload.

Infection Chain

The ZIP archive downloaded from the hiring portal contains three files. By default, only `setup.exe` is visible because the remaining files are unpacked with the hidden attribute.

The `setup.exe` file is a renamed legitimate Microsoft Visual Studio component originally named `ServiceHub.VSDetouredHost.exe`. The threat actor abuses this .NET-based application through a manipulated executable configuration file. The modified configuration instructs the .NET runtime to load a custom `AppDomainManager` implemented in `TOTPGuard.dll` instead of following the normal execution flow.

In the `setup.exe.config` file, the operators set the probing path to the current directory and add binding directives that cause the runtime to resolve the attacker-controlled assembly named `TOTPGuard`. When the application starts, it follows the configuration directives and loads the malicious payload. Since the renamed `setup.exe` is signed by Microsoft, the initial process appears less suspicious and is less likely to trigger detections.

Once the victim executes `setup.exe`, the attacker code inside `TOTPGuard.dll` is loaded. The victim is then presented with a form impersonating Ebix again. The form asks the user to enter a secret key, although the provided input is not relevant for malware execution.

After the victim enters a “secret key,” the application displays a functional one-time password generator.

2FA app with fake Ebix branding

Unlike malware that only displays an error message to reduce user suspicion, this campaign maintains the appearance of a legitimate workflow. The fake application continues the hiring portal flow and provides enough functionality to make the 2FA prompt appear credible.

During stager analysis, we saw something rather amusing: the PDB path contains the term `AppDomainInjection`, which directly references the technique used by the malware. Thereby also revealing that we are definitely not dealing with a legitimate 2FA app. We can also leverage this keyword to create a generic YARA rule that searches for binaries and archive files.

```
rule SUSP_AppDomainInjection_Keyword_May26 {
  meta:
    description = "Detects link files, archives and binaries that contain keywords related to AppD
    author = "Jonathan Peters (cod3nym)"
    date = "2026-05-27"
```

```

reference = "https://attack.mitre.org/techniques/T1574/014/"
hash = "eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71"
score = 70
strings:
  $x1 = "AppDomainInjection" ascii wide fullword
  $x2 = "AppDomainHijack" ascii wide fullword
condition:
  (
    uint16(0) == 0x5a4d // PE
    or uint16(0) == 0x4b50 // ZIP
    or uint32(0x8000) == 0x30444301 // ISO
    or uint16(0) == 0x004c and uint32(4) == 0x00021401 // LNK
  )
  and 1 of ($x*)
}

```

Hunting with the above rule revealed several malicious samples and offensive security tooling. While most were not related to this specific APT campaign, the matches show that the rule can be used for generic detection of suspicious artifacts.

Technical Analysis

As mentioned before the staging payload uses a custom AppDomainManager.

When we inspect the code in dnSpyEx we found that the domain initialization referred execution to a method named `Execute` in `TOTPGuardRunner`.

The method checks for command-line arguments if the stager is invoked with the argument `doit` it will skip showing the graphical interface for the 2FA app and immediately execute the main payload. We will first checkout the path that is executed when no argument is provided.

The `Run` function initializes the one time password GUI and then proceeds to decrypt the embedded payload and drop it to disk.

The payload is encrypted using AES. The keys and IV are generic and hardcoded into the stager without obfuscation. We used the encrypted `MZ` header as a hunting indicator. Since the first bytes of a PE file are predictable before encryption, the corresponding encrypted bytes can be used to identify related samples that reuse the same key and IV. The following YARA rule matches the first bytes of the AES-encrypted `MZ` header observed in this stager:

```

rule SUSP_PE_Contains_Encrypted_Executable_May26 {
  meta:
    description = "Detects executables containing an encrypted embedded payload using parameters c
    author = "Jonathan Peters (cod3nym)"
    date = "2026-05-20"
    reference = "Internal Research"
    hash = "eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71"

```

```

    score = 70
strings:
    // MZ header AES encrypted with key: 1234567890123456 and IV: abcdefghijklmnop
    $op = { ae b6 8d 86 71 f0 a9 c8 90 66 53 31 ef 7f 1f d2 b4 a8 21 bc 39 77 c2 c2 60 db 24 4a 12
condition:
    uint16(0) == 0x5a4d
    and 1 of them
}

```

Hunting with this rule did not identify additional Nimbus Manticore artifacts. However, it did match several other malicious payloads, including AsyncRAT and an unidentified credential stealer.

Although the rule did not identify further campaign-specific samples, the results show that this pattern can still be useful as a generic hunting indicator. In particular, AES-encrypted PE headers with weak, hardcoded cryptographic material remain suspicious and may help identify staged payloads across unrelated malware families.

After decrypting the payload the file is written to disk under the following path:

`\AppData\Roaming\2FAGuard\main.dll` . Giving us an artifact that can be used for detection. Once the payload has been written to disk. The stager copies itself and the `setup.exe.config` file to the same directory.

Next the stager uses the TaskScheduler via COM to create a scheduled task that runs at logon with the least privileges for current user. We suspect the attackers use standard privileges to appear less suspicious. The task executes the copied stager located at `\AppData\Roaming\2FAGuard\setup.exe` with the argument `doit` . This brings us back to the `Execute` function. The scheduled task executes the other flow that loads the main payload without showing the 2FA app. The task name is hardcoded to “BackupCheck”. The hardcoded name and the rather unique argument can also be used as detection vectors.

```

rule SUSP_ScheduledTasks_Nimbus_Manticore_Persistence_May26 {
    meta:
        description = "Detects scheduled task used for persistence by Nimbus Manticore (UNC1549). The
        author = "Jonathan Peters (cod3nym)"
        date = "2026-05-27"
        reference = "Internal Research"
        score = 75
    strings:
        $a0 = "<Task version=" wide
        $a1 = "xmlns=\"http://schemas.microsoft.com/windows/" wide

        $x1 = "<Arguments>doit" wide
    condition:
        uint16(0) == 0xfeff
        and all of them
}

```

THOR APT Scanner can apply YARA rules to scheduled task definitions, which helps analysts quickly identify suspicious task configurations. The rule above has been modified to work with most common Yara scanners by targeting the XML Task definition files.

Next we will explore what happens when the stager is invoked with the `doit` argument. When executed the stager calls `LoadAnCallTestFunction`. Which loads and executes the native main payload.

The function assumes that the `main.dll` is present on disk. Loading it via `LoadLibrary` from the expected path `\AppData\Roaming\2FAGuard\main.dll` and then resolving the exported function via ordinal. Since `main.dll` only exports a single function the ordinal is simply 1. The stager creates a delegate for the native function in order to call it directly from the managed code. It then clears the references to the native module and stops execution. Execution continues in the native payload.

Before analyzing the native implant, we add a separate YARA rule for the stager.

When reviewing earlier reporting on Nimbus Manticore, we found that the operators tend to change the stager between campaigns while keeping the broader execution pattern consistent. They modify cryptographic routines, loader logic, and obfuscation levels, but the overall role of the stager remains the same.

For this reason, the following rule is intentionally specific to the stager used in the campaign we observed. It is intended for campaign-level hunting rather than broad detection of Nimbus Manticore activity.

```
rule MAL_APT_Nimbus_Manticore_Stager_May26 {
  meta:
    description = "Detects .NET based stager using AppDomain Hijacking observed to be used by Nimbus Manticore"
    author = "Jonathan Peters (cod3nym)"
    date = "2026-05-20"
    reference = "https://research.checkpoint.com/2026/fast-and-furious-nimbus-manticore-operations/"
    hash = "eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71"
    score = 80
  strings:
    $x1 = "MyCompany-Product-TOTP-Salt-2024!@# $" wide fullword
    $x2 = "TOTPGuardRunner" ascii fullword
    $x3 = "\\AppDomainInjection-metlifeScenario\\TOTP" ascii fullword

    $sa1 = "EncData" ascii fullword
    $sa2 = "DecryptAndSaveToDesktop" ascii fullword
    $sa3 = "CopyHelloToDesktop" ascii fullword

    $sb1 = "doit" wide fullword
    $sb2 = "DailyTrigger" wide fullword
    $sb3 = "GetTypeFromCLSID" ascii fullword
    $sb4 = "yyyy-MM-ddTHH:mm:ss" wide fullword
  condition:
    uint16(0) == 0x5a4d
    and
```

```
(
    1 of ($x*)
    or all of ($sa*)
    or all of ($sb*)
)
}
```

The implant

The native implant is invoked through the exported function called `CheckForUpdates` this is where the execution begins. At startup, the implant performs a basic environment check: it verifies that its module name is `setup.exe` and that the parent process is not `svchost.exe`, a common anti-sandbox check. If the check fails the implant will abort further execution.

The screenshot shows a less obfuscated variant of the payload as discovered in previous attacks. The sample we analyzed used heavier control-flow obfuscation, including dynamically calculated indirect jumps and opaque predicates. It also contained a large amount of junk code, likely intended to complicate static analysis and slow down reverse engineering. The operators appear to have increased the level of obfuscation after earlier public reporting by other security vendors.

This screenshot shows the `CheckForUpdates` function from `dfa1e3137a032ee8561a1cd5e1a0f71a10bebb36aef7c336c878638a9c1239ee` the sample we analyzed. The code calculates a jump target address using large, seemingly random constants, a base address, and an input parameter. We can also see an example of the opaque predicate obfuscation `v9` is an algebraic tautology evaluating to either 1 or 0 it looks like legitimate logic at first but its only purpose is to waste analysis time, alter the CPU state slightly, and set up the register `rax` for the final indirect jump (`jmp rax`). This is a classic obfuscation technique designed to prevent static analysis tools from identifying and analyzing the target code. The payload also does several inline anti debug checks by checking the Process Environment Block (PEB) for the `NtGlobalFlag` which is usually set by debuggers to enable heap checking. If a debugger is detected execution is aborted.

Apart from the increased obfuscation, the payload's core functionality remains consistent with previous Nimbus Manticore activity reported [in detail by Check Point Research](#). The implant follows the same general execution and C2 communication flow observed in earlier operations.

The C2 addresses and API routes used for command-and-control communication remain unobfuscated. This makes them straightforward to extract during static analysis and provides useful network indicators for detection and infrastructure correlation.

Throughout their campaigns, the operators have repeatedly used Azure-hosted infrastructure for command and control. This allows them to benefit from the trusted reputation of Microsoft cloud services and makes their traffic less likely to stand out in environments where Azure access is expected. The C2 domains are generic and aligned with the campaign theme, making them less suspicious during a quick manual review.

Using these identifiers, we can write a YARA rule that detects both this implant and previously reported variants. The rule focuses on stable C2-related strings, including API routes, the user-agent, and the Azure base domain.

```
rule MAL_APT_Nimbus_Manticore_Agent_May26 {
  meta:
    description = "Detects Nimbus Manticore (UNC1549) agent implant featuring data exfiltration and C2 communication"
    author = "Jonathan Peters (cod3nym)"
    date = "2026-05-28"
    reference = "Internal Research"
    hash = "dfa1e3137a032ee8561a1cd5e1a0f71a10bebb36aef7c336c878638a9c1239ee"
    score = 80
  strings:
    $a1 = "Chrome/146.0.0.0 Safari/537.36" wide
    $a2 = ".azurewebsites.net" wide

    $s1 = "/agent/poll?token=" wide fullword
    $s2 = "/agent/init" wide fullword
    $s3 = "/agent/result" wide fullword
  condition:
    uint16(0) == 0x5a4d
    and 1 of ($a*)
    and 1 of ($s*)
    or 3 of them
}
```

You can also use the from this article rules to scan your own environment with **THOR Cloud Lite**. It is free to use and does not require setting up extra infrastructure.

Sign up, download the launcher, run the scan, and review the results in the cloud-based web interface.

THOR Cloud Lite is a practical option for quickly checking systems with public rules and known indicators. For broader coverage and additional private detection logic, we recommend THOR Cloud or THOR for on premise scanning.

Detection strategies

Domain age can also support detection and mitigation. At the time of our investigation, the domain used for the initial phishing campaign had been registered only ten days earlier.

Defenders should treat freshly registered domains with caution, especially when they are used for authentication flows, file downloads, or externally hosted portals. For many organizations, blocking or temporarily restricting access to newly created domains can be an effective mitigation.

A practical first step is to apply stricter controls in sensitive business areas such as finance, HR, marketing and executive support functions. These teams are frequent targets for phishing and social engineering, and restricting newly registered domains can reduce exposure to recently prepared attacker infrastructure.

For the Azure-hosted C2 domains, we estimated infrastructure age using available certificate data. The certificates were issued on 15 April 2026, which gives us an approximate lower bound for when the infrastructure may have become active. However, certificate issuance alone is not conclusive. Azure-managed certificates can be renewed or reissued, and the underlying domains may have existed before the observed certificate issuance date. Based on the campaign timeline and infrastructure patterns, we nevertheless assess that the domains were likely prepared shortly before use.

The domains we identified closely match C2 infrastructure previously reported by Google and Check Point. They use benign-looking names that align with the lure context and do not immediately stand out during manual review.

This naming pattern makes the infrastructure more difficult to distinguish from legitimate cloud-hosted services, especially in environments where Azure traffic is common.

Additionally, we identified a consistent marker in the PDFs used in the current phishing attempt. This marker can be used to hunt for related lure documents and to identify PDFs likely created by the same operators.

The following YARA rule is intended to detect lure PDFs associated with this campaign.

```
rule SUSP_Nimbus_Manticore_PDF_Indicators_May26 {
  meta:
    description = "Detects indicators found in PDF files created by Nimbus Manticore (UNC1549). The"
    author = "Jonathan Peters (cod3nym)"
    date = "2026-05-25"
    reference = "Internal Research"
    hash = "620c51f4376cb79f0109c21971c28661418ae50b119585e3ffdb8011189fcb7b"
    score = 70
  strings:
    $op = { 3C 3C 2F 41 75 74 68 6F 72 28 4A 65 72 72 79 29 20 2F 43 72 65 61 74 6F 72 28 FE FF 00
  condition:
    1 of them
}
```

The threat actors created the lure PDF with a distinctive author string: `Jerry /Creator Microsoft Word LTSC .` Special characters and spacing were removed here for readability. We used this metadata as a hunting indicator and identified additional lure documents offering positions at Airbus. The hashes are listed in the IOC section at the end of this article.

Prevention

Hunting for artifacts inside the environment is an important step when assuming compromise. However, we believe several measures can help prevent attacks like this or significantly raise the barrier for the operators.

As observed in this and other incidents, phishing is not limited to email. Threat actors increasingly use social media platforms such as LinkedIn, as well as job portals, to approach selected targets and establish trust before delivering malware.

Security awareness programs should reflect this shift. Traditional phishing training often focuses mainly on email-based lures, which can leave employees with an incomplete understanding of social engineering risks. Employees should be trained to recognize suspicious recruiter outreach, externally hosted hiring portals, unusual authentication prompts, and requests to download tools as part of an application process.

As discussed earlier, domain age was a relevant suspicious indicator in this campaign. We recommend establishing a policy to block or restrict access to freshly registered domains. Where full blocking is not feasible, organizations can apply partial restrictions. For example, access to newly registered domains from sensitive or high-risk business functions such as HR, finance, legal, and executive support can require manual approval or additional security review.

Another quickly actionable measure is to use built-in Windows application control features. AppLocker can be configured to block execution from user-writable locations such as:

```
%TEMP%  
%APPDATA%  
%LOCALAPPDATA%  
Downloads
```

This reduces the likelihood that payloads delivered through archives, browser downloads, or fake application installers can execute directly from common staging locations.

Some legitimate applications, especially Electron-based applications, may run from `AppData`. These should not be broadly allowed by path alone. Instead, organizations should approve them on an application-by-application basis, with stricter policies for higher-risk business units such as HR, finance, legal, and marketing.

Conclusion

This incident again shows the risk of hiring-themed social engineering, especially in a tense job market. The operators used trusted brands, professional recruiter personas, and a convincing hiring portal to establish credibility with the victim. The fake but functional 2FA application added another layer to the pretext and helped keep the workflow believable after execution.

The core logic and functionality of the implants have not changed significantly compared to previously reported Nimbus Manticore activity. However, the operators increased the level of obfuscation, making static analysis more difficult and time-consuming. We suspect that with future iterations they will also improve on their current shortcomings.

For defenders, the main challenge remains the combination of targeted social engineering, multi-stage payloads, and abuse of trusted host binaries. Detection should therefore focus not only on static indicators, but also on behavioral patterns such as suspicious sideloading chains, unusual .NET configuration abuse, newly registered domains, and cloud-hosted C2 infrastructure.

To support defenders, we included YARA rules and IOCs for the current campaign with this article. The rules are available throughout the article and on [GitHub](#); the indicators are listed below and are also available in our

[GitHub repository](#).

These public rules are intended to help analysts hunt for artifacts related to this campaign and identify similar activity in their environments. Nextron customers using THOR or Valhalla have access to additional private rules with broader and more resilient coverage for this and similar campaigns. [Learn more about THOR](#).

Hash	Description / Filename
06d12a4c4e3cc725dba37445cebeba41803718ccdb63d9d637355a241f651668	Fake Airbus Job Description PDF
9b63b744dc1f3a24f057a404c5622ed0ca933752a00ce05117727c7d11f05536	Fake Airbus Job Description PDF
620c51f4376cb79f0109c21971c28661418ae50b119585e3ffdb8011189fcb7b	Fake Ebix Job Description PDF
d1f525eb9347133b92e9558e1413558c8348c0f35a62577f60a5192ba38eb776	TOTPGuard.zip
8e5fc0998838559ca8611e6c03fd998a17ffc2eade24715b2fc3e723c712eb8b	setup.exe.config
eee657ffdb2af8ed6412221e7d5fbf4f5742f2ac2c88f43f12db46af0697de71	TOTPGuard.dll
dfa1e3137a032ee8561a1cd5e1a0f71a10bebb36aef7c336c878638a9c1239ee	main.dll
3628d13d2f8af7663d58dd1aa352c8f12d12233a7318ee203f01f195573a2ed2	EbixExam.Desktop.zip
c7ef2ec19d158301773b1590f5b5eeb362a30f725acad8f5b3a230e9f26d14be	EbixExam.Updater.dll
072744ce205bb89a36e563a86f30df5689e64eee75106b97ce708551c8194bbc	EbixExam.Updater.ServiceHub.dll
Domain	Associated Payload
globalitconsultants[.]azurewebsites[.]net	main.dll
globalbusiness-checkers-it[.]azurewebsites[.]net	main.dll
global-check-business-it[.]azurewebsites[.]net	main.dll
global-check-itbusiness[.]azurewebsites[.]net	main.dll
global-it-checkbusiness[.]azurewebsites[.]net	main.dll
global-it-consultants[.]azurewebsites[.]net	main.dll
globalit-consultants[.]azurewebsites[.]net	main.dll
global-it-checkers[.]azurewebsites[.]net	main.dll
business-dns-ns-joiners[.]azurewebsites[.]net	EbixExam.Updater.ServiceHub.dll
ebix-exam-join-from-app[.]azurewebsites[.]net	EbixExam.Updater.ServiceHub.dll
business-joiners-exam[.]azurewebsiets[.]net	EbixExam.Updater.ServiceHub.dll

Domain	Associated Payload
join-exam-now-ebix[.]azurewebsites[.]net	EbixExam.Updater.ServiceHub.dll
Path/Filename	Associated Payload
\AppData\Local\VirtualStore\result.con	main.dll
\CKAConsent.dll	main.dll
\2FAGuard\main.dll	TOTPGuard.dll
\2FAGuard\setup.exe.config	TOTPGuard.dll

Source: https://www.nexttron-systems.com/2026/06/01/detecting-nimbus-manticore-and-their-sideloaded-infection-chains/?hss_channel=tw-2727675700