# RudePanda owns IIS servers like it's 2003

Published: 2025-10-21 · Archived: 2026-04-05 16:26:20 UTC

Published on 21 October, 2025 37min


*Identifier: TRR251001*.

**Summary**

> Late August and early September 2025, our security product detected the compromises of IIS servers with a previously undocumented malicious module which we call “HijackServer”. The associated

> infection chain involved the use of previously exposed ASP .NET machine keys and a ready-made toolset which notably includes a customised but publicly available rootkit.
>
> Investigating the case, we discovered variants of the HijackServer module (a .NET alternative for IIS and a PHP version targeting the Apache server), an extensive operation which infected hundreds of servers around the world, as well as additional and likely associated infrastructure.
>
> While the malicious operators appear to be using Chinese as main language and leveraging the compromises to support search engine optimisation (SEO), we notice that the deployed module offers a persistent and unauthenticated channel which allows any party to remotely execute commands on affected servers.
>
> NB: on the day we were wrapping this blog post up, Elastic Security Labs in cooperation with 2 other organisations released an article which describes the same operation and refers to the IIS modules as "TOLLBOOTH".


- Background: exploiting viewstate and compromising servers for SEO
- Infection chain
  - Overview
  - Initial infection: it's not a secret if it has been public for 20 years
  - Attackers' toolset
  - Rootkit and associated usermode command-line tool
  - GUI deployment tool and associated scripts
- HijackServer – IIS module
  - Overview
  - Features
  - Implementation details on configuration files
  - Additional samples and likely related variants
- Infrastructure
- Targets
- Attribution: similarities with previously reported Larva-25003 activity
- Conclusion: a pool of initial accesses is now available for exploitation
- Appendix: indicators and detection rules
  - Indicators of compromise (IOCs)
  - Yara rules


## Background: exploiting viewstate and compromising servers for SEO

The exploitability of application data which is stored on the client side (eg. in a "viewstate") has been thoroughly documented starting 2010 for ASP .NET. Microsoft has been providing safeguards to mitigate such risk, including improvements and patches over the years. But exploiting the ASP .NET viewstate remains possible if secrets known as "machine keys are known to the attackers.

Early 2025, Microsoft alerted about the abuse of exposed ASP .NET machine keys to execute malicious code through viewstate manipulation, leading to the compromise of IIS servers. At the time of their publication, Microsoft had identified more than 3,000 publicly exposed machine keys in code repositories or programming forums. During the summer of 2025, the exploitation of SharePoint vulnerabilities notably supported the exfiltration of ASP .NET machine keys, demonstrating that the attackers still put an interest in viewstate exploitation.

At the same time, the compromise of Microsoft IIS servers by financially motivated threat actors for Search Engine Optimization (SEO) fraud has been described by several vendors. In 2021, ESET documented several families of IIS malware, some of which aimed to improve the popularity of certain websites by modifying the HTTP responses sent to search engines web crawlers. In September last year, Cisco Talos described the activity of a threat actor they named *DragonRank* who deploys IIS modules known as *BadIIS* in order to boost the visibility of particular websites. In February 2025 and in April 2025, similar activity has been documented by Trend Micro and AhnLab. In that latter case, the threat actor used a rootkit to conceal the presence of the IIS module they had installed.

In recent weeks, several other publications reported the compromise of IIS servers for SEO poisoning, suggesting a growing prevalence of such activity.

## Infection chain

### Overview

Between late August and September this year, our product detected compromises of IIS web servers. Our investigation revealed that the attackers first exploited the ASP .NET viewstate to achieve remote code execution, then leveraged privileges escalation techniques known as “Potatoes” (EfsPotato and DeadPotato) to create an additional “hidden” local administrator ( `admin$` ).

The attackers further dropped a remote access tool (GotoHTTP) to interact with the graphical interface of the compromised servers, and ultimately deployed malicious IIS modules (later referred to as “HijackServer”). The threat actor relied on pre-packaged tools and scripts to partially automate the process, and tried to conceal the presence of deployed modules using a rootkit.

The install process includes a noisy deletion of every single Windows Event log file, which is at odds with the attempt to conceal the presence of modules using a rootkit.

In one case (see Fig. 1), we noticed a failed name resolution during the exploitation process which may have been an attempt from the attackers to communicate with incident responders.


Figure 1 – Profanity

In the following section, we further detail the root cause of the IIS servers compromises – an issue which we believe affects a significant number of organisations – and later describe the toolset used by the attacker.

### Initial infection: it’s not a secret if it has been public for 20 years

As part of the malicious IIS modules deployment process, the attackers leveraged a script to delete IIS log files in the standard location. However, in both cases we observed, the targeted IIS web applications were setup to save logs in custom locations, so they were still available.

The attackers exploited ASP .NET web applications to initially execute ASP payloads on targeted servers. We could not retrieve these payloads, but our security product still provided relevant information. The available logs for corresponding applications included multiple lines of suspicious POST requests at the time the first malicious activities were detected. The HTTP requests had various user agents (the client language was set to `zh-tw` when specified) and targeted the root pages ( `/` ) of the applications:

```
 2025-08-21 XX:Y1:46 <IP1> POST / - 80 - <IP2> Mozilla/5.0+(Windows;+U;+Windows+NT+6.1;+en-US;+rv:1.9.2.13)+Geck
2025-08-21 XX:Y1:48 <IP1> POST / - 80 - <IP3> Mozilla/5.0+(Macintosh;+Intel+Mac+OS+X+10_6_0)+AppleWebKit/537.4+(
2025-08-21 XX:Y2:08 <IP1> POST / - 80 - <IP2> Mozilla/5.0+(Windows+NT+6.1)+AppleWebKit/537.2+(KHTML,+like+Gecko)
2025-08-21 XX:Y2:10 <IP1> POST / - 80 - <IP3> Mozilla/5.0+(Macintosh;+Intel+Mac+OS+X+10_6_0)+AppleWebKit/537.4+(
[...]
2025-08-21 XX:Y4:15 <IP1> POST / - 80 - <IP2> Mozilla/5.0+(Windows;+U;+Windows+NT+6.1;+en-US;+rv:1.9.2.13)+Gecko
2025-08-21 XX:Y4:21 <IP1> POST / - 80 - <IP3> Mozilla/5.0+(Windows;+U;+Windows+NT+6.1;+en-US;+rv:1.9.2.13)+Gecko
2025-08-21 XX:Y5:04 <IP1> POST / - 80 - <IP2> Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+Trident/5.0)+ch
```

The only data submission path that existed for such root pages is a default ASP .NET viewstate form:

```
 <form method="post" action="./" id="aspnetForm">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="[REDACTED]" />
</div>
```

Further analysing the configurations of the ASP .NET applications ( `Web.config` ), we could retrieve the secrets that are used to validate and encrypt the associated viewstates:

```
 <machineKey
    validationKey="C50B[TRUNCATED]"
    decryptionKey="8A9B[TRUNCATED]"
    validation="SHA1"
/>
```

We were very surprised to find that the said “secrets” were actually examples from a MSDN help page which was already publicly available in 2003:

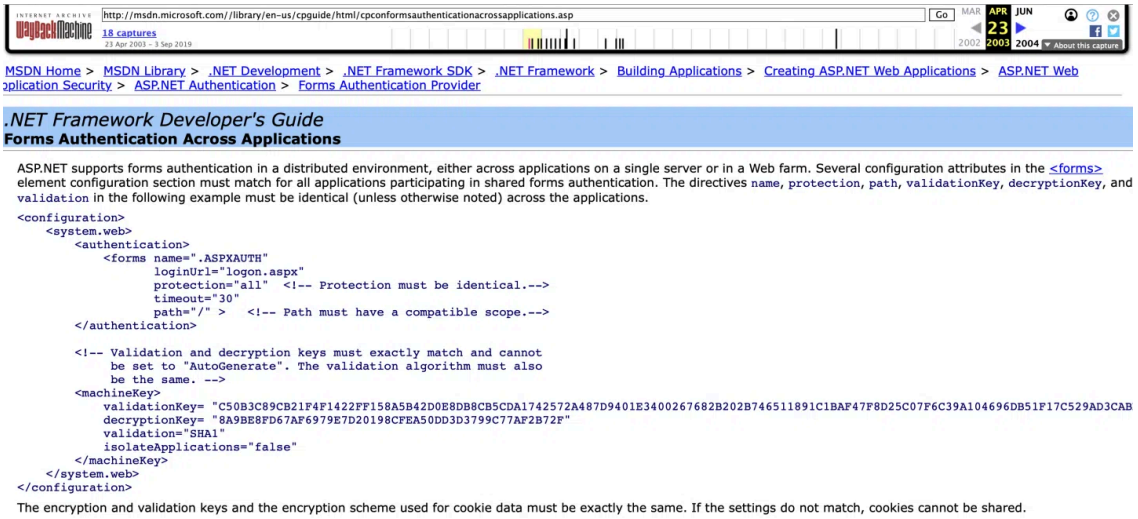


Figure 2 – MSDN article originally referencing the exploited encryption secrets

As we previously mentioned in the Background section, attackers can trivially exploit ASP .NET viewstate deserialization to remotely execute code if they know the associated cryptographic secrets.

It should be noted that this original help page (see Fig. 2) reads "*The directives name, protection, path, validationkey, decryptionKey, and validation in the following example must be identical (unless otherwise noted)*". This has likely and unfortunately been applied to the letter by many IIS users for years, as it can further be confirmed seeing how many times this configuation snippet was suggested on various public forums, including StackOverflow.

### Attackers' toolset

As mentioned before, the attackers relied on pre-packaged tools and scripts to partially automate the infection process. This toolset was initially deployed on a targeted server as a ZIP archive, which is described below.

| | |
|---|---|
| Filename | `sys-tw-v1-6-1-clean-log.zip` |
| File type | Zip archive data, at least v2.0 to extract, compression method=store |
| Hash (SHA-256) | `7cc8b4206e87788b8403500f37bb8b5cfb71d3c26d49365ccc9c36b688c7428a` |

The `sys-tw-v1.6.1-clean-log.zip` archive contains the following folders and files, some of which are further described in the following sections, and the most recent files are dated from August 20, 2025:

- in the `IIS` folder:
  - `x86.dll` and `x64.dll` : 32 and 64-bit malicious IIS modules (HijackServer);
  - `install.bat` : a script which installs the IIS modules;
  - `install_bak.bat` and `install_bak - \u00a9\u2592\u2592\u00a5.txt` (identical files): install script, not used in the cases we observed, possibly an older variant;
  - `hello.bat` : install script, not used in the cases we observed, possibly an older variant;
  - `heoo.docx` : document listing the same commands as those found in `hello.bat` ;
  - `uninstall.bat` : an uninstall script, which unloads and deletes the IIS modules;

- in the `x86` folder:
  - `Wingtb.sys` : 32-bit Windows kernel driver serving as a rootkit;
  - `Hidden.inf` : setup information file for the driver;
  - `Wingtb.cat` : catalog file for the driver;
- in the `x64` folder: similar files than in the `x86` folder but for a 64-bit variant of the rootkit;
- `WingtbCLI.exe` : a usermode client command-line tool for the included rootkit;
- `HijackDriverManager.exe` : GUI deployment tool which is aimed at easing the install process of the malicious IIS modules;
- `lock.bat` : a post-installation script which is run once the IIS modules have been deployed, in order to hide deployed files and delete logs.

The `heoo.docx` document has a first modification date set to 2024-11-09, and was uploaded to an online multiscanner in December 2024, indicating that parts of the same toolkit were likely in use in late 2024 already. The last modifying user of the Microsoft document is set to `807751673[@]qq.com`.

### Rootkit and associated usermode command-line tool

| | |
|---|---|
| Filename | `Wingtb.sys` |
| File type | PE32+ executable (native) x86-64, for MS Windows |
| Compilation time | 2024-11-05 07:33:46 UTC |
| Hash (SHA-256) | `f9dd0b57a5c133ca0c4cab3cca1ac8debdc4a798b452167a1e5af78653af00c1` |

The `Wingtb.sys` file is a signed Windows kernel driver whose original file name is `Winkbj.sys` and which serves as a rootkit. Its companion setup information file ( `Hidden.inf` ) defines the corresponding service name as “Wingtb”. An associated usermode command-line tool which is named `WingtbCLI.exe` (SHA-256 `913431f1d36ee843886bb052bfc89c0e5db903c673b5e6894c49aabc19f1e2fc` ) can interact with the driver.

It appears that both the rootkit and its associated usermode tool are derived from the open-source “Hidden” project. Hidden is mainly aimed at hiding artefacts (such as files, registry keys and processes) from a Windows system, and must first be “enabled” (activated) to operate. Among the differences observed in our samples, the following changes have been made, likely to simplify their use for an operator:

- the names of the commands of the usermode tool have been translated from English to Chinese (transliterated into the Latin alphabet);
- an additional `/shanchu` command has been included in the usermode tool, and may be intended to delete files and directories – but the corresponding feature is not implemented in our samples of the driver;
- logging messages have been customised in the driver.

The driver is signed with an expired code-signing certificate which is issued to “Anneng electronic Co. Ltd.” (thumbprint: `9A6EE51A6A437603ACEE9ADC5F1A5F13329A7E59` ) and was valid from 2013-05-06 00:00:00 to 2014-05-06 23:59:59. However, this certificate meets the requirements that are set by Microsoft for exceptions to its driver

signing policy, and, as a result, may still be loaded on Windows systems: "*Driver was signed with an end-entity certificate issued prior to July 29th 2015 that chains to a supported cross-signed CA*".

We identified more than 400 samples that were signed with the same certificate, most of them being drivers, including 10 samples of the Hidden rootkit. In addition, we also found other samples of the `WingtbCLI.exe` command line tool (see Appendix).

### GUI deployment tool and associated scripts

| | |
|---|---|
| Filename | `HijackDriverManager.exe` |
| File type | PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows |
| Compilation time | 2099-04-12 16:40:58 UTC |
| Hash (SHA-256) | `7260f09e95353781f2bebf722a2f83c500145c17cf145d7bda0e4f83aafd4d20` |

`HijackDriverManager.exe` provides a GUI which facilitates the installation (and removal) of the malicious IIS modules, as well as the concealement of associated artefacts. The tool does so by by running scripts and executing a limited subset of the commands that are provided by the rootkit's usermode command-line tool ( `WingtbCLI.exe` ). The GUI also offers buttons, text fields and dialogs to browse folders and copy files:


Figure 3 – Main windows for the GUI deployment tool

The `一键部署` ("One-click deployment") button runs the `IIS\install.bat` script to deploy malicious IIS modules, while `卸载插件` ("Uninstall the plugin") runs the `IIS\uninstall.bat` script to remove them.

`启用保护` enables the rootkit by running the `/yinshen on` command from `WingtbCLI.exe` . Prior to this execution, it starts the service which is supposedly associated with the driver ( `sc.exe start winkbj` ) and runs the post-installation ( `lock.bat` ) script. However, the name of the service which is associated to the driver has been

changed in the analyzed toolkit ( `wingtb` intead of `winkbj` ), so this `sc` command will have no effect (but a command in `lock.bat` will successfully load the driver).

The `锁定` button hides a previously selected file by executing the `/xiaoshi file <file-path>` from the rootkit usermode tool, while the `解锁全部` button stops hiding all previously hidden files by executing `/buxiaoshi file all` .

`暂停保护` (“Suspend protection”) and `结束保护` (“End protection”) disable the rootkit by running the `/yinshen off` command from the usermode tool. In addition, clicking on `结束保护` executes `sc.exe stop winkbj` to stop the service. However, as with the hardcoded command which is issued to start the service, this command will have no effect.

**IIS modules installation script**

Executing `IIS/install.bat` (SHA-256 `e6a9bf90accf17355a1f779d480a38838b2bbb2877cde095c7c139e041c50d71` ) performs the following operations:

- stop the IIS server (using `iisreset /STOP` );
- delete IIS log files in their standard location ( `C:\inetpub\logs\LogFiles\` ), in a likely attempt to remove traces of the initial infection;
- copy the IIS application configuration ( `%windir%\System32\inetsrv\config\applicationHost.config` ) in `%windir%\SysWOW64\inetsrv\Config\` . This is likely done to ensure the later IIS module installation commands work for both the 32 and 64-bit modules regardless of the environment (the installation commands will result in modifications of `applicationHost.config` );
- deploy the malicious IIS modules (HijackServer):
  - copy the 32 and 64-bit malicious IIS module files as `scripts.dll` and `caches.dll` namely, in both `%windir%\System32\inetsrv\` and `%windir%\SysWOW64\inetsrv\` folders;
  - modify access control lists (using `icacls` ) on module files to grant everyone a full access on them;
  - install the malicious modules in IIS as `ScriptsModule` and `IsapiCachesModule` namely (using the `appcmd.exe install module` command);
- prepare the HijackServer working directory:
  - modify access control lists (using `icacls` ) on `%windir%\Temp` to grant regular users and IIS users a full access on it;
  - create `C:\Windows\Temp\_FAB234CD3-09434-8898D-BFFC-4E23123DF2C` (the working directory for HijackModule) and modify its access control lists as for `%windir%\Temp` ;
- setup Windows to store plaintext logon credentials in memory (by setting the `UseLogonCredential` value for the `WDigest` registry key). This might be done to facilitate later attempts to gather credentials from the compromised server;
- start the IIS server again (using `iisreset` ).

**IIS modules removal script**

Executing `IIS/uninstall.bat` (SHA-256 `a96e1643dedd472e5712282904110ee948592fab722dc87d8f1e7658d3d8449d` ) performs the following operations:

- stop the IIS server;
- uninstall the malicious modules from IIS (using the `appcmd.exe uninstall module` command);
- delete the 32 and 64-bit malicious IIS modules files in both `%windir%\System32\inetsrv\` and `%windir%\SysWOW64\inetsrv\`;
- start the IIS server again.

**Post-installation script**

Running the `lock.bat` (SHA-256 `8ed76396e11d1c268b6a80def8b57abacf4ea1ac059838bd858c8587c26b849c`) script performs the following operations:

- load the rootkit, by starting the `wingtb` service;
- hide the following files (by running `WingtbCLI.exe /xiaoshi file <file-path>`):
  - the IIS modules files as deployed;
  - the custom IIS error pages `403.htm`, `404.htm` and `500.htm` (in `C:\inetpub\custerr\en-US\`). Those might be replaced by backdoored files in some circumstances, but were not found in the cases we observed;
  - the IIS applications configuration (as it is modified during the installation of malicious modules), `C:\Windows\System32\inetsrv\config\applicationHost.config`;
  - the driver file as deployed;
- hide the registry key which is associated to the rootkit service (by running `WingtbCLI.exe /xiaoshi regkey HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Wingtb`);
- actually enable the rootkit (by running `WingtbCLI.exe /yinshen on`;
- delete every Windows Event log file (`for /f "tokens=*" %%1 in ('wevtutil el') do wevtutil cl "%%1"`).

## HijackServer – IIS module

| | |
|---|---|
| Filenames | `caches.dll`, `x64.dll` |
| File type | PE32+ executable (DLL) (GUI) x86-64, for MS Windows |
| Compilation time | 2025-08-12 06:23:42 |
| Hash (SHA-256) | `c1ca053e3c346513bac332b5740848ed9c496895201abc734f2de131ec1b9fb2` |

This is a malicious module for IIS server. Its main purpose appears to be SEO for dubious cryptocurrencies investment schemes… but it also enables unauthenticated remote command execution, turning it into an easily actionable backdoor.

### Overview

The module uses `HijackServer`, `Hijackbot` and `hj-plugin-iis-cpp-v1.6.1` (`hj` likely standing for “hijack”) as most distinctive internal names – we will later refer to the module as “HijackServer” in consequence, and consider the analyzed sample is of version 1.6.1.

HijackServer is a "native" IIS module, developed using C++, which hooks all HTTP requests for all applications (provided it is a globally installed module) at the very first stage of their processing by the IIS server ( `GL_PRE_BEGIN_REQUEST` ). Should the global hooking fails, the module registration also hooks requests for the current IIS application only, at the request ( `RQ_BEGIN_REQUEST` ) and response stages ( `RQ_SEND_RESPONSE` ) – whichever works first.

The operators of HijackServer use the content of HTTP requests (URL path, user agent and referer headers) to the compromised IIS server as a command and control (C2) channel. The malicious module is flexible by design, and several of its features can be controlled by a JSON configuration file, text lists and HTML templates, which are all downloaded from external staging servers ( `c.cseo99[.]com` , `f.fseo99[.]com` ). Downloaded files are stored in a working directory ( `C:\Windows\Temp\_FAB234CD3-09434-8898D-BFFC-4E23123DF2C\` ).

The main purpose of HijackServer appears to be Google search engine optimisation for questionable cryptocurrencies-related websites. HijackServer answers HTTP requests that come from Google to most[1] URL paths on the compromised server with dynamically generated HTML pages.

```
 <a href="/market-outlook/Is-KDLYW-stock-a-smart-retirement-pick">Is KDLYW stock a smart retirement pick </a>
<a href="/blank/Why-is-MLYS-stock-going-down">Why is MLYS stock going down </a>
<a href="/video/What-analysts-say-about-MBAVW-stock">What analysts say about MBAVW stock </a>
<a href="/bullish-on/Should-I-hold-or-sell-AleAnna-Inc.-Equity-Warrant-now">Should I hold or sell AleAnna, Inc.
<a href="/video/What-are-the-analyst-revisions-for-ZIMVIs-AEye-Inc.-Equity-Warrant-stock-a-good-investment-in-YE
<a href="/blank/Can-CCBG-beat-the-S&P-500">Can CCBG beat the S&P 500 </a>
```

Those HTML pages contain links on various investment-related sentences to specifically crafted redirection URLs on the same compromised server. Those redirection URLs in turn lead to questionable cryptocurrency-related websites.

```
 <!DOCTYPE html>
<html>
<head>
    <script>
        window.location.href = 'hxxps://j.betkr[.]cc/';
    </script>
</head>
<body></body>
</html>
```

It should be noted that this Google SEO approach seems to be working to some extent, as can be witnessed by searching for some sentences or keywords amongst those that appear in generated SEO HTML pages:

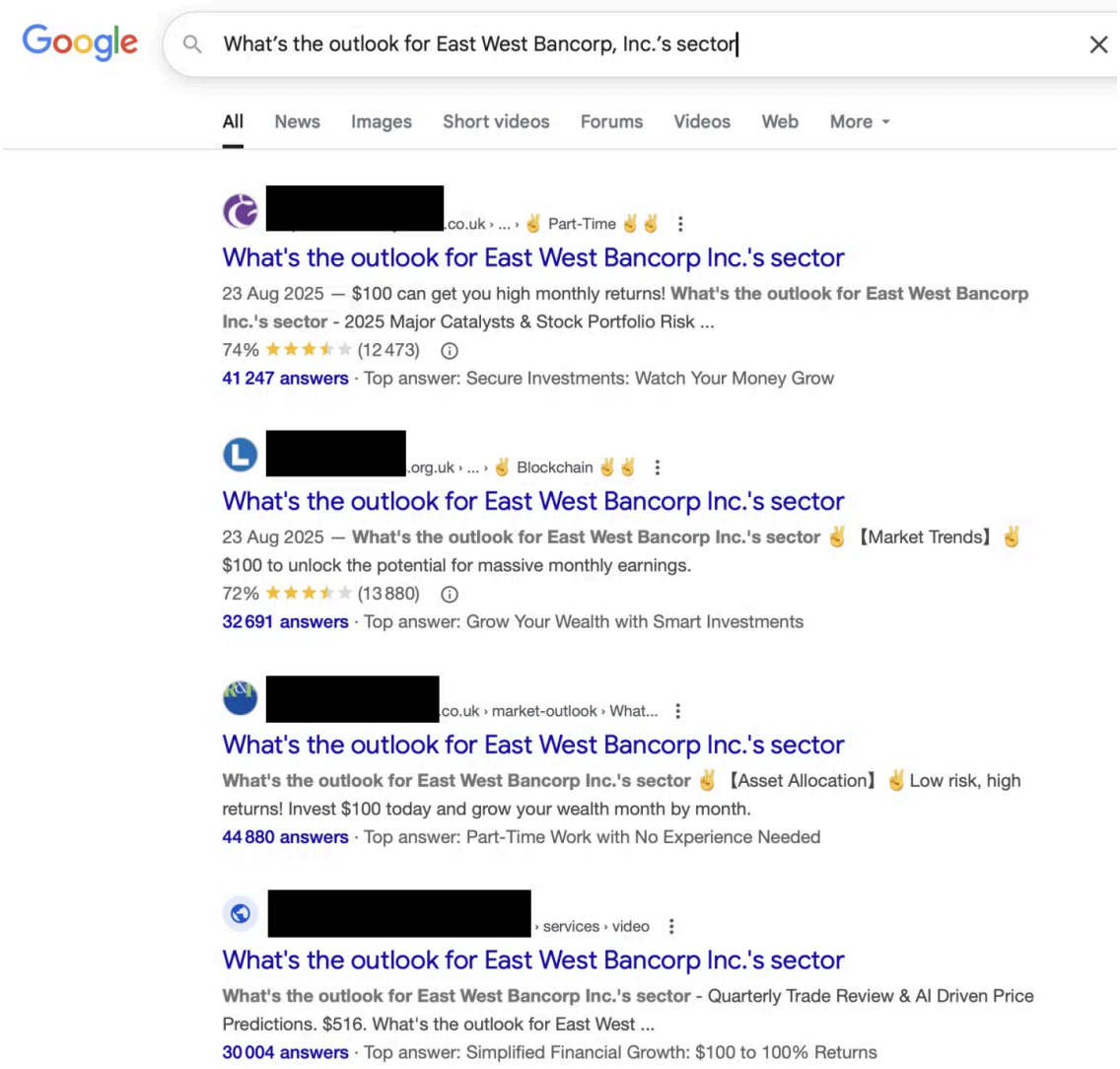


Figure 4 – Google SEO results

## Features

HijackServer ultimately offers the following functionalities (which are implemented in distinct C++ classes of a common interface matching the given names):

**AffLinkServer**: generate SEO HTML pages to answer HTTP requests that come from Google (based on the `User-Agent` header value). HTML pages contain links to redirection URL paths. Each HTML page generation is logged to a C2 API endpoint ( `api.aseo99[.]com` ), using a JSON document.

**RedirectServer**: generate HTML/JavaScript redirections to external websites, depending on URL and if the referer of the HTTP request is Google (based on the `Referer` header value).

**UploadServer**: file upload and Windows shell command execution capability, which is aimed to be exposed through an HTML form ( `/mywebdll` URL path, see Fig. 5), and should be password protected – but the shell command execution capability can be triggered without any sort of authentication when bypassing the form ( `/scjg` URL path).

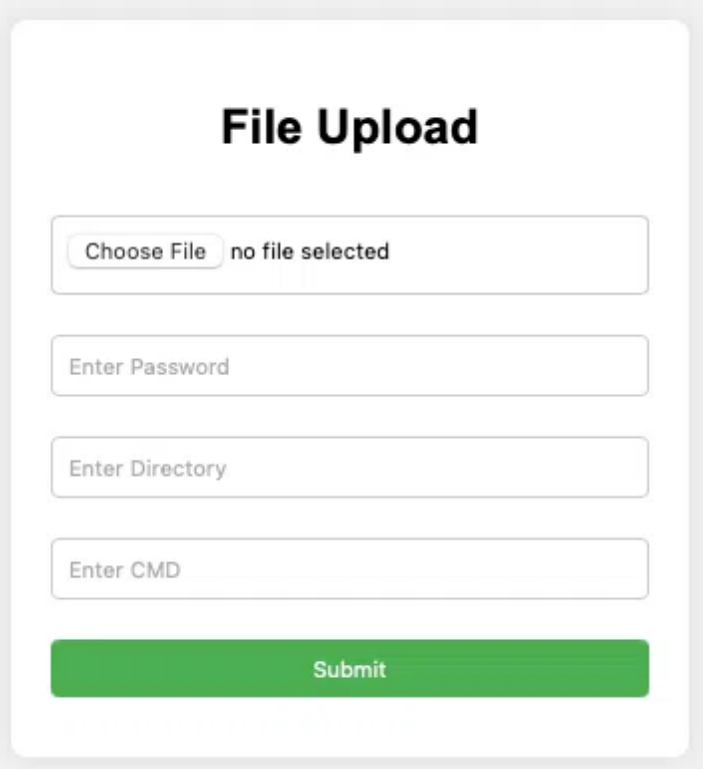


Figure 5 – File upload form from the IIS HijackServer module

The UploadServer class can additionally process HTTP requests targeting a `/xlb` path, by returning an HTML page which loads a JavaScript from `mlxya.oss-accelerate.aliyuncs[.]com` (such ressource is delivered from Alibaba content delivery network). Unfortunately we could not retrieve the associated payload if any.

**WebdllServer**: load and execute ASP .NET payloads from the `C:\inetpub\wwwroot\` folder (`/web.dll` URL path). Payloads can be uploaded (as ASP .NET libraries) with the previously described arbitrary file upload capability (UploadServer).

**HijackServer**: management interface for the operators of the module. It is exposed to HTTP requests that use a specific User Agent header value (amongst a set predefined but dynamically modifiable values). It offers the following features:

- retrieving detailed HTTP request and module state information (`/well-known/acme-challenge/<predefined base64 value>` or `/debug` URL paths);
- retrieving elementary information about available disk space, version and module status (`/health` URL path);
- retrieving the currently implemented configuration (`/conf` URL path);
- deleting files that have been stored in the working directory (`/clean` URL path, with additional `type` query parameter to delete `all`, `conf` or `tmp` files).

### Implementation details on configuration files

The downloaded files that are stored in the working directory (including configuration files) are Zlib-compressed (with additional 4 random bytes before the Zlib header).

Configuration for a given host is downloaded from `hxxps://c.cseo99[.]com/config/<hostname>.json`, where `<hostname>` is the hostname (or IP address) that has been reached to access the underlying application on the compromised IIS server. HijackServer attempts to download (and possibly update) the configuration several times upon failure, and for each HTTP request that is processed – which makes it an incredibly noisy and slow module.

Configuration files are stored to `<working directory>/conf/<MD5 hash>`, where `<MD5 hash>` is the MD5 hash of `<hostname>.json`. A default configuration is further copied from the first successfully downloaded configuration file to `<working directory>/conf/f19ae30a014229b59e40b60ef1b7ee44`, where `f19ae30a014229b59e40b60ef1b7ee44` is the MD5 hash for `system.json`.

Configuration files further points to HTML templates and words list that are hosted at `f.fseo99[.]com`. Associated content is download as needed, and stored under `<working directory>/tmp/` and `<working directory>/remote/`, using MD5 hashes for filenames.

It should also be noted that in configuration files we retrieved, all variable names for SEO templates are written in Chinese (for instance: `{网页模版}` – “Web template”, or `{正文内容}` – “Main content”).

### Additional samples and likely related variants

On top of the 2 files that were provided in the analyzed toolkit, we could identify additional samples of the same HijackServer module:

| Hash (SHA-256) | Bitness | Compilation Date | Internal Version |
|---|---|---|---|
| `c348996e27fc14e3dce8a2a476d22e52c6b97bf24dd9ed165890caf88154edd2` | 32-bit | 2025-02-08 | 1.6.0 |
| `bd2de6ca6c561cec1c1c525e7853f6f73bf6f2406198cd104ecb2ad00859f7d3` | 64-bit | 2025-02-08 | 1.6.0 |
| `7a10207a430234b448f692a534cea16d400858c5fdda014c786fbf97127dce88` | 64-bit | 2024-11-25 | 1.4.0 |

All of those samples use `iismodrqf.dll` as an internal name and `Dongtai.pdb` as a PDB filename like the ones we analyzed. Most samples use the same configuration staging servers than the sample we analyzed (`c.cseo99[.]com`), except for `7a10207a430234b448f692a534cea16d400858c5fdda014c786fbf97127dce88` which uses `f.zseo8[.]com`.

We identified another sample (SHA-256 `64d0a4703ec976b0e0db4e193b9ccdf4ef6f34d24c32274579ee028a67bfa3a9`) which also uses `iismodrqf.dll` as an internal name and implements common features, but is not exactly HijackServer. As its compilation date is set to 2024-05-07, we believe with low to medium confidence it is a previous variant of C++ HijackServer, or a another variant which is based on a common source code. We note that this sample is mentioned as a reference in a public Yara rule (`Malware_IIS_Dongtai_Module`) from PwC, along with 2 other samples which do not appear to be variants of HijackServer but also use `Dongtai.pdb` as a PDB filename.

We also identified a .NET-developed variant of HijackServer (SHA-256 `915441b7d7ddb7d885ecfe75b11eed512079b49875fc288cd65b023ce1e05964`, which according to its internal version

string ( `hj-iis-cim-v1.6.1` ) matches the version of the C++ sample we initially analyzed. This .NET alternative also uses `c.cseo99[.]com` as a staging server.

Last but not least, we identified a PHP downloader (SHA-256 `665234a6627269ba0b3816a6a29ede4fc72d36f34978f5ba1410e63d968d3d62` ) for a PHP version of HijackServer. The latter is destined to be deployed on Apache servers with PHP on Windows. PHP HijackServer is loaded from `php.ini` using the PHP `auto_prepend_file` directive, and is delivered in a Base64-encoded and compressed form from one of the following URLs:

- `hxxps://f.zseo8[.]com/uploads/2024-10-24/48c3a008cd9ccfa5fd3bdb69ed6d12ce.txt` . This PHP HijackServer sample (SHA-256 `e3bfd9aca49726556f6279aad2ab54ca9c1f0df22bcad27aa7e1ba3234f8eaff` ) is internally named “hj-plugin-php-systmp” and is of version “1.2.5”. Its configuration staging server is set to `c.cseo8[.]com` , and its C2 API to `api.xseo8[.]com` (the latter is only used to report an absence of configuration file).
- `hxxps://f.zseo8[.]com/uploads/2024-10-24/99749da89ec4d1e3f3179f119f2a955b.txt` . This PHP HijackServer sample (SHA-256 `e107bf25abc1cff515b816a5d75530ed4d351fa889078e547d7381b475fe2850` ) is named “hj-plugin-php-forcehttp” and is of version “1.2.51”. It is virtually the same than the previous sample, except that all communications with staging servers are done using HTTP (instead of HTTPS).

## Infrastructure

Domain names of the staging servers of the C++, PHP and .NET variants of HijackServer are all registered with “Dominet (HK) Limited” (which is a name provider for Alibaba Cloud) or with “Eranet International Limited” in Hong-Kong, and point to Cloudflare infrastructure.

| Domain | Registration | Stagers resolution (during known activity) |
|---|---|---|
| `cseo99[.]com` | 2025-01-02, Dominet (HK) Limited | Cloudflare |
| `fseo99[.]com` | 2025-01-02, Dominet (HK) Limited | Cloudflare |
| `aseo99[.]com` | 2025-01-02, Dominet (HK) Limited | Cloudflare |
| `cseo8[.]com` | 2024-08-01, Eranet International Limited | Cloudflare |
| `zseo8[.]com` | 2024-08-01, Eranet International Limited | Cloudflare |
| `xseo8[.]com` | 2024-08-01, Eranet International Limited | Cloudflare |

Error pages that are returned by staging servers that are still online show a next hop behind Cloudflare that is likely hosted by Alibaba cloud:

```
 <?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The specified key does not exist.</Message>
  <RequestId>[REDACTED]</RequestId>
```

```
  <HostId>c.cseo99.com</HostId>
  <Key>[REDACTED]</Key>
  <EC>0026-00000001</EC>
  <RecommendDoc>https://api.alibabacloud.com/troubleshoot?q=0026-00000001</RecommendDoc>
</Error>
```

We noticed that the additional staging servers that appear in the possibly related but older HijackServer variant (SHA-256 `64d0a4703ec976b0e0db4e193b9ccdf4ef6f34d24c32274579ee028a67bfa3a9`) also exhibit similar characteristics:

| Domain | Registration | Stagers resolution (during known activity) |
|---|---|---|
| `gov[.]land` | 2024-01-23, Eranet International Limited | Cloudflare |
| `cn[.]lol` | 2023-03-15, Eranet International Limited | Cloudflare |
| `org[.]cfd` | 2023-03-15, Eranet International Limited | Cloudflare |

Finally, looking for domains which matched a similar name pattern and that were registered from known registrars, we identified the following ones. We assess, with medium confidence, that these domains were or will be leveraged as staging servers for HijackServer:

| Domain | Registration | Possible role |
|---|---|---|
| `lseo99[.]com` | 2025-05-30, Dominet (HK) Limited | Unknown |
| `jseo99[.]com` | 2025-05-21, Dominet (HK) Limited | SEO telemetry/reporting endpoint |
| `wseo99[.]com` | 2025-01-02, Dominet (HK) Limited | Unknown |
| `wseo88[.]com` | 2024-12-15, Dominet (HK) Limited | Unknown |
| `fseo88[.]com` | 2024-12-15, Dominet (HK) Limited | Files stager |
| `cseo88[.]com` | 2024-12-15, Dominet (HK) Limited | Configuration stager |
| `aseo88[.]com` | 2024-12-15, Dominet (HK) Limited | Unknown |
| `wseo8[.]com` | 2024-08-01, Eranet International Limited | Unknown |

## Targets

The described malicious IIS module (HijackServer) specifically processes HTTP requests from predefined “User-Agent” values and to predefined URLs, but also generates identifiable SEO pages. We looked for such specific signatures in the wild in late August and early September, and could reliably identify IIS servers that were compromised by HijackServer.

We identified 171 distinct instances of the HijackServer module, affecting websites on approximately 240 server IP addresses (see Fig. 6) and 280 domain names. It should be noted that counting “compromised servers” from these statistics is not trivial, as a single compromised server with a single HijackServer module can affect multiple websites. Conversely, a single server can sometimes be reached through different IP addresses, or several IIS reverse proxies which could all be compromised by HijackServer.

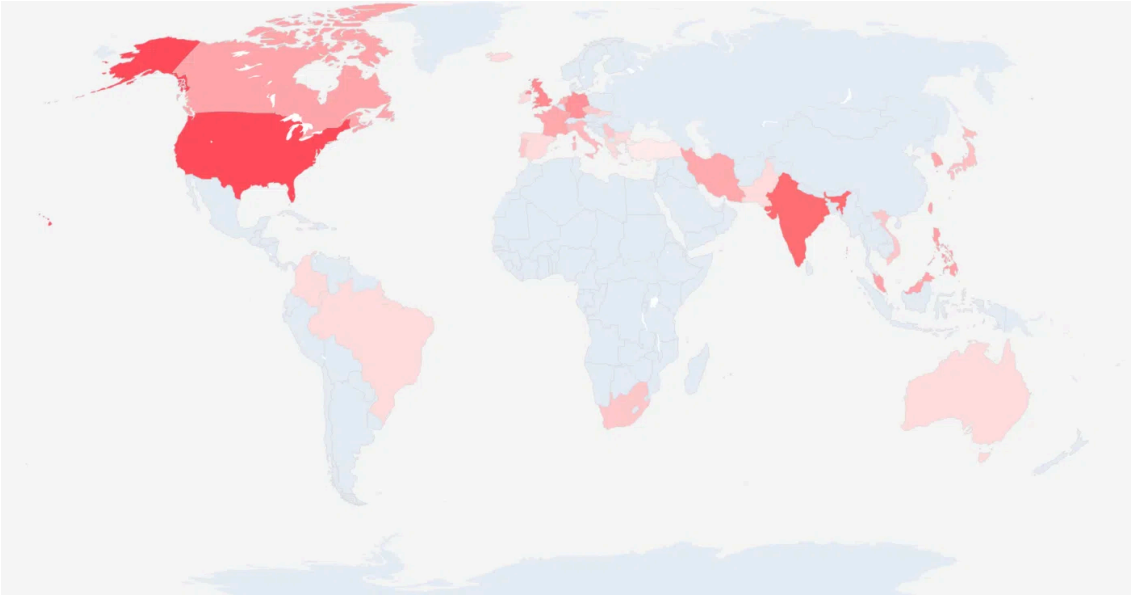

Figure 6 – Location of the IP addresses of the compromised servers we identified. The location is provided by public databases as of September 2025

The oldest instance of HijackServer that we could identify is installed on a server whose IP address is geolocated in Singapore, and appears to be a legitimate website. The installation date in its configuration is set to 2024 (but the same server might still have been compromised before with an older version of the module).

HijackServer does not appear to target any specific vertical and affect very distinct types of websites from various organisations, including small online shops, personal websites, SMBs websites and government websites. This indicates that compromises might be opportunistic: the threat actor might just be targeting servers that are vulnerable.

## Attribution: similarities with previously reported Larva-25003 activity

The `HijackDriverManager.exe` tool provides the user with a GUI in Chinese and allows to perform the install process of the IIS modules and to interact with the rootkit. In addition, the command-line tool used to interact with the latter, derived from an open source project, has been adapted to offer commands in Chinese (transliterated into the Latin alphabet). For these reasons, we believe that the IIS modules, as well as the helper scripts and tools, are intended to be distributed to Chinese speaking users.

We noticed similarities with the tools used by a threat actor tracked as Larva-25003 by AhnLab following the compromise of an IIS web server. In particular, the rootkit we retrieved in the archive left by the threat actor is identical to the one mentioned by AhnLab in their blog post from April 30, 2025. In addition, the IIS module they described is similar to the samples we retrieved in the threat actor’s toolset.

However, due to limited visibility, we do not know if the same operator is responsible for the intrusion described by AhnLab and the compromises we observed. We also noticed some differences such as the use of GotoHTTP for remote access instead of Gh0st RAT.

## Conclusion: a pool of initial accesses is now available for exploitation

The HijackServer deployment cases initially appeared as benign, opportunistic and financially motivated compromises to us. Operators with very limited skills used known exploitation techniques and deployed ready-made tools on the IIS servers of a seemingly non-strategic organisation, to facilitate cryptocurrency scams. Our security product quickly detected the multiple steps of the infection attempts.

Further analysing the initially identified toolset, we nonetheless discovered several variants of the HijackServer module, as well as an extensive operation which affected hundreds of servers around the world in a relatively short timespan. The threat actor also tried to maintain and conceal accesses to the IIS servers, leveraging a customised variant of a publicly available rootkit. The overall operation demonstrates a level of determination and capability that is aligned with that of an organisation. The latter could however still largely rely on poorly skilled operators.

Whatever the threat actor's goal, the operation is effective and leaves hundreds of servers exposed to unauthenticated and trivial remote command execution, even if the initially exploited vulnerabilities have been patched. Any third party, whether cooperating with the threat actor or not, could now profit from such compromises, for espionage or malicious infrastructure development.

We should remind all IIS servers administrators and owners again that it is still of utmost importance to rotate all "machine keys" of all Internet-facing ASP.NET applications and/or IIS servers. After decades of secrets reuse and recent secrets extractions (notably through Sharepoint vulnerabilities exploitation), configurations of IIS servers should be analyzed to identify suspicious modules.

## Appendix: indicators and detection rules

### Indicators of compromise (IOCs)

Associated IOCs are also available on our GitHub repository.

#### Hashes (SHA-256)

```
 82b7f077021df9dc2cf1db802ed48e0dec8f6fa39a34e3f2ade2f0b63a1b5788|IIS HijackServer C++ module 1.6.1, x86.dll and
c1ca053e3c346513bac332b5740848ed9c496895201abc734f2de131ec1b9fb2|IIS HijackServer C++ module 1.6.1, x64.dll and
c348996e27fc14e3dce8a2a476d22e52c6b97bf24dd9ed165890caf88154edd2|IIS HijackServer C++ module 1.6.0
bd2de6ca6c561cec1c1c525e7853f6f73bf6f2406198cd104ecb2ad00859f7d3|IIS HijackServer C++ module 1.6.0
7a10207a430234b448f692a534cea16d400858c5fdda014c786fbf97127dce88|IIS HijackServer C++ module 1.4.0
915441b7d7ddb7d885ecfe75b11eed512079b49875fc288cd65b023ce1e05964|IIS HijackServer .NET module 1.6.1
665234a6627269ba0b3816a6a29ede4fc72d36f34978f5ba1410e63d968d3d62|PHP downloader for Apache HijackServer PHP modu
e3bfd9aca49726556f6279aad2ab54ca9c1f0df22bcad27aa7e1ba3234f8eaff|Apache HijackServer PHP module 1.2.5
e107bf25abc1cff515b816a5d75530ed4d351fa889078e547d7381b475fe2850|Apache HijackServer PHP module 1.2.51
7260f09e95353781f2bebf722a2f83c500145c17cf145d7bda0e4f83aafd4d20|GUI deployment tool, HijackDriverManager.exe
```

```
8ed76396e11d1c268b6a80def8b57abacf4ea1ac059838bd858c8587c26b849c|Post-installation script, lock.bat
913431f1d36ee843886bb052bfc89c0e5db903c673b5e6894c49aabc19f1e2fc|Usermode command-line tool for rootkit, WingtbC
4e24349b61c5af60a5e7f543c86963087ca6d6078378f83c8fe55b36dc6331f4|Usermode command-line tool for rootkit, WinkbjC
5113d2da6cd9f4a4a9123a3547b01250659dcc349c36159ee11b93805ce51105|Usermode command-line tool for rootkit, WinszBu
e6a9bf90accf17355a1f779d480a38838b2bbb2877cde095c7c139e041c50d71|IIS modules installation script, install.bat
ed2c4429cf27e19aa6881d86bc5b42c21470525564fc53be688b9b26c83db766|IIS modules installation script, hello.bat
4c6703c7435759dbe0c889474a5fae4ca86e491ca45887a0dae3fcd4649e79c5|IIS modules installation script, install.bat, i
0d07b8485145e0ea6789570b9ab476d8e1604110a9c45c9c753ef7bc5edfd539|Document listing IIS modules installation comma
a96e1643dedd472e5712282904110ee948592fab722dc87d8f1e7658d3d8449d|IIS modules removal script, uninstall.bat
f9dd0b57a5c133ca0c4cab3cca1ac8debdc4a798b452167a1e5af78653af00c1|Customized Hidden rootkit, Wingtb.sys
88fd3c428493d5f7d47a468df985c5010c02d71c647ff5474214a8f03d213268|Customized Hidden rootkit, Wingtb.sys
af05f1b780a14583887857cb87d697d985ce172abb1d57e4108cac5e5aaca136|Customized Hidden rootkit, Wingtb.sys
83620389548516c74b40f9067ca20b7cc641a243c419d76ab2da87f8fd38e81c|Customized Hidden rootkit, Winkbj.sys.0xc8046ed
a8498295ec3557f1bf680a432acf415abf108405063f44d78974a4f27c27dd20|Driver setup information file for rootkit, Hidd
fc16cb7949b0eb8f3ffa329bef753ee21440638c1ec0218c1e815ba49d7646bb|Driver setup information file for rootkit, Hidd
82a1f8abffbd469e231eec5e0ac7e01eb6a83cbeb7e09eb8629bc5cc8ef12899|Driver catalog file for rootkit, Wingtb.cat
13ebf6422fe07392c886c960fafb90ef1ba3561f00eedb121a136e7f6c29c9ee|Driver catalog file for rootkit, Wingtb.cat
```

### Possibly related hashes (SHA-256)

```
64d0a4703ec976b0e0db4e193b9ccdf4ef6f34d24c32274579ee028a67bfa3a9|Variant of IIS HijackServer C++ module
```

### File paths

```
C:\Windows\System32\inetsrv\scripts.dll|IIS HijackServer C++ module
C:\Windows\SysWOW64\inetsrv\scripts.dll|IIS HijackServer C++ module
C:\Windows\System32\inetsrv\caches.dll|IIS HijackServer C++ module
C:\Windows\SysWOW64\inetsrv\caches.dll|IIS HijackServer C++ module
C:\Windows\System32\drivers\Wingtb.sys|Customized Hidden rootkit
```

### Hostnames

```
c.cseo99[.]com|HijackServer configuration stager
f.fseo99[.]com|HijackServer files stager
api.aseo99[.]com|HijackServer SEO telemetry endpoint
c.cseo8[.]com|HijackServer configuration stager
f.zseo8[.]com|HijackServer files stager
api.xseo8[.]com|HijackServer reporting endpoint
mlxya.oss-accelerate.aliyuncs[.]com|HijackServer JavaScript distribution stager (on legitimate Alibaba CDN)
```

### Possibly related domains

```
 gov[.]land|Staging server in older variant of IIS HijackServer C++ module (early and mid-2024)
cn[.]lol|Staging server in older variant of IIS HijackServer C++ module (early and mid-2024)
org[.]cfd|Staging server in older variant of IIS HijackServer C++ module (early and mid-2024)
lseo99[.]com|Likely HijackServer infrastructure (2025)
jseo99[.]com|Likely HijackServer infrastructure (2025)
wseo99[.]com|Likely HijackServer infrastructure (2025)
wseo88[.]com|Likely HijackServer infrastructure (2024)
fseo88[.]com|Likely HijackServer infrastructure (2024)
cseo88[.]com|Likely HijackServer infrastructure (2024)
aseo88[.]com|Likely HijackServer infrastructure (2024)
wseo8[.]com|Likely HijackServer infrastructure (2024)
```

### URLs

```
 hxxps://f.zseo8[.]com/uploads/2024-10-24/48c3a008cd9ccfa5fd3bdb69ed6d12ce.txt|Apache HijackServer PHP module
hxxps://f.zseo8[.]com/uploads/2024-10-24/99749da89ec4d1e3f3179f119f2a955b.txt|Apache HijackServer PHP module
```

## Yara rules

```
rule iis_module_hijackserver_native {
   meta:
       description = "Matches the IIS HijackServer module"
       references = "TRR251001"
       hash = "c1ca053e3c346513bac332b5740848ed9c496895201abc734f2de131ec1b9fb2"
       date = "2025-08-25"
       author = "HarfangLab"
       context = "file"
   strings:
       $c1 = ".?AVCHttpModule@@" ascii fullword
       $c2 = ".?AVCGlobalModule@@" ascii fullword
       $m1 = "RegisterModule" ascii fullword
       $s1 = "hack1234" ascii
       $s2 = "<!- GP -->" ascii fullword
       $s3 = /\.cseo\d{1,3}\.com\/config\// ascii
       $s4 = ":(80|443)(?=/|$)" ascii fullword
       $s5 = "TryCleanTmp:" ascii
       $s6 = "no excute " ascii
       $s7 = "\\b(\\d{1,2})-(\\d{1,2})-(\\d{4})\\b" ascii fullword
       $s8 = "/Tqpn0tGX550fVwt5D6g4CGWP6" ascii
       $s9 = "\\IISCPP-GM\\" ascii
       $s10 = "\\Dongtai.pdb\x00" ascii
       $s11 = "_FAB234CD3-09434-88" ascii
       $s12 = "<input type='text' name='cmdml' place" ascii
       $s13 = ".?AVHiJackServer@@" ascii fullword
       $s14 = ".?AVWebdllServer@@" ascii fullword
```

```
        $s15 = ".?AVAffLinkServer@@" ascii fullword
    condition:
        uint16be(0) == 0x4D5A
        and filesize > 200KB and filesize < 2MB
        and $m1
        and (any of ($c*))
        and (4 of ($s*))
}

rule iis_module_hijackserver_dotnet {
    meta:
        description = "Matches the IIS HijackServer .NET module"
        references = "TRR251001"
        hash = "915441b7d7ddb7d885ecfe75b11eed512079b49875fc288cd65b023ce1e05964"
        date = "2025-10-14"
        author = "HarfangLab"
        context = "file"
    strings:
        $dotNet = ".NETFramework,Version=" ascii
        $c1 = "HttpApplication" ascii fullword
        $c2 = "IHttpModule" ascii fullword
        $s1 = "YourSecretKey123" wide fullword
        $s2 = "<!- GP -->" wide fullword
        $s3 = /\.cseo\d{1,3}\.com\/config\// wide
        $s4 = ":(80|443)(?=/|$)" wide fullword
        $s5 = "clean?type=all" wide fullword
        $s6 = "DealRequest" ascii
        $s7 = "\\Tiquan\\CustomIISModule\\" ascii
        $s8 = "\\CustomIISModule.pdb\x00" ascii
        $s9 = "\\Temp\\AcpLogs\\conf\\" wide
        $s10 = "RobotTxtServer" ascii fullword
        $s11 = "HijackServer" ascii fullword
        $s12 = "WebdllServer" ascii fullword
        $s13 = "AffLinkServer" ascii fullword
    condition:
        uint16be(0) == 0x4D5A
        and filesize > 200KB and filesize < 2MB
        and $dotNet
        and (all of ($c*))
        and (4 of ($s*))
}

rule apache_module_hijackserver_php_decoded {
    meta:
        description = "Matches the decompressed and decoded Apache HijackServer PHP module"
        references = "TRR251001"
        hash = "e107bf25abc1cff515b816a5d75530ed4d351fa889078e547d7381b475fe2850"
```

```
        date = "2025-10-15"
        author = "HarfangLab"
        context = "file"
    strings:
        $php = /\$_SERVER\[\s*['"]PHP_SELF['"]\s*\]/ ascii wide fullword
        $s1 = "hj_clean_cache_dir" ascii wide fullword
        $s2 = "hj_get_file_content" ascii wide fullword
        $s3 = "\"清理目录空间 目录:\"" ascii wide fullword
        $s4 = "'/:(80|443)$/'" ascii wide fullword
        $s5 = "isCleanRequest()" ascii wide fullword
        $s6 = "shuffle_file_current_line" ascii wide fullword
        $s7 = "self::replaceAffLinkUrl" ascii wide fullword
        $s8 = "/Tqpn0tGX550fVwt5D6g4CGWP6" ascii wide
        $s9 = "HJ_CONFIG_URL_FORMAT" ascii wide fullword
        $s10 = "HJ_DEFAULT_LOCAL_LINK_NUM" ascii wide fullword
        $s11 = "renderHealthCheck" ascii wide fullword
        $s12 = "renderRedirect" ascii wide fullword
        $s13 = "renderAffLink" ascii wide fullword
    condition:
        filesize > 50KB and filesize < 600KB
        and $php
        and (4 of ($s*))
}

rule apache_module_hijackserver_php {
    meta:
        description = "Matches the encoded Apache HijackServer PHP module"
        references = "TRR251001"
        hash = "e107bf25abc1cff515b816a5d75530ed4d351fa889078e547d7381b475fe2850"
        date = "2025-10-15"
        author = "HarfangLab"
        context = "file"
    strings:
        $s1 = "\"display_errors\"" ascii wide fullword
        $s2 = /\$code\s*=\s*['"]eJztvWl7XMXRMPydX3GsKJmRGS22sQF5IbIkYwVZ/ ascii wide
        $s3 = /eval\(\s*gzuncompress\(\s*base64_decode\(\s*\$code\s*\)\s*\)\s*\);/ ascii wide nocase fullword
    condition:
        filesize > 10KB and filesize < 200KB
        and (all of them)
}

rule wingtb_rootkit {
    meta:
        description = "Matches the customized Hidden rootkit, Wingtb.sys."
        references = "TRR251001"
        hash = "f9dd0b57a5c133ca0c4cab3cca1ac8debdc4a798b452167a1e5af78653af00c1"
        hash = "88fd3c428493d5f7d47a468df985c5010c02d71c647ff5474214a8f03d213268"
```

```
        date = "2025-10-15"
        author = "HarfangLab"
        context = "file"
    strings:
        $a1 = "\\Device\\WinkbjDamen" wide fullword
        $a2 = "\\DosDevices\\WinkbjDamen" wide fullword
        $s1 = "Kbj_Zhuangtai" wide fullword
        $s2 = "Kbj_YinshenMode" wide fullword
        $s3 = "Kbj_WinkbjFsDirs" wide fullword
        $s4 = "Kbj_WinkbjFsFiles" wide fullword
        $s5 = "Kbj_WinkbjRegKeys" wide fullword
        $s6 = "Kbj_WinkbjRegValues" wide fullword
        $s7 = "Kbj_FangxingImages" wide fullword
        $s8 = "Kbj_BaohuImages" wide fullword
        $s9 = "Kbj_WinkbjImages" wide fullword

        $pdb = "D:\\DriverSpace\\hidden\\x64\\Release\\Winkbj.pdb" fullword

    condition:
        uint16be(0) == 0x4d5a and
        filesize < 1MB and
        ((1 of ($a*) and 6 of ($s*)) or $pdb)
}

rule wingtb_rootkit_commandline_tool_wingtbcli {
    meta:
        description = "Matches the usermode command-line tool for rootkit, WingtbCLI.exe."
        references = "TRR251001"
        hash = "913431f1d36ee843886bb052bfc89c0e5db903c673b5e6894c49aabc19f1e2fc"
        date = "2025-10-15"
        author = "HarfangLab"
        context = "file"
    strings:
        $s1 = ".?AVCommandUnignore@@" fullword
        $s2 = ".?AVCommandUnprotect@@" fullword
        $s3 = ".?AVCommandYinshen@@" fullword
        $s4 = "System\\CurrentControlSet\\Services\\Wingtb" wide fullword
        $s5 = "/buxiaoshi" wide fullword
        $s6 = "/fangxing" wide fullword
        $s7 = "/bufangxing" wide fullword
        $s8 = "/bubaohu" wide fullword
        $s9 = "/zhuangtai" wide fullword
        $s10 = "/yinshen" wide fullword
        $s11 = "Kbj_ShanchuFile" wide fullword
        $s12 = "Kbj_ShanchuDir" wide fullword
        $s13 = "Kbj_WinkbjRegValues" wide fullword
        $s14 = "Kbj_FangxingImages" wide fullword
```

```
        $s15 = "Kbj_Zhuangtai" wide fullword
        $s16 = "\\\\.\\WinkbjDamen" wide fullword
        $pdb = "D:\\DriverSpace\\hidden\\x64\\Release\\HiddenCLI.pdb" fullword
    condition:
        uint16be(0) == 0x4d5a and
        filesize < 1MB and
        (8 of ($s*) or $pdb)
}
```

---

Source: https://harfanglab.io/insidethelab/rudepanda-owns-iis-servers-like-2003/