



We use optional cookies to improve your experience on our websites, such as through social media connections, and to display personalized advertising based on your online activity. If you reject optional cookies, only cookies necessary to provide you the services will be used. You may change your selection by clicking “Manage Cookies” at the bottom of the page. [Privacy Statement](#) [Third-Party Cookies](#)

Accept

Reject

Manage cookies



Microsoft

Microsoft Security

Solutions

Products

Services

Partners

|



All Microsoft



Light



Dark

[Blog home](#) / Threat intelligence

Search the blog



[Research](#) [Threat intelligence](#) [Threat actors](#)

10 min read

IIS modules: The evolution of web shells and how to detect them

By [Microsoft Incident Response](#)

December 12, 2022



Vulnerabilities and exploits

The Microsoft Detection and Response Team (DART) has been renamed to Microsoft Incident Response (Microsoft IR). For more information on IR services, go to [Microsoft Incident Response](#)

Web exploitation and web shells are some of the most common entry points in the current threat landscape. Web servers provide an external avenue directly into your corporate network, which often results in web servers being an initial intrusion vector or mechanism of persistence. Monitoring for exploitation and web shells should be a high priority for all networks, and while these detection techniques are targeted towards malicious IIS modules, a lot of these techniques will also provide general web shell detections.

IIS modules and the creation of persistent backdoors by malicious IIS modules has recently been addressed in the Microsoft Security blog titled [Malicious IIS modules creating persistent backdoors](#). In this blog by Microsoft Detection and Response Team (DART), we aim to provide further guidance on detecting malicious IIS modules and other capabilities (such as logging) that you can use during your own investigations. While we will cover Microsoft Defender for Endpoint detections in this blog, these detection methods should be tool agnostic. The queries listed are not definitive detection queries, but rather a base query that you can build on to suit your specific environment.

A history of malicious IIS modules

The concept of malicious IIS has been around since at least 2013. Historical malware analysis shows how crimeware groups used IIS modules to intercept client logons and payment details by using the BeginRequest triggers to read user-provided parameters before the webserver processes them.

One of the first examples of sophisticated IIS modules was discovered in late 2021. The vendor’s [ICEAPPLE report](#) details an IIS module that was used by an actor to reflectively load further .NET modules, which by extension, loads further .NET

capability. This allowed the actor to have minimal malicious indicators in the base IIS module, then load further capabilities as required.

A more recent example was provided in a [recent Microsoft blog](#), where the actor instead opted for child process execution rather than loading the capability directly into w3wp.exe. This blog delves further into the capability to discuss the different capabilities that malicious IIS modules have access to.

Malicious IIS modules techniques

Event handlers

A core part of IIS module functionality is event handling. Event triggers provide opportunities for code to be called when specific actions happen. IIS modules have access to 27 event triggers by default, including:

- Begin Request: When a request is received by the webserver.
- End Request: When a response is about to be sent to the client.
- Error: When an error happens.
- Log Request: When a request is about to be logged.

Event handlers, which run when their associated trigger is fired, can allow an actor to essentially setup a proxy on the IIS server. By setting up an event handler on the BeginRequest, EndRequest, and Error event triggers, the actor can intercept all requests before the web service can process them and before the response is sent to the client.

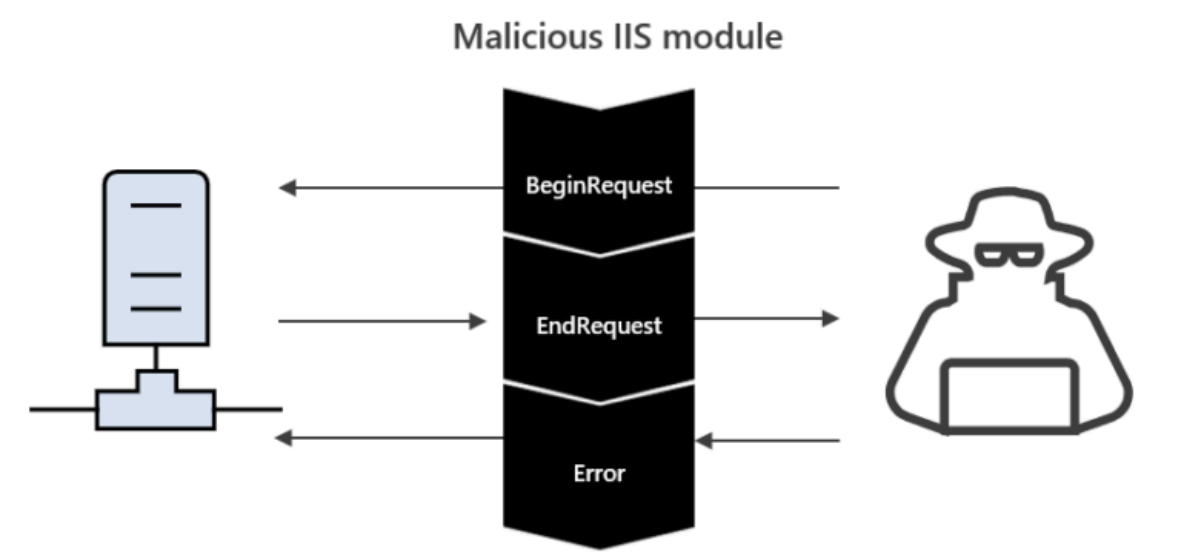


Figure 1. Diagram showing how the malicious IIS module sits between the web server and the client

Event handlers are given full read and write access to the requests, which allows malicious IIS modules to hide web shell communications within any aspect on the web request, turning every page that does and does not exist into a web shell. This can include hiding web shell communications in the parameters, body, headers, or HTTP methods.

These aspects of malicious IIS modules make them very hard to detect in standard IIS logs. You cannot rely on usual web shell detection strategies, such as high frequency page requests or URI patterns. Malicious IIS modules instead require new detection techniques and use of advanced IIS logging.

Request and response tampering

An additional difficulty with malicious IIS modules is that they can tamper with any aspect of the request and/or response. This could include removing web shell commands from parameters or headers and preventing web shell commands from being logged.

IIS Modules can also intercept responses before they are sent, which presents the opportunity for an actor to serve malicious payloads into any response from the website, potentially infecting viewers of the website.

Process creation


‘W3wp.exe’, also known as IIS worker processes, is utilized by web applications that run within IIS. Process creation is the most common indication of a web shell on IIS servers. Monitoring for the creation of common shell tooling (cmd, PowerShell, rundll32, mshta) with the parent process w3wp.exe can help detect low-sophistication IIS modules.


This should not be considered a strong detection for IIS modules. With full integration with C# and the .NET framework, a large amount of functionality can be integrated to execute directly within the IIS process without relying on creating child processes.

.NET assembly loading

A common execution path for actors is to load .NET modules directly into memory through [reflectively loading assemblies](#). This can allow common tools, such as SharpHound or the [Potato PrivEsc family](#), to be loaded without being written to disk. This is also seen as a stealthier alternative to process creation because it’s within the context of w3wp.exe rather than within a child process.

w3wp.exe created named pipe \\Device\\NamedPipe\\dbc4eb6b-96c0-4c8f-8632-e28d2885cf1b\\pipe\\spoolss

 Hunt for related events

Event info 

Event

w3wp.exe created named pipe
\\Device\\NamedPipe\\dbc4eb6b-96c0-4c8f-8632-e28d2885cf1b\\pipe\\spoolss


Event time

9/3/2022, 7:00:25.004 PM

Action type

NamedPipeEvent

User


 IIS APPPOOL\\defaultapppool

Initiating process

w3wp.exe

Pipe name

\\Device\\NamedPipe\\dbc4eb6b-96c0-4c8f-8632-e28d2885cf1b\\pipe\\spoolss



Desired access

1180063

Thread id

5116

File operation

Create

Remote client access

True

Session id

0

Entities




 services.exe >  svchost.exe >  w3wp.exe

Figure 2. SweetPotato named pipes being created from within w3wp.exe

As mentioned in the vendor paper earlier, assemblies can be provided arbitrarily to deliver additional functionality. This may be through providing the assembly through the web request or downloading the assembly from an actor-controlled C2. The figure below shows:

1. SharpHound being downloaded from an external C2 and loaded through the Reflection Assembly Load method.
2. Two methods being invoked within the binary and the output directory being set to ProgramData.

Figure 3. Example of an IIS module remotely downloading SharpHound and reflectively invoking it

With access to .NET, malicious actors can add additional layers of evasion to prevent detection of their IIS modules, such as encoding or encryption. In the figure below, we can see:

1. A base64 encoded blob and the size of the decoded assembly.
2. A new memory allocation is made, where the assembly is decoded and deflated into the new allocation.
3. The assembly is loaded and invoked, executing the command whoami.

Figure 4. Example of SweetPotato being reflectively loaded and invoked

Logging and monitoring

Advanced IIS Logs

IIS logs are a great place to start hunting for web shells, but advanced IIS logging is recommended because IIS modules can remove malicious traffic from the standard IIS logs. The IIS Service can provide additional advanced logging, such as the Microsoft IIS Configuration Operational log, which can be enabled through the event log tool by using the following commands:

- Lists additional logs available for IIS: ``wevtutil el | findstr -i IIS``
- Configuration for the selected log: ``wevtutil gl Microsoft-IIS-Configuration/Operational``
- Enable the selected log: ``wevtutil sl /e:true Microsoft-IIS-Configuration/Operational``

Figure 5. Example showing wevtutil querying the IIS Configuration Operational event log

The log that we will be focusing on in this blog is the Microsoft IIS Configuration Operational log. When enabled, the default path for this log is ``C:\Windows\System32\winevt\Logs\Microsoft-IIS-Configuration%4Operational.evtx`` (shown in the figure above).

The Microsoft IIS Configuration Operational log captures the additional and removal of IIS modules (Event ID 29). IIS modules are not commonly added to a production IIS server, so alerting on this event ID should be enabled within your SIEM or security products.

Figure 6. Event ID 29 showing the IIS module ‘ProxyShell’ being added to the default website

Note: This IIS module has no correlation with the Exchange Vulnerability ProxyShell.

Figure 7. Event ID 29 showing the IIS module ‘ProxyShell’ being removed from the default website

IIS module listing

IIS modules can be installed at a global level or at a site level. In detecting malicious IIS modules, it is important to check both the global and site level for unauthorized modules. Regular monitoring of these locations for such modules and comparing

against a known good list can help detect and identify malicious IIS modules.

[Appcmd](#) (path: %windir%\system32\inetsrv\appcmd.exe), a command line tool for managing your IIS servers, can be used to that purpose. The command ‘appcmd list modules’ will list global IIS modules on your server. The command ‘appcmd list modules /app.name:<appName>/’ will let you search specific websites.

Figure 8. Appcmd listing the modules for Default Web Site and showing two malicious modules: “ProxyShell” and “Malicious IIS Module”

Modules listed through appcmd will be ordered based on the order of installation. In Figure 9, the two malicious IIS modules, ProxyShell and Malicious IIS Module, were the two most recent IIS modules installed and therefore the last two on the list. The type parameter also shows the class that is called when the module is loaded.

Web.config

The web.config file, which details the settings for a website, can include modules that the website loads and should therefore be monitored when detecting malicious IIS modules. Monitoring of web.config should primarily focus on tracking modifications to the file, and can be done through multiple tools and sources. For example, the Microsoft IIS Configuration Operational event log produces Event ID 50 when a modification is made to a website. Because the content of the modification is not captured, a backup of the web.config should be stored for easy comparison between the modified file and the original.

Figure 9. Event ID 50 showing that a modification has been made to default website

Most EDRs capture file modification events as well. Enabling an alert for the modification of web.config, especially from the w3wp.exe process, will enable detection of unwarranted changes to the config.

Hunting for malicious IIS modules

IIS module loading

While loaded IIS modules are standardly loaded DLLs, not all tools list .NET modules that are loaded into w3wp.exe. One tool that does show IIS modules loaded into w3wp.exe is Process Hacker, which if used with administrative privileges, will show them under the Modules tab.

Figure 10. Malicious ProxyShell IIS module loaded within the w3wp.exe process

In Microsoft Defender for Endpoint, an IIS module that is loaded into w3wp.exe will appear twice: First when loaded from the bin directory from which it resides, then immediately after from the temporary ASP.NET directory.

Figure 11. Malicious IIS module ProxyShell being listed in Defender for Endpoint

By default, IIS modules are loaded when the w3wp.exe process is created. If an IIS module is loaded while the w3wp.exe process is already executing and at a different

time than the rest of the module, it can be an indicator for malicious IIS module loading. Monitoring for abnormal module loads may help detect malicious IIS modules. Using a query like the KQL one below can group together modules loaded into w3wp.exe at the same second. In Figure 12, we can see a large number of modules being loaded within a three second time period, followed by the malicious ProxyShell IIS module three hours later.

```
DeviceImageLoadEvents
| where InitiatingProcessFileName has "w3wp.exe"
| summarize loaded_modules=make_set(FileName) by
format_datetime(Timestamp, 'yy-MM-dd HH:mm:ss')
| project Timestamp, loaded_modules,
count=array_length(loaded_modules)
```

Figure 12. Anomalous module loading based on timeframe of other IIS modules

Assembly loading

While IIS modules have the capability to load .NET modules arbitrarily and reflectively within the context of w3wp.exe, the AppDomains are still registered within the hosting process. By listing AppDomain loaded within an assembly through a tool like Process Hacker, you'll be able to find the loaded IIS module and any .NET modules that have been loaded.

Figure 13. Malicious ProxyShell IIS module, SharpHound and SweetPotato App Domains

In the figure above, the malicious IIS module ProxyShell can be seen alongside the loaded assemblies SharpHound and SweetPotato. Another thing to note is that reflectively loaded modules usually do not have Flags: in Figure 13, all the assemblies without Flags are either loaded through the malicious IIS module or through Visual Studio debugging.

The ETW provider Microsoft-Windows-DotNETRuntimeRundown provides a snapshot in time of the loaded .NET modules within active processes. Two events can help detect malicious assemblies loaded within IIS:

- Event ID 151 lists loaded AppDomains.
- Event ID 155 enumerates assemblies loaded at the time of the rundown.

The ModuleILPath field shows the path of the loaded assembly; however, if this assembly is loaded reflectively, rather than a path to the file, it will instead just list the name of the assembly. The figure below shows how SharpHound and SweetPotato, both with reflectively loaded assemblies, do not have paths while other events do.

Figure 14. Example of reflectively loaded assemblies not having a file path within the ModuleILPath field

The Assembly Flags field may also be 0, similar to Figure 13 where Process Hacker shows empty Flags for the assemblies.

Figure 15. Example of empty assembly flags for .NET rundown

IIS module installation

Processes which contain appcmd or gacutil within the command line and the parent process w3wp.exe should be investigated for potential installation of malicious IIS modules. The following Defender for Endpoint queries can help detect such malicious IIS module installation:

```
DeviceProcessEvents
| where ProcessCommandLine has "appcmd.exe add module"
| where InitiatingProcessParentFileName == "w3wp.exe"

DeviceProcessEvents
|where ProcessCommandLine has "\\gacutil.exe /I"
| where InitiatingProcessParentFileName == "w3wp.exe"
```

Process creation

Process creation events with the parent process of w3wp.exe should be monitored for abnormal child processes. For IIS servers that require child processes of w3wp.exe, ignore lists should be created for these child processes to prevent false flags.

```
DeviceProcessEvents
| where Timestamp > ago(7d)
| where InitiatingProcessFileName in~ ('w3wp.exe', 'httpd.exe')
| where FileName in~ ('cmd.exe', 'powershell.exe', 'cscript.exe',
'wscript.exe', 'net.exe', 'net1.exe', 'ping.exe', 'whoami.exe')
| summarize instances = count() by ProcessCommandLine, FolderPath,
DeviceName, DeviceId
| order by instances asc
```

Query source: [Azure-Sentinel/Webserver Executing Suspicious Applications.yaml](#)

If one of the Potato-spoits are being used to create processes, the AccountName field may also be "System" while the InitiatingProcessAccountName will be the Application Pool.

Conclusion

Threat actors use a variety of techniques to conceal their activity when using malicious IIS modules. Enabling additional logging in your environment is recommended to facilitate the detection of these harmful modules. Specifically, IIS servers should enable the optional event log Microsoft IIS Configuration Operational. This log can provide insight into IIS modules being added or removed from the IIS server and track any modifications made to web.config.

IIS Servers should have their web.config and IIS modules monitored for malicious modifications being made. For example, Gacutil.exe and appcmd.exe being executed from w3wp.exe should be monitored for potential installation of IIS modules. Additionally, the bin directories of websites and the default GAC path should be monitored and regularly scanned for malicious modules being created.

Hunting for malicious IIS modules can be performed through Microsoft Defender for Endpoint or your EDR of choice, and it should be conducted regularly to detect abnormal w3wp.exe interactions. This can include, but is not limited to, process creation, file creation and named pipe creation. Due to the wide flexibility around how IIS modules can execute malicious code, it’s important to look for irregularities in behavior.

Detecting web exploitation and malicious IIS modules should be a priority for all organizations, and Microsoft DART believes that following the recommendations provided in this blog and along with the monitoring and detection strategies will assist your organization in the detection and response of these threats.

Get started with Microsoft Security

Microsoft is a leader in cybersecurity, and we embrace our responsibility to make the world a safer place.

Learn more

Connect with us on social



What's new

- Surface Pro
- Surface Laptop
- Surface Laptop Studio 2
- Surface Laptop Go 3
- Microsoft Copilot
- AI in Windows
- Explore Microsoft products
- Windows 11 apps

Business

- Microsoft Cloud
- Microsoft Security
- Dynamics 365
- Microsoft 365
- Microsoft Power Platform
- Microsoft Teams

Microsoft Store

- Account profile
- Download Center
- Microsoft Store support
- Returns
- Order tracking
- Certified Refurbished
- Microsoft Store Promise
- Flexible Payments

Developer & IT

- Azure
- Developer Center
- Documentation
- Microsoft Learn
- Microsoft Tech Community
- Azure Marketplace

Education

- Microsoft in education
- Devices for education
- Microsoft Teams for Education
- Microsoft 365 Education
- How to buy for your school
- Educator training and development
- Deals for students and parents
- Azure for students

Company

- Careers
- About Microsoft
- Company news
- Privacy at Microsoft
- Investors
- Diversity and inclusion

