Anatomy of native IIS malware

By Zuzana HromcováAnton Cherepanov

Archived: 2026-04-05 17:39:20 UTC

ESET researchers have discovered a set of previously undocumented malware families, implemented as malicious extensions for *Internet Information Services* (IIS) web server software. Targeting both government mailboxes and e-commerce transactions, as well as aiding in malware distribution, this diverse class of threats operates by eavesdropping on and tampering with the server's communications.

Along with a complete breakdown of the newly discovered families, our new paper, *Anatomy of native IIS malware*, provides a comprehensive guide to help fellow security researchers and defenders detect, dissect and mitigate this class of server-side threats. In this blogpost, we summarize the findings of the white paper.

Today, we are also launching a series of blogposts where we introduce the most notable of the newly discovered IIS malware families, as case studies of how this type of malware is used for cybercrime, cyberespionage and SEO fraud.

The findings of our IIS malware research were first presented at Black Hat USA 2021 and will also be shared with the community at the Virus Bulletin 2021 conference on October 8th.

IIS is Microsoft Windows web server software with an extensible, modular architecture that, since v7.0, supports two types of extensions – *native* (C++ DLL) and *managed* (.NET assembly) modules. Focusing on *malicious native* IIS modules, we have found over 80 unique samples used in the wild and categorized them into 14 malware families – 10 of which were previously undocumented. ESET security solutions detect these families as Win{32,64}/BadIIS and Win{32,64}/Spy.IISniff.

How IIS malware operates

IIS malware is a diverse class of threats used for cybercrime, cyberespionage, and SEO fraud – but in all cases, its main purpose is to intercept HTTP requests incoming to the compromised IIS server and affect how the server responds to (some of) these requests.

With the default installation, IIS itself is persistent, so there is no need for extension-based IIS malware to implement additional persistence mechanisms. Once configured as an IIS extension, the malicious IIS module is loaded by the IIS Worker Process (w3wp.exe), which handles requests sent to the server – this is where IIS malware can interfere with the request processing.

We identified five main modes in which IIS malware operates, as illustrated in Figure 1:

- *IIS backdoors* allow their operators to remotely control the compromised computer with IIS installed
- *IIS infostealers* allow their operators to intercept regular traffic between the compromised server and its legitimate visitors, to steal information such as login credentials and payment information. Using HTTPS doesn't prevent this attack, as IIS malware can access all data handled by the server – which is where the data is processed in its unencrypted state.
- *IIS injectors* modify HTTP responses sent to legitimate visitors to serve malicious content
- *IIS proxies* turn the compromised server into an unwitting part of the C&C infrastructure for another malware family, and misuse the IIS server to relay communication between victims of that malware and the real C&C server
- *SEO fraud IIS malware* modifies the content served to search engines to manipulate SERP algorithms and boost the ranking for other websites of interest to the attackers

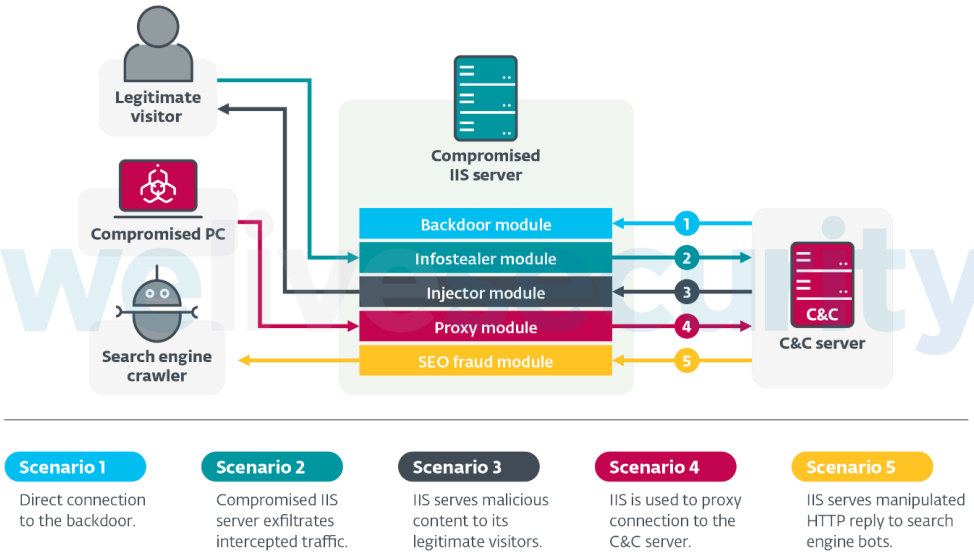

Figure 1. Overview of IIS malware mechanisms

All of these malware types are discussed at length in the paper.

How (and where) it spreads

Native IIS modules have unrestricted access to any resource available to the server worker process – thus, administrative rights are required to install native IIS malware. This considerably narrows down the options for the initial attack vector. We have seen evidence for two scenarios:

- IIS malware spreading as a trojanized version of a legitimate IIS module
- IIS malware spreading through server exploitation

For example, between March and June 2021, we detected a wave of IIS backdoors spread via the Microsoft Exchange pre-authentication RCE vulnerability chain (CVE-2021-26855, CVE-2021-26857, CVE-2021-26858, and CVE-2021-27065), aka ProxyLogon. Targeted specifically were Exchange servers that have *Outlook on the web* (aka OWA) enabled – as IIS is used to implement OWA, these were a particularly interesting target for espionage.

After our colleagues reported the first such case in March 2021, we have detected four more campaigns of various IIS backdoors spreading to Microsoft Exchange servers through the same vulnerability. To complement our telemetry, we have performed internet-wide scans to detect the presence of these backdoors, which allowed us to identify and notify other victims of the malware.

Figure 2 shows the geographical locations of servers affected by these five campaigns, using data from our telemetry and internet-wide scans.

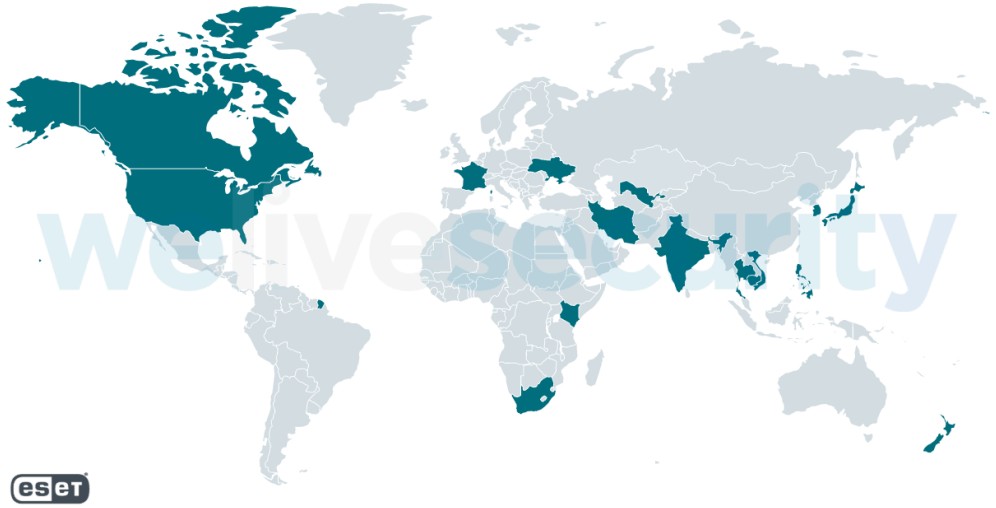

Figure 2. Victims of native IIS backdoors spread via the ProxyLogon vulnerability chain

The following entities were among the victims:

- Government institutions in three countries in Southeast Asia
- A major telecommunications company in Cambodia
- A research institution in Vietnam
- Dozens of private companies in a range of industries, located mostly in Canada, Vietnam and India, and others in the USA, New Zealand, South Korea, and other countries

Note that while IIS backdoors may be well-suited for spying on high-profile mailboxes, victims of IIS malware are not limited to compromised servers – all legitimate visitors of the websites hosted by these servers are potential targets, as the malware can be used to steal sensitive data from the visitors (IIS infostealers) or serve malicious content (IIS injectors). Please refer to the full white paper for the details on the targets of the other analyzed IIS families.

The insides of native IIS malware

From the technical perspective, all types of native IIS malware are implemented as dynamic-link libraries (DLLs), written using the IIS C++ API. Any such DLL must:

- Implement a class inherited from either the CHttpModule or CGlobalModule class (or both), and override a number of that class's methods (*event handlers*)
- Export the RegisterModule function, which is the library entry point, responsible for creating the instances of these classes and registering the implemented handlers for *server events*, as illustrated in Figure 3.

```
.text:000007FEFB1F17D0 ; Exported entry   1. RegisterModule
.text:000007FEFB1F17D0
.text:000007FEFB1F17D0
.text:000007FEFB1F17D0 ; rdx [in] = IHttpModuleRegistrationInfo
.text:000007FEFB1F17D0
.text:000007FEFB1F17D0 public RegisterModule
.text:000007FEFB1F17D0 RegisterModule proc near
.text:000007FEFB1F17D0 push    rbx
.text:000007FEFB1F17D2 sub     rsp, 20h
.text:000007FEFB1F17D6 mov     ecx, 8          ; Size
.text:000007FEFB1F17DB mov     rbx, rdx
.text:000007FEFB1F17DE call    MyHttpModuleFactory_ctor
.text:000007FEFB1F17E3 lea     rcx, ??_7CMyHttpModuleFactory@@6B@ ; const CMyHttpModuleFactory::`vftable'
.text:000007FEFB1F17EA mov     cs:pFactory, rax
.text:000007FEFB1F17F1 xor     r9d, r9d
.text:000007FEFB1F17F4 mov     r8d, RQ_SEND_RESPONSE
.text:000007FEFB1F17FA mov     rdx, rax
.text:000007FEFB1F17FD mov     [rax], rcx
.text:000007FEFB1F1800 mov     rcx, rbx
.text:000007FEFB1F1803 mov     r10, [rbx]
.text:000007FEFB1F1806 add     rsp, 20h
.text:000007FEFB1F180A pop     rbx
.text:000007FEFB1F180B jmp     [r10+IHttpModuleRegistrationInfoVtbl.SetRequestNotifications]
.text:000007FEFB1F180B RegisterModule endp
.text:000007FEFB1F180B
```

Figure 3. A typical RegisterModule function of native IIS malware

Server events refer to the steps that the IIS server takes during request processing (see Figure 4), but also to other actions taken by the server (for example, sending an HTTP response). These events generate *event notifications*, which are handled by *event handlers* implemented in the server's modules (see Figure 5).

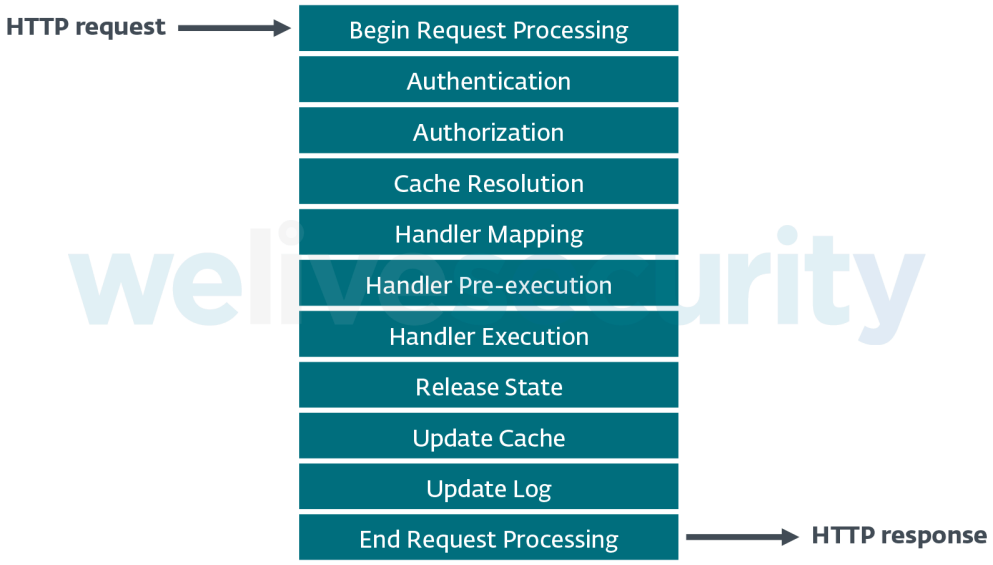

Figure 4. HTTP request-processing pipeline in IIS

In short, the event handlers (or the methods of IIS module core classes) are where the IIS malware functionality is implemented and where any reverse engineers should focus their analysis. For a deep dive into IIS malware essentials and how to analyze such binaries, refer to the *Anatomy of native IIS malware* section of our white paper.

```
; const HttpModule::`vftable'
??_7HttpModule@@6B@ dd offset OnBeginRequest
                                        ; DATA XREF: sub_7454A310+19↑o
                                        ; sub_7454A360+9↑o
                dd offset OnPostBeginRequest
                dd offset OnAuthenticateRequest
                dd offset OnPostAuthenticateRequest
                dd offset OnAuthorizeRequest
                dd offset OnPostAuthorizeRequest
                dd offset OnResolveRequestCache
                dd offset OnPostResolveRequestCache
                dd offset OnMapRequestHandler
                dd offset OnPostMapRequestHandler
                dd offset OnAcquireRequestState
                dd offset OnPostAcquireRequestState
                dd offset OnPreExecuteRequestHandler
                dd offset OnPostPreExecuteRequestHandler
                dd offset OnExecuteRequestHandler
                dd offset OnPostExecuteRequestHandler
                dd offset OnReleaseRequestState
                dd offset OnPostReleaseRequestState
                dd offset OnUpdateRequestCache
                dd offset OnPostUpdateRequestCache
                dd offset OnLogRequest
                dd offset OnPostLogRequest
                dd offset OnEndRequest
                dd offset OnPostEndRequest
                dd offset OnSendResponse
                dd offset OnMapPath
                dd offset OnReadEntity
                dd offset OnCustomRequestNotification
                dd offset OnAsyncCompletion
                dd offset Dispose
                dd offset sub_7454A360
                dd offset ??_R4HttpModuleFactory@@6B@ ; const HttpModule
; const HttpModuleFactory::`vftable'
??_7HttpModuleFactory@@6B@ dd offset sub_7454A310
```

```
; const CMyGlobalModule::`vftable'
??_7CMyGlobalModule@@6B@ dq offset OnGlobalStopListening
                                        ; DATA XREF: DNameNode
                                        ; Terminate+5↑o
                dq offset OnGlobalCacheCleanup
                dq offset OnGlobalCacheOperation
                dq offset OnGlobalHealthCheck
                dq offset OnGlobalConfigurationChange
                dq offset OnGlobalFileChange
                dq offset OnGlobalPreBeginRequest
                dq offset OnGlobalApplicationStart
                dq offset OnGlobalApplicationResolveModules
                dq offset OnGlobalApplicationStop
                dq offset OnGlobalRSCAQuery
                dq offset OnGlobalTraceEvent
                dq offset OnGlobalCustomNotification
                dq offset Terminate
                dq offset OnGlobalThreadCleanup
                dq offset OnGlobalApplicationPreload
                dq offset OnSuspendProcess
```

Figure 5. Event handlers: methods of the module classes, CHttpModule and CGlobalModule

Network communication

A notable feature of IIS malware is how it communicates with its operators. Malicious IIS modules, especially IIS backdoors, don't usually create new connections to their C&C servers. They work as *passive implants*, allowing the attackers to control them by providing some "secret" in an HTTP request sent to the compromised IIS web server. That's why IIS backdoors usually have a mechanism to recognize *attacker requests* that are used to control the server and have a predefined structure, such as:

- URL or request body matching a specific regex
- A specific custom HTTP header present
- An embedded token (in the URL, request body or one of the headers) matching a hardcoded password
- A hash value of an embedded token matching a hardcoded value
- A more complex condition – for example, a relationship between all of the above

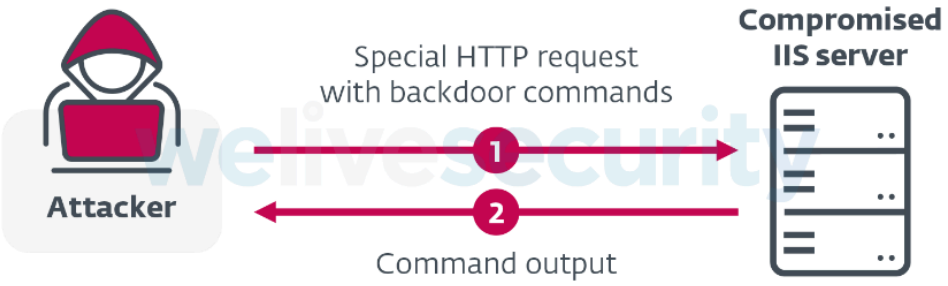

Figure 6. Passive C&C communication channel (IIS backdoors)

On the other hand, some IIS malware categories *do* implement an alternative C&C channel – using protocols such as HTTP or DNS – to obtain the current configuration on the fly. For example, an IIS injector contacts its C&C server every time there is a new request from a legitimate visitor of the compromised website, and uses the server response to modify the content served to that visitor (such as malicious code or adware).

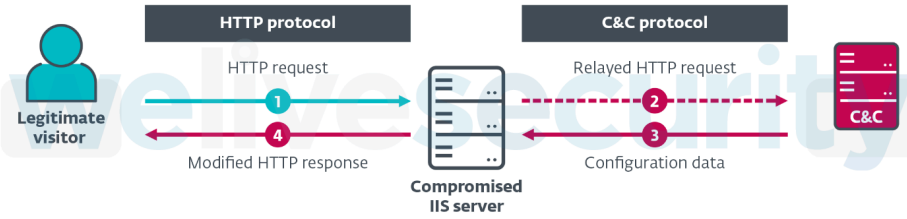

Figure 7. Alternative C&C communication mechanism (IIS injectors)

Table 1 summarizes how the C&C channels, as well as other notable techniques, are implemented by the 14 analyzed IIS malware families.

Table 1. Summary of obfuscations implemented, and functionalities supported by analyzed IIS malware families

Group #	**Functionality**	#colspan#	#colspan#	#colspan#	#colspan#	**C&C channel**	#colspan#	#colspan#	**Dete evas and obfu tech**
Backdoor	**Infostealer**	**Proxy**	**SEO fraud**	**Injector**	**Attacker request verification (e.g. specific header present, specific URI, query string parameter)**	**Encryption/ encoding**	**Alternative channel protocol**	#rowspan#	
	Group 1	✅	✅	❌	❌	❌	HTTP header with hardcoded password	base64	❌
Group 2	✅	❌	❌	❌	❌	HTTP header with hardcoded password	RSA + AES-CBC	❌	❌
Group 3	✅	❌	❌	❌	❌	HTTP header present	base64	❌	❌
Group 4	✅	❌	❌	❌	❌	HTTP header with hardcoded password	XOR + base64	❌	Anti logg
Group 5	❌	✅	❌	❌	❌	URI and HTTP header with hardcoded password	❌	❌	Strir stacl
Group 6	❌	✅	❌	❌	❌	Query string parameter	❌	❌	❌
Group 7	✅	❌	❌	❌	❌	Relationship between	AES-CBC	❌	Anti logg

Group #	Functionality	#colspan#	#colspan#	#colspan#	#colspan#	C&C channel	#colspan#	#colspan#	Dete evas and obfu tech
						HTTP headers, HTTP body format			
Group 8	✅	❌	❌	❌	❌	HTTP header with hardcoded password	❌	❌	❌
Group 9	❌	❌	✅	✅	❌	No support for attacker requests	❌	HTTP	Encr strin (XO
Group 10	❌	❌	❌	✅	❌	No support for attacker requests	❌	HTTP to obtain JavaScript config	❌
Group 11	✅	❌	✅	✅	✅	HTTP header with hardcoded password	❌	DNS TXT to obtain config, HTTP for C&C	Strir encr (AD 0x0
Group 12, variant A	✅	❌	✅	✅	✅	HTTP header with password whose MD5 hash is hardcoded	❌	HTTP	Strir encr (AD 0x0
Group 12, variant B	✅	❌	❌	✅	✅	#rowspan#	❌	HTTP	UPX pack
Group 12, variant C	❌	❌	❌	✅	❌	No support for attacker requests	❌	HTTP	Strir encr (XO 0x0
Group 13	✅	❌	❌	✅	❌	Query string parameter	❌	HTTP	❌
Group 14	❌	❌	❌	✅	✅	No support for attacker requests	❌	HTTP	❌

Mitigation

Since native IIS modules can only be installed with administrative privileges, the attackers first need to obtain elevated access to the IIS server. The following recommendations could help make their work harder:

- Use dedicated accounts with strong, unique passwords for the administration of the IIS server. Require multifactor authentication (MFA) for these accounts. Monitor the usage of these accounts.
- Regularly patch your OS, and carefully consider which services are exposed to the internet, to reduce the risk of server exploitation.

- Consider using a web application firewall, and/or endpoint security solution on your IIS server.
- Native IIS modules have unrestricted access to any resource available to the server worker process; you should only install native IIS modules from trusted sources to avoid downloading their trojanized versions. Be especially aware of modules promising too-good-to-be-true features such as magically improving SEO.
- Regularly check the IIS server configuration to verify that all the installed native modules are legitimate (signed by a trusted provider, or installed on purpose).

For details on how to detect and remove IIS malware, refer to the Mitigation section of the white paper. We are also publishing a set of YARA rules that you can leverage to detect all the 14 analyzed IIS malware families.

Conclusion

Internet Information Services web servers have been targeted by various malicious actors, for cybercrime and cyberespionage alike. The software's modular architecture, designed to provide extensibility for web developers, can be a useful tool for attackers to become a part of the IIS server, and intercept or modify its traffic.

It is still quite rare for endpoint (and other) security software to run on IIS servers, which makes it easy for attackers to operate unnoticed for long periods of time. This should be disturbing for all serious web portals that want to protect their visitors' data, including authentication and payment information. Organizations that use OWA should also pay attention, as it depends on IIS and could be an interesting target for espionage.

While IIS server threats are not limited to native IIS malware, we believe this paper will be a helpful starting point for defenders for understanding, identifying, and removing IIS threats, and a guide to our fellow researchers to reverse engineer this class of threats and understand their common tactics, techniques and procedures.

Additional technical details on the malware and Indicators of Compromise can be found in our comprehensive white paper, and on GitHub. For any inquiries, or to make sample submissions related to the subject, contact us at: threatintel@eset.com.

Acknowledgements to fellow ESET malware researchers Marc-Étienne Léveillé and Mathieu Tartare for their work on this investigation.

Read next:
IIStealer: A server-side threat to e-commerce transactions
IISpy: A complex server-side backdoor with anti-forensic features
IISerpent: Malware-driven SEO fraud as a service