

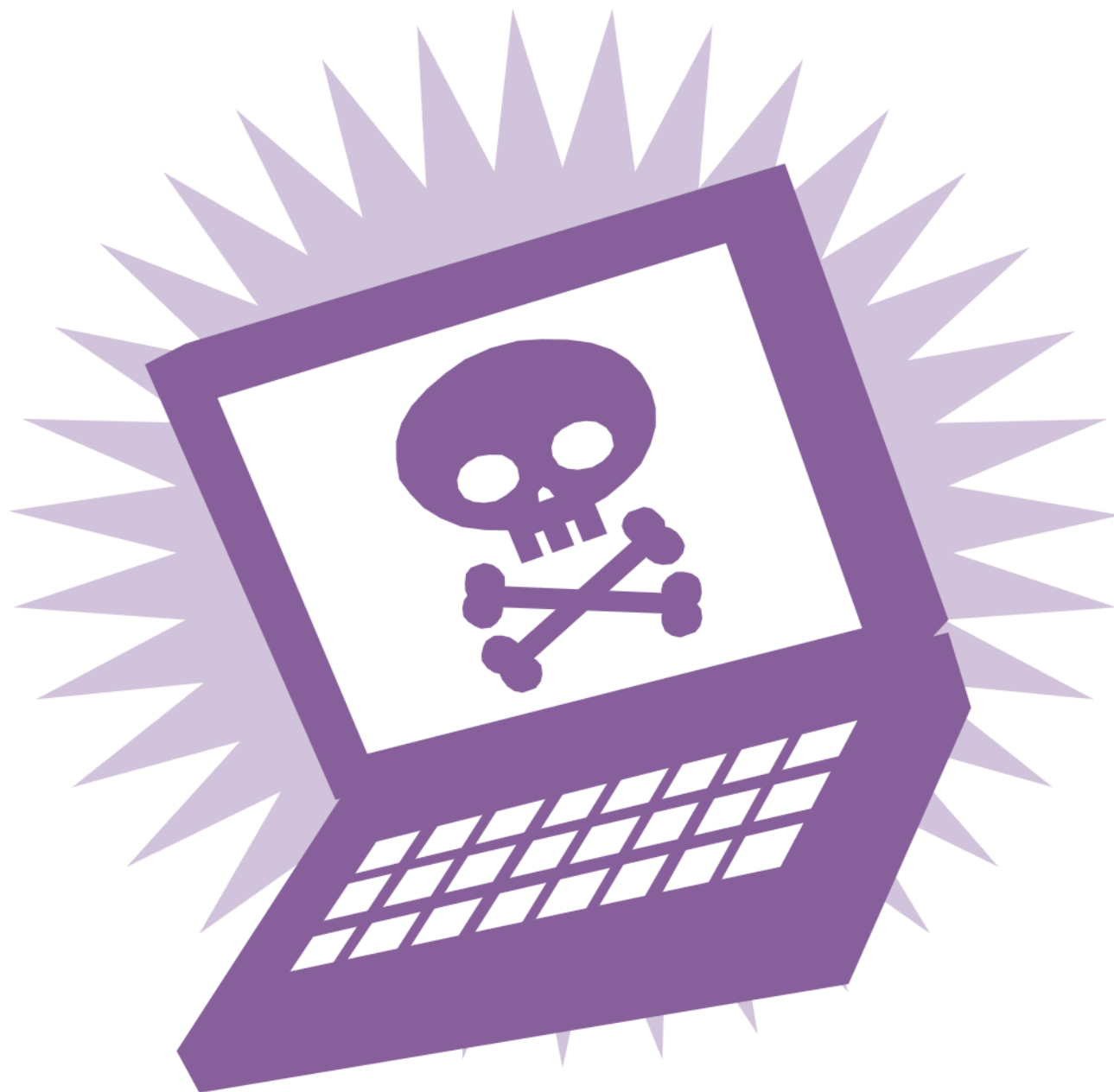
Ping32 RMM and ValleyRAT

By Erik Hjelmvik

Published: 2026-06-25 · Archived: 2026-09-15 02:02:03 UTC

,

Thursday, 25 June 2026 09:27:00 (UTC/GMT)



Fareed Radzi recently [blogged about a malware campaign](#) observed earlier in June by Kaspersky's GReAT team. The malware campaign embedded malicious code in VBScripts, which were distributed through WhatsApp DMs.

The VBScript then dropped the legitimate Remote Monitoring and Management (RMM) tool ManageEngine Endpoint Central.

Fareed included the IOCs for the following Endpoint Central server IP addresses:

- 202.61.160.208
- 202.61.160.202
- 202.61.160.201
- 202.61.160.160
- 202.61.160.137
- 38.55.151.63

He also noted a link to [ValleyRAT](#):

Notably, 202.61.160[.]201 had previously been observed as command-and-control infrastructure associated with ValleyRAT and Gh0st RAT activity. Although the overlap raises the possibility of the VBS campaign being linked to the operator of these known malware families, the available evidence is insufficient to confidently attribute the campaign to a known threat actor.

Attribution is difficult, so it makes sense not to call out any specific threat actor just because of a single overlapping IP address. Nevertheless, the threat actor that typically comes to mind when talking about ValleyRAT is [Silver Fox](#) (银狐).

Retrohunting in Sandboxes

I searched various online sandboxes for the IP addresses and MD5 hashes that were published in Fareed's blog post. To my delight I found plenty of samples on ANY.RUN as well as Triage. But what was even more interesting was the sandbox executions on Triage for the sample with MD5 hash d43fdaa1f0ee09d7e5f0f94ee9df7b6c. One of the known filenames for this sample was "Bitte füllen Sie das Formular für Umsatzsteuer-Nullsatz-Verkäufe aus..vbs".

Sample executions on Recorded Future Triage Sandbox:

- <https://tria.ge/260325-z4xp4sew7y>
- <https://tria.ge/260325-z763ysex41>

I can't determine how this sample was originally connected to the ManageEngine Endpoint Central campaign, but it shared several traits with what was described in Fareed's Securelist write-up. However, this particular VBScript didn't install the ManageEngine RMM. Instead it reached out to f004.backblazeb2[.]com and downloaded a dropper.

	DNS	● f004.backblazeb2.com	CRYPTNET.DLL	▼
	GET	● https://f004.backblazeb2.com/file/fadaoxiao/uamcd.pdf	NETAPI.DLL	▼
	DNS	● fadaoxiao.s3.us-west-004.backblazeb2.com	NETAPI.DLL	▼
	GET	● https://fadaoxiao.s3.us-west-004.backblazeb2.com/pacc.vbs	NETAPI.DLL	▼
	GET	● https://f004.backblazeb2.com/file/gaosu2/CoreShield.msi	CRYPTNET.DLL	▼

The dropper then deployed NSecRTS.exe, which turned out to be another RMM tool called “Ping32” from the Chinese company Shandong Anzai Information Technology, aka NSecsoft. This RMM tool has a [history](#) of [being abused](#) as a Remote Access Trojan (RAT) by hackers.

The Ping32 RMM used HTTP over multiple TCP ports on 143.92.37.168, and it also communicated via UDP port 18987 on the same server.

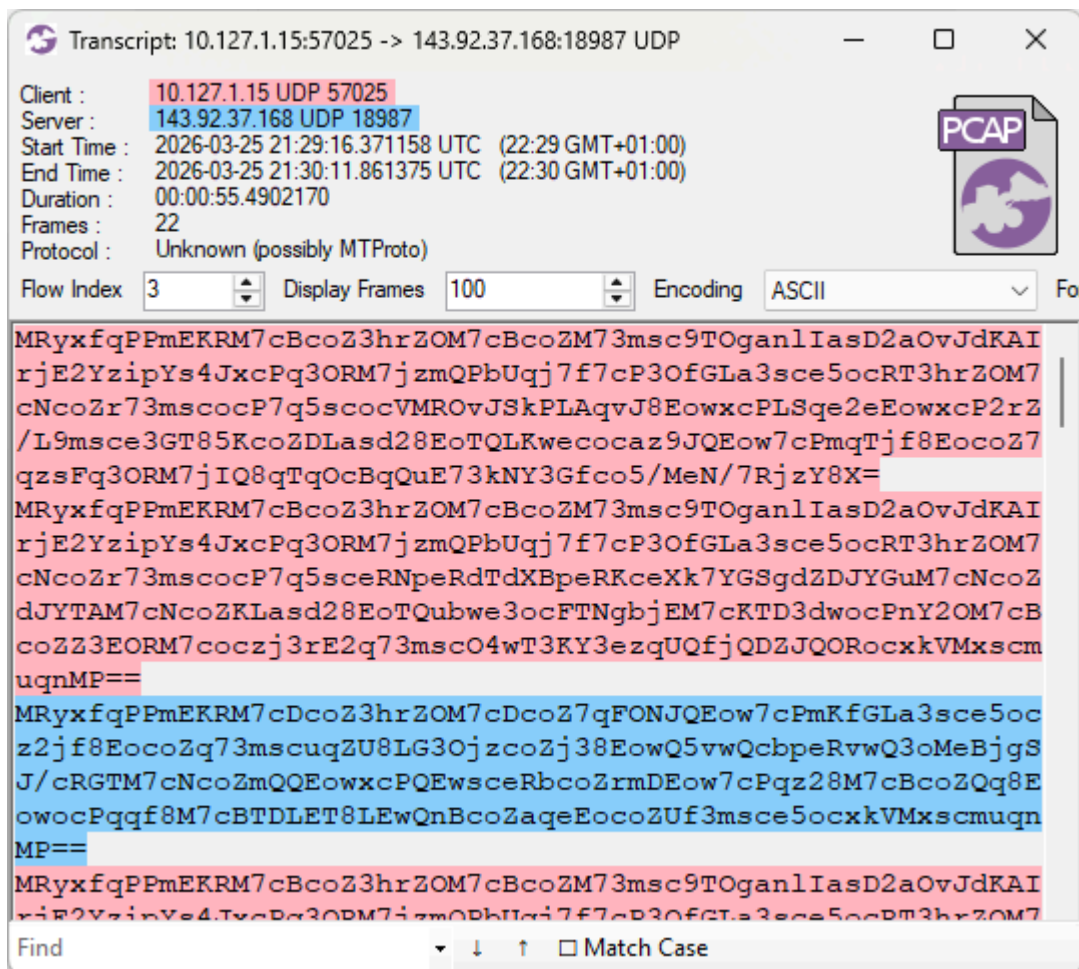


Image: UDP traffic to 143.92.37.168:18987

Pivot to ValleyRAT

I pivoted on the C2 IP 143.92.37.168, which was used by the malicious Ping32 RMM, and got a [hit on Triage Sandbox](#). Triage classified this sample as DonutLoader and ValleyRAT, and its malware config extractor identified

the following attributes:

Family	valleyrat_s2
Version	1.0
C2	143.92.37.168:10086
Campaign date	2026-02-02

This is interesting, because this is another link between the campaign mentioned in Fareed's blog post and ValleyRAT. When I examined the ValleyRAT C2 traffic from the Triage sandbox execution I noticed that [CapLoader](#) as well as [FlowCarp](#) identified it as [Gh0stKCP](#), which is a UDP-based protocol that ValleyRAT sometimes uses to transport its C2 traffic.

CapLoader 2.1.0.0

File Edit View Tools Help

Input Settings

- ☒ Identify protocols
- ☒ Parse DNS ☒ Parse TLS
- Frames limit: 0
- Enter input filter in BPF format

File ID | Filename

1 | 260514-agrsxacw6n-behavioral1.pcapng

☒ Auto-extract Selection PCAP

Flows (40) Services (9) Hosts (9) Alerts (2)

Hide Selected Flows | Invert Hiding | Show All Flows | Selected Flows: 0

Display Filter (BPF) Clear Apply 40

Row Filter Contains Phrase Clear Apply

Flow_ID	Client_IP	Client_Port	Server_IP	Server_Port	Transport	Protocol	Sub_Prot
0	10.127.0.22	63810	143.92.37.168	10086	UDP	Gh0stKCP	
1	10.127.0.22	63811	143.92.37.168	10086	UDP	Gh0stKCP	
2	10.127.0.22	63812	143.92.37.168	10086	UDP	Gh0stKCP	
3	10.127.0.22	52670	8.8.8.8	53	UDP	DNS	
4	10.127.0.22	49924	20.190.159.131	443	TCP	TLS	
5	10.127.0.1	3	10.127.0.22	4	ICMP		
6	10.127.0.22	52671	143.92.37.168	10086	UDP	Gh0stKCP	
7	10.127.0.22	64870	8.8.8.8	53	UDP	DNS	
8	10.127.0.22	49925	20.223.35.26	443	TCP	TLS	
9	10.127.0.22	64871	143.92.37.168	10086	UDP	Gh0stKCP	
10	10.127.0.22	64872	143.92.37.168	10086	UDP	Gh0stKCP	
11	10.127.0.22	51898	8.8.8.8	53	UDP	DNS	
12	10.127.0.22	49928	52.111.227.14	443	TCP	TLS	
13	10.127.0.22	51899	143.92.37.168	10086	UDP	Gh0stKCP	
14	10.127.0.22	51900	143.92.37.168	10086	UDP	Gh0stKCP	
15	10.127.0.22	51901	143.92.37.168	10086	UDP	Gh0stKCP	
16	10.127.0.22	51902	143.92.37.168	10086	UDP	Gh0stKCP	
17	10.127.0.22	51903	143.92.37.168	10086	UDP	Gh0stKCP	
18	10.127.0.22	51904	143.92.37.168	10086	UDP	Gh0stKCP	
19	10.127.0.22	51905	143.92.37.168	10086	UDP	Gh0stKCP	
20	10.127.0.22	51906	143.92.37.168	10086	UDP	Gh0stKCP	
21	10.127.0.22	62476	8.8.8.8	53	UDP	DNS	
22	10.127.0.22	49937	142.250.151.94	80	TCP	HTTP	CRL
23	10.127.0.22	51897	8.8.8.8	53	UDP	DNS	

Use this oneliner to upload the PcapNG file from Triage to the free [FlowCarp](#) demo server and extract IP:port IOCs from FlowCarp alerts.

```
curl -fSs --data-binary @260514-agrsxacw6n-behavioral1.pcapng https://demo.flowcarp.com | jq -s -c  
'map(select(.event_type=="alert"))[(.dest_ip + ":" + (.dest_port|tostring)), .alert.signature)]|unique[]'  
  
["143.92.37.168:10086","MALWARE protocol detected: Gh0stKCP"]
```

If you prefer Suricata, use these custom signatures to detect Gh0stKCP:

<https://github.com/Netresec/Suricata/blob/main/netresec.rules>

You can then use the same jq query as in the FlowCarp example to extract the alert IOCs from Suricata's eve.json output.

```
cat eve.json | jq -s -c 'map(select(.event_type=="alert"))[(.dest_ip + ":" + (.dest_port|tostring)),  
.alert.signature)]|unique[]'  
  
["143.92.37.168:10086","Gh0stKCP / HP-Socket ARQ handshake"]  
["143.92.37.168:10086","Gh0stKCP close"]
```

Silver Fox

It is difficult to attribute the analyzed malware samples to a specific threat actor, but it is possible that they were used by the notorious Silver Fox group, which is one of China's largest and most active cybercrime groups.

On a positive note, China Daily recently [reported](#) that Chinese police have taken “criminal compulsory measures” against 27 suspects linked to Silver Fox. The same article also stated that “The gang allegedly sent phishing emails in bulk, stole corporate data and built fraud scenarios to carry out criminal activities totaling more than 7 million yuan (\$1 million)”.

Let's hope this puts a stop to, or at least significantly reduces, the massive flood of malware that has been coming from this threat actor.

IOC List

Unknown Downloader

- d43fdaa1f0ee09d7e5f0f94ee9df7b6c (Bitte füllen..vbs)
- hxxps://f004.backblazeb2[.]com/file/fadaoxiao/uamcd.pdf
- hxxps://f004.backblazeb2[.]com/file/gaosu2/CoreShield.msi
- hxxps://fadaoxiao.s3.us-west-004.backblazeb2[.]com/pacc.vbs
- ac63eb8814f20ffd89ce81f51cba6916 (uamcd.pdf)
- 9ca134a5ed592a0fb57e2ad910a71c80 (pacc.vbs)

NSecsoft Ping32 RMM C2

- 143.92.37.168:18987 (UDP)
- 143.92.37.168:38987 (TCP)
- 143.92.37.168:48988 (TCP)
- 143.92.37.168:48991 (TCP)

- 143.92.37.168:48992 (TCP)

DonutLoader/ValleyRAT

- 8266b00c4e45d728cef78b3f5a865f68 (ManagementTool.exe)
- 143.92.37.168:10086 (UDP)

Posted by Erik Hjelmvik on Thursday, 25 June 2026 09:27:00 (UTC/GMT)

Tags: #[Gh0stKCP](#)#[ValleyRAT](#)#[Suricata](#)#[FlowCarp](#)#[CapLoader](#)

Source: <https://www.netresec.com/?page=Blog&month=2026-06&post=Ping32-RMM-and-ValleyRAT>