Harvesting Credentials: Phantom Stealer’s Agricultural Obfuscation Campaign

Defentive ⋮ ⋮ 10/28/2025

In late October 2025, Defentive’s threat hunting team intercepted a curious VBScript sample reported by an enterprise manufacturing client. The initial analysis looked almost playful — variables evocatively named “SheepBase” and “ThreshingZone,” junk strings like “mango” sprinkled everywhere. It quickly became clear, however, that this wasn’t a case for the farm — it was the start of an advanced, multi-stage infostealer campaign. What followed was a technical cat-and-mouse story rife with anti-analysis tricks, staged delivery, and the unmistakable fingerprints of the Phantom Stealer malware family.

Technical Timeline: Step-by-Step Analysis

Stage 1: VBScript Dropper

- MD5: `aa3a8dc8406e4998da963b0401376c8d`
- SHA1: `7d9ec8c5f8c75b97346517370c9147f84d8d0f9b`
- SHA256: `048b16aa8b4c5fcfddc7609999446ff888f7106e2f4fa106a459fdeb21c81e82`
  1.1MB

Our analysis began with static triage. The script's surface was a tangled web of `Chr()` arithmetic, farm-themed variable names, and aggressive string concatenation—all hallmarks of current Phantom Stealer and Stealerium variants. Through careful resolution, Defentive's researchers mapped out the core execution sequence:

- The dropper constructs `Scripting.FileSystemObject` and writes a heavily obfuscated batch file to `C:\Users\Public\SeedComplex.bat`.
- Activation is handled via WMI:

```
GetObject("winmgmts:").Get("Win32_Process").Create("cmd.exe /c
C:\Users\Public\SeedComplex.bat")
```

- All disk paths, execution flows, and method calls are heavily layered in junk strings (“mango”, “avocadopapaya”), mirrored in recent proof-of-concept Stealerium/Phantom campaigns.

Stage 2: The Batch File — Layered Obfuscation

Upon detonation in a sandbox, the batch file proved a masterclass in string pollution (205 lines):

- Junk strings replace executable commands, only later sanitized for actual variable expansion and execution.
- Contains and decodes over of base64 payload, split into multiple fragments and littered with “mango”/“avocadopapaya” to defeat automated forensics.
- Assembles and executes a PowerShell payload responsible for the next stage of infection and, crucially, the reflective code injection that is Phantom’s trademark.

Key artifacts left behind:

- `C:\Users\Public\SeedComplex.bat`
- `aoc.bat` in the `USERPROFILE` directory

Stage 3: PowerShell Loader — .NET Magic and In-Memory Mischief

The decoded PowerShell script gave us our strongest attribution anchor.

- It dynamically generates .NET assemblies (Reflection.Emit), defines Windows API P/Invokes (`VirtualProtect`, `VirtualQuery`, `OpenProcess`).
- Implements advanced memory scanning, process hollowing, and fully in-memory payload execution — the same routine used by Phantom Stealer to avoid endpoint security detection.
- The script’s logic and binary-embedded shellcode show with open-source Stealerium but with distinctive Phantom loader fingerprints: anti-VM checks, persistence via registry or scheduled task, and classic process hollowing against `MSBuild.exe` or similar.

Attribution: Is it Phantom Stealer?

Defentive Threat Research Attribution Synopsis:

> *Through cross-reference with recent technical writeups and Proofpoint reporting, we confirm with this sample is a variant, built atop Stealerium’s open-source codebase but weaponized with its own loader, injection, and obfuscation techniques. Signature behaviors — multi-stage*

dropper, farm-themed obfuscation, reflective PowerShell loader, string pollution — closely match both historical and new Phantom/Stealerium campaigns as documented by Palo Alto Networks, Proofpoint, and threat researcher communities.

Why not “just Stealerium”?

Stealerium is the “base” open infostealer project. It does not natively tie together the magic of farm-themed VBS drops, .NET dynamic P/Invokes, and specialized process hollowing targeting MSBuild. That’s Phantom Stealer’s signature — the hands-on-keyboard operator’s refinement that takes commodity malware and turns it into a bespoke threat for high-value intrusions.

Real-World Impact & Detection

What’s at Risk?

- Immediate compromise of browser credentials, session tokens, crypto wallets.
- Full device fingerprinting — hardware, software, Wi-Fi profiles, environment info.
- Potential webcam and screenshot exfiltration for blackmail or further intrusion.
- Persistence through task scheduler, registry, or dropped scripts — often invisible to default EDR.

Detection Opportunities

- Flag unusual file creations like `SeedComplex.bat` and batch files in public dirs.
- Monitor WMI usage for process creation with suspicious scripts.
- Look for and strings, or agricultural variable patterns in any scripting telemetry/logs.
- Log large PowerShell commands and scripts that use .NET `DefineDynamicAssembly`, `VirtualProtect`, or are associated with `MSBuild.exe` child processes.

Defentive Threat Research: Our Recommendations

- Disable or restrict WMI and PowerShell script execution for non-admin users wherever practical.
- Use EDRs that scan command line and in-memory behaviors, not just on-disk artefacts.
- Apply custom YARA and string-matching rules against known pollution terms (“mango”, `Chr()` obfuscation, farm-themed variables).
- Monitor for anomalous persistence (user tasks/registry keys invoking script interpreters from public/temp folders).

For IR Practitioners:

- Review the hashes and IOCs below.
- Isolate infected systems, review all scheduled tasks and scripts in public locations, and preserve memory for advanced shellcode Hunt.

Indicators of Compromise (IOCs)

File Hashes (VBScript Dropper):

- MD5: `aa3a8dc8406e4998da963b0401376c8d`
- SHA1: `7d9ec8c5f8c75b97346517370c9147f84d8d0f9b`
- SHA256: `048b16aa8b4c5fcfddc7609999446ff888f7106e2f4fa106a459fdeb21c81e82`

Dropped Files:

- `C:\Users\Public\SeedComplex.bat`
- `%USERPROFILE%\aoc.bat`

Behavioral:

- Strings: “mango”, “avocadopapaya”
- VBScript with farm variable terminology
- WMI-based PowerShell and batch execution
- Large base64 PowerShell payloads in batch, followed by in-memory .NET loader activity

Conclusion

This campaign is a textbook example of effective, adaptively maintained criminal infostealer operations — combining open-source code (Stealerium base), deep obfuscation, and skilled loader engineering. Phantom Stealer and its siblings continue to push the boundary of what’s possible with “just scripts,” reinforcing the need for defense-in-depth and vigilant threat hunting.

Get Defentive’s stories in your inbox

Join Medium for free to get updates from this writer.

Defentive Threat Research is tracking this threat cluster aggressively — stay tuned for updates, IOCs, and new analytic rules as Phantom Stealer continues to evolve in the wild.