Lorem Ipsum Malware: Trojanized MS Teams Installers Deliver…

By Thomas Elkins and Joshua Green

Archived: 2026-05-21 02:01:36 UTC

Lorem Ipsum Malware: Trojanized MS Teams Installers Deliver Multi-Stage Loader and Backdoor

May 4, 2026 | 18 min read

What Happened?

BlueVoyant Security Operations Center (SOC) and Threat Fusion Cell (TFC) security researchers have been tracking an emerging, rapidly maturing threat group conducting a global SEO-poisoning campaign that distributes trojanized Microsoft Teams installers. These installers ultimately deploy a multi-stage shellcode loader and backdoor BlueVoyant has designated **Lorem Ipsum**. Active since at least February 2026, the campaign opportunistically targets users searching for Microsoft Teams across at least six countries, with a US-based healthcare-sector client confirmed as targeted, with successful BlueVoyant interdiction. In roughly ten weeks, the operators evolved from a minimally obfuscated test build into an operationally mature loader chain featuring substitution cipher decoding, XOR-encrypted shellcode stubs, DLL sideloading, JFIF-disguised C2 traffic, and a per-victim UUID-tracked callback architecture. Most distinctively, the loader abuses ***letsdiskuss[.]com***, a legitimate India-based question-and-answer/blogging platform, as a dead-drop resolver for C2 infrastructure across at least four attacker-controlled profiles.

The combination of systematically procured Microsoft ID verified code-signing certificates with three-day burn cycles, continuously registered NameCheap infrastructure weaponized within hours, and sustained SEO-poisoning operations suggests a meaningfully funded mid-tier criminal actor whose development velocity indicates possible LLM-assisted tooling and warrants proactive defender attention.

Key Findings:

- *letsdiskuss[.]com* dead-drop resolver embedded within attacker-controlled profiles used to segment C2 across campaign waves.
- Validly signed MSI installers using Microsoft ID Verified certificates with three-day validity issued under multiple individual identities, suggesting a systematic certificate procurement pipeline.
- JFIF-disguised C2 traffic with payloads appended beyond image boundaries and *Content-Type: image/jpeg* headers replacing traditional HTTPS callbacks.
- Per-victim UUID-tracked C2 architecture using /api/init/{UUID} callback paths across dedicated domains.
- Architectural maturation in payload staging, shifting from plainraw[.]com-hosted gzip/hex PowerShell payloads in March to dedicated UUID-tracked C2 endpoints by mid-April.

- Externalized AES key/initialization vector passed via MSI custom action arguments rather than embedded in the PowerShell loader, forcing analysts to recover both artifacts to reconstruct decryption.
- Disposable infrastructure tradecraft: NameCheap domains with Withheld-for-Privacy registration weaponized within hours allocated one IP per domain across multiple provider ranges.
- Ten-week development velocity from test build to mature loader chain, consistent with possible LLM-assisted coding workflows observed across emerging threat groups.

SEO Poisoning: Initial Attack Vector

The attack begins when users search Bing and Google to find Microsoft Teams installers. Using SEO poisoning, the threat group ensures their attacker-controlled websites remain as top search results for users. While interacting with the fake Microsoft Teams websites, the only functional links on their websites are the large download button. All the other links keep returning the user to the main page.

Figure 1 Fake Microsoft

The download button interacts with a backend PHP script which serves the user the trojanized Microsoft Teams installer. BlueVoyant researchers observed that the script appears to keep track of IP addresses downloading their software. This ensures that the updated installers are only delivered once to each IP address. If multiple download attempts are made, the script defaults to serve a previous version.

The threat group has been very active in terms of updating their installers and delivery mechanisms. Older versions, starting towards the middle of March 2026, contained fewer anti-analysis methods. Updated versions, starting around the beginning of April, contained more anti-analysis techniques including stripping parts of their command line arguments for their PowerShell scripts, which are used as part of the key and initialization vector (IV) for decrypting the embedded payloads and encrypting the core functionality of their shellcode.

Initial Installer Analysis

While analyzing the installers, BlueVoyant researchers identified that each trojanized installer was validly signed with a digital certificate used to ensure their installers appeared legitimate to Windows and EDR tooling. Also, each digital certificate was only registered for a maximum of 3 days.

Figure 2 Digital Certificate

Contained within each installer is a PowerShell script loader along with a legitimate Microsoft Teams installer, both of which are dropped into the user's ***AppData/Roaming*** folder during installation. Each stage exists primarily to peel back another layer of obfuscation, culminating in stage three which performs persistence and assembles the final payload. The MSI file contains a custom action that executes the PowerShell loader component silently by incorporating the ***-WindowStyle Hidden*** flag, concealing it from the end user while the Microsoft Teams installer runs in the foreground as a distraction.

The PowerShell script decrypts a hardcoded Base64-encoded payload embedded within itself using AES encryption. The key and IV required for decryption are not contained within the script. Instead, they are passed in as command-line arguments directly from the custom action string within the MSI at install time.

After decrypting the embedded Base64-encoded payload, BlueVoyant researchers observed that it contained a second layer of PowerShell code. The second PowerShell code is responsible for decoding another Base64-encoded payload embedded within itself, decompress it using gzip, and reflectively loads it directly into memory for execution, ensuring it is never written to disk.

Figure 3 Snippet of second stage Power Shell script

This revealed a third PowerShell script, containing the ultimate loading code for the embedded payload. That script first establishes persistence by duplicating the original PowerShell command line and writing it to a Run key in the Windows registry, ensuring the malware executes again, even if the user logs out and logs back in.

Figure 4 Snippet of third Power Shell script

Older iterations of the third PowerShell loader stored the embedded payload as decimal values. However, newer versions employ a more sophisticated approach, rebuilding the payload through a substitution cipher-based decoding algorithm. The first variable holds the ciphertext while the second acts as the key. Both strings are then split into arrays using the split operator, ***-split '\s+'***, where the ciphertext array serves as an alphabet style lookup table and the key array drives the decoding sequence.

The core of the routine iterates over each token in the key array, locating its positional index within the ciphertext array using the ***[Array]::IndexOf()*** method. Each recovered index is then taken modulo 256 to ensure it falls within a valid byte range, before being appended to a growing list of bytes. In essence, each token's position within the ciphertext array encodes a numeric value, and the key array provides an ordered sequence of references into that alphabet to progressively reconstruct the final byte stream. This approach is a deliberate obfuscation strategy, as neither the ciphertext nor the key resembles recognizable shellcode or executable content in isolation, with the malicious payload only emerging when both variables are processed together.

BlueVoyant researchers observed this final embedded payload consisted of encrypted bytes accompanied by a small shellcode stub. Additionally, the decrypted payload was found to include a JFIF file structure. Upon review of the small shellcode stub, it was determined that the code is responsible for decrypting the remainder of the shellcode using XOR encryption with the hardcoded key JPEG.

Figure 5 Lorem Ipsum Loader decryption routine

Notably, earlier versions of this shellcode did not employ XOR decryption at all, a clear indication that this threat group is actively iterating on their tooling to improve evasion and complicate analysis.

Beginning in late April, BlueVoyant researchers observed newer iterations of the Lorem Ipsum loader being executed via DLL sideloading. In these cases, an MSI installer establishes persistence by creating a Run registry key with the value **"Microsoft Revo Health"**, which launches a binary named **Microsoft Teams Revo Helper ZnrAq.exe**. This binary is then used to sideload a malicious DLL masquerading as **msvcp140.dll**.

Analysis of the DLL revealed ciphertext stored within the .init and .bss sections, matching data previously observed in the third PowerShell script. The DLL contains logic to decode the data in the .bss section using the ciphertext embedded in .init. Once decoded, execution is transferred to the recovered Lorem Ipsum loader.

Lorem Ipsum Loader

The Lorem Ipsum loader begins by resolving various Windows libraries using the API ***LoadLibraryA***. After loading the Windows Libraries, Lorem Ipsum loader then resolves specific corresponding APIs using ***GetProcAddress***.

Figure 6 Windows Library resolution via Load Library A

Instead of storing the hashes of known API names, the APIs are hardcoded into the Lorem Ipsum loader and resolved at runtime.

Following API resolution, the shellcode creates a mutex via ***CreateMutexA*** to prevent concurrent execution. Notably, the mutex name is hard-coded and matches the <UUID> portion of the C2 initialization endpoint (***/api/init/<UUID>***), indicating deliberate reuse of the same identifier for both execution control and C2 communication.

One of the most distinctive characteristics of the Lorem Ipsum loader is its abuse of a legitimate third-party social media platform as an intermediary dead-drop resolver for command-and-control infrastructure. Rather than hardcoding C2 domains directly into the malware where they could be quickly extracted through static analysis and blocked, the threat group instead stores encoded C2 information within the profile description fields of attacker-controlled accounts on ***letsdiskuss[.]com***, an India-based question-and-answer and blogging service. The platform itself is entirely legitimate and has not been compromised; it is being abused in much the same way that threat actors have historically abused Pastebin, GitHub Gists, Telegram channels, and Google Docs, serving as a low-friction, trusted-traffic conduit for staging operational data. The profiles configured to support this campaign appear to incorporate the Lorem Ipsum placeholder text commonly used in design and publishing to demonstrate visual layout without meaningful content — providing both the campaign's namesake and a means of disguising the encoded C2 data as innocuous filler text. The decoding routine the loader applies to this content closely mirrors the substitution cipher logic observed in the third-stage PowerShell script analyzed earlier in this report, providing further evidence of shared tooling lineage and reinforcing BlueVoyant's assessment that these components were developed by the same operator or development team.

Figure 7 Attacker controlled letsdiskuss com profile used to stage encoded C2 information

The mechanics of the dead-drop are straightforward but operationally effective. The threat actor seeds a user profile on ***letsdiskuss[.]com*** with encoded C2 data embedded within the profile description, delimited by the markers ***-=(*** and ***)=-***. Upon execution on a victim host, the Lorem Ipsum loader issues a standard HTTP request to the attacker's profile URL, traffic that is largely indistinguishable from legitimate user browsing of the platform. The loader then parses the returned HTML, extracts the content located between the delimiters, and stores those encoded strings in memory for further processing. Each extracted string is compared against a hardcoded reference list within the loader, and the index distance between the extracted and hardcoded values is converted into hexadecimal byte values which, when concatenated, reconstruct the actual C2 domains the loader will ultimately contact. Notably, this decoding routine closely mirrors the substitution cipher logic observed in the third-stage PowerShell script analyzed earlier in this report, providing further evidence of shared tooling lineage and reinforcing BlueVoyant's assessment that these components were developed by the same operator or development team.


Figure 8 Sample Python code replicating decoding Lorem Ipsum C2s

After decoding its C2 endpoints, the Lorem Ipsum loader attempts to establish communication by transmitting its embedded JFIF image to the C2 server using a ***Content-Type: image/jpeg*** header rather than a conventional HTTPS request. What distinguishes this routine is how the loader repurposes the JFIF image itself as a bidirectional communication channel.

The loader contains logic to determine the raw size of the embedded image and compare it against a hard-coded expected value. If the actual image size exceeds this value, the loader interprets the additional bytes beyond the expected image boundary as appended data. For outbound communications, these extra bytes are obfuscated using a custom XOR-based encryption routine prior to transmission. For inbound communications, the loader applies a corresponding routine to identify appended data within received JFIF images and decrypts it using a separate custom decryption function.

As a result, both outbound and inbound C2 communications are fully encapsulated within JFIF image files, allowing command and data payloads to be covertly transmitted and extracted from seemingly benign image responses.


Figure 9 Network traffic capture via Adversary in the Middle Ait M proxy

If the C2 server is active, it responds with a JFIF file containing additional data appended beyond the image boundary including a new UUID.

After retrieving the payload from its C2 server, the loader creates a file within the user's ***%TEMP%*** directory and writes the UUID retrieved from the server into it. While the precise purpose of this file is not entirely clear, it appears to function as a tag because the loader contains logic to check for the presence of this file. It is possible it is used as a victim/session identifier or as a sandbox detection marker.

The loader then converts the UUID string into its binary representation using ***UuidFromStringA***. Following this, the loader calls ***VirtualProtect*** to mark the memory region containing the newly retrieved shellcode as executable, before transferring execution to the beginning of the decoded payload, completing the handoff to the next stage.

Lorem Ipsum Backdoor

Similar to the Lorem Ipsum loader component, the backdoor resolves its hardcoded libraries and APIs using a similar routine via ***LoadLibraryA*** and ***GetProcAddress***. Once the libraries and APIs are resolved, Lorem Ipsum backdoor generates an AES 32-byte key and IV using ***CryptoGenRandom*** API.

Once a key and IV are generated, the backdoor copies and appends them to its embedded image. As in the loader component, the backdoor contains similar functionality where it applies a custom XOR-based encryption routine to obfuscate the data before transmitting it to its hard-coded C2 server. If the C2 server is active, it responds with a JFIF image, indicating successful communication.

Next, the backdoor gathers information about the impacted host, storing the information in JSON.


Figure 10 AP Is used by Lorem Ipsum backdoor to identify information about compromised host

Each of the returned values are encoded using Base64 via the API ***CryptBinaryToStringA***.

Once the backdoor has gathered host information, the backdoor encrypts the data using the random key and IV that it generated and sends it to the hard coded C2. If the C2 is active, it will respond with another JFIF image file. This communication continues to cycle until the C2 is no longer responsive.

Although no further payloads were observed during research, the Lorem Ipsum backdoor contains functionality to execute additional code using ***VirtualProtect***, consistent with techniques seen in the associated loader. This capability provides the threat actor with flexibility to deploy additional tooling if a compromised host is identified as a high-value target.

Campaign Overview & Adversary Assessment

BlueVoyant identified first evidence associated with this threat group in February 2026, when researchers analyzed a malicious binary masquerading as the legitimate utility ***PE Detective***. Unlike later iterations, this early sample was minimally obfuscated — the attacker simply modified the binary's Address of Entry Point (AEP) to redirect execution into appended malicious code. BlueVoyant assesses with high confidence that this binary represents one of the threat group's earliest test builds, based on code lineage connecting it to the later Lorem Ipsum loader, infrastructure reuse across multiple campaign iterations, and an observable pattern of iterative refinement. Each subsequent variant has introduced additional anti-analysis layers — substitution cipher decoding, XOR-encrypted shellcode stubs, and DLL sideloading — indicating an actively maintained development pipeline, with the current Lorem Ipsum loader and backdoor variants representing the most operationally mature toolset observed to date.

The campaign targets users searching for Microsoft Teams installers through Bing and Google, using SEO poisoning to ensure attacker-controlled websites appear as top search results and serving trojanized installers through fake download portals where only the prominent download button is functional. The victimology appears broad and opportunistic rather than narrowly targeted, with public scanning telemetry showing campaign indicators surfaced by researchers and automated pipelines across at least six countries spanning North America, Europe, and Asia between March and late April 2026. The BlueVoyant client targeted victim for this attempted attack was US-based and linked to the healthcare sector. This broad approach is consistent with global SEO-driven targeting of any user searching for Microsoft Teams regardless of geography or industry.

The trojanized installers are validly signed with digital certificates registered for a maximum of three days, appearing legitimate to Windows and EDR tooling while minimizing exposure from certificate revocation. Campaign infrastructure is consistently registered through NameCheap with Withheld-for-Privacy domain privacy service out of Iceland and weaponized within hours to days of registration, with URL strings following templated conventions and rotating version-style filenames (***Setup_MT_V14.63.msi***, ***MTSetup_v15.3.7110.msi***, ***SetupMT_V5_7765.msi***) designed to defeat hash-based detection rather than reflect genuine software versioning.

Analysis of campaign telemetry reveals at least four distinct attacker-controlled ***letsdiskuss[.]com*** profiles serving as dead-drop resolvers — ***joeblack1673***, ***stevenblake8483***, ***dhuahsd12d2752***, and ***stevenseagal4596*** — with staggered first-seen activity from mid-March through late April 2026. BlueVoyant assesses with moderate confidence that the threat group uses multiple profiles to segment campaign waves or compartmentalize victim

clusters, with each profile resolving to its own C2 subset. For example, samples contacting the stevenblake8483 profile consistently resolve to the C2 cluster ***biblegodlike[.]com***, ***semigoddess[.]com***, and ***reeeeealy[.]com*** under a shared per-victim UUID, suggesting a deliberate one-profile-to-one-C2-cluster operational mapping. The deliberate selection of a less-trafficked regional platform rather than globally prominent services such as GitHub, Pastebin, or Twitter may further reflect an intentional effort to avoid platforms with mature, well-resourced abuse-monitoring programs, extending the operational lifespan of each dead-drop account.

The campaign's payload-staging architecture has itself evolved meaningfully across the operational window. In the March 2026 iteration of the campaign, the operators relied on ***plainraw[.]com*** — a recently registered domain offering a plaintext pasting service — to serve gzip-compressed, hex-encoded PowerShell payloads under disposable ***/raw/<hex>*** URL paths embedded in the trojanized MSI installers. By the April 15–16 iteration of the campaign, ***plainraw[.]com*** no longer appears in the telemetry of newer samples, with the operators having shifted to dedicated C2 domains using a ***/api/init/{UUID}*** callback architecture. This transition from a single shared paste-style staging host to per-victim UUID-tracked C2 endpoints represents a meaningful architectural maturation, providing both better victim tracking and improved infrastructure segmentation.

Temporal analysis of sample metadata reveals signing activity tightly clustered between approximately 10:14 UTC and 17:16 UTC, with all observed signing events occurring on weekdays. This narrow daytime band is consistent with a structured weekday work schedule operating from a time zone in the UTC+2 to UTC+5 range, broadly encompassing Eastern Europe, Russia, the Middle East, and parts of South Asia. While time-zone inference from signing timestamps is a soft attribution signal that sophisticated actors can deliberately manipulate, the consistency of the pattern across multiple distinct signers and certificates supports the assessment with low-to-moderate confidence and serves as a useful corroborating data point.

Overall, the Lorem Ipsum campaign reflects a capable, rapidly maturing, and meaningfully funded mid-tier threat actor rather than a commodity-level operation or a top-tier APT. The operation's development velocity is striking: from a minimally obfuscated PE Detective test build in February, to a functional multi-stage PowerShell loader by mid-March, to a substantially more sophisticated April variant featuring substitution cipher decoding, XOR-encrypted shellcode stubs, DLL sideloading, and a re-architected per-victim C2 callback model — all within roughly ten weeks. This pace, combined with the architectural sophistication of the staged loader chain, raises the question of whether the operators are leveraging large language model (LLM) tooling to accelerate development, consistent with what BlueVoyant has begun to observe more broadly across emerging threat groups augmenting traditional development with AI-assisted coding workflows.

Equally notable is the operational funding profile the campaign exposes: continuous registration of disposable NameCheap domains weaponized within hours, systematic procurement of Microsoft ID Verified code-signing certificates under multiple individual identities with three-day burn cycles, dedicated single-purpose hosting infrastructure allocated one IP per domain across multiple provider ranges, and sustained SEO-poisoning campaigns competitive enough to outrank legitimate Microsoft Teams results — all of which represent meaningful and recurring operational expenditure that distinguishes this actor from low-budget operators. The reliance on SEO poisoning for initial access and the use of hardcoded API strings rather than hashed resolution still suggest the group lacks the sophistication characteristic of advanced persistent threats, but the trajectory is clearly upward. BlueVoyant assesses with moderate confidence this is a relatively new but rapidly developing and adequately resourced criminally motivated actor, potentially operating as an initial access broker establishing footholds for

sale to downstream actors, though the absence of observed final-stage payloads precludes definitive attribution of ultimate objectives.

Conclusion

The Lorem Ipsum campaign represents the emergence of a capable, well-resourced, and rapidly maturing mid-tier criminal threat actor whose operational tempo warrants proactive defender attention. The combination of validly signed installers, multi-stage in-memory loader chains, JFIF-disguised C2 traffic, per-victim UUID tracking, and creative dead-drop abuse of a legitimate regional platform produces a meaningfully harder-to-detect delivery and command profile than typical commodity operations, while sustained operational expenditure on disposable domains, short-lived signing certificates, and dedicated hosting points to a meaningful and recurring budget.

Defenders should treat SEO-poisoned fake Microsoft Teams installer portals as a current and elevated initial-access risk and prioritize behavioral detections — non-browser process traffic to ***letsdiskuss[.]com/user/****, the ***/api/init/{UUID}*** callback pattern, JFIF-formatted POSTs to non-image endpoints, ***-WindowStyle Hidden*** PowerShell launched from msiexec writing into ***%AppData%/Roaming***, and short-validity Microsoft ID Verified developer certificates — over any single static IOC. Given the demonstrated development velocity and likely AI-assisted tooling pipeline, defenders should expect continued capability improvements at a pace meaningfully faster than historically typical for mid-tier criminal operations and treat the underlying behavioral patterns documented in this report as the durable detection surface for tracking this threat group going forward.

MITRE ATT&CK Techniques

T1102.001 (Dead Drop Resolver)

T1584.006 (Compromise Infrastructure: Web Services)]

T1583.006 (Acquire Infrastructure: Web Services)

T1102 (Web Service)

T1608 (Stage Capabilities)

T1608.002 (Upload Tool)

T1195 (Supply Chain Compromise)

T1588.002 (Tool)

T1587.001 (Malware)

T1027.004 (Compile After Delivery)

T1140 (Deobfuscate/Decode Files or Information)

T1127 (Trusted Developer Utilities Proxy Execution)

T1564 (Hide Artifacts)

T1584 (Compromise Infrastructure)

T1583 (Acquire Infrastructure)

T1518 (Software Discovery)

T1562.001 (Disable or Modify Tools)

T1592.003 (Firmware)

Indicators

hxxps://www[.]letsdiskuss[.]com/user/dhuahsd12d2752

hxxps://official-teams-storage[.]com/files_dws_arch/MTSetup_v15.3.71194.msi

official-teams-storage[.]com

graburban[.]com

valeurban[.]com

semigoddess[.]com

reeeeealy[.]com

biblegodlike[.]com

letsdiskuss[.]com

plainraw[.]com

82ebca8612e203f6d8a2dcdc5e586095ebf94e5e29724ba92cd8bd090df47eb2 - Setup_MT_V14.63.msi

ba5d73ca2c5aced43c7605e5652ba31fc63ca9b1f419ee4b934757c010c60f75 - MTSetup_v15.3.7110.msi

448afbdb6752c86e627d269ea244994d2c072d5110b490232dd7834943b043cb - SetupMT_V5_7765.msi

045b76fa552dbfdfb7e5de66c9c599fe91151384be6a9849ec8965aa7251b818 - Tms_Setup__V5.4.140.msi

---

Source: https://www.bluevoyant.com/blog/lorem-ipsum-trojanized-microsoft-teams-installers-multi-stage-loader-backdoor