

Same packet, different magic: Mustang Panda hits India's banking sector and Korea geopolitics

 acronis.com/en/tru/posts/same-packet-different-magic-mustang-panda-hits-indias-banking-sector-and-korea-geopolitics



Summary

Acronis Threat Research Unit (TRU) identified a new variant of the [LOTUSLITE](#) backdoor with a theme related to India's banking sector, delivered via DLL sideloading using a legitimate Microsoft-signed executable.

The backdoor communicates with a dynamic DNS-based command-and-control server over HTTPS and supports remote shell access, file operations and session management, indicating a continued espionage-focused capability set rather than financially motivated objectives.

Code-level analysis confirms direct lineage to LOTUSLITE, including identical command structures, shared persistence mechanisms, and a residual **KugouMain** and **multiple other exports** carried over from the original codebase, establishing this as an evolved build from the same developer rather than an independent tool.

This variant demonstrates incremental improvements over its predecessor, suggesting the developer is actively maintaining and evolving the implant between campaigns.

The campaign reflects a shift in delivery tradecraft of Mustang Panda's cluster delivering LOTUSLITE, which is moving from CHM-based delivery to JavaScript loaders to DLL sideloading across recent operations, while also pivoting geographically from U.S. government entities to India's financial sector.

Attribution to Mustang Panda is assessed with moderate confidence based on shared code lineage, residual build artifacts and consistent operational patterns observed across both campaigns.

Introduction

Acronis Threat Research Unit (TRU), following the discovery of the LOTUSLITE backdoor, has focused on actively monitoring the implant, along with tracking malware campaigns and threat activity linked to recent geopolitical developments in the West Asian region. During this monitoring, TRU identified the use of the LOTUSLITE implant in [targeted campaigns](#) leveraging these developments, with activity observed in March this year.

The current activity, which we talk about in this report stands out due to its sudden deviations from the usual targeting vectors, which are mostly government and similar entities part of the global crisis to targeting, which we believe to be the banking sector of India, with a little deviation to the TTPs. These observations prompted further analysis to understand the purpose of the change, the relationship between its components and the infrastructure used to operate the backdoor.

This report focuses on documenting the delivery mechanism, execution flow and command-and-control (C2) behavior associated with this variant of LOTUSLITE, as well as examining how this activity aligns with broader espionage-oriented threat trends. And, just like the previous campaign which focuses on original discovery of LOTUSLITE, documented by the TRU team, this variant also demonstrates limited technical sophistication, its targeting, and thematics and other interesting artifacts.

Through this research, TRU outlines the analytical basis for attributing this activity to a cluster tied to Mustang Panda with moderate confidence, primarily based on observed overlaps in terms of **shared-codebase, infrastructural-patterns, deployment patterns** and operational characteristics and OPSEC mistakes consistent with our findings and previously documented campaigns. The report details the indicators and behaviors that informed this assessment, providing defenders with context for understanding how such overlaps can support attribution.

Attack chain

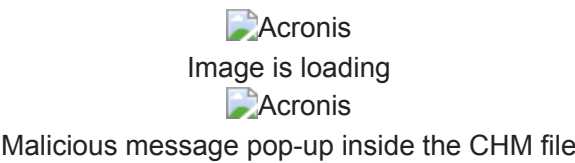


Image is loading



Technical details

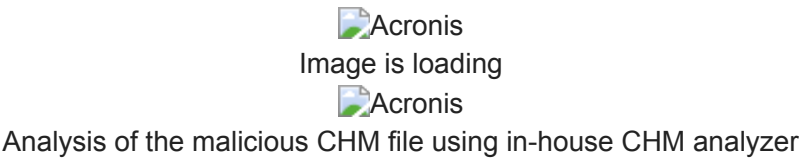
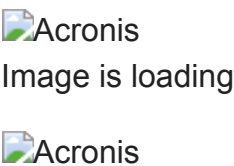
Initial discovery and delivery mechanism



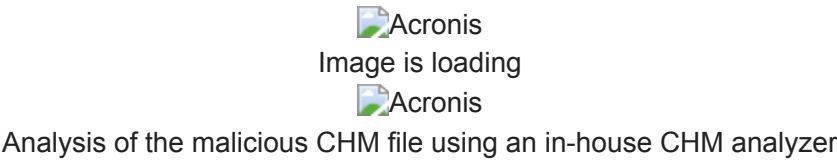
Our investigation began following the surfacing and detection of a suspicious and fresh CHM or Compiled HTML file, which we believe have been delivered via spear phishing, known as Request for Support.chm, uploaded to the public sandbox known as VirusTotal.

Further, looking at the naming convention, this stood out due to the help-oriented ticketing nomenclature often used at banking-based institutions and similar conventions, which downloaded a malicious implant and a signed executable along with execution facilitated by a JavaScript file. These observations prompted a deeper analysis of the campaign artefacts and changes into the payload and its execution flow along with the infrastructure used to control the implant, with pivotal intelligence to hunt for other campaigns-in-the-wild (ITW) alongside its attributional artifacts.

Malicious CHM file

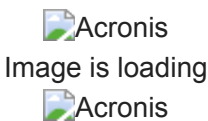


Initially, looking inside the CHM File using our in-house CHM analysis tool, we found that the CHM file contained a total of six objects, out of which the first interesting one was HTML present inside the archive_1.html file containing a pop up that prompted the user to click on **Yes**, while prompting for further malicious actions contained a redirect to another web-server that downloaded and executed a malicious JS File.



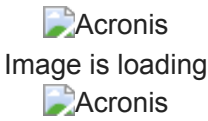
Also, we found that the CHM file contained a Microsoft-signed executable and the malicious implant — the updated version of the LOTUSLITE backdoor. In the next section, we will look at the malicious JavaScript file.

Malicious JavaScript File



Code structure of the malicious JavaScript file

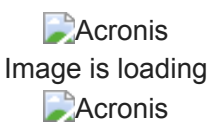
Finally, looking inside the JavaScript code, which is named as music.js, served from the domain **hxxps[:]//cosmosmusic[.]com**, we found that it contained a total of three functions, which are **aownroia**, **getCwd** and **calniown**.



Analysis of the malicious JavaScript function

We found that the function **aownroia** was basically responsible for dropping and running the malware hidden inside the CHM file. It built a chunk of HTML containing three ActiveX objects and injected it into a hidden div on the page. The first object told [hh.exe](#) to decompile the CHM and dump its contents, including **Microsoft_DNX.exe**, into **C:\Users\Public\Documents**. The second one simply ran that extracted executable from the same path. There was also a third object using the **Scriptlet.TypeLib CLSID**, which was there to get around the security restrictions that would normally block the other two from running. After injecting all of this into the page, it waited 600 milliseconds and then called **calniown()**, which clicked both objects in order, first extracting the files, then executing the payload. So, in short, this function handled the full drop-and-execute chain by abusing [hh.exe](#) as the extractor and the **ShortcutCommand** ActiveX control as the launcher.

DLL sideloading



DLL sideloading into a signed executable

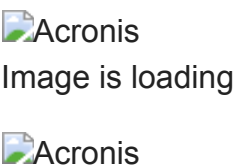




Image is loading



Code related to DLL sideloading

Once the CHM file dropped the executable and the DLL file in the same directory and the JS file completed the execution, we found that the executable was basically a Microsoft-signed binary known as Microsoft DNX, which is responsible for running .NET applications in a lightweight runtime environment. It's a legitimate developer tool from Microsoft that was part of the older ASP.NET ecosystem. The reason the threat actor chose this specific binary is because it's signed and trusted by [Windows](#), so it won't raise any red flags with security products. Looking further into the executable launcher named **Microsoft_DNX**, we found that it dynamically loaded the DLL at runtime using [LoadLibraryExW](#) and resolved the exported function DnxMain via [GetProcAddress](#), rather than relying on the import table for implicit loading. Since the binary loaded the DLL by name without verifying its authenticity or specifying a full path, the attacker can place their own crafted DLL, which is the LOTUSLITE variant in this case, with the same filename alongside the executable. When Microsoft_DNX.exe runs, it picks up the malicious DLL from its own directory and transfers execution directly into the attacker-controlled **DnxMain** function, all under the context of a legitimate, Microsoft-signed binary.

LOTUSLITE backdoor - v1.1

As we continued tracking this implant, we identified notable changes in its code structure, C2 communication and several smaller implementation details. We'll cover these changes below while continuing to track this variant under the designation **LOTUSLITE v1.1**, reflecting its iterative evolution from the original.



Image is loading



Differences between the number of export functions

The first update we noticed was right in the DLL's export table. In version 1.0, the implant exported 16 functions, with DataImporterMain sitting at the [ordinal](#) 4 as the main entry point containing the core operational logic. Version 1.1 bumped that count to 22 exports, and DataImporterMain was nowhere to be found. Instead, we saw DnxMain taking its place at ordinal 4, which now acts as the routing function that calls into **HDFCBankMain**, a new target-specific export whose name alone made the campaign's focus on India's banking sector clear.

Even the Evt functions got a subtle touch, being renamed from EvtClose, EvtNext, EvtQuery, EvtRender to Evt1Close, Evt1Next, Evt1Query, Evt1Render, which hints at some kind of internal versioning the developers are maintaining.

Overall, the developers have moved from a monolithic design in v1.0 to something more modular in v1.1, splitting things out into dedicated functions rather than cramming everything into a single-entry point.



Image is loading



Difference and updates related to code artefacts

Further, looking at the updated version of the implant, we found that the threat actor removed the comments such as “**Goofball**” and many other sarcastic remarks inside the payload, keeping the updated version slightly professional, while just adding the prefix ‘0’ into the decoy name, while, just like the previous version, the decoy does not spawn, once the payload is being executed.



Image is loading



Difference and updates related to Mutex Nomenclature

In a subsequent update, the threat actor introduced a new mutex named “[mdseccoUk](#),” which appears to reference a U.K.-based security company specializing in adversary simulation.



Image is loading



Differences and updates related to flags.

The next update in the implant was the change in execution of flags. The original LOTUSLITE used **--DATA** as its internal switch to decide whether the malware should install itself or start beaconing, while this variant swaps it out for **--ZoneMAX**. Functionally, nothing changes; both flags do the exact same job. But the fact that we think that the developer bothered to change it at all tells us they're aware that defenders may have written detection rules around the **--DATA flag** from the first campaign, and rotating it is a lazy update to stay ahead.



Image is loading



Difference and updates related to command-and-control (C2) infrastructure

Then, looking into the other update, we found that the threat actor had updated the C2-based infrastructure related to the fresh set of implant, but hardcoded it inside the implant, just like the initial version, where the fresh set of infrastructure is **editor[.]gleeze[.]com**.



Image is loading



Image is loading



Differences and updates related to network-based artefacts

Along with that, we also saw that there have been changes in terms of the other network-related artefacts such as the User-Agent, Host-Header and Session-Cookie, while the Referrer remained the same. The new request looked like this:



Image is loading



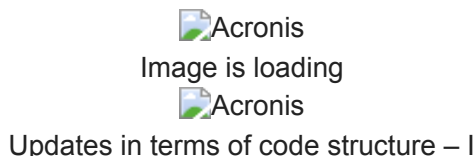
Differences and updates related to magic value

Then, looking into one of the most interesting changes we noticed in this updated version of the LOTUSLITE backdoor was the change in the magic value. In the original version, the function responsible for building the raw C2 packet used the hardcoded magic value **0x8899AABB** as the first four bytes of every packet header, acting as a signature that tells the C2 server the incoming data is from a valid implant. In v1.1, this value had been swapped to **0xB2EBCFDF**. While the packet structure itself remained identical, magic, followed by the command or response code, followed by the data length, followed by the payload, the change in magic value means that any network-based detection rules written against the original **0x8899AABB** marker will no longer match traffic from this variant. Beyond just the magic value, the way the packet was assembled also changed; the original version directly wrote the header into memory using a simple memmove, while the new version resolves memcpy from ntdll.dll at runtime through the LdrLoadDll resolution chain, avoiding a direct import which we believe had been added against the measure that static analysis tools could flag.

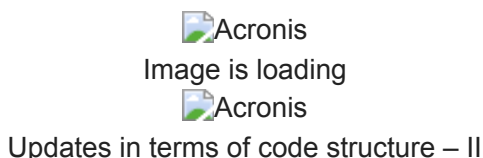
Beyond the magic value change, the rest of the implant functionality remained almost identical to LOTUSLITE v1.0, in terms of capabilities, the same command set, the same packet structure and the same file operations. What has changed is how the implant calls Windows APIs. In the original version, functions like WinHttpOpen, WinHttpConnect, CreatePipe, CreateProcessA, and even memcpy were called directly or through simple imports. In v1.1, every single API call is wrapped in the same resolution pattern that is to get a handle to ntdll.dll via GetModuleHandleA, resolve LdrLoadDll and RtlInitUnicodeString from it, then use LdrLoadDll to load the target library, whether that's winhttp.dll, kernel32.dll, advapi32.dll, or shlwapi.dll and then finally resolve the actual function by name.

This means the DLL's import table stays clean and gives nothing away about its real capabilities, making it harder for static analysis tools and endpoint detection to flag it based on suspicious API imports alone. The developer applied this pattern uniformly across the entire codebase, even for trivial functions like memcopy, which suggests this was a deliberate move to avoid static engines alone — for the time being.

Persistence

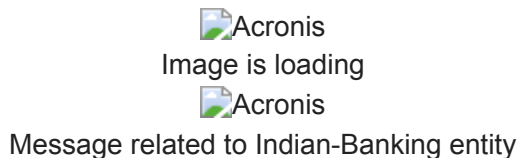


Just like the above updates, another notable change was how persistence was handled. In v1.0, the registry key was written directly inside the install function using a plain call to SHSetValueA, sitting right next to the file copying logic with no effort to obscure it. In v1.1, the developer moved the persistence logic into its own separate function and wrapped the SHSetValueA call behind the same **ntdll** resolution chain used across the rest of the implant. Same registry key, same Run key path, same technique, just modularized and hidden from static analysis.



There was also a small but interesting difference in how the two versions handled the victim's experience after installation. In v1.0, the malware displayed a fake error message saying the **Word file was corrupted**, giving the victim a plausible reason why nothing seemed to happen when they opened the file. In v1.1, that social engineering step is missing entirely. Whether that was a deliberate choice or simply the developer, cutting corners between campaigns is unclear as of now.

Message via Exports



One of the most interesting messages via the exports was the pop-up window created via the MessageBox API, which prints "**HDFC Bank Limited**", disguising itself as a legitimate connection to the [banking institution](#). However, we believe that the export carrying this message is still named KugouMain, a residual artifact from the original LOTUSLITE codebase where the DLL was named kugou.dll after the Tencent music application used for sideloading.

The developer updated the message strings for the new Indian Banking-focused campaign but forgot to rename the export itself, leaving behind one of the strongest code-lineage indicators linking this variant directly back to the original LOTUSLITE build, which was [discovered](#) by the Acronis TRU Team.



Image is loading



Image is loading



Message related to researcher tracking Mustang Panda

Another interesting artefact which is present inside this updated version is the pop-up which contains the message “**goldenjackel12**”; with a little research, we found out that this string corresponds to the X handle of an independent threat researcher known as [R3BELFOX](#), who has been actively and publicly tracking Mustang Panda and UNC6384 campaigns, regularly posting indicators such as LNK filenames and sample hashes. We think that the developer is not only aware of being tracked but comfortable enough to call it out directly within their tooling. We saw similar behaviors in LOTUSLITE v1.0, where the developer embedded identity-related messages within dummy export functions. It seems to be a recurring habit of this particular operator.

Infrastructural artefacts

The implant we analyzed communicated with a C2 server at editor[.]gleeze[.]com, a free subdomain provided by Gleeze, a dynamic DNS service run by Dynu Systems Incorporated. Notably, this is the same DNS provider behind the original LOTUSLITE C2 infrastructure, where the implant communicated with 172[.]81[.]60[.]97 under Dynu's network (ASN AS398019). The reuse of Dynu-managed infrastructure across both campaigns further supported the connection between the two operations. As with the previous version, the implant connected over TCP port 443 to blend in with normal HTTPS traffic.

In the next section, we will encounter the overlaps in terms of OPSEC blunders from the maintainer of the implant, which will be leveraged to pivotal intelligence.

Pivotal intelligence

Old Habits, Die Hard – South Korean campaign



Image is loading



Image is loading



Pivotal intelligence and findings

As discussed in the previous sections, we found that the threat actor carried over multiple export names from previous versions of the implant into newer campaigns, which we believe have been targeting South Korean policy and diplomatic circles, based on some interesting artefacts, which we found, along with this campaign, such as material referencing Victor Cha, a prominent figure in U.S.-Korea policy and Senior Vice President at the Center for Strategic and International Studies (CSIS). Cha also served on the U.S. National Security Council as Director for Asian Affairs under President George W. Bush and is widely regarded as one of the most influential voices on North Korea policy, U.S.-South Korea relations and Indo-Pacific security.

During our hunting efforts, we came across another sample that continued this pattern. This time, the archive was themed around an invitation letter for a committee meeting, with the legitimate executable renamed to match the lure. The malicious DLL shipped alongside it still contains the HDFCBankMain export from the banking-themed campaign, even though the lure has nothing to do with HDFC Bank. The legitimate executable loads the DLL through the MddBootstrapInitialize2 export, which is simply a thunk that jumps straight into the main backdoor logic.

Once again, the developer swapped the lure and adjusted the exports needed for the new sideloading host but left the old ones sitting in the binary untouched. It is the same carelessness we saw with KugouMain carrying over from LOTUSLITE v1.0 into the HDFC campaign, a pattern that at this point appears to be a **signature** of this particular developer.

Similar victimology



Image is loading



Contents of a spear-phishing campaign-based ZIP file

While we have previously seen that this group had been targeting important meetings and summits across the globe at various targeting nations, we also found in this case that it had been using the well-known technique DLL sideloading with renamed binaries placed inside the same directory for the implant to be executed.

Actual Updates



Image is loading



Updates in decoy and mutex

Upon analyzing the implant, which was targeted for South Korean entities, we found out that there were no changes to the implant, except the name of the Mutex and the decoy, which were different in this case — where the mutex was 1ac5e7ee1a107499, and the decoy was 323.txt in this campaign.

South Korean connection



Image is loading



Discovery of the attacker's infrastructure, used to host malicious spear-phishing elements

While we discovered the file on March 24, 2026, we also saw that the malicious file was being delivered through a Google Drive account, which contained a folder known as "March 30", which we believe is the staging directory used by the threat actor. Apart from that, we also found that the threat actor used a dummy email mimicking Victor Cha. The spoofed email address, victorcha707@gmail.com, was used to deliver the archive to what we assess are individuals within the Korea policy and diplomatic community, further reinforcing the espionage-driven nature of this campaign.



Image is loading



Further interesting artefacts related to the Google account

Further, we also found that the threat actor had been using Victor Cha's personal photo as the profile picture of this fake Gmail account, which was pretending to be Victor Cha; also, we found that the threat actor had been active at Google Meet services as well as they also have a Google Maps [account](#), which had been active, but now has been stripped of comments and other information — we believe by the threat actor.

With this information, we believe that the group had been targeting certain entities belonging to the South Korean and U.S. diplomatic and policy communities, specifically those involved in Korean peninsula affairs, North Korea policy discussions and Indo-Pacific security dialogues.

Attribution and confidence

Based on our analysis, we assess with moderate confidence that this campaign has been attributable to a cluster belonging to Mustang Panda. This assessment is based on multiple independent lines of evidence, as outlined below.

The implant shares direct code-level lineage with LOTUSLITE v1.0, including identical command IDs across the entire command set (0x0A, 0x0B, 0x01, 0x06, 0x03, 0x0D, 0x0E, 0x0F), the same binary TLV packet structure, the same persistence mechanism using SHSetValueA to write a Run key under HKCU, and the same two-phase execution model controlled by a command-line flag. The residual KugouMain export found in the HDFC campaign DLL directly ties this variant to the original LOTUSLITE codebase, where the DLL was named kugou.dll after the Tencent music application used for sideloading.

The C2 infrastructure for this campaign is hosted under editor[.]gleeze[.]com, a free dynamic DNS subdomain operated by Dynu Systems Incorporated (ASN AS398019). This is the same DNS provider and the same ASN used in the original LOTUSLITE campaign, where the C2 was hosted at 172[.]81[.]60[.]97 under Dynu's network. The continued reliance on Dynu-managed disposable infrastructure is a consistent operational pattern across both campaigns.

The delivery tradecraft remains aligned with established Mustang Panda behavior, including the use of DLL sideloading via legitimate signed executables, staging files under C:\ProgramData, deploying thematically relevant lures tied to real-world geopolitical events and figures, and targeting government, diplomatic and policy-related entities. The impersonation of Victor Cha and the use of a spoofed Gmail account to deliver the payload through Google Drive further reflects the group's known social engineering methods.

The developer exhibited a recurring habit of embedding messages and references within dummy export functions, a behavioral trait first observed in LOTUSLITE v1.0 where identity-related statements were placed inside the EvtNext and EvtQuery exports. In this variant, the same pattern continues with the HDFC Bank-themed MessageBox pop-ups and the goldenjackel12 reference to an independent researcher tracking Mustang Panda campaigns. This is not a common practice among threat actors and serves as a distinctive behavioral fingerprint linking both campaigns to the same developer.

The campaign has demonstrated an active and evolving development cycle, with incremental changes between builds such as rotating execution flags (--DATA to --ZoneMAX), changing magic values (0x8899AABB to 0xB2EBCFDF), upgrading API resolution to a layered ntdll chain, and cycling through different legitimate binaries for sideloading (KuGou to Microsoft DNX to kwpswnssserver). Despite these updates, the core implant architecture, command protocol, and operational playbook remain unchanged, indicating a single developer or closely coordinated team iterating the same codebase across campaigns.

Conclusion

This campaign has demonstrated how Mustang Panda continues to rely on simple, well-tested techniques that remain effective when paired with targeted delivery and relevant lures. The LOTUSLITE variant we uncovered retains the same core architecture, command set and operational playbook as its predecessor, while introducing incremental improvements to evade detection.

What stands out is the broadening of the group's targeting, from U.S. government entities with geopolitical lures, to India's banking sector through implants embedded with HDFC Bank references and pop-ups masquerading as legitimate banking software, and now to South Korean and U.S. policy circles through the impersonation of a prominent figure in Korean peninsula diplomacy, delivered via spoofed Gmail accounts and Google Drive staging. With moderate confidence, we attribute this activity to Mustang Panda based on shared code lineage, overlapping infrastructure, residual build artifacts and consistent behavioral patterns observed across all three campaigns.

IoCs

SHA256 Hashes

EXEs

Af31ebe9085df408bedcf8f027fb60389897e5c8d3b0e9695fea29774f9d3aec -
Microsoft_DNX.exe

cc0ff7e25ea686171919575916e2d9ebaeb5800a063f370a6980ea791f8851b8 -
kwpswnssserver.exe (renamed to INVITATION LETTER lure)

DLLs

7beede15ecdc7d3f01db4b699e5fe5f4f2e7c79cd7ef0e918ed0583bf621de7d -
dnx.onecore.dll

9bf2f3b15a621789f898f9bd7710ba857e3f238a4937b64fdc47ef9a92e0b05d
Microsoft.WindowsAppRuntime.Bootstrap.dll

Archives

18bc0e0f627d90fb283aa243055b46d0bfb5d85a7240d8f63ec2d1c8a2c15893 -
Request_for_Support.chm

6d22d50634c2c2fc853bfd2b564e1837d51087aa684a9c4415634c8c13c44135 -MARCH
30.zip

Domains

editor[.]gleeze[.]com

www[.]cosmosmusic[.]com

Mutex

mdseccoUkFuiCkTrump

1ac5e7ee1a107499

Paths used for persistence

C:\ProgramData\Microsoft_DNX\