

Behind Khmer Shadow: Targeted espionage against Cambodian government entities

 acronis.com/en/tru/posts/behind-khmer-shadow-targeted-espionage-against-cambodian-government-entities

Acronis TRU has identified two targeted campaigns against Cambodian government entities, with focus on defense and military intelligence sectors, delivered by a cluster we've tracked as **Khmer Shadow**.

Both campaigns used a meeting-themed SFX archive to sideload a custom C++ loader we tracked as **NIGHTFORGE**, which decrypts and executes a [Havoc Demon](#) payload in memory.

NIGHTFORGE has demonstrated a moderate level of sophistication, combining advanced defense-evasion techniques such as NTDLL unhooking and [Hell's Gate syscall resolution](#), a method that enables direct system calls and helps evade user-mode monitoring, with operational shortcomings that suggest the tool is still under active development.

The group reused identical payloads and infrastructure across both campaigns with minimal retooling between targets.

Based on targeting patterns, lure construction and operational characteristics, TRU has assessed with moderate confidence that the activity is espionage motivated and likely aligned with regional intelligence collection interests in Southeast Asia.

Introduction

Acronis [Threat Research Unit](#) (TRU) has identified two espionage-focused campaigns targeting Cambodian government entities in the defense and public works sectors. Both campaigns delivered a previously undocumented DLL-based loader, which we've tracked as **NIGHTFORGE**, through government-themed lures packaged in self-extracting (SFX) archives. Once executed, NIGHTFORGE decrypts and launches a Havoc Demon payload directly in memory.

The campaigns shared nearly identical tooling and infrastructure, with minimal changes between targets, suggesting a low retooling tempo by the operator. Based on targeting patterns, lure construction and infrastructure characteristics, TRU has assessed with moderate confidence that the activity is espionage-motivated and likely aligned with regional

intelligence collection interests in Southeast Asia. We've tracked this previously unreported cluster as **Khmer Shadow**, based on its observed focus on Cambodian government entities.

Attack chain

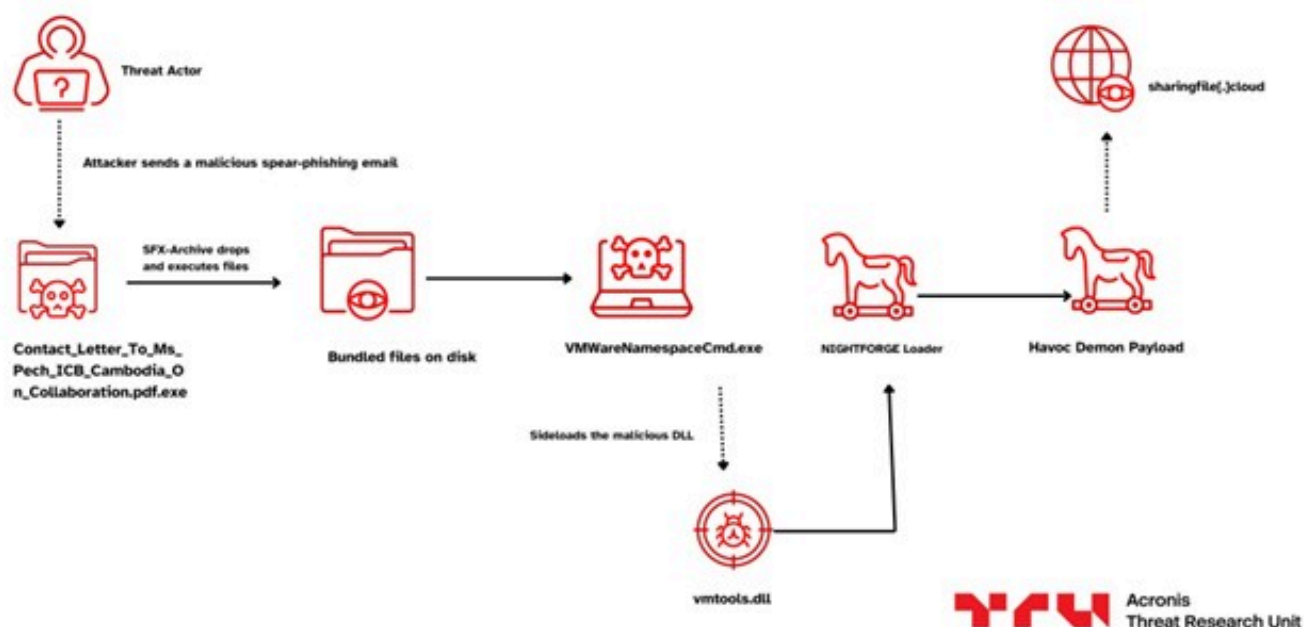


Figure 1: Attack-Chain demonstration of the payload execution

The attack chain begins with a suspected spear-phishing email delivering a compressed SFX archive named *Contact_Letter_To_Ms_Pech_ICB_Cambodia_On_Collaboration.pdf.exe*. The archive bundles multiple benign library files alongside a malicious DLL used for sideloading. The legitimate executable is *VMwareNamespaceCmd.exe* and the malicious DLL is *vmtools.dll*.

Once the victim executes the archive, an embedded script launches *VMwareNamespaceCmd.exe*, which sideloads *vmtools.dll*. NIGHTFORGE then establishes persistence, drops the Havoc Demon payload, and connects to the command-and-control (C2) server.

The key difference between the two campaigns was the decoy document. The campaign targeting Cambodia's Information Collection Bureau (ICB) included a decoy PDF, while the campaign targeting the Ministry of Public Works and Transport did not.

Technical details

Initial analysis and delivery mechanism

Figure 2: Execution script found inside the SFX-compressed executable

The investigation began with the detection of *Contact_Letter_To_Ms_Pech_ICB_Cambodia_On_Collaboration.pdf.exe* on [VirusTotal](#), a SFX archive masquerading as a PDF document. Once extracted, the archive contained a malicious script responsible for executing the loader, deleting all DLL files, the decoy PDF and remaining artifacts including the SFX archive itself from disk, a basic cleanup routine used by the threat actor to hide its tracks.

Pivoting on the *vmtools.dll* hash, the *VMwareNamespaceCmd.exe* binary, the SFX file structure, and the upload time, we identified a second sample uploaded to VirusTotal from Cambodia within a similar timeframe. The second archive, named **CN_Contact_Work_Cambodia's_Ministry_of_Public_Works_and_Transport.pdf**, shared an identical file structure and the same sideloading chain.

Victimology for both campaigns was deduced from the archive filenames, the decoy document content and the named recipient in the lure letter, which collectively pointed to the Information Collection Bureau under Cambodia's Ministry of National Defense for the first campaign, and the Ministry of Public Works and Transport for the second. The only notable difference between the two was the absence of a decoy document in the second campaign.

Looking into the decoy

Of the two campaigns, only the operation targeting Cambodia's military intelligence apparatus included a decoy document. No decoy was present in the campaign targeting the Ministry of Public Works and Transport.

Figure 3a, 3b and 3c: Decoy letter snippet (Chen Minglong / Ms. Pech Nget)

When analyzing the decoy document, we found that the letter is addressed to Ms. Pech Nget, who appears to work under the Information Collection Bureau (ICB) of Cambodia's Ministry of National Defense. The sender signs off as Chen Minglong from a "Development and Investment Division" based in Beijing. We could not find any public trace of either individual, which suggests the operator may have fabricated both personas to lend credibility to the lure.

Looking at the content of the decoy, the letter talks about an upcoming "mission in May 2026" and lays out three areas of proposed coordination: setting up a data exchange protocol between the sender's division and the ICB, discussing what technical resources the ICB team can provide and sorting out clearances and meetings for a site visit with MOD personnel. It

reads like any routine bilateral cooperation email between China and Cambodia, exactly the kind of thing someone in defense coordination would open without a second thought. Notably, the letter references an individual named "Youthy" and an "EOD Administrative team." Although we could not independently verify either reference, their inclusion suggests the operator attempted to incorporate organization-specific details into the lure to enhance its credibility.

DLL sideloading

Figure 4: VMwareNamespaceCmd.exe digital signature

Figure 5: DLL export table

When the victim executes the SFX archive, *VMwareNamespaceCmd.exe* and the malicious *vmtools.dll* are placed in the same directory. *VMwareNamespaceCmd.exe* is a legitimate VMware-signed binary that statically imports several functions from *vmtools.dll*. As a result, Windows automatically loads the malicious DLL before the application begins execution.

One of the first functions called by the binary is **VmCheck_IsVirtualWorld**. The attacker-controlled DLL exports this function and returns TRUE, causing the VMware application to continue operating normally while NIGHTFORGE executes under the context of a trusted VMware-signed process.

NIGHTFORGE loader

Analysis of *vmtools.dll* showed that it acts as a custom loader, which decrypts and executes a shellcode payload stored in a companion file, ultimately launching the next stage of the infection chain.

Figure 6: IDA Imports view showing vmtools.dll imports

Despite minor differences in how the payload is stored across the two campaigns, the NIGHTFORGE loader is nearly identical in both samples. While the DLL exports multiple functions, execution ultimately passes through *VmCheck_IsVirtualWorld*, which serves as the loader's primary entry point.

Figure 7: PEB ImagePathName resolution

Rather than calling *GetModuleFileName*, the loader resolves the full image path of the running executable directly from the Process Environment Block (PEB) via *ImagePathName*, retrieving it as a Unicode string.

Figure 8: IsRunningFromStagedLocation and other code logic

Once the path is retrieved, the loader runs through three key functions: checking if it's running from the intended persistence directory, hiding the foreground window, then loading and executing the shellcode from disk.

Figure 9

Next, the *IsRunningFromStagedLocation* function compares the current image path against the temp directory and the Windows directory. Running from temp means it hasn't staged yet. Running from the Windows directory confirms it's already sitting in its persistence location under *VMwareNamespace* and can proceed with payload decryption and execution.

Figure 10: HideForegroundWindowAndCheckInteractive function inside the loader

The function we renamed *HideForegroundWindowAndCheckInteractive* checks whether an active foreground window is present and hides it to reduce the loader's visibility to the user. It also verifies that the current session is interactive, ensuring the payload executes only when a user is actively logged in rather than in a headless or sandboxed environment. If these checks succeed, execution is transferred to *LoadDecryptAndExecutePayload*.

LoadDecryptAndExecutePayload is responsible for the core stages of the infection chain. It first unhooks NTDLL (NT Layer Dynamic-Link Library), then decrypts the encoded shellcode stored on disk, and finally executes the payload in memory using direct syscalls.

Figure 11a and 11b: LoadDecryptAndExecutePayload and NTDLL unhooking

The loader obtains a handle to the in-memory copy of ntdll, loads a clean copy directly from disk, and maps it into memory using the *CreateFileMappingW* and *MapViewOfFile* APIs. It then walks the PE section headers to locate the .text section and overwrites the in-memory copy with the clean version from disk. This technique of unhooking the NTDLL is therefore used to remove the inline hooks that have been placed on the NTDLL functions, which is basically an anti-malware bypass technique. Then it goes ahead and reads the file size of the encoded shellcode file on the disk using APIs such as *GetFileSize*.

Figure 12a and 12b: Hell's Gate syscall resolution

Next, it resolves syscall numbers at runtime using **Hell's Gate**, locating *NtAllocateVirtualMemory*, *NtWriteVirtualMemory*, *NtCreateThreadEx* and *NtWaitForSingleObject* via *GetProcAddress* and scanning each stub for the **4C 8B D1 B8** signature to extract the syscall number and syscall instruction address.

Figure 13: XOR Key Derivation

Next, once the syscalls have been resolved via the Hell's Gate technique, the code decodes the encoded shellcode blob present on disk using a simple XOR method. It reads the first 8 bytes of the encrypted file and XORs them against a hardcoded magic value (**42 3A 4B 5C F0 EA 72 18**) to derive an 8-byte key, then decrypts the remaining contents using a rolling 8-byte XOR with that derived key.

Figure 14: Shellcode injection via direct syscalls

Further the decrypted shellcode is then injected into the current process by allocating RWX memory via *NtAllocateVirtualMemory*, writing the payload into it with *NtWriteVirtualMemory*, and spawning a new thread through *NtCreateThreadEx*. At that point the loader's job is done, and execution moves to establish persistence.

Persistence

Before diving into the next stage involving the shellcode, we found that the loader uses a slightly interesting way to establish persistence.

Figure 15a, 15b and 15c: COM-based persistence used in NIGHTFORGE

We saw that it is using COM APIs for persistence. Initially it calls *CoInitialize* to set up the COM library, then uses *CoCreateInstance* to create an instance of the Task Scheduler service. It then calls *ITaskService::Connect* through the *vtable* to establish a connection, followed by *ITaskService::GetFolder* to access the root task folder with the path "\\".

From there it builds the scheduled task definition with a registration trigger configured with a start boundary of "2010-01-01T00:00:00", a repetition interval of "PT10M" (every 10 minutes), and a duration of "P1D" (one day). It sets the executable action pointing to the loader's path in the staged directory, and finally calls *ITaskFolder::RegisterTaskDefinition* with the *TASK_CREATE_OR_UPDATE* flag to register the task under the name "VMwareNamespase", which blends in with legitimate VMware entries in the task scheduler.

KaynLdr and the Demon

Figure 16: Code depicting KaynLdr

Analysis of the decoded payload revealed that the penultimate stage is [KaynLdr](#), a well-known open-source reflective loader used to load the final-stage Havoc Demon implant.

Figure 17: Code depicting KaynLdr's FindEmbeddedPE function

The first part of this reflective loader looks for an embedded PE file inside its own code address and validates the outcome until a valid MZ header is found. For ease, we have renamed this function to *FindEmbeddedPE*.

Figure 18a and 18b

Followed by another function, which uses the PEB and walks the InLoadOrderModuleList via PEB_LDR_DATA to enumerate all loaded modules in the current process. For each module, it hashes the BaseDllName using a DJB2 hashing algorithm with a hardcoded seed constant of **5381** followed by, comparing the result against a precomputed hash value to locate ntdll.dll.

Figure 19: Code depicting walking the PEB InLoadOrderModuleList and hashing each module's BaseDllName

Once the base address of ntdll.dll is resolved, the loader walks its export table to find three functions it needs: **LdrLoadDll**, **NtAllocateVirtualMemory** and **NtProtectVirtualMemory**, matching each export name against precomputed hash values. With these in hand, it parses the embedded PE's headers, allocates a fresh region of memory and copies each section into place at its expected virtual address. If the image lands at a different base than the author originally intended, the loader patches up all the absolute addresses by walking the relocation table.

The loader then sets the right page protections on each section, executable for code, read-write for data and read-only where appropriate, before handing off execution to the DLL's DllMain entry point. From this point on, the actual Havoc Demon implant is running entirely in memory, with no trace of the decrypted payload ever written to disk. Well, an important aspect of KaynLdr being used is that it erases the DOS and NT Header. Therefore, we extracted the final payload using a simple python script to get the last stage Havoc Demon.

Figure 20: Code depicting API Hashing

As analysis of the [Havoc Demon](#) payload revealed numerous capabilities consistent with the open-source Havoc C2 framework, including API hashing and other standard implant functionality. Given that these features have been extensively documented elsewhere, we focus on extracting and analyzing the agent configuration embedded within the payload.

Figure 21a and 21b: Havoc Demon config extractor output

Upon extracting the Demon agent configuration from the embedded DLL, we found that the implant communicates with its C2 server at **www[.]sharingfile[.]cloud** over HTTPS on port 443 using POST requests. The operator has set up the agent to look like normal browser traffic, using a Chrome user-agent string and standard browser headers that all point back to the same domain. The C2 traffic rotates across five endpoints designed to look like sections

of a news website: **/national**, **/world**, **/sport**, **/download** and **/regular**. For post-exploitation tasks, the agent spawns *gpupdate.exe* as its sacrificial process for injection, which is a legitimate Windows binary that wouldn't look out of place on the target machine.

Figure 22: Havoc Demon HTTP headers

Followed by which, we found that the demon implant is configured with HTTP headers that make its traffic look like someone browsing a website in Chrome. The user-agent mimics Chrome 131 on Windows 10, and the Origin and Referrer headers both point to `sharingfile[.]cloud`. It also sends Sec-Fetch headers that browsers normally include on every page of visit. The goal is simple: make the C2 traffic blend in with everyday web browsing, so it slips past network monitoring without raising suspicion.

Infrastructural artifacts

The Havoc Demon, dropped from the NIGHTFORGE Loader via KaynLdr, was observed communicating with `sharingfile[.]cloud` over port 443, extracted directly from the implant's config.

Figure 23: Information related to the C2 domain

Further pivoting, we found that the domain `sharingfile[.]cloud` was registered through NameSilo and currently sits behind CloudFlare's CDN, which acts as a reverse proxy to mask the true origin of the C2 server. On the CloudFlare edge, the domain resolves to `172.67.129[.]134` and returns a 403 Forbidden response with the page title "Attention Required! | CloudFlare", suggesting the operator configured CloudFlare's bot protection to filter out unwanted visitors and scanners.

Figure 24: CriminalIP view of `sharingfile[.]cloud`

Further investigation using CriminalIP revealed the origin server behind CloudFlare at `193.169.240[.]138`, hosted by SERVER.UA LLC in Mykolayiv, Ukraine. The server runs Microsoft-HTTPAPI/2.0, exposes only port 443, and returns a bare 404 response with a content length of just 18 bytes when accessed directly. A CloudFlare Origin Certificate tied to `sharingfile[.]cloud` is installed on the server, which we assess with moderate confidence confirms it as the backend for the C2 domain. The JARM fingerprint `3fd21b20d00000021c43d21b21b43de0a012c76cf078b8d06f4620c2286f5e` was then used for further pivoting.

Figure 25: Origin server details using CriminalIP

Figure 26: JARM pivot on Shodan

Pivoting on the JARM fingerprint through Shodan returned only two results at the time of our search. The first was our known C2 server at 193.169.240[.]38, hosted in Kyiv, Ukraine, serving sharingfile[.]cloud on port 443. The second was a previously unknown server at 104.193.255[.]99, hosted in Santa Clara, United States, serving linkednewsapi[.]top on port 8443.

Both servers returned identical responses: a bare 404 with the same server header, the same Vary: Cookie header, and a body of exactly 18 bytes. Both ran Microsoft-HTTPAPI/2.0 and were protected by CloudFlare Origin Certificates issued by the same certificate authority. The combination of an identical JARM fingerprint, a matching server stack, an identical response structure, and shared certificate characteristics leads us to assess with moderate confidence that linkednewsapi[.]top is operated by the same threat cluster. The domain had no prior public reporting or [VirusTotal](#) detections at the time of discovery. While none of these indicators alone are sufficient to establish common ownership, their combination provides moderate-confidence evidence of operational linkage.

Figure 27: C2 infrastructure overlap timeline

Here is an overlap timeline between the two C2 servers we identified. The operator set up sharingfile[.]cloud first, with its CloudFlare Origin Certificate going live in early December 2025. About eight weeks later, a second certificate was issued for linkednewsapi[.]top, which tells us they were already spinning up a second server by late January. Things picked up in March: on March 10, the SSL certificate for sharingfile[.]cloud showed up in historical records for the first time, and that same day lines up with the compile timestamp on the Havoc Demon DLL we extracted, which had the domain baked into its config. Three weeks after that, on March 31, the certificate on the same domain was rotated.

Attribution and confidence

Attribution in threat intelligence is never straightforward. Threat groups routinely borrow, share, and outright copy each other's tools and techniques, and a TTP match alone is rarely enough to pin an operation on a specific actor. With that in mind, we approached this campaign cautiously, looking at the full picture rather than any single artifact.

The use of vmtools.dll as a sideloaded DLL is worth calling out specifically because it has been seen before, most notably by APT29 in a campaign against European diplomats in early 2025, and by groups like [Mustang Panda](#) and PatchWork. The filename is a natural choice for anyone abusing VMware-signed binaries since vmtools.dll ships with VMware tools and appears on HijackLibs as a known sideloading target. We noted this overlap but do not treat it

as an attribution indicator, as independent selection of the same well-known sideloading target is entirely plausible.

The targeting profile, lure construction and operational characteristics observed across both campaigns are consistent with an espionage-focused actor operating in Southeast Asia. The campaigns were deliberately aimed at Cambodian government entities in the defense and public works sectors, indicating a clear intelligence collection objective. At the same time, the combination of a custom C++ loader, KaynLdr and Havoc Demon does not closely resemble any previously documented cluster we track, and the infrastructure shows no meaningful overlap with known threat groups.

As a result, we are not attributing this activity to any known actor at this time. **Acronis TRU tracks the activity as a previously unreported cluster, which we designate Khmer Shadow**, and will continue monitoring for additional activity that may provide stronger attribution opportunities.

Conclusion: Advanced evasion, limited operational security

Ultimately, this campaign reflects a cluster that prioritized stealth over operational security. While Khmer Shadow employed advanced evasion techniques, including NTDLL unhooking, Hell's Gate syscall resolution, and DLL sideloading, it repeatedly reused the same payloads and infrastructure across multiple targets. This inconsistency enabled infrastructure correlation and ultimately led to the identification of additional cluster-controlled assets through JARM-based pivoting.

Detection by Acronis

Acronis detects and blocks both the **NIGHTFORGE** loader and the **Havoc Demon payload** as high-severity threats.

Indicators of compromise

SHA256: Contact_Letter_To_Ms_Pech_ICB_Cambodia_On_Collaboration.pdf.exe
1852120a84a328edd1995e633dfd2009867898a8e3f0b385e2490cf21c77a994

CN_Contact_Work_Cambodia's_Ministry_of_Public_Works_and_Transport.pdf
b3e853eee14fb7948c6907888ee07139085ba9af4231c30e97ff6236b86ca024

Havoc Payloads 90bbfa9e7af176b85d110f4f1789cae6777fcb60813b047133c8f12caa344a17
15278c52f4e0d8b5bbfe288a5e826ab2ebeaedb7fb85572940cf1263e384761f

Paths used for persistence: %LOCALAPPDATA%\VMwareNamespace\

Scheduled task: Task name: VMwareNamespace

Trigger: Daily, starting 2010-01-01T00:00:00, repeat every PT10M for P1D

Logon type: TASK_LOGON_INTERACTIVE_TOKEN

Previous task name (cleaned up by loader): driver

C2 domains:

www[.]sharingfile[.]cloud

linkednewsapi[.]top

C2 origin IPs:

193[.]169[.]240[.]38 (SERVER.UA LLC, Kyiv, Ukraine)

104[.]193[.]255[.]99 (Hosting Solution Ltd., Santa Clara, U.S.)

© 2003–2026 Acronis International GmbH.