Sindoor Dropper: New Phishing Campaign

By Pierre-Henri Pezier

Published: 2026-03-30 · Archived: 2026-04-05 21:22:28 UTC

Our analysis uncovered a phishing campaign targeting organizations in India, leveraging spear-phishing techniques reminiscent of Operation Sindoor. What makes this activity stand out is the use of a Linux-focused infection method that relies on weaponized .desktop files. This technique has been linked to APT36 (aka

Transparent Tribe, Mythic Leopard, G0134)

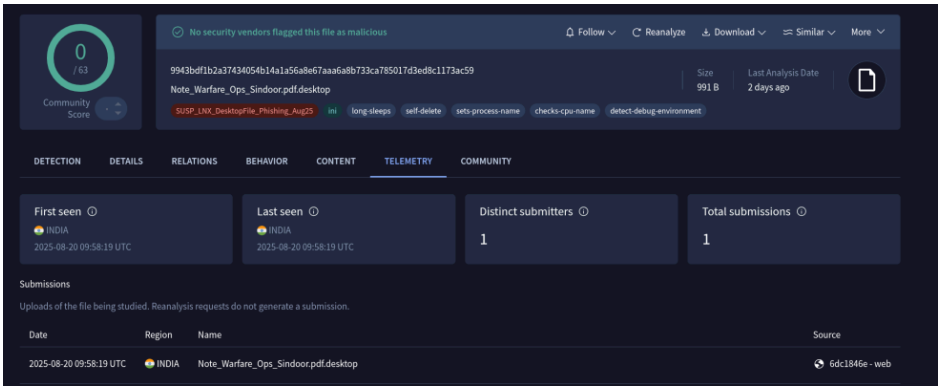
in the past, suggesting the same group may be behind the campaign while also adapting its methods.

When opened, these `.desktop` files trigger a heavily obfuscated execution chain built to evade both static and dynamic detection. The chain ultimately delivers a MeshAgent payload, which gives the attacker full remote access to the system. This access includes the ability to monitor activity, move laterally, and potentially exfiltrate data.

The campaign highlights an evolution in regional threat actor tradecraft, particularly in its targeting of Linux environments, which have historically received less attention from phishing operators. By combining localized spear-phishing lures with advanced obfuscation techniques, the adversaries increase their chances of bypassing defenses and gaining footholds in sensitive networks.

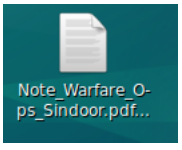
This report provides a technical breakdown of the payload delivery campaign, along with indicators of compromise (IOCs) and recommendations for mitigation.

At the time this article was written, the initial payload was not flagged by any antivirus vendors on VirusTotal.



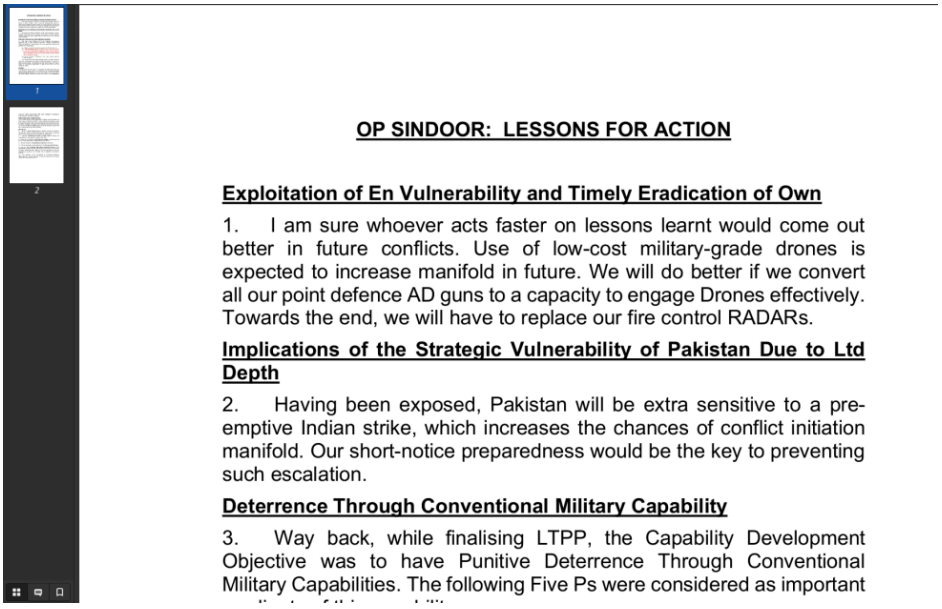
Smuggling

The malicious `.desktop` file is crafted to appear harmless by masquerading as a legitimate document. On the victim's desktop, it displays an icon resembling a PDF file (image below), luring the user into executing it.



Upon launch, the file opens a benign-looking decoy PDF (image below) to reinforce the illusion of legitimacy.

While the user is focused on the opened document, a background process silently runs the obfuscated routines that initiate the deployment chain, ultimately installing the MeshAgent payload. This approach blends social engineering with stealth, hiding malicious activity behind what appears to be a harmless file.



OP SINDOOR: LESSONS FOR ACTION

Exploitation of En Vulnerability and Timely Eradication of Own

1. I am sure whoever acts faster on lessons learnt would come out better in future conflicts. Use of low-cost military-grade drones is expected to increase manifold in future. We will do better if we convert all our point defence AD guns to a capacity to engage Drones effectively. Towards the end, we will have to replace our fire control RADARs.

Implications of the Strategic Vulnerability of Pakistan Due to Ltd Depth

2. Having been exposed, Pakistan will be extra sensitive to a pre-emptive Indian strike, which increases the chances of conflict initiation manifold. Our short-notice preparedness would be the key to preventing such escalation.

Deterrence Through Conventional Military Capability

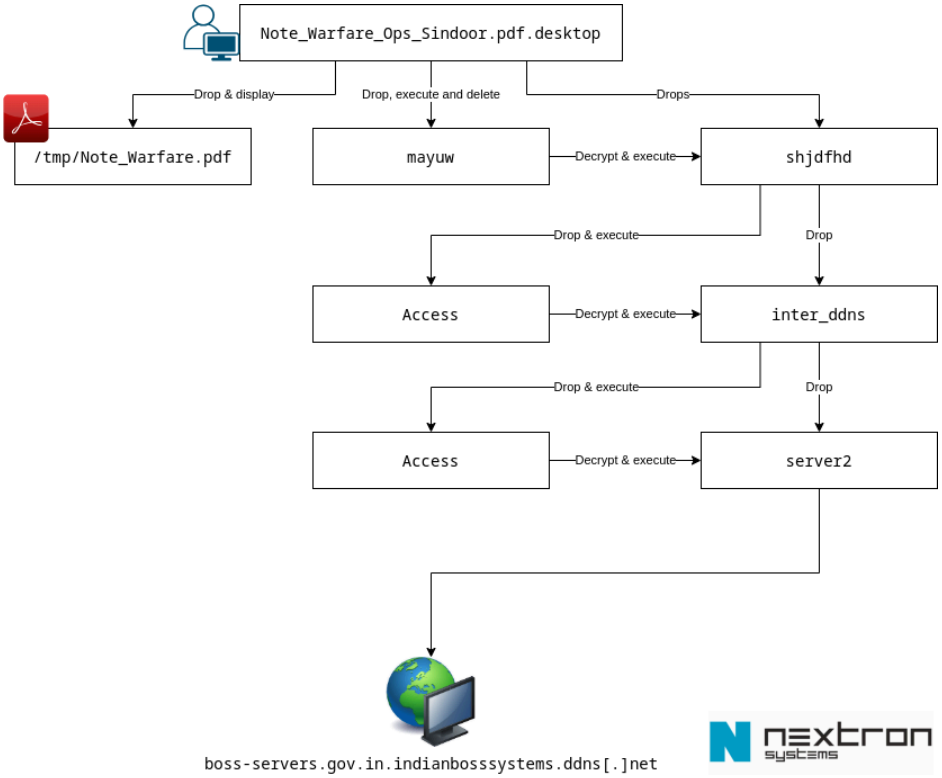
3. Way back, while finalising LTPP, the Capability Development Objective was to have Punitive Deterrence Through Conventional Military Capabilities. The following Five Ps were considered as important

By using a .desktop file, the attacker can place executable files on the victim's system without needing elevated privileges (chmod).

Technical Analysis

Infection Chain Overview

To better illustrate the complexity of this execution chain, the following schema outlines the infection process from the initial `.desktop` file launch to the final MeshAgent deployment.



Process Analysis

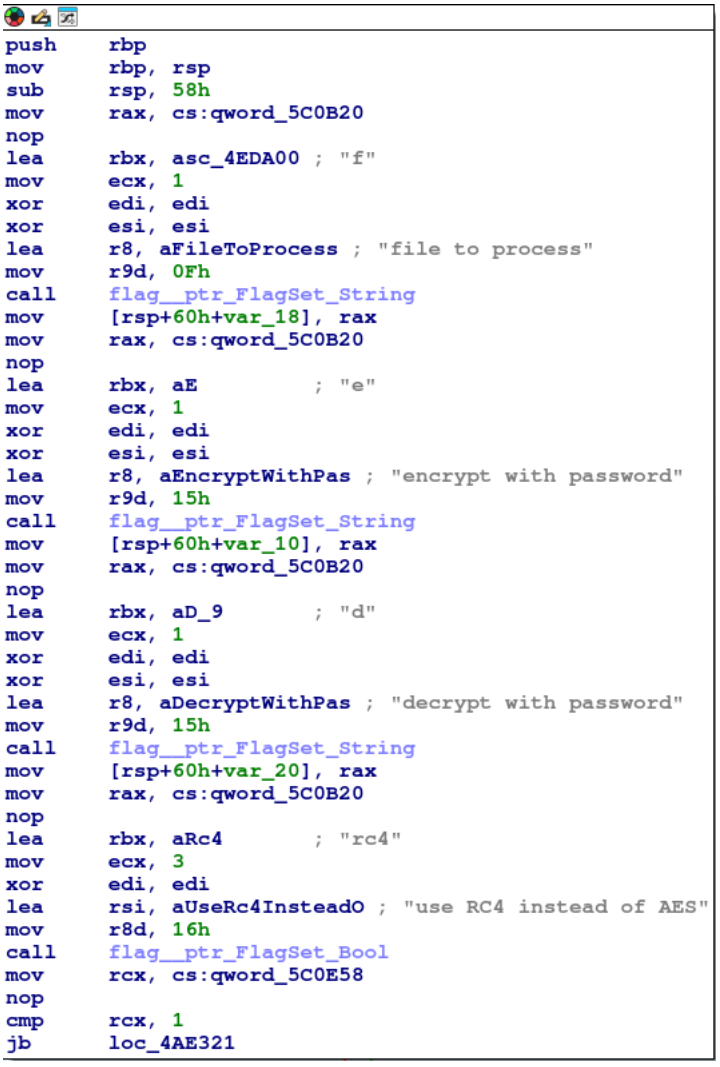
The original desktop file downloads the decoy document, a corrupted decryptor, and an encrypted downloader.

File name	sha256	URL
Note_Warfare_Ops_Sindoor.pdf.desktop	9943bdf1b2a37434054b14a1a56a8e67aaa6a8b733ca785017d3ed8c1173ac59	N/A
/tmp/Note_Warfare.pdf	ba5b485552ab775ce3116d9d5fa17f88452c1ae60118902e7f669fd6390eae97	https://docs.google[.]com/ export=download&id=1fz
mayuw	6879a2b730e391964afe4dbbc29667844ba0c29239be5503b7c86e59e7052443	https://drive.google[.]com export=download&id=1M
shjdfhd	6b1420193a0ff96e3a19e887683535ab6654b2773a1899c2ab113739730924a1	https://drive.google[.]com export=download&id=19(

The decryptor is a Go binary packed using UPX. To avoid detection, this binary has its ELF magic bytes stripped off, which appears to be necessary to evade scanning by the Google Docs platform. The magic bytes are restored on the fly by the `.desktop` file to make it executable again:

```
printf '\x7FELF' | dd of=mayuw bs=1 count=4 conv=notrunc
```

This decryptor is responsible for AES decryption and execution of the payload. There is also an option to use DES instead of AES:



```
push    rbp
mov     rbp, rsp
sub     rsp, 58h
mov     rax, cs:qword_5C0B20
nop
lea     rbx, asc_4EDA00 ; "f"
mov     ecx, 1
xor     edi, edi
xor     esi, esi
lea     r8, aFileToProcess ; "file to process"
mov     r9d, 0Fh
call    flag__ptr_FlagSet_String
mov     [rsp+60h+var_18], rax
mov     rax, cs:qword_5C0B20
nop
lea     rbx, aE         ; "e"
mov     ecx, 1
xor     edi, edi
xor     esi, esi
lea     r8, aEncryptWithPas ; "encrypt with password"
mov     r9d, 15h
call    flag__ptr_FlagSet_String
mov     [rsp+60h+var_10], rax
mov     rax, cs:qword_5C0B20
nop
lea     rbx, aD_9       ; "d"
mov     ecx, 1
xor     edi, edi
xor     esi, esi
lea     r8, aDecryptWithPas ; "decrypt with password"
mov     r9d, 15h
call    flag__ptr_FlagSet_String
mov     [rsp+60h+var_20], rax
mov     rax, cs:qword_5C0B20
nop
lea     rbx, aRc4       ; "rc4"
mov     ecx, 3
xor     edi, edi
lea     rsi, aUseRc4InsteadO ; "use RC4 instead of AES"
mov     r8d, 16h
call    flag__ptr_FlagSet_Bool
mov     rcx, cs:qword_5C0E58
nop
cmp     rcx, 1
jb      loc_4AE321
```

The decryption process is straightforward and can be achieved with the following command line:

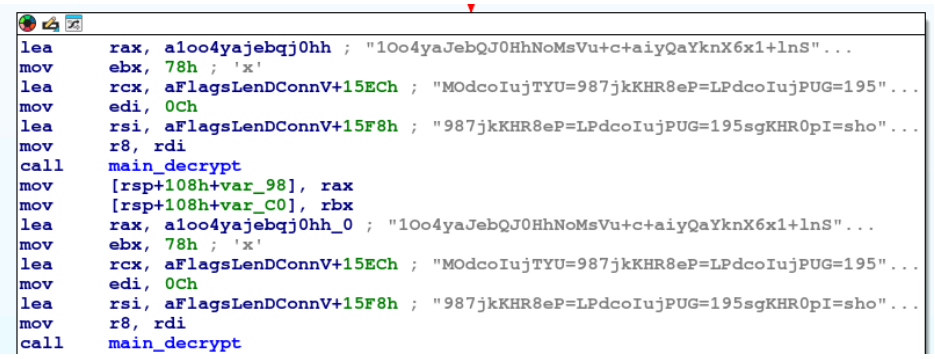
```
./mayuw -f shjdfhd -d 'NIC0fficialDB_Auth' && rm -r mayuw && ./shjdfhd
```

Once decrypted, the second-stage payload is a UPX-packed Go dropper that drops another decryptor along with another AES-encrypted payload (with the password `WOrkiNgtoDesksSS8123whyme?youseethis`).

The dropper contains basic anti-VM tricks:

1. The process checks that every value in `/sys/class/dmi/id/board_name`, `/sys/class/dmi/id/bios_vendor`, `/sys/class/dmi/id/board_vendor`, `/sys/class/dmi/id/sys_vendor`, and `/sys/class/dmi/id/product_name` does not match any of the following values: `VBOX_QEMU`, `QEMU`, `KVM_XEN`, `XEN`.
2. The following MAC address prefixes are blacklisted:
   - `00:05:69`
   - `00:0c:29`
   - `00:1c:14`
   - `00:50:56`
   - `08:00:27`
   - `00:15`
3. It attempts to execute the following processes to check if it is inside a VM:
   - `vboxservice`
   - `vboxtray`
   - `vmtoolsd`
   - `vmwaretray`
   - `xenservice`
4. The dropper checks that `/etc/os-release` does not contain any of the following values: `boss`, `vbox`, `qemu`, `KVM_`, `XEN_`.
5. The machine's uptime must be over 10 minutes to continue execution.

All strings are obfuscated using a combination of Base64 encoding and DES-CBC encryption:



```
lea     rax, a1oo4yajebqj0hh ; "1Oo4yaJebQJ0HhNoMsVu+c+aiyQaYknX6x1+lnS"...
mov     ebx, 78h ; 'x'
lea     rcx, aFlagsLenDConnV+15ECh ; "MOdcoIujTYU=987jkKHR8eP=LPdcoIujPUG=195"...
mov     edi, 0Ch
lea     rsi, aFlagsLenDConnV+15F8h ; "987jkKHR8eP=LPdcoIujPUG=195sgKHR0pI=sho"...
mov     r8, rdi
call    main_decrypt
mov     [rsp+108h+var_98], rax
mov     [rsp+108h+var_C0], rbx
lea     rax, a1oo4yajebqj0hh_0 ; "1Oo4yaJebQJ0HhNoMsVu+c+aiyQaYknX6x1+lnS"...
mov     ebx, 78h ; 'x'
lea     rcx, aFlagsLenDConnV+15ECh ; "MOdcoIujTYU=987jkKHR8eP=LPdcoIujPUG=195"...
mov     edi, 0Ch
lea     rsi, aFlagsLenDConnV+15F8h ; "987jkKHR8eP=LPdcoIujPUG=195sgKHR0pI=sho"...
mov     r8, rdi
call    main_decrypt
```

The process repeats with another download and decryption stage, this time using the password `WOrkiNgtoDesksSS8123`:

File name	sha256	URL
access	231957a5b5b834f88925a1922dba8b4238cf13b0e92c17851a83f40931f264c1	https://drive.google[.]com/uc?export=download&id=1g1AgwMnUTEV1HHmQkcH-eTww3w8et82m
inter_ddns	a6aa76cf3f25c768cc6ddcf32a86e5fcf4d8dd95298240c232942ce5e08709ec	https://drive.google[.]com/uc?export=download&id=1UFD10tcoPJZIBpF4hcz6orM1C

The decryption process concludes with the deployment of the final payload, a MeshAgent.

Mesh Agent

The final payload delivered by the Sindoor dropper is a MeshAgent binary, a legitimate remote administration tool that has been repurposed for malicious use. MeshAgent provides the attacker with full remote access to the compromised system, enabling a wide range of post-exploitation activities such as activity monitoring, lateral movement, data exfiltration, and persistent access.

File name	sha256	URL
server2	b46889ed27b69b94fb741b4d03be7c91986ac08269f9d7c37d1c13ea711f6389	https://drive.google[.]com/uc?export=download&id=1ygXFO_RLAfvjfBS1go5qWX93o7

File name	sha256	URL
server2	05b468fc24c93885cad40ff9ecb50594faa6c2c590e75c88a5e5f54a8b696ac8	N/A

Once executed, the MeshAgent connects to its command and control (C2) server at:

```
wss://boss-servers.gov.in.indianbosssystems.ddns[.]net:443/agent.ashx
```

All observed subdomains under `indianbosssystems.ddns[.]net` resolve to the IP address `54.144.107.42`. Analysis indicates that this command-and-control (C2) infrastructure is hosted on an Amazon Web Services (AWS) EC2 instance. According to registration records from validin.com, these subdomains were all registered on 2025-08-15, suggesting a coordinated setup of the infrastructure immediately prior to the campaign's activity.

This connection allows the attacker to issue commands, transfer files, and maintain persistence on the infected host. The use of a legitimate tool like MeshAgent complicates detection and response, as its functionality overlaps with legitimate remote administration.

IOCs

In the following section, we provide Indicators of Compromise (IOCs) and detection rules for the Sindoor dropper campaign. Security teams and incident responders can use them to help spot potential infections in their environments. The list includes file hashes, filenames, and YARA rules designed to detect the obfuscation methods and payload delivery techniques observed in this campaign.

Organizations are encouraged to monitor for these IOCs in their security solutions, SIEMs, and endpoint detection platforms. Regularly updating detection signatures and correlating these indicators with network and host activity can help to identify and contain infections at an early stage.

Sha-256 of the decrypted and unpacked samples

File name	sha256	Description
access	231957a5b5b834f88925a1922dba8b4238cf13b0e92c17851a83f40931f264c1	AES decryptor
mayuw	9a1adb50bb08f5a28160802c8f315749b15c9009f25aa6718c7752471db3bb4b	AES decryptor
shjdfhd	0f4ef1da435d5d64ccc21b4c2a6967b240c2928b297086878b3dcb3e9c87aa23	Stage2 downloader
inter_ddns	38b6b93a536cbab5c289fe542656d8817d7c1217ad75c7f367b15c65d96a21d4	Stage3 downloader
server2	05b468fc24c93885cad40ff9ecb50594faa6c2c590e75c88a5e5f54a8b696ac8	MeshAgent

Files

- `mayuw`
- `shjdfhd`
- `inter_ddns`
- `Note_Warfare_Ops_Sindoor.pdf.desktop`
- `/tmp/Note_Warfare.pdf`

YARA

```
rule SUSP_LNX_Sindoor_ELF_Obfuscation_Aug25 {
   meta:
      description = "Detects ELF obfuscation technique used by Sindoor dropper related to APT 36"
      author = "Pezier Pierre-Henri"
      date = "2025-08-29"
      score = 70
      reference = "Internal Research"
      hash = "6879a2b730e391964afe4dbbc29667844ba0c29239be5503b7c86e59e7052443"
   strings:
      $s1 = "UPX!"
   condition:
      filesize < 10MB
      and uint16(0) == 0
      and uint16(4) > 0
```

```
      and $s1 in (0xc0..0x100)
}

rule SUSP_LNX_Sindoor_DesktopFile_Aug25 {
   meta:
      description = "Detects ELF obfuscation technique used by Sindoor dropper related to APT 36"
      author = "Pezier Pierre-Henri"
      date = "2025-08-29"
      score = 70
      reference = "Internal Research"
      hash = "9943bdf1b2a37434054b14a1a56a8e67aaa6a8b733ca785017d3ed8c1173ac59"
   strings:
      $hdr = "[Desktop Entry]"
      $s1 = "printf '\\\\x7FELF' | dd of"
      $s2 = "Future_Note_Warfare_OpSindoor.pdf"
   condition:
      filesize < 100KB
      and $hdr
      and any of ($s*)
}

rule MAL_Sindoor_Decryptor_Aug25 {
   meta:
      description = "Detects AES decryptor used by Sindoor dropper related to APT 36"
      author = "Pezier Pierre-Henri"
      date = "2025-08-29"
      score = 80
      reference = "Internal Research"
      hash = "9a1adb50bb08f5a28160802c8f315749b15c9009f25aa6718c7752471db3bb4b"
   strings:
      $s1 = "Go build"
      $s2 = "main.rc4EncryptDecrypt"
      $s3 = "main.processFile"
      $s4 = "main.deriveKeyAES"
      $s5 = "use RC4 instead of AES"
   condition:
      filesize < 100MB
      and (
         uint16(0) == 0x5a4d // Windows
         or uint32be(0) == 0x7f454c46  // Linux
         or (uint32be(0) == 0xcafebabe and uint32be(4) < 0x20)  // Universal mach-O App with dont-match-java-c
         or uint32(0) == 0xfeedface  // 32-bit mach-O
         or uint32(0) == 0xfeedfacf  // 64-bit mach-O
      )
      and all of them
}

rule MAL_Sindoor_Downloader_Aug25 {
   meta:
      description = "Detects Sindoor downloader related to APT 36"
      author = "Pezier Pierre-Henri"
      date = "2025-08-29"
      score = 80
      reference = "Internal Research"
      hash = "38b6b93a536cbab5c289fe542656d8817d7c1217ad75c7f367b15c65d96a21d4"
   strings:
      $s1 = "Go build"
      $s2 = "main.downloadFile.deferwrap"
      $s3 = "main.decrypt"
      $s4 = "main.HiddenHome"
      $s5 = "main.RealCheck"
   condition:
      filesize < 100MB
      and (
         uint16(0) == 0x5a4d // Windows
         or uint32be(0) == 0x7f454c46  // Linux
         or (uint32be(0) == 0xcafebabe and uint32be(4) < 0x20)  // Universal mach-O App with dont-match-java-c
         or uint32(0) == 0xfeedface  // 32-bit mach-O
         or uint32(0) == 0xfeedfacf  // 64-bit mach-O
      )
```

```
      and all of them
}
```

Appendices

Module decryption:

```
import zlib
import magic
import tempfile
import re
import base64
import pathlib
import sys

try:
   data = pathlib.Path(sys.argv[1]).read_bytes()
except IndexError:
   print("Usage: python decrypt_mesh_modules.py <path_to_encrypted_file>")
   sys.exit(1)


def decode_module_data(encoded_str: str) -> str:
   data = base64.b64decode(encoded_str)
   while magic.from_buffer(data) == "zlib compressed data":
      data = zlib.decompress(data)
   return data.decode()

with tempfile.TemporaryDirectory() as _tempdir:

   #for module in re.findall(rb"addCompressedModule.*?'(.*?').*Buffer.from\('(.*?)'\)", data):
   for module_name, module_data in re.findall(rb"addCompressedModule.*?'(.*?').*?'(.*?')", data):
      module_name = module_name.decode()
      print("Decoding module:", module_name)
      module_data = decode_module_data(module_data.decode())

      pathlib.Path(_tempdir, module_name + '.js').write_text(module_data)
   input(f"Data saved to {_tempdir}. Press a key to delete...")
```

About the author:

Pierre-Henri Pezier

Pierre-Henri Pezier is an IT Security Engineer and Threat Researcher with over a decade of experience in offensive security, reverse engineering, malware analysis and secure software development. He began reverse-engineering software in the early 2010s, a passion that expanded into analyzing advanced threats, developing decryptors, and writing detection rules. With a background in both offensive and defensive security, Pierre-Henri has worked on malware classification engines, sandbox environments, and EDR evasion techniques.

Source: https://www.nextron-systems.com/2025/08/29/sindoor-dropper-new-phishing-campaign/