

SloppyLemming Deploys BurrowShell and Rust-Based RAT to Target Pakistan and Bangladesh

arcticwolf.com/resources/blog/sloppylemming-deploys-burrowshell-and-rust-based-rat-to-target-pakistan-and-bangladesh

Arctic Wolf Labs

March 2, 2026



Summary

Between January 2025 and January 2026, Arctic Wolf tracked an extensive cyber espionage campaign that we assess was conducted by SloppyLemming (also known as Outrider Tiger and Fishing Elephant), an India-nexus threat actor, targeting government entities and critical infrastructure operators in Pakistan and Bangladesh. To our knowledge, this report documents previously unreported tooling, infrastructure, and targeted organizations that significantly expand the prior public understanding of this actor's capabilities and operational scope.

The campaign employed two distinct attack chains. The primary vector delivered PDF lure documents that redirected victims to ClickOnce application manifests, which deployed a DLL sideloading package consisting of a legitimate Microsoft .NET runtime executable (NGenTask.exe) and a malicious loader (mscorsvc.dll). This loader decrypted and executed a custom x64 shellcode implant that Arctic Wolf has named **BurrowShell**.

BurrowShell is a full-featured backdoor providing the threat actor with file system manipulation, screenshot capture capabilities, remote shell execution, and SOCKS proxy capabilities for network tunneling. The implant masquerades its command-and-control (C2) traffic as Windows Update service communications and employs RC4 encryption with a 32-character key for payload protection.

The secondary attack chain leverages macro-enabled Excel documents to deliver a Rust-based keylogger with extended reconnaissance capabilities, including port scanning and network enumeration. The use of the Rust programming language represents a notable evolution in SloppyLemming's tooling, as prior reporting documented the actor using only traditional compiled languages and borrowed adversary simulation frameworks such as Cobalt Strike, Havoc, and the custom NekroWire RAT.

Infrastructure analysis identified 112 Cloudflare Workers domains registered between January 2025 and January 2026, representing an eight-fold expansion from the 13 Workers domains [documented](#) in Cloudflare's September 2024 public reporting. Three of these Workers exhibited open directory misconfigurations that exposed staged malware, including Havoc framework loaders with distinct RC4 encryption keys. Peak infrastructure registration occurred in July 2025 with 42 new domains, suggesting intensified operational tempo during this period.

Arctic Wolf assesses with moderate confidence that this activity is attributable to SloppyLemming, based on continued exploitation of Cloudflare Workers infrastructure with government-themed typo-squatting patterns, deployment of the Havoc C2 framework previously linked to this actor, DLL sideloading techniques consistent with documented tradecraft, and victimology focused on South Asian government and critical infrastructure entities matching established targeting priorities.

In particular, the targeting of Pakistani nuclear regulatory bodies, defense logistics organizations, and telecommunications infrastructure – alongside Bangladeshi energy utilities and financial institutions – aligns with intelligence collection priorities consistent with regional strategic competition in South Asia.

Key Findings

- **Campaign continuation:** This activity represents a direct continuation of SloppyLemming operations [documented by Cloudflare's CloudForce One](#) in September 2024, with expanded tooling and infrastructure.
- **Multi-stage execution chains:** The threat actor employs ClickOnce application manifests and macro-laden Office documents to deliver payloads through DLL search order hijacking, abusing legitimate Microsoft binaries including NGenTask.exe and phoneactivate.exe.
- **Dual payload deployment:** Two custom implants were identified – an in-memory x64 shellcode (BurrowShell) implant providing C2 communication with SOCKS proxy capabilities. BurrowShell supports fifteen distinct commands, enabling operators to manipulate the file system, capture screenshots, execute shell commands, and establish SOCKS proxy tunnels for lateral network access. The second implant is a Rust-based keylogger with extended information-stealing features, including remote command execution, file operations, and network reconnaissance.
- **Exploitation of Cloudflare Workers:** The campaign leverages 112 unique Cloudflare Workers domains mimicking Pakistani and Bangladeshi government entities for both payload delivery and C2 communication.
- **Operational security (OPSEC) failures:** Multiple C2 domains were discovered operating as open directories, exposing malware components including Havoc C2 framework artifacts. This gave Arctic Wolf researchers an insight into its infrastructure.
- **Targeted sectors:** Government agencies, defense organizations (Pakistan Navy, National Logistics Corp), energy utilities (DESCO, PGCB), telecommunications providers (SCO, PTCL), and nuclear regulatory bodies (PNRA) across Pakistan and Bangladesh.

How Arctic Wolf Protects its Customers

Arctic Wolf is committed to ending cyber risk, and when active campaigns are identified, we move quickly to protect our customers. Arctic Wolf Labs has leveraged threat intelligence around SloppyLemming activity to implement new detections in the [Arctic Wolf® Aurora™ Platform](#) to protect customers.

As we track this campaign and discover new information, we will continue to enhance our detections to account for additional indicators of compromise (IOCs) and techniques leveraged by the threat group behind this malicious activity.

SloppyLemming Threat Actor Profile

[SloppyLemming](#) (also tracked as Outrider Tiger and Fishing Elephant) is a suspected India-aligned cyber-espionage group that has been active since at least 2021. The group's operations primarily target entities within government, telecommunications, defense, and critical infrastructure sectors across South and East Asia, with particular focus on Pakistan and neighboring countries.

Previously documented campaigns have seen SloppyLemming leveraging adversary emulation frameworks such as Cobalt Strike Beacon and Havoc, alongside custom-developed tooling.

Our attribution of this new campaign to SloppyLemming is based on overlapping tactics, techniques, and procedures (TTPs) with previously documented activity, consistent victimology patterns, and infrastructure characteristics that align with the group's established operational tradecraft.

Campaign Overview and Attack Methodology

The campaign Arctic Wolf observed employs two primary attack vectors, both initiated through [spear-phishing](#). The first vector uses PDF documents containing embedded malicious URLs that redirect victims to ClickOnce application manifest files, which orchestrate the download and execution of a multi-component payload chain. The second vector uses macro-laden Excel spreadsheets that directly download and execute malicious binaries.

Both attack chains ultimately achieve code execution through DLL search order hijacking, whereby legitimate, digitally signed Microsoft executables are placed alongside malicious DLLs that the executables automatically load during initialization. This technique allows the threat actor to execute malicious code within the context of trusted processes, potentially bypassing security controls that rely on process reputation.

The campaign infrastructure makes extensive use of Cloudflare Workers subdomains, with domain naming conventions carefully crafted to impersonate legitimate Pakistani and Bangladeshi government entities. Analysis of the domain registration timeline reveals sustained operational activity, with 112 unique domains identified between January 2025 and January 2026, peaking in July 2025 with 42 new domains registered.

Technical Analysis

Attack Vector 1: PDF Lure with ClickOnce Execution Chain

Stage 1: Initial Access via Malicious PDF

The first attack vector begins with a PDF document designed to deceive victims through [social engineering](#). When opened, the document displays a blurred-out document with a prominent “Download file” button, along with text claiming “PDF reader is disabled” and instructing the user to enable the reader through browser settings.

Field	Value
SHA-256	8faeea306a331d86ce1acb92c8028b4322efbd11a971379ba81a6b769ff5ac4b
File Type	PDF
Size	123,495 bytes
Creator	pdf-lib (https://github.com/Hopding/pdf-lib)
First Seen	2026-01-07 10:55:31 UTC

The document contains an embedded malicious URL – `hxtps://webmail-pnra[.]gov-pk[.]workers[.]dev/ftp[.]pnra.org.application` – designed to mimic the Pakistan Nuclear Regulatory Authority (PNRA), which operates at the legitimate site `pnra.org`. When the victim clicks the “Download file” button, they are redirected to the malicious URL, which serves a ClickOnce application manifest file.

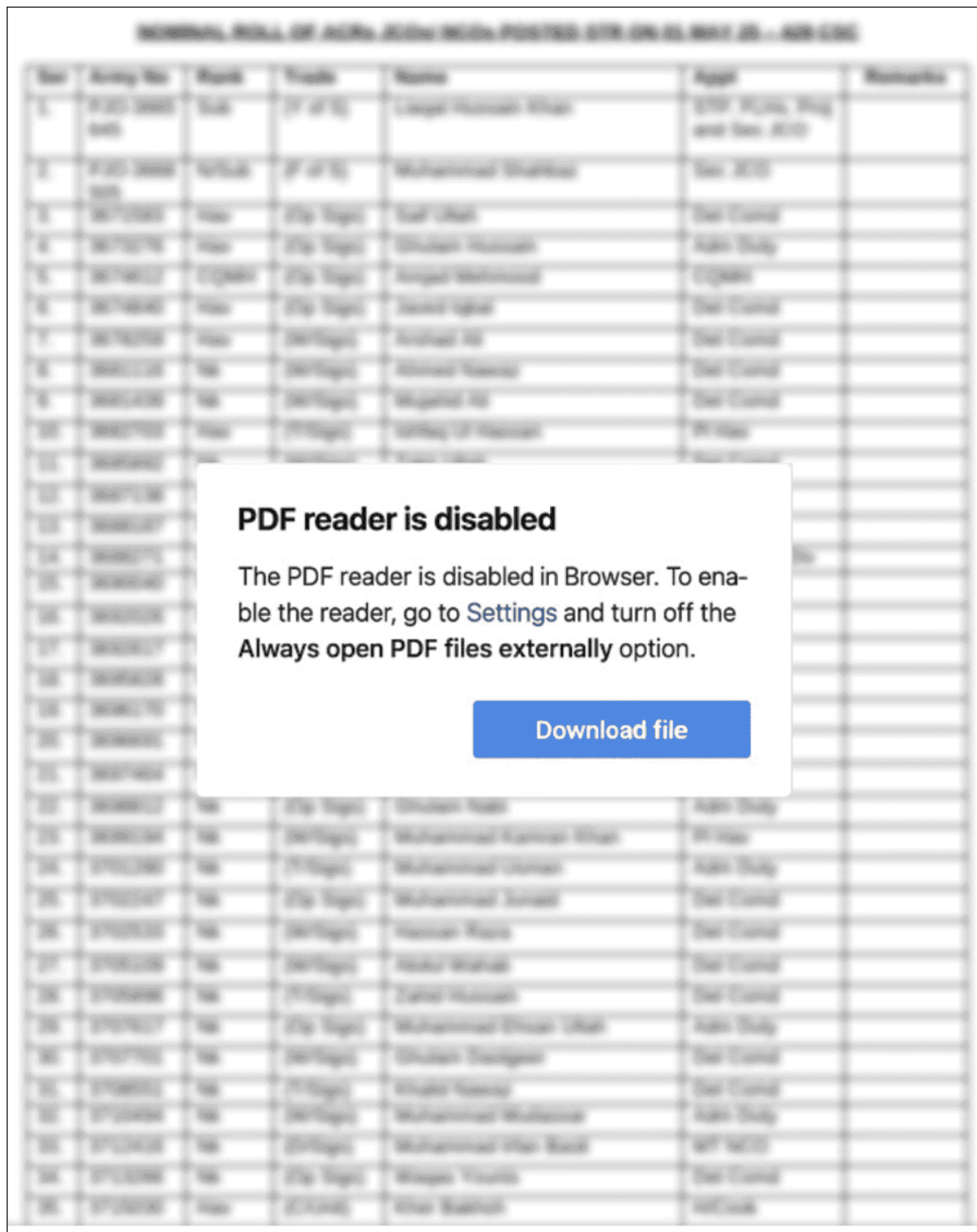


Figure 1: PDF lure displaying blurred document with social engineering message “PDF reader is disabled”.

Stage 2: ClickOnce Application Manifest Chain

The initial URL serves a ClickOnce application manifest (.application file) that references a secondary manifest file (ftp[.]pnra.org.exe[.]manifest). This secondary manifest specifies three Windows Portable Executable (PE) files to be downloaded:

1. **OneDrive.exe**: A renamed copy of the legitimate Microsoft NGenTask.exe
2. **Mscorsvc.dll**: A malicious DLL loader
3. **System32.dll**: An RC4-encrypted shellcode blob

The manifest files contain an embedded hostname reference – CN=DESKTOP-7LTGBEK\pakis – which proved to be a useful pivot point during the investigation.

```
<dependentAssembly dependencyType="install" codebase="ftp.pnra.org.exe.manifest" size="7709">
  <assemblyIdentity name="ftp.pnra.org.exe" version="1.2.0.0" publicKeyToken="ca9923741b74f6d5" language="neutral"
processorArchitecture="amd64" type="win32" />
  <hash>
    <dsig:Transforms>
      <dsig:Transform Algorithm="urn:schemas-microsoft-com:HashTransforms.Identity" />
    </dsig:Transforms>
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256" />
    <dsig:DigestValue>xgPnoQGPezoWhAS88vcJlQx0KeB2bHiCNke6qtrzf8c=</dsig:DigestValue>
  </hash>
</dependentAssembly>
</dependency>
<publisherIdentity name="CN=DESKTOP-7LTGBEK\pakis" issuerKeyHash="c451af0d4c4185fe571a4ec9b81f8e58527c824a" /><Signature
Id="StrongNameSignature" xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /><SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256" /><Reference
```

Figure 2: First XML manifest file showing deployment provider and dependent assembly configuration.

All execution chain components are saved to a ClickOnce application directory following the naming convention: C:\Users\
<Username>\AppData\Local\Apps\2.0\<12randomchars>\<12randomchars>\ftp...app_ca9923741b74f6d5_0001.0002_<16random-hex>

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1 assembly.adaptive.xsd" manifestVersion="1.0" xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns="urn:schemas-microsoft-com:asm.v2" xmlns:asmv2="urn:schemas-microsoft-com:asm.v2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:co.v1="urn:schemas-microsoft-com:clickonce.v1" xmlns:asmv3="urn:schemas-microsoft-com:asm.v3"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns:co.v2="urn:schemas-microsoft-com:clickonce.v2">
  <asmv1:assemblyIdentity name="ftp.pnra.org.exe" version="1.2.0.0" publicKeyToken="ca9923741b74f6d5" language="neutral"
processorArchitecture="amd64" type="win32" />
  <application />
  <entryPoint>
    <assemblyIdentity name="NGenTask" version="4.0.0.0" publicKeyToken="B77A5C561934E089" language="neutral" processorArchitecture="amd64" />
    <commandLine file="OneDrive.exe" parameters="" />
  </entryPoint>
```

Figure 3: Stage 2 – Second XML manifest file showing entry point configuration with OneDrive.exe.

```

<dependentAssembly dependencyType="install" allowDelayedBinding="true" codebase="OneDrive.exe" size="79848">
  <assemblyIdentity name="NGenTask" version="4.0.0.0" publicKeyToken="B77A5C561934E089" language="neutral" processorArchitecture="amd64" />
  <hash>
    <dsig:Transforms>
      <dsig:Transform Algorithm="urn:schemas-microsoft-com:HashTransforms.Identity" />
    </dsig:Transforms>
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256" />
    <dsig:DigestValue>n9EzsRq8u+0zz0pxvUd06PNeQs1jf7dj9asqj7ubYmE=</dsig:DigestValue>
  </hash>
</dependentAssembly>
</dependency>
<file name="mscorlib.dll" size="8704">
  <hash>
    <dsig:Transforms>
      <dsig:Transform Algorithm="urn:schemas-microsoft-com:HashTransforms.Identity" />
    </dsig:Transforms>
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256" />
    <dsig:DigestValue>gdGmLAByTB37wFp5rErpIcRZNQoqSpM2bAhC+txAsBE=</dsig:DigestValue>
  </hash>
</file>
<file name="system32.dll" size="81008">
  <hash>
    <dsig:Transforms>
      <dsig:Transform Algorithm="urn:schemas-microsoft-com:HashTransforms.Identity" />
    </dsig:Transforms>
    <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256" />
    <dsig:DigestValue>Pb9k2jdhasvha8a9BqMg/tQWxMjsN6BPgRoy0Jrz16w=</dsig:DigestValue>
  </hash>
</file>

```

Figure 4: Final payload components showing all downloaded files, including manifest files and executables.

Stage 3: Payload Execution via DLL Search Order Hijacking

OneDrive.exe (NGenTask.exe)

The first component, delivered as OneDrive.exe, is a legitimate Microsoft .NET Framework binary (NGenTask.exe) responsible for optimizing .NET application performance.

Field	Value
Name	OneDrive.exe
SHA-256	9fd133b11abcbbed33ccea71bd4743e8f35e42cd637fb763f5ab2a8fbb9b6261
File Type	Win32 executable
Size	79,848 bytes
Certificate Serial Number	33000002511476ccc585836e0b00000000251

The binary is signed with a legitimate but expired Microsoft digital certificate (valid from September 2, 2021, to September 1, 2022). While the file itself is benign, it is susceptible to [DLL search order hijacking](#) as it attempts to load mscorsvc.dll from its current directory before checking system directories.

```

Compile date:      2022-04-08 21:27:51
Certificate:
  Issuer:          Microsoft Code Signing PCA 2011 (Organization=Microsoft Corporation / Unit=??? / Country=US)
  Subject:         Microsoft Corporation
  Org Details:     Microsoft Corporation / Unit=??? / State=Washington / Locality=Redmond / Country=US / Email=???
  Validity:        from 2021-09-02 to 2022-09-01
  SerialNumber:    33000002511476ccc585836e0b00000000251
  HashAlgorithm:   SHA256
  CryptAlgorithm:  RSA
VersionInfo:
  CompanyName:     Microsoft Corporation
  FileDescription: Microsoft .NET Framework optimization service
  FileVersion:     4.8.9032.0 built by: NET481REL1
  InternalName:    NGenTask.exe
  LegalCopyright:  © Microsoft Corporation. All rights reserved.
  OriginalFilename: NGenTask.exe
  ProductName:     Microsoft® .NET Framework
  ProductVersion:  4.8.9032.0
  Comments:        Flavor=Retail
  PrivateBuild:    DDBLD344B
Debug:
  Date.Debug.Codeview: 2022-04-08 21:27:51
  Path:              NGenTask.pdb
DotNet:
  Module name:      NGenTask.exe

```

Figure 5: Expired digital signature showing Microsoft Corporation certificate details.

This technique was [documented](#) in 2024 describing how legitimate executables within the trusted Windows Side-by-Side (WinSxS) folder can be leveraged for DLL search order hijacking. The approach allows attackers to execute malicious code without requiring elevated privileges and to do so from arbitrary filesystem locations. It remains effective on Windows 10 and 11.

mscorsvc.dll (Malicious loader)

The second component masquerades as the legitimate Microsoft Common Object Runtime Service .NET Framework component, but is, in fact, a malicious DLL loader.

Field	Value
Name	mscorsvc.dll
SHA-256	81d1a62c00724c1dfbc05a79ac4ae921c459350a2a4a93366c0842fadc40b011
File Type	Win32 DLL
Size	8,704 bytes

Upon execution of NGenTask.exe, this DLL is loaded into the process address space, whereupon it calls its sole export function CorInitSvcLogger. The loader performs several operational checks before proceeding:

1. Path validation: The loader obtains the current process path and compares it against an allowlist:

- C:\PROGRAMDATA
- C:\PROGRAM FILES
- C:\PROGRAM FILES (X86)
- \APPPDATA\
- C:\WINDOWS

If the process is not executing from one of these paths, execution terminates. This check ensures the malware only runs when deployed in expected locations.

2. Persistence establishment: The loader adds OneDrive.exe to the Windows startup folder by creating a registry entry under Software\Microsoft\Windows\CurrentVersion\Run, ensuring the execution chain survives system reboots.

```

80 }
81 while ( *v10 );
82 Data[(unsigned int)(v7 + 13)] = 0;
83 if ( !RegOpenKeyExA((HKEY)0x80000001, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, 2u,
84 {
85     LODWORD(v14) = 0;
86     if ( Data[0] )
87     {
88         do
89             v14 = (unsigned int)(v14 + 1);
90         while ( Data[v14] );
91     }
92     v15 = RegSetValueExA(hKey, "OneDrive", 0, 1u, Data, v14 + 1);
93     RegCloseKey(hKey);
94     LOBYTE(v2) = v15 == 0;
95     result = v2;
96 }
97 else
98 {

```

Figure 6: Persistence mechanism code showing registry key manipulation.

3. Payload decryption: The loader locates system32.dll in the same directory as the parent process, reads the encrypted blob, and decrypts it using RC4 with a hardcoded 32-character key.

Type	Value
RC4 Decryption Key	boikztaigkuneapfvpesuabfmpxgwnad

```

MultiByteToWideChar(0, 0, MultiByteStr, -1, WideCharStr, 260);
v28 = 0;
running_path = CreateFileW(WideCharStr, 0x80000000, 0, 0LL, 3u, 0x80u, 0LL); // Get handle to enc payload system32.dll
v14 = running_path;
if ( running_path != -1LL )
{
    FileSize_0 = GetFileSize_0(running_path, 0LL);
    v16 = FileSize_0;
    if ( FileSize_0 != -1 )
    {
        alloc_buff = VirtualAlloc_01(0LL, FileSize_0, 0x3000u, 4u);
        ptr_encPayload = alloc_buff;
        if ( alloc_buff )
        {
            if ( ReadFile_0(v14, alloc_buff, v16, &v28, 0LL) && v28 == v16 )
            {
                CloseHandle_0(v14);
                v19 = 0LL;
                byte_1800043C0 = byte_180004088 ^ byte_180004070;
                byte_1800043C1 = 0;
                *SystemFunction033 = _mm_xor_ps(_mm_loadu_si128(&xmmword_180004060), _mm_loadu_si128(&xmmword_180004078));
                do
                {
                    advapi32_dll[v19] = byte_180004020[v19 + 32] ^ byte_180004020[v19 + 48];
                    ++v19;
                }
                while ( v19 < 12 );
                byte_18000438C = 0;
                MultiByteToWideChar(0, 0, advapi32_dll, -1, &advapi32_dll_, 13);
                advapi32_base = LoadLibraryW(&advapi32_dll_);
                SystemFunction033_RC4 = GetProcAddress(advapi32_base, SystemFunction033);
                if ( SystemFunction033_RC4 )
                {
                    strcpy(rc4_key, "boikztaigkuneapfvpsuabfmpxgwnad");
                    ::rc4_key = rc4_key;
                    key_len = 32;
                    max_key_len = 32;
                    data_buff = ptr_encPayload;
                    data_length = v16;
                    max_data_length = v16;
                    (SystemFunction033_RC4)(&data_length, &key_len);
                    v30 = 0;
                    if ( VirtualProtect_01(ptr_encPayload, v16, 0x20u, &v30) )
                    {
                        v22 = GetTickCount_0() & 3;
                        if ( !v22 || (v23 = v22 - 1) == 0 || v23 - 1 <= 1 )
                            ptr_encPayload();
                    }
                }
                LODWORD(running_path) = VirtualFree_0(ptr_encPayload, 0LL, 0x8000u);
                return running_path;
            }
            VirtualFree_0(ptr_encPayload, 0LL, 0x8000u);
        }
        LODWORD(running_path) = CloseHandle_0(v14);
    }
}
return running_path;

```

RC4 Key

Figure 7: RC4 decryption code showing key initialization and SystemFunction033 call.

system32.dll (Encrypted Shellcode Payload)

The third component is an RC4-encrypted blob containing the final payload.

Field	Value
Name	system32.dll
SHA-256	3dbf64da37616acbe16bc6bd06a320fed416c4c8ec37a04f811a32389af3d46c
File Type	Encrypted blob
Size	81,008 bytes

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	8a	25	8c	9f	95	5a	ab	cb	22	15	03	c8	60	a5	aa	ad	. % . . . Z . . . " . . . ` . . .
00000010	c2	c0	d2	f6	4c	35	a7	c2	14	ba	88	5d	f7	de	3e	37	 L 5] . . . > 7
00000020	d3	b7	2e	24	09	27	11	82	31	ae	24	b3	25	42	7b	6c	. . . \$. ' . . . 1 . \$. % B { l
00000030	20	81	23	71	c4	35	98	8f	c6	8b	de	c3	9b	7a	d4	d2	. # q . 5 z . .
00000040	f4	87	24	42	e2	71	58	23	9f	cb	8d	04	a5	11	1b	d9	. . \$ B . q X #
00000050	48	c7	71	af	ea	a7	24	5b	9f	68	0f	23	82	c2	e1	57	H . q . . . \$ [. h . # . . . W
00000060	c4	40	a3	53	aa	55	3e	fb	ff	89	f2	94	aa	ee	02	8e	. @ . S . U >
00000070	fb	17	c5	05	96	af	16	bd	45	b0	81	08	06	13	8e	b8	 E
00000080	d3	5b	0a	9a	b4	73	d5	a5	51	4d	35	50	29	3b	77	0f	. [. . . s . . Q M 5 P) ; w .
00000090	eb	3e	8e	ac	fc	e7	24	ee	6c	26	ab	36	ca	36	9f	13	. > \$. l & . 6 . 6 . .
000000A0	2a	f4	fb	8b	79	8b	2c	60	ba	42	9e	51	49	0e	1b	41	* . . . y . , ` . B . Q I . . A
000000B0	71	af	3c	8f	b0	97	f1	72	e3	24	15	88	ba	fb	05	83	q . < r . \$
000000C0	f2	54	17	81	18	93	26	0e	56	96	01	5b	6f	17	93	6c	. T & . V . . [o . . l
000000D0	93	ef	23	51	f0	e2	07	41	ea	ce	72	01	31	c3	9c	cc	. . # Q A . . r . 1 . . .
000000E0	56	f1	0e	9d	d5	49	5f	23	61	a2	10	e5	3b	7e	f1	27	V I _ # a . . . ; - . '
000000F0	55	6d	b3	cd	c1	4c	7f	6e	e4	6f	53	c1	c2	b4	4b	a7	U m . . . L n . o S . . . K .
00000100	53	96	1b	b8	4b	92	a9	d3	23	0f	19	14	b6	f6	63	9e	S . . . K . . . # c .
00000110	63	88	fa	30	8a	98	07	98	f4	91	f6	c6	b7	e6	6c	9a	c . . 0 l .
00000120	83	50	03	72	4c	07	8c	5b	40	df	2d	b6	60	8d	65	fe	. P . r L . . . [@ . . . ` . e .
00000130	00	28	a8	9a	3f	0b	4b	19	90	a2	0e	a4	36	62	a7	12	. (. . ? . K 6 b . .
00000140	2b	a7	a9	65	3c	31	e9	98	03	ff	9e	4a	7e	9f	2f	a4	+ . . e < 1 J ~ . / .
00000150	a7	85	d8	58	d1	be	e3	1f	18	b1	b3	e4	2c	9d	66	03	. . . X , . f .
00000160	33	cb	2f	f7	f7	d7	76	7f	d5	b1	11	67	22	6c	ea	b5	3 . / . . . v . . . g " l . .
00000170	0d	0b	2a	72	c6	6d	85	a5	53	79	13	b5	5d	ec	9d	01	. . * r . m . . S y . .] . . .
00000180	2a	f1	0d	97	07	a8	6d	a6	f8	fa	b6	3f	1f	13	19	77	* m ? . . . w
00000190	c1	a1	b7	61	52	dc	23	57	dc	f9	0b	97	b0	26	41	90	. . . a R . # W & A .
000001A0	e7	1c	b3	d5	bf	79	6a	a0	db	8b	c4	4c	a2	a1	47	b5	 y j L . . G .
000001B0	b3	eb	c7	1b	ad	ef	95	bf	14	ef	14	64	86	2b	60	95	 d . + ` .
000001C0	e9	7c	65	9b	6d	39	b0	59	bc	87	c2	c7	35	3f	f0	34	. e . m 9 . Y 5 ? . 4
000001D0	4b	4d	e2	e2	ad	bb	cf	2d	1c	17	42	00	ab	33	fe	db	K M B . . 3 . .
000001E0	11	f6	6b	e6	76	11	ec	58	51	8e	d1	c3	93	94	46	86	. . k . v . . X Q F .
000001F0	4d	81	ac	0f	f4	32	3f	49	a6	fa	91	0d	77	94	0b	c9	M 2 ? I w . .
00000200	90	ff	50	56	1e	37	e7	01	0d	a8	3f	4d	96	e8	b9	95	. . P V . 7 ? M . . .
00000210	47	bd	93	c3	61	8e	63	a8	7b	8e	13	98	ca	4c	09	72	G . . . a . c . { L . r
00000220	ed	b0	72	b8	14	f5	1e	7b	2f	99	7d	23	4b	66	6b	7f	. . r { / . } # K f k
00000230	f9	94	e2	1b	b9	f6	0b	ba	cf	a4	2c	82	1c	3e	2d	a0	 , . . > . .
00000240	96	15	10	0f	e5	e2	f1	ce	7d	7b	7f	88	9b	53	a8	32	 } { . . S . 2
00000250	aa	7a	a7	2b	1a	cd	f6	8d	ab	3c	04	ae	78	df	c2	05	. z . + ... < . . x . .
00000260	0a	dd	32	20	ab	ea	33	5b	56	67	d0	aa	98	99	90	2f	. . 2 . . 3 [V g /

Figure 8: RC4 encrypted blob showing raw hexadecimal content.

Upon decryption, the blob yields x64 shellcode to which execution flow is transferred.

In-Memory BurrowShell Implant Analysis

The decrypted x64 BurrowShell functions as an in-memory shellcode implant designed to provide the operator with remote command execution and network pivoting capabilities. The implant establishes C2 communication and, upon instruction, can activate SOCKS-based tunneling that allows the compromised host to function as a proxy for relaying network traffic.

Dynamic API Resolution

The malware dynamically resolves required Windows API functions at runtime by hashing exported function names and comparing them against hardcoded values. This technique hinders static analysis and evades signature-based detection. The implant resolves APIs from the following Dynamic Link Libraries:

- KERNEL32.dll

- WINHTTP.dll
- ADVAPI32.dll
- USER32.dll
- GDI32.dll
- WS2_32.dll



Figure 9: BurrowShell dynamically resolving APIs at runtime through hash comparison.

The API resolution function iterates through export tables, computing hashes for each function name and matching against an internal table of 50+ hardcoded hash values for required APIs including CreateFileA, WinHttpConnect, GetUserNameW, BitBlt, and socket operations.

Network Communication: Initial Connection and Registration

After API resolution, the shellcode initiates an outbound HTTPS connection to its C2 server at `www[.]gov-pk[.]workers[.]dev` over port 443 using the WinHTTP API. The connection uses a custom User-Agent string designed to blend with legitimate Windows Update traffic:

User-Agent: `Windows-Update-Agent/10.0.10011.Client-Protocol/2.50`

As part of the initial check-in, the malware collects basic host and process metadata, and constructs a JSON-formatted registration payload. This data is transmitted via HTTP POST to the endpoint `/beta/deviceManagement/managedDevices`:

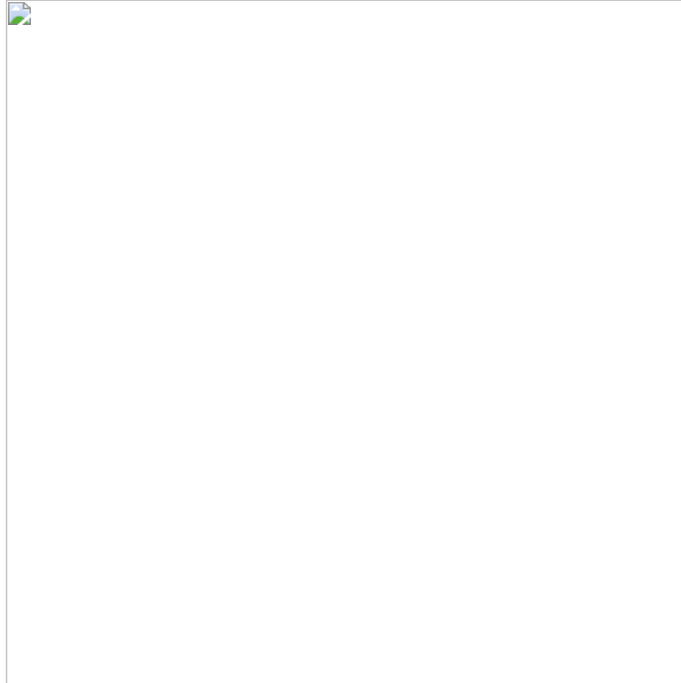


Figure 10: Example (fictional) victim information sent to C2 for registration.

The `device_id` value is generated using a combination of the current process ID and system time values according to the following formula:



Figure 11: Device_id generation formula using process ID and timestamp components.

Notably, while the implant gathers system information via Windows APIs, the arch and os_info fields are hardcoded to "x64" and "Windows" respectively, rather than using the dynamically collected values.

Network Communication: C2 Commands

The implant enters an infinite loop, periodically sending heartbeat ("alive") POST requests to the endpoint `/v10/WindowsUpdate/ClientWebService/ClientService.asmx/SyncUpdates` with a JSON payload containing the device_id and status. The C2 response contains a task array with commands to execute:

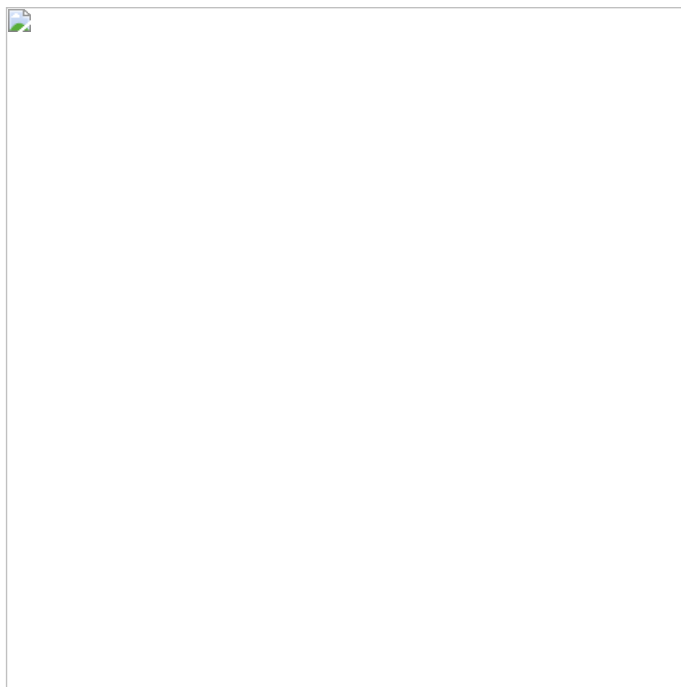


Figure 12: Implant command C2 response.

If the C2 response contains “re-register”, the implant repeats the initial registration process.

The implant supports the following command set:

Command	Purpose	Arguments	Behavior/Effect
config	Update implant configuration	sleep_time, jitter	Updates beacon sleep interval and jitter values in memory
pwd	Print working directory	none	Returns current working directory via GetCurrentDirectoryA
cd	Change directory	<path>	Changes current working directory via GetFileAttributesA & SetCurrentDirectoryA.
ls/dir	List directory contents	[path]	Lists files/directories via FindFirstFileA & FindNextFileA.
download	Exfiltrate file	<path>	Reads file via CreateFileA and ReadFile, Base64-encodes contents, sends to C2.
upload	Write file to disk	<path>, <base64 data>	Decodes Base64 data and writes file via CreateFileA & WriteFile.
rm	Delete file or directory	<path>	Deletes directory via RemoveDirectoryA or file via DeleteFileA.
mkdir	Create directory	<path>	Creates directory via CreateDirectoryA; optionally marks hidden using SetFileAttributesA.
cp	Copy file	<source> <destination>	Copies file via CopyFileA.
mv	Move/rename file	<source> <destination>	Moves or renames file via MoveFileExA.
screenshot	Capture screen	none	Takes screenshot via GetDC, CreateCompatibleDC, CreateDIBSection, and BitBlt; Base64 encodes BMP data and sends to C2.
perun	Execute process (no output)	<command>	Launches process via CreateProcessW.
shell	Execute shell command	<command>	Executes command via CreateProcessW and returns output.

socks_connect	Open SOCKS tunnel	<channel_id> <host> <port>	Establishes outbound TCP connection via WSASStartup, socket, getaddrinfo, and connect.
socks_data	Transfer SOCKS data	<channel_id> <base64 data>	Relays data to remote socket via send, receives response via recv, returns Base64-encoded data to C2.
socks_close	Close SOCKS tunnel	<channel_id>	Terminates SOCKS proxy channel via closesocket.

Event Messaging (OneCollector)

The implant includes an internal event messaging mechanism referred to as “OneCollector” in the code, likely designed to mimic legitimate Microsoft telemetry endpoints. This functionality transmits predefined status messages to the operator via the /OneCollector/1.0/ endpoint using a simple JSON structure:

```
{"device_id": <id>, "result": "<message>"}
```

Status messages include success notifications (example: “[+] SOCKS tunnel connected”, “[+] File copied:”) and error messages (example: “[–] Download failed: File too large or empty (max 100MB)”, “[–] Failed to create directory:”).



Figure 13: Execution chain diagram showing complete attack flow from PDF lure to C2 communication.

Attack Vector 2: Excel Macro with Keylogger Delivery

Stage 1: Initial Access via Macro-Laden Excel Document

The second attack vector uses a malicious Excel spreadsheet containing a VBA macro that executes upon document opening.

Field	Value
SHA-256	1946315d645d9a8c5114759b350ec4f85dba5f9ee4a63d74437d7a068bff7752

File Type	Microsoft Excel sheet
Size	803,840 bytes
First Seen	2026-01-07 10:42:58 UTC

Upon execution, the macro defines a User-Agent string (Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)) and downloads two files from external URLs:

1. **favicon.ico** from `hxxps://ftp[.]desco-gov-bd[.]workers[.]dev/favicon.ico` – saved as `C:\ProgramData\sppc.dll`
2. **audiodg.pdf** from `hxxps://fancy-voice-b182[.]goldibrowhoami[.]workers[.]dev/audiodg.pdf` – saved as `C:\ProgramData\audiodg.exe`

After downloading both files, the macro executes `audiodg.exe` to initiate the next stage of the attack chain.



Figure 14: Malicious macro executed via Excel file and running audiodg.exe.

Stage 2: DLL Sideloadng with Legitimate Microsoft Binary

audiodg.exe (phoneactivate.exe)

The file delivered as audiodg.exe is a legitimate Microsoft binary – phoneactivate.exe – normally located at C:\Windows\System32\phoneactivate.exe.

Field	Value
Name	audiodg.exe (phoneactivate.exe)

SHA-256	8cc46f6ef1b659fa463b7eb343b4ca033de89c313af2e68e2cc7ce08eaff88de
File Type	PE64 executable
Size	133,688 bytes

This legitimate binary attempts to load sppc.dll from its current directory, enabling the same DLL search order hijacking technique observed in Attack Vector 1. When audiodg.exe executes from C:\ProgramData\, it loads the malicious sppc.dll placed in the same directory by the macro.

sppc.dll (Rust-based keylogger)

The primary payload is a Rust-based keylogger with remote access capabilities.

Field	Value
Name	sppc.dll
SHA-256	4f1628821c13cc27fd4134301cc93a1ad32b2a3f7066c3d90f7ba89e02180754
File Type	Win32 DLL x64
Size	429,568 bytes
Compilation Timestamp	2025-08-18 11:40:42 UTC

Analysis reveals a Rust compilation path artifact: C:\Users\pakis\.cargo\registry, consistent with the hostname reference found in the ClickOnce manifest files (DESKTOP-7LTGBEK\pakis).

The malware loads the following DLLs for functionality:

- gdi32.dll
- ntdll.dll
- winhttp.dll
- user32.dll
- ws2_32.dll
- bcrypt.dll
- kernel32.dll
- dbghelp.dll

Rust Keylogger Capabilities

The keylogger provides an extended set of capabilities compared to the in-memory shellcode implant:

- **Keylogging:** Captures keystrokes with special key handling ([BACK], [ESC], [LEFT], [UP], [RIGHT], [DOWN], [DEL], [HOME], [END], [PGUP], [PGDN], [ENTER])
- **Persistence:** Establishes startup persistence
- **Remote command execution:** Executes arbitrary commands
- **File system operations:** ls, rm, cp, mv, mkdir, upload, download
- **Network reconnaissance:** Port scanning capability
- **Process execution:** Launch arbitrary processes
- **Screenshots:** Capture screen contents
- **Configurable timing:** Adjustable sleep and jitter intervals

Source code module references identified in the binary include:

- src\cp.rs
- src\ls.rs
- src\key.rs
- src\lib.rs
- src\shell.rs
- src\check.rs
- src\download.rs
- src\portscan.rs
- src\move_file.rs
- src\polymorph.rs

Keylogger Network Communication

The initial C2 check-in mirrors the behavior of the shellcode implant. Upon execution, the malware establishes an outbound HTTPS connection to the embedded C2 at `api[.]desco-gov-bd[.]workers[.]dev:443`.

The connection uses the User-Agent string: `Mozilla/5.0 (compatible; SecureNet/1.2)`

After establishing connectivity, the malware registers the infected host via POST request to `/api/analytics/collect`:

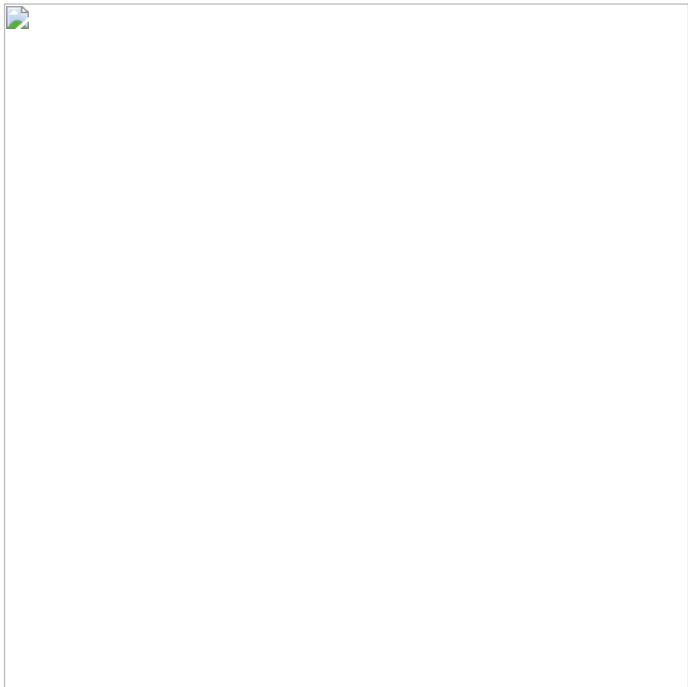


Figure 15: Example JSON payload and C2 response message.

This identifier is subsequently used to poll the C2 for tasking via `/api/content/updates/{networklog_id}`.

Additional C2 references within the malware:

- `/api/assets/chunked_upload` (Chunked Files Upload)
- `/api/feedback/submit` (Output results submission)
- `/api/assets/content` (File Content Upload)
- `/api/content/updates/` (Command Retrieval)
- `/api/analytics/collect` (Beacon Checkin)

Network Infrastructure Analysis

Cloudflare Workers Abuse

SloppyLemming extensively abuses Cloudflare Workers as part of its C2 and delivery infrastructure. Cloudflare Workers is a serverless platform that allows deployment of code across Cloudflare’s global content delivery network, providing the threat actor with low-latency, resilient infrastructure that blends in with legitimate traffic.

This abuse pattern is not novel for SloppyLemming; the group has previously leveraged Cloudflare Workers for credential harvesting and C2 relay operations, as [documented in public reporting](#).

The threat actor establishes domains following naming conventions designed to impersonate the following Pakistani and Bangladeshi government agencies and organizations:

Domain Pattern	Impersonated Entity
webmail-pnra[.]gov-pk[.]workers[.]dev	Pakistan Nuclear Regulatory Authority
info[.]sco-gov-pk[.]workers[.]dev	Special Communications Organization

ftp[.]desco-gov-bd[.]workers[.]dev	Dhaka Electric Supply Company
api[.]desco-gov-bd[.]workers[.]dev	Dhaka Electric Supply Company
support[.]paknavy-gov-pk-fd9[.]workers[.]dev	Pakistan Navy
xen[.]pgcb-gov-bd[.]workers[.]dev	Power Grid Company of Bangladesh
vrms[.]bangladeshbaank-gov-bd[.]workers[.]dev	Bangladesh Bank
cms[.]indu-edu-gov[.]workers[.]dev	National Defense University
ntsoc[.]pta-gov-pk[.]workers[.]dev	Pakistan Telecommunication Authority
uploads[.]ptcl-gov-pk[.]workers[.]dev	PTCL (Pakistan Telecommunication Company)

Infrastructure Timeline

Analysis of domain registration and activity timelines reveals sustained operational cadence:

Period	New Domains	Notes
January 2025	6	Campaign infrastructure establishment begins.
February 2025	4	
March 2025	5	
April 2025	7	
May 2025	8	
June 2025	1	
July 2025	42	Peak activity – significant infrastructure expansion.
August 2025	2	
September 2025	2	
October 2025	13	
November 2025	16	
December 2025	3	
January 2026	3	Campaign continues.
Total	112	

The significant spike in July 2025 (42 new domains) suggests either a major operational expansion or preparation for a sustained campaign phase.

Open Directory Exposure

Three domains were discovered configured as open directories – an operational security failure that allowed Arctic Wolf researchers to download and analyze malware components directly:

1. **webmail-pnra[.]gov-pk[.]workers[.]dev**
2. **info[.]sco-gov-pk[.]workers[.]dev**
3. **file-super-net-pk[.]workers[.]dev**

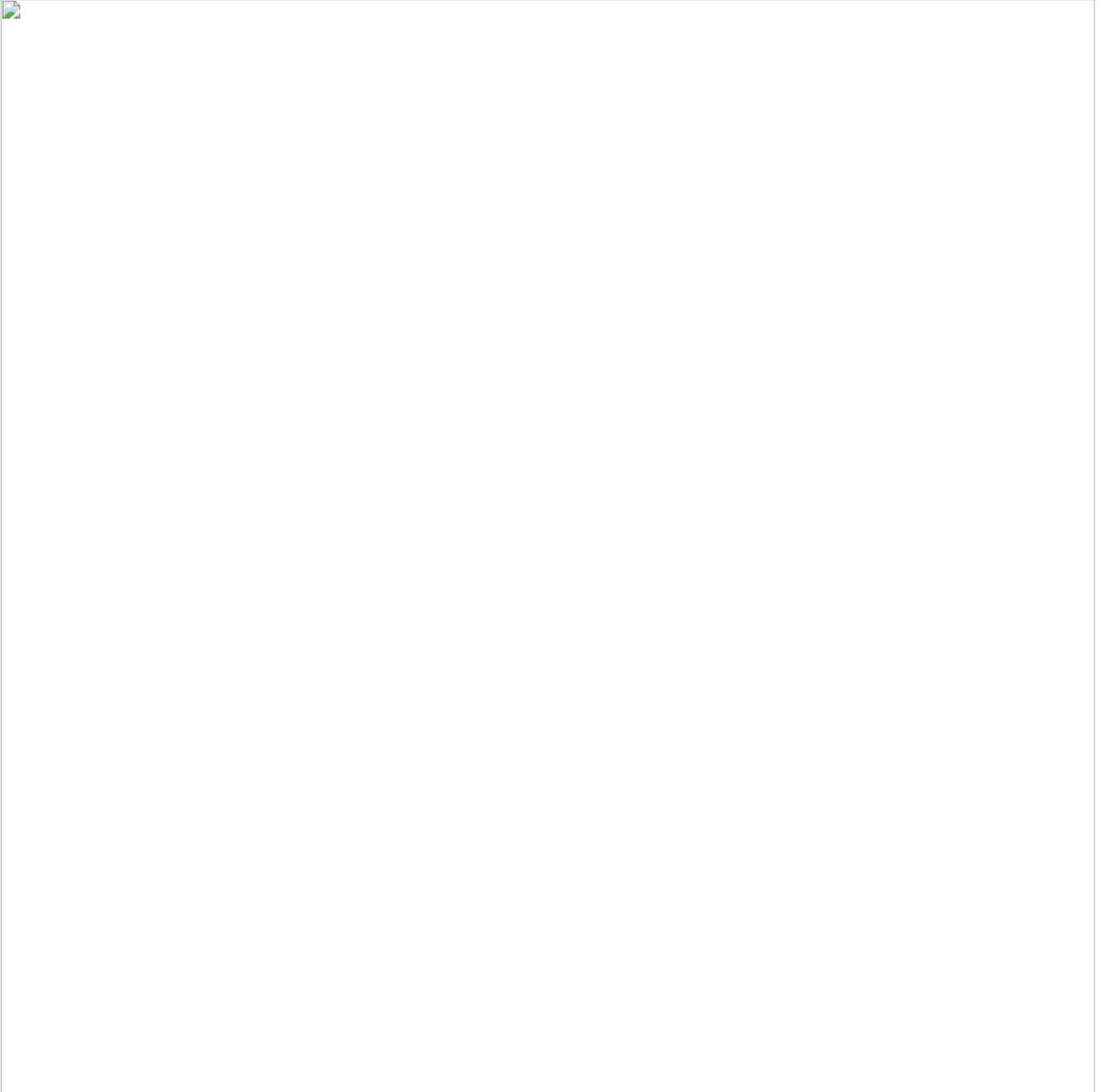


Figure 16: info[.]sco-gov-pk[.]workers[.]dev open directory showing hosted files.

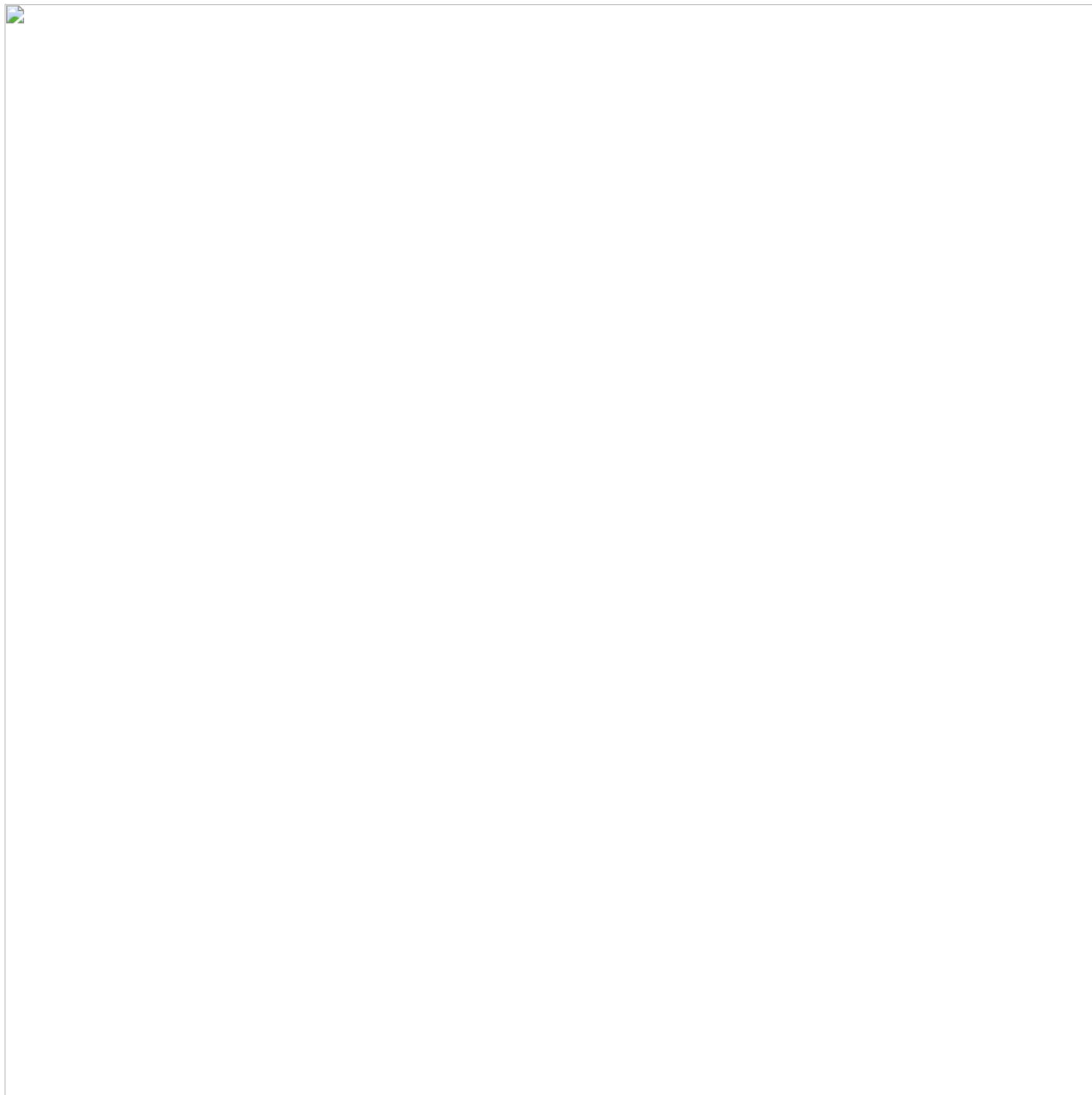


Figure 17: webmail-pnra[.]gov-pk[.]workers[.]dev open directory showing hosted files.

Files observed in these directories included:

- OneDrive.exe
- OneDrive.pdf
- ftp.pnra.org.application
- ftp.pnra.org.exe.manifest
- mscorsvc.dll
- sppc.pdf
- system32.dll
- system32.pdf
- cryptbase.dll
- test.dll
- vault.dll

- PDF-Reader.application
- PDF-Reader.exe.manifest
- OneDriveStandAlone.exe

Havoc Framework Components

Analysis of files recovered from open directories revealed several components associated with the Havoc post-exploitation C2 framework. Two distinct DLL loaders and a Havoc shellcode payload were identified:

SHA-256	Name	Type	RC4 Key
6ea8fd10725676c886692d3acda9782e044c9f3988276360c87559dcdf1a3123	cryptbase.dll	Havoc Loader	oudabiaxuixskxmdwrnomhwomgdguszp
87822f0b579c6c123c72971ee524a2d977ba4f02027f32d57a533d8f123183c3	test.dll	Havoc Loader	oudabiaxuixskxmdwrnomhwomgdguszp
7e16fc7603e450b28f06e55748ef65204f8685b0f75e963da997192fdec5f96e	vault.dll / system32.dll	Havoc Shellcode x64	—

The Havoc shellcode contains a unique C2 domain:

Type	Value
C2 Domain	info[.]bangladesh-islamic-baank[.]workers[.]dev
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36

This domain impersonates a Bangladeshi financial institution, consistent with the campaign’s targeting pattern.

The use of Havoc is a recurring TTP for SloppyLemming. The group was documented leveraging Havoc in [2024 campaigns targeting entities in Southeast Asia](#). However, at the time of our analysis, no evidence was found indicating these specific Havoc components were actively deployed against targets.

Victimology and Target Analysis

Inferred Targeting

Based on multiple lure documents content, C2 domain naming conventions, VirusTotal submission metadata, and infrastructure analysis, this threat actor targets the following sectors and entities:

Pakistan:

- Nuclear Sector
- Defense
- Telecommunications
- Government

Bangladesh:

- Energy
- Financial
- Media

Sri Lanka (Secondary):

Defense

Attribution Assessment

moderate confidence that this campaign is attributable to SloppyLemming. This assessment is based on multiple observed factors, including continuity in tactics, tooling, and operational tradecraft that align with the group’s previously documented behavior and assessed motivation.

Supporting Evidence

TTP Overlap with Documented Activity: The tactics, techniques, and procedures observed in this campaign closely resemble those used in earlier SloppyLemming activity documented by [Cloudflare’s CloudForce One in September 2024](#). Key overlaps include:

- Extensive abuse of Cloudflare Workers for C2 and payload delivery
- Domain naming conventions mimicking government entities
- Use of ClickOnce application deployment for payload delivery
- DLL sideloading using legitimate Microsoft binaries
- Targeting of government, telecommunications, and critical infrastructure sectors in Pakistan and Bangladesh

Tooling Consistency:

- Use of the Havoc C2 framework, consistent with documented SloppyLemming activity in 2024 Southeast Asia campaigns
- Custom Rust-based tooling with similar capability sets
- Multi-stage execution chains using DLL search order hijacking

Victimology Alignment: Inferred targeting based on C2 domain naming conventions indicates focus on government, defense, law enforcement, telecommunications, financial, and critical infrastructure sectors across Pakistan and Bangladesh. This sectoral and geographic distribution closely mirrors SloppyLemming’s historically documented operations in South Asia.

Alternative Hypotheses Considered

SideWinder Overlap: Elements of the infection chain and initial access methodology are consistent with TTPs associated with recent SideWinder operations [reported](#) in October 2025, particularly the use of PDF-based lures and ClickOnce-enabled execution flows. However, meaningful distinctions exist:

- The SideWinder APT group has not been documented using the Havoc framework
- SideWinder typically employs more polished operational security
- The use of Rust-based custom tooling is more consistent with SloppyLemming
- Infrastructure registration patterns and domain naming conventions align with SloppyLemming tradecraft

Shared/Leaked Tooling: The techniques employed (DLL sideloading, ClickOnce abuse) are well-documented and could be adopted by multiple threat actors. However, the combination of these techniques with specific infrastructure patterns, victimology, and tooling choices provides sufficient uniqueness for attribution.

Confidence Assessment

Evidence Type	Weight	Finding
Infrastructure	High	Cloudflare Workers abuse pattern, domain naming conventions consistent with documented SloppyLemming activity.
TTPs	Medium-High	Execution chain methodology matches previous campaigns.
Tooling	Medium	Havoc framework use documented in prior SloppyLemming operations.
Victimology	Medium	Targeting profile consistent with group’s known interests.
Code Artifacts	Low-Medium	“pakis” username reference provides geographic indicator.

Confidence Level: MODERATE*

Attribution is assessed as moderate rather than high, due to limited unique technical indicators (publicly documented techniques), potential for TTP mimicry given public reporting on SloppyLemming operations, and absence of direct infrastructure overlap with previously attributed campaigns.

**Attribution reflects Arctic Wolf Labs’ assessment as of the report period and may evolve with new evidence. References to threat actor identity, nexus, and intent are analytical judgments, not statements of legal fact.*

Mitigation Recommendations

Detection and Prevention

Email and Document Security:

- Block or quarantine PDF files containing embedded URLs pointing to Cloudflare Workers subdomains.
- Implement or maintain macro execution control for Office documents from external and unknown sources.
- Deploy email security solutions capable of analyzing embedded URLs within document content.
- Configure endpoints to block ClickOnce application downloads from untrusted sources.

Network-Based Controls:

- Monitor for and alert on connections to *.workers.dev domains, particularly those matching patterns like *-gov-pk.workers.dev or *-gov-bd.workers.dev.
- Block known malicious domains at the network perimeter.
- Implement SSL/TLS inspection to analyze encrypted traffic to suspicious destinations.
- Monitor for the specific User-Agent strings:
 - Windows-Update-Agent/10.0.10011.Client-Protocol/2.50
 - Mozilla/5.0 (compatible; SecureNet/1.2)

Endpoint Detection:

- Monitor for DLL sideloading activity involving legitimate Microsoft binaries (NgenTask.exe, phoneactivate.exe) executing from non-standard locations.
- Alert on processes creating persistence via the Software\Microsoft\Windows\CurrentVersion\Run registry key.
- Detect RC4 decryption patterns using SystemFunction033 in non-standard contexts.
- Monitor for files named mscorsvc.dll, sppc.dll, cryptbase.dll, or system32.dll in user-writable directories.

File System Indicators:

- Monitor for file creation in C:\ProgramData\ matching patterns: audiodg.exe, sppc.dll.
- Alert on ClickOnce application cache directories containing suspicious executables.
- Detect PDF files created using pdf-lib library (metadata indicator).

Security Awareness Training:

- Organizations should consider implementing regular user awareness training to make users aware of the [typical phishing red flags](#).
- For those without the time to create security training resources from scratch, the [Arctic Wolf Managed Security Awareness® training solution](#) delivers easily digestible security lessons for employees, including regular phishing simulations and a “Report Phish” button.

Conclusions

This investigation documents an ongoing cyber-espionage campaign attributed by Arctic Wolf with moderate confidence to the suspected India-aligned threat actor SloppyLemming. The campaign represents a continuation and evolution of activity documented by Cloudflare’s CloudForce One in September 2024, with the threat actor expanding both its infrastructure and tooling while maintaining consistent targeting patterns focused on Pakistani and Bangladeshi government and critical infrastructure entities.

The technical analysis reveals a threat actor operating with moderate capability: the multi-stage execution chains demonstrate understanding of defense evasion techniques and shows familiarity with Windows internals, while the operational security failures – particularly the open directory exposures – indicate areas where tradecraft falls short of the capabilities of more disciplined threat actors. This assessment aligns with the “Sloppy” designation in the group’s name, which references their historically inconsistent operational security.

The deployment of dual payloads – the in-memory shellcode BurrowShell for C2 and SOCKS proxy operations, and a Rust-based keylogger for information stealing – suggests the threat actor maintains flexibility to deploy appropriate tools based on target value and operational requirements. BurrowShell represents a purpose-built implant with mature operational features including network tunneling via SOCKS proxy. The implant’s design choices of dynamic API resolution, traffic masquerading as Windows Update communications, and RC4 encryption, indicate deliberate effort to evade detection and maintain persistent access. The presence of the Havoc C2 framework components indicates the group continues to leverage commodity post-exploitation tools alongside custom implants.

The sustained infrastructure development spanning January 2025 through January 2026, with 112 unique domains identified and peak registration activity in July 2025, demonstrates dedicated operational commitment to this campaign. The choice of Cloudflare Workers provides the threat actor with resilient, legitimate-appearing infrastructure that complicates network-based detection and blocking efforts.

From a geopolitical perspective, the targeting of Pakistani nuclear regulatory bodies, defense logistics organizations, and telecommunications infrastructure – alongside Bangladeshi energy utilities and financial institutions – aligns with intelligence collection priorities consistent with regional strategic competition in South Asia. Organizations within these sectors should consider themselves potential targets and implement appropriate defensive measures.

APPENDIX

Indicators of Compromise (IOCs)

NOTE: The IOCs mentioned below are referential. The full list has been provided to Arctic Wolf customers. These technical indicators are intended for defensive use. Do not use these indicators or techniques for offensive purposes.

File Indicators

Referential Initial Access Documents:

SHA-256	Name	Type
8faeea306a331d86ce1acb92c8028b4322efbd11a971379ba81a6b769ff5ac4b	N/A	PDF
1946315d645d9a8c5114759b350ec4f85dba5f9ee4a63d74437d7a068bff7752	N/A	XLS

Loaders and DLLs:

SHA-256	Name	Description
81d1a62c00724c1dfbc05a79ac4ae921c459350a2a4a93366c0842fadca40b011	mscorsvc.dll	Shellcode loader
4f1628821c13cc27fd4134301cc93a1ad32b2a3f7066c3d90f7ba89e02180754	sppc.dll	Keylogger
c57baa17321257ea1915ba0336a89f63975e6ed612a89c9888be7067222bef38	cryptbase.dll	Keylogger variant
67c341e187ddfd5a4a7df8743ae82e72db1e5c3747d5c4e185d99f54182f093	cryptbase.dll	Keylogger variant
6ea8fd10725676c886692d3acda9782e044c9f3988276360c87559dcdf1a3123	cryptbase.dll	Havoc loader
87822f0b579c6c123c72971ee524a2d977ba4f02027f32d57a533d8f123183c3	test.dll	Havoc loader

Encrypted Payloads:

SHA-256	Name	Description
3dbf64da37616acbe16bc6bd06a320fed416c4c8ec37a04f811a32389af3d46c	system32.dll	RC4-encrypted shellcode
7e16fc7603e450b28f06e55748ef65204f8685b0f75e963da997192fdec5f96e	vault.dll / system32.dll	Havoc shellcode x64
bb83cd7ebe75cf62f06859ab2166a35a16cac924f874109b78dd5c4b653d6d44	system32.pdf	Encrypted shellcode
3269829b50da5b3c4120a103ef72b09a8bbbf258ab3086ca24b2aa24dc00039b	sppc.pdf	Custom implant loader

Legitimate Binaries (Abused for Sideloadng):

SHA-256	Name	Legitimate Name
9fd133b11abcbbed33ccea71bd4743e8f35e42cd637fb763f5ab2a8fbb9b6261	OneDrive.exe	NGenTask.exe
8cc46f6ef1b659fa463b7eb343b4ca033de89c313af2e68e2cc7ce08eaff88de	audiodg.exe	phoneactivate.exe

ClickOnce Manifests:

SHA-256	Name
c4cea4147719c3abe7eb6c7c7e3420480361773b602d4270af0a607d29f8771f	ftp[.]pnra[.]org.application
c603e7a1018f7b3a168404bcf2f709950c4e29e0596c78823647baaadaf317c7	ftp[.]pnra[.]org.exe.manifest
1f79f88e97e60bc431ab641ccbbfb09e9d2633d258d3d4bc8b0cb5b9adbc9a4a	PDF-Reader.application
7a34070f98bd129764f053d8003b402975f73e85da87eebdfcc718ac7c8bb0bb	PDF-Reader.exe.manifest

d071ea65ea30df38623afe959ccc142f14dc4659dce21c2d7195e31245ee2df1	MicrosoftPDF-Reader.application
9dca24630c06463a01ca6d38b73987589bbe68650b0ff893770eab9ff6ec581a	MicrosoftPDF-Reader.exe.manifest

Network Indicators

Malicious C2 Domains:

Domain	Purpose
www[.]gov-pk[.]workers[.]dev	Shellcode implant C2
api[.]desco-gov-bd[.]workers[.]dev	Keylogger C2
info[.]bangladesh-islamic-baank[.]workers[.]dev	Havoc C2

User-Agent Strings

Windows-Update-Agent/10.0.10011.Client-Protocol/2.50

Mozilla/5.0 (compatible; SecureNet/1.2)

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Host Indicators

Registry Keys Created

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\OneDrive

File Paths

C:\ProgramData\audiodg.exe

C:\ProgramData\sppc.dll

C:\Users\<Username>\AppData\Local\Apps\2.0\<12randomchars>\<12randomchars>\ftp...app_*

RC4 Decryption Keys

boikztaigkuneapfypesuabfmpxgwnad (Shellcode loader)

oudabiauxisksxmdwrnomhwomdgduszp (Havoc loader)

gzmzsduyrttrnwlpjfgylwwinlopsznc (Custom implant loader)

C2 References

/beta/deviceManagement/managedDevices

/v10/WindowsUpdate/ClientWebService/ClientService.asmx/SyncUpdates

/OneCollector/1.0/

/api/analytics/collect

/api/content/updates/{id}

/api/assets/chunked_upload

/api/feedback/submit

/api/assets/content

Applied Countermeasures

YARA Rules

```
rule targeted_SloppyLemming_Rust_Keylogger_RAT
{
    meta:
        description = "Rule to detect SloppyLemming Keylogging Remote Access tool used to target Bangladesh based entities"
        author = "Arctic Wolf"
        distribution = "TLP:CLEAR"
        version = "1.1"
        creation_date = "2026-01-08"
        last_modified = "2026-01-08"
        hash256 = "4f1628821c13cc27fd4134301cc93a1ad32b2a3f7066c3d90f7ba89e02180754"
    strings:
        $a1 = "=== KEYLOGGER SUMMARY ===" ascii wide
        $a2 = "\\c.cargo\\registry\\" ascii wide
        $a3 = "[BACK][ESC][LEFT][UP][RIGHT][DOWN][DEL][HOME][END][PGUP][PGDN]" ascii wide
        $a4 = "Failed to create log file: " ascii wide
    condition:
        uint16(0) == 0x5a4d and ((filesize < 15000KB) and all of ($a*))
}

rule targeted_SloppyLemming_BurrowShell_loader
{
    meta:
        description = "Rule to detect sloppylemming BurrowShell loader including Encrypted Shellcode"
        author = "Arctic Wolf"
        distribution = "TLP:CLEAR"
        version = "1.0"
        creation_date = "2026-01-08"
        last_modified = "2026-01-27"
        hash256 = "f46dd8154f963a8d49c4503bcfb93caf6551f4c845377c95fdde52ce9ca9798b"
    strings:
        $a1 = "Less than 2 GB of RAM detected. Possible virtual environment" ascii wide
        $a2 = "Less than 2 processors detected. Possible virtual environment" ascii wide
        $a3 = "GlobalMemoryStatusEx Failed With Error" ascii wide
        $a4 = "SystemFunction033" ascii wide
    condition:
        uint16(0) == 0x5a4d and ((filesize < 15000KB) and all of ($a*))
}
```

MITRE ATT&CK® Mapping

Tactic	Technique	Sub-Technique / Procedure Context
Resource Development	T1583.001 – Acquire Infrastructure: Domains	Registered 112+ Cloudflare Workers subdomains mimicking Pakistani and Bangladeshi government entities.
Resource Development	T1587.001 – Develop Capabilities: Malware	Developed custom Rust-based keylogger and x64 shellcode implant.
Initial Access	T1566.001 – Phishing: Spearphishing Attachment	Distributed malicious PDF and Excel documents via email.
Initial Access	T1566.002 – Phishing: Spearphishing Link	PDF lures contain embedded malicious URLs to ClickOnce manifests.
Execution	T1204.002 – User Execution: Malicious File	Victims must click “Download file” button in PDF or enable macros in Excel.
Execution	T1059.005 – Command and Scripting Interpreter: Visual Basic	Excel documents execute VBA macros to download payloads.
Persistence	T1547.001 – Boot or Logon Autostart Execution: Registry Run Keys	Creates Run key entry for OneDrive.exe persistence.
Defense Evasion	T1574.001 – Hijack Execution Flow: DLL Search Order Hijacking	Places malicious DLLs alongside legitimate Microsoft binaries.
Defense Evasion	T1574.002 – Hijack Execution Flow: DLL Side-Loading	Abuses NGenTask.exe and phoneactivate.exe to load malicious DLLs.

Defense Evasion	T1140 – Deobfuscate/Decode Files or Information	RC4 decryption of shellcode payload using hardcoded key.
Defense Evasion	T1027.002 – Obfuscated Files or Information: Software Packing	Shellcode payload delivered as encrypted blob.
Defense Evasion	T1036.005 – Masquerading: Match Legitimate Name or Location	DLLs named mscorsvc.dll, sppc.dll, system32.dll to appear legitimate.
Defense Evasion	T1218 – System Binary Proxy Execution	Abuse of ClickOnce deployment mechanism.
Credential Access	T1056.001 – Input Capture: Keylogging	Rust-based implant captures keystrokes.
Discovery	T1082 – System Information Discovery	Collects computer name, username, architecture, admin status.
Discovery	T1083 – File and Directory Discovery	Directory listing commands (ls, dir).
Discovery	T1057 – Process Discovery	Gathers process information.
Collection	T1113 – Screen Capture	Screenshot capability in both implants.
Collection	T1560 – Archive Collected Data	Base64 encoding of exfiltrated data.
Command and Control	T1071.001 – Application Layer Protocol: Web Protocols	HTTPS communication to C2 servers.
Command and Control	T1573.001 – Encrypted Channel: Symmetric Cryptography	TLS-encrypted C2 communications.
Command and Control	T1090.001 – Proxy: Internal Proxy	SOCKS proxy capability for network pivoting.
Command and Control	T1102.002 – Web Service: Bidirectional Communication	Abuse of Cloudflare Workers for C2.
Command and Control	T1571 – Non-Standard Port	C2 communication over port 443 mimicking legitimate HTTPS.
Exfiltration	T1041 – Exfiltration Over C2 Channel	Data exfiltrated via C2 HTTP POST requests.

Legal disclaimer: This Arctic Wolf Labs report is provided for informational purposes only and does not constitute a guarantee of detection or prevention. Defensive effectiveness varies by environment, configuration, and available telemetry.

About the Authors

Arctic Wolf Labs

[Arctic Wolf Labs](#) is a group of elite security researchers, data scientists, and security development engineers who explore security topics to deliver cutting-edge threat research on new and emerging adversaries, develop and refine advanced threat detection models with artificial intelligence and machine learning, and drive continuous improvement in the speed, scale, and detection efficacy of Arctic Wolf's solution offerings.

Arctic Wolf Labs brings world-class security innovations to not only Arctic Wolf's customer base, but the security community at large. Sign up for our Threat Intelligence newsletter [here](#).