

Ink Dragon's Relay Network and Stealthy Offensive Operation

 research.checkpoint.com/2025/ink-dragons-relay-network-and-offensive-operation

samanthar@checkpoint.com

December 16, 2025



Inside Ink Dragon: Revealing the Relay Network and Inner Workings of a Stealthy Offensive Operation

December 16, 2025

Key Findings

- In recent months, Check Point Research has identified a new wave of attacks attributed to the Chinese threat actor **Ink Dragon**. Ink Dragon overlaps with threat clusters publicly reported as **Earth Alux**, **Jewelbug**, **REF7707**, **CL-STA-0049**, among others.
- **Ink Dragon has expanded its operational focus to new regions** – In the last few months, the threat actor's activities show increased focus on government targets in Europe in addition to continued activities in Southeast Asia and South America.
- **Ink Dragon builds a victim-based relay network** – Ink Dragon leverages a custom ShadowPad IIS Listener module to turn compromised servers into active nodes within a distributed mesh, allowing each victim to forward commands and traffic, effectively transforming targets into part of their C2 infrastructure.

- **Ink Dragon continues to exploit long-known IIS misconfigurations for initial access** – Despite years of public reporting and awareness within the security community, Ink Dragon still relies on predictable or mismanaged ASP.NET machineKey values to perform ViewState deserialization attacks against vulnerable IIS and SharePoint servers.
- **Ink Dragon is evolving its operations with new TTPs and tools** – The cluster has introduced a new variant of **FinalDraft** malware with enhanced stealth and higher exfiltration throughput, along with advanced evasion techniques that enable stealthy lateral movement and multi-stage malware deployment across compromised networks.

Introduction

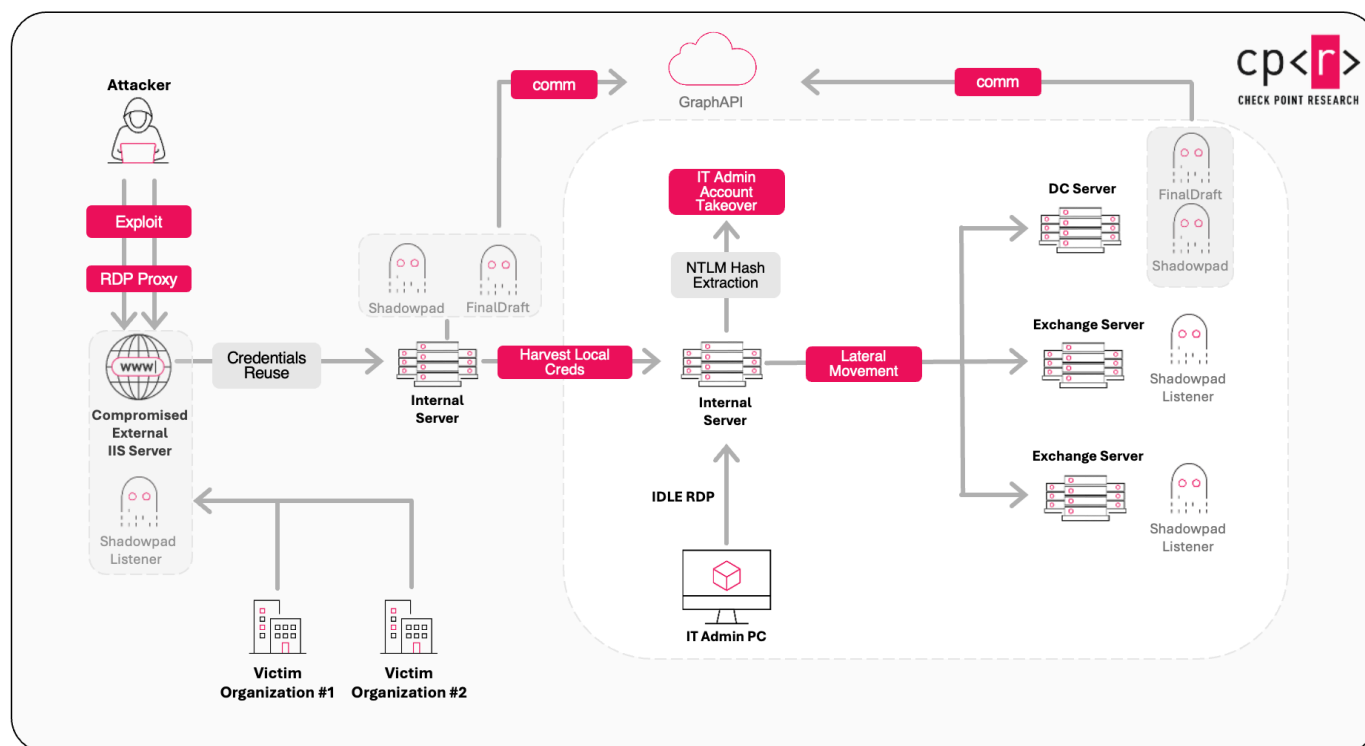
Check Point Research tracks a sustained, highly capable espionage cluster, which we refer to as Ink Dragon, and is referenced in other reports as CL-STA-0049, Earth Alux, or REF7707. This cluster is assessed by several vendors to be PRC-aligned. Since at least early 2023, Ink Dragon has repeatedly targeted government, telecom, and public-sector infrastructure, initially concentrating on Southeast Asia and South America, but with an increasing footprint in Europe and other regions. The actor's campaigns combine solid software engineering, disciplined operational playbooks, and a willingness to reuse platform-native tools to blend into normal enterprise telemetry. This mix makes their intrusions both effective and stealthy.

A notable characteristic of Ink Dragon's operations is their tendency to convert compromised environments into part of a larger, distributed relay network. By deploying a ShadowPad IIS Listener Module across multiple victims, the group effectively turns each breached server into a communication node capable of receiving, forwarding, and proxying commands. This design allows attackers to route traffic not only deeper inside a single organization's network, but also across different victim networks entirely. As a result, one compromise can quietly become another hop in a global, multi-layered infrastructure supporting ongoing campaigns elsewhere, blending operational control with strategic reuse of previously breached assets.

This blog also presents the forensic story of a high-stakes compromise of a European government office, highlighting recurring methods observed across different victims. We walk through the entire kill chain observed in the field, including web-centric initial access, hands-on-keyboard activity, staged loaders, privilege escalation, and credential-harvesting components, as well as aggressive lateral movement that culminated in domain dominance. We also document multiple delivery and persistence patterns that Ink Dragon favors, and unpack a new variant of the FinalDraft backdoor, which is used as a resilient, cloud-native command-and-control platform.

Beyond the technical details, this article shows how Ink Dragon's tooling and repeatable TTPs reflect a mature, modular development model that steadily expands in capability while maintaining a consistent operational philosophy.

Attack Chain



Attackers begin by gaining initial access through ViewState deserialization or ToolShell-based exploits, then deploy ShadowPad on the compromised server. They harvest IIS worker credentials and establish an RDP proxy to move laterally, using RDP and ShadowPad's built-in capabilities along with reused credentials. After obtaining access to a domain admin account, they achieve domain dominance. From there, they deploy FinalDraft on strategic machines and install a ShadowPad IIS listener on public-facing servers, enabling new victims to connect to the attackers' infrastructure as the campaign continues.

Initial Access

In the environments we investigated, the common initial access vector is exploitation of ASP.NET ViewState deserialization via [publicly disclosed machine keys](#). In this scenario, the **__VIEWSTATE** parameter, normally protected using the application's machineKey, can be forged if the key is copied from public sources. Once the attacker can generate a valid signature, they can inject a crafted ViewState payload that the server deserializes, leading to remote code execution.

In other cases, we've also observed the actor abusing the ToolShell SharePoint vulnerability. [ToolShell](#) is an exploit chain targeting on-premises Microsoft SharePoint that combines authentication bypass and unsafe deserialization (CVE-2025-49706 / CVE-2025-53771 and CVE-2025-49704 / CVE-2025-53770, among others) to enable unauthenticated remote code

execution and web shell deployment on vulnerable servers. In July 2025, we observed the actor conducting mass scanning for the ToolShell vulnerability during the initial waves of exploitation, indicating the actor was among a limited set of actors with early access to the exploit.

This demonstrates that the attackers have multiple web-facing options for initial compromise. The practical workflow is straightforward: enumerate internet-facing IIS/SharePoint servers, test for predictable machine keys or vulnerable SharePoint endpoints (often using publicly available fuzzing lists), and submit crafted POSTs or payloads that trigger deserialization/RCE. These attacks are stealthy and scale well against large organizations with inconsistent web configurations.

Internal Operation & Kill Chain Overview

Lateral Movement

In these campaigns, the adversary leverages two complementary strengths of a web compromise after gaining an initial foothold: privileged local artifacts (the IIS application/service credentials and configuration) and visibility into active administrative sessions.

By obtaining the IIS `machineKey/DecryptionKey` or otherwise recovering the site's cryptographic secrets, the attacker can decrypt locally stored configuration blobs and credentials that the site or its worker processes store. In practice, this frequently yields the IIS worker/app-pool account password or other local secrets that carry elevated rights on the host and often across other IIS servers that reuse the same service account or credential material.

With a local administrative credential in hand, Ink Dragon escalates from code execution in `w3wp.exe` to full system control, and then leverages that control to create a persistent remote access channel (commonly an RDP tunnel or scheduled task that launches an administrative payload). Because many organizations reuse service credentials across web farms for management convenience, obtaining a single IIS credential can allow the actor to authenticate to sibling IIS hosts and pivot laterally with minimal network noise. Ink Dragon frequently tunnels RDP traffic to reach internal hosts from a remote workstation, exposing their machine names and enabling direct, interactive sessions that appear superficially legitimate.

From there, the attacker's lateral playbook becomes straightforward: stage a resilient implant (ShadowPad/FinalDraft variants are common) and propagate it using native protocols such as SMB. The operator copies the triad (EXE + side-loaded DLL + encrypted blob) into writable shares, creating a service or scheduled task to run the payload, and attempting to disguise the service/process under a plausible name.

Persistence

The most common patterns we observed:

1. **Scheduled tasks** – the actor created tasks with benign-looking names (notably `SYSCHECK`) set to run under `SYSTEM` and pointing to staged loader hosts such as `conhost.exe`. Tasks were often created to run once at a chosen time, allowing the operator to bootstrap wide-scale re-execution while minimizing noisy periodic callbacks.
2. **Service installation** – on several machines, Ink Dragon installed services to launch their loaders as persistent system services, with service names disguised as Windows updates or temporary maintenance (e.g., `WindowsTempUpdate`). These services were used to run side-loaded triads (EXE + malicious DLL + encrypted blob), guaranteeing automatic restart and `SYSTEM` execution after reboots.

It's essential to note that many of the staged executables (e.g., `conhost.exe`) were renamed to resemble native Windows binaries, yet they were digitally signed by legitimate vendors such as Advanced Micro Devices, Realtek Semiconductor Corp, and NVIDIA.

Their `OriginalFileName` metadata differed from the on-disk names, indicating deliberate masquerading and abuse of trusted signatures to blend with the operating system.

Privilege Escalation

Ink Dragon combines targeted exploitation with heavy credential harvesting to quickly escalate local control into domain-level dominance.

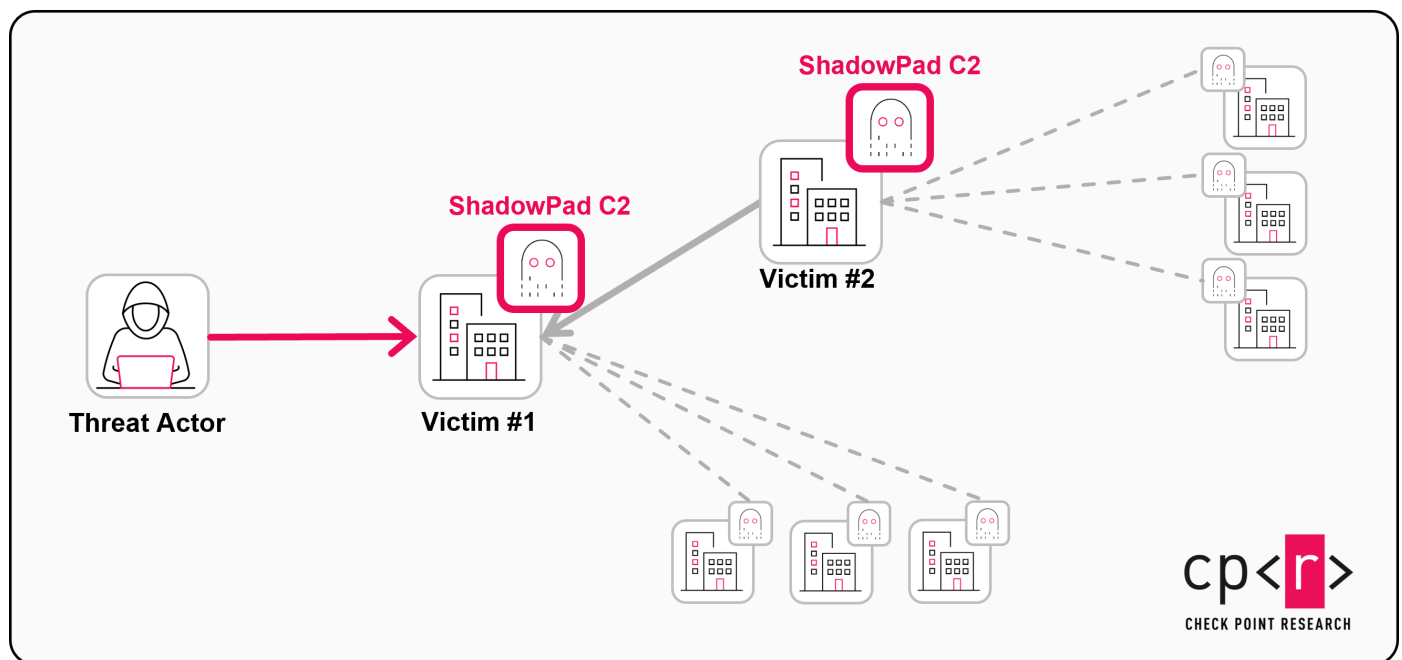
1. **Local escalation via exploitation:** In multiple incidents, the initial `__VIEWSTATE` RCE was followed by local escalation tooling such as `PrintNotifyPotato` to obtain `SYSTEM` from a web server context. This allowed full control over the host and the rights to create persistent services and change firewall settings.
2. **Credential harvesting and LSASS dumping:** the operators deployed custom tools (we observed variants of `LalsDumper`) to create LSASS dumps and extract registry hives (`SAM`, `SYSTEM`) into ProgramData or user-profile directories for offline cracking. Dump files and registry hive exports were then used to recover NTLM hashes and Kerberos material.
3. **Leveraging idle sessions:** Ink Dragon actively enumerated Remote Desktop sessions and administrative consoles on key servers. In at least one instance, the actor located an idle RDP session belonging to a Domain Administrator that had authenticated via Network Level Authentication (CredSSP) using NTLMv2 fallback. Since the session remained disconnected but not logged off, it is highly likely that LSASS retained the associated logon token and NTLM verifier in memory. Ink Dragon obtained `SYSTEM`-level access to the host, extracted the token (and possibly the NTLM key material), and reused it to perform authenticated SMB operations. Through these actions, they were able to write to administrative shares and exfiltrate `NTDS.dit` and registry hives, marking the point at which they achieved domain-wide privilege escalation and control.

Enabling Egress

Ink Dragon modified host firewall rules to permit broad outbound traffic and effectively turned compromised hosts into unconstrained exfiltration/proxy nodes.

The group created a permissive outbound rule labeled to resemble legitimate software (we observed a rule named **Microsoft MsMpEng**, associated with **MsMpEng.exe**) that allowed Any → Any outbound traffic across all profiles. The rule was created locally (not via GPO), enabled, and applied to the Defender process in **SYSTEM** context. This bypasses upstream egress controls that would otherwise block custom ports or tunneling traffic.

Building the Relay Network



During the investigation, we discovered that Ink Dragon is actively converting compromised organizations into functional communication nodes within a distributed ShadowPad relay network. This capability is powered by an IIS Listener Module. Instead of serving as a traditional backdoor, the module enables the malware to register new URL listeners directly through the **HttpAddUrl** API, which lets processes bind HTTP(S) endpoints dynamically, including wildcard patterns that match all sub-paths or hostnames. This means the malicious listener seamlessly coexists with legitimate IIS behavior, silently intercepting any incoming HTTP requests whose URL matches its configuration. When a request arrives, the module decrypts the payload and evaluates whether the structure fits its proprietary protocol. If not, it falls back to genuine IIS logic, serving real web content or returning legitimate error codes. The result is a hard-to-detect implant that blends into the server's normal traffic patterns while retaining full control over its hidden C2 channel.

```

if ( GetFileAttributesW(FileName) == -1 )
{
    LODWORD(v31) = 0x4D13BC19;
    // HTTP/1.1 404 Not Found
    v25 = decrypt_string_(&v32, 0x44FF9F1E, 893756311, 1580527907, v31, 0xF7431E46, 0xF85A5EB3, 33966);
    v26 = move_to_wide(v25, 0);
    v27 = send_file_to_client(listener_struct, error_filename, 0x194u, v26);
    if ( v32 )
    {
        free_wrapper(v32);
        v32 = 0;
        v33 = 0;
    }
    if ( v34 )
    {
        free_wrapper(v34);
        v34 = 0;
        v35 = 0;
    }
}
else
{
    ok = decrypt_string_(&v32, 0x27E281E9, 0x9190);
    v29 = move_to_wide(ok, 0);
    v27 = send_file_to_client(listener_struct, FileName, 0xC8u, v29);
    if ( v32 )
    {
        free_wrapper(v32);
        v32 = 0;
        v33 = 0;
    }
    if ( v34 )
    {
        free_wrapper(v34);
        v34 = 0;
        v35 = 0;
    }
}
}

```

Communication with a regular client via the IIS module

Where this becomes significantly more strategic is in how the module manages remote peers. The listener can categorize external IP addresses into two functional roles: servers and clients. It automatically pairs nodes from each list to relay traffic between them. Once two peers are matched, the compromised host becomes a live forwarding point: data arriving from the “server” side is streamed directly to the “client,” and responses are returned in the same manner, effectively turning the victim into a transparent communication bridge. We observed multiple instances where government entities were inadvertently serving as C2 relays. In some instances, the victim machine is leveraged as a hop, serving as an access node for ShadowPad clients active in other, unrelated target environments. This chaining effect forms a multi-layered mesh of compromised infrastructure, allowing the operator to issue commands to downstream implants without direct communication with them.

Beyond its role in constructing a large-scale external relay network, the ShadowPad IIS module also exposes a conventional internal proxy capability, designed to route commands toward ShadowPad nodes located deeper within the same network. When the attackers need to deliver

instructions to implants that do not have direct external reachability, the IIS module simply relays the traffic internally, behaving much like a pivot point inside the compromised environment. This capability is not new in concept, but its integration into a legitimate IIS worker process makes it extremely difficult to distinguish malicious lateral communication from normal internal service traffic.

Beyond the relay logic itself, the module leaves behind an unusually rich forensic artifact: debug strings that document the number of bytes it forwards between external and internal IP addresses. These strings include source, destination, and payload size. During our investigation, this telemetry proved essential in reconstructing the attackers’ communication graph, enabling us to map exactly how commands entered the victim, how they were relayed, and which internal hosts were drawn into the chain. The presence of such granular logging underscores just how central the relay mechanism is to the operator’s workflow: the victim is not merely compromised, but actively repurposed as a communication bridge that keeps the broader ShadowPad infrastructure stitched together.

```
062CD51F: Transfer [      :    <-  ] <- [    <-      :    ]      43 bytes...
062D9293: Transfer [      :    ->  ] -> [    ->      :    ]      16 bytes...
062D92B2: Transfer [      :    <-  ] <- [    <-      :    ]      52 bytes...
062E93D6: Transfer [      :    ->  ] -> [    ->      :    ]      16 bytes...
062E93F5: Transfer [      :    <-  ] <- [    <-      :    ]      42 bytes...
062F0165: recv(      :    ->  )=
062F0165: recv( <-    :    )=
062FC62C: recv(      :    ->  )=
062FC62C: recv( <-    :    )=
062FD08C: recv(      :    ->  )=
062FD08C: recv( <-    :    )=
062FD2BF: recv(      :    ->  )=
```

Debug log strings dumped from the IIS Listener

This architecture enhances stealth and survivability. By routing commands through unrelated victims, the true controlling IP is never exposed, and network defenders have difficulty distinguishing malicious cross-organization traffic from legitimate inter-government or inter-infrastructure communication. Most significantly, every newly compromised perimeter system can be repurposed immediately as another hop, allowing Ink Dragon to expand its operational reach while obscuring both the origin and the direction of command flow.

Tools & Post-Exploitation Components

As we expanded our analysis beyond the initial access vector, a broader toolset began to emerge. Ink Dragon does not operate with a single backdoor or a monolithic framework; instead, the intrusions feature a sequence of purpose-built components that activate at different stages of the operation. What follows is a breakdown of the post-exploitation tooling we recovered, from IIS-embedded ShadowPad modules to debugger-based loaders, credential-harvesting implants, and long-term command-and-control platforms.

Summary of Observed Components

Category	Name	Function	Brief Description
IIS Backdoors & C2	ShadowPad IIS Listener Module	Core C2 & relay node	Intercepts selected IIS traffic, decrypts commands, builds a distributed relay network, and exposes a full ShadowPad backdoor on IIS servers.
Loader	ShadowPad Loader	Payload delivery	Triad structure (EXE + DLL + TMP) that decrypts and runs the ShadowPad core in memory while deleting artifacts.
Loader	CDBLoader	Memory-resident payload execution	Uses <code>cdb.exe</code> scripts to patch memory, run shellcode, and load AES-encrypted payloads via a debugger session.
Credential Access	LalsDumper	LSASS dump extraction	Registers a malicious SSP DLL in LSASS, extracts a compressed LSASS memory dump via custom direct-syscall logic.
Loader	032Loader	Host-dependent payload loader	RC4-like decryption using the system's InstallDate as entropy, delivering payloads via shared memory mapping.
Modular RAT	FinalDraft	Long-term espionage & cloud C2	Modular RAT using Microsoft Graph API; supports exfiltration, RDP history harvesting, tunneling, scheduling, and mailbox-based command exchange.

ShadowPad IIS Listener Module

Module Initialization

The ShadowPad IIS Listener Module deployed in this intrusion is a fully integrated component designed to masquerade as part of the legitimate IIS stack while quietly providing command-and-control and relay capabilities. Its configuration block defines several operational parameters that determine both how it responds to benign web traffic and how it handles attacker-controlled messages. These include a *server type* string (used in HTTP response headers), a *document root* path, and a fallback *error page* path. Even when configuration fields are absent, the module falls back to realistic defaults — "`C:\\inetpub\\wwwroot`", "`C:\\inetpub\\custerr\\en-US\\404.htm`", and "`Microsoft-IIS/10.0`". Those values allow the module to blend seamlessly with a standard Windows Server installation.

The most important part of the configuration is the list of URL patterns the module will intercept. These patterns are stored as wildcard-enabled strings, separated by semicolons, and are used to register listeners via the Windows `HttpAddUrl` API. This API allows the module to attach itself to

arbitrary HTTP URL prefixes, including those containing wildcards for hostnames or paths, making the listener hard to detect unless the exact bindings are enumerated at the OS level. Any inbound request matching one of these patterns is captured and passed through the module's internal decryption and message-parsing routine.

```
def decrypt_first_packet(buf: bytearray, seed: int, length: int):  
    """  
  
    buf = bytearray starting at a1->type (first byte = LOBYTE(seed), second = HIBYTE(seed))  
  
    length = total length of buffer (rdx)  
    """  
  
    count = length - 2  
  
    seed_lo = buf[0]  
    seed_hi = buf[1]  
  
    num = (seed_hi << 8) | seed_lo  
    num &= 0xFFFFFFFF  
  
    pos = 2  
  
    for _ in range(count):  
  
        hi = (num >> 16) & 0xFFFF  
  
        num = (hi * 0x7093915D - num * 0x6EA30000 + 0x06B0F0E3) & 0xFFFFFFFF  
  
        buf[pos] ^= num & 0xFF  
  
        pos += 1  
  
    return buf
```

Requests that do not conform to the attacker's expected encrypted structure are quietly handled as normal IIS traffic: static files are served from the configured webroot folder when available, and legitimate error pages are returned otherwise. This fallback behavior ensures the listener remains operationally stealthy, presenting a legitimate façade while still acting as a covert interception point.

Split Command Architecture

Once a request passes the module's URL-matching, decryption, and structural checks, the ShadowPad IIS Listener evaluates the embedded command ID. These commands fall into two broad categories, both of which are handled by the same component.

The first category contains the instructions responsible for building and maintaining the distributed relay network. These include commands that register an endpoint as a "server," commands that register an endpoint as a "client," and the logic that pairs nodes, forwards traffic, and maintains the two queues that drive the hop-to-hop communication model.

The second category is entirely separate in purpose: a full backdoor command set enabling the operator to interact directly with the local system. These commands cover host reconnaissance, file operations, data collection, payload staging, configuration updates, process control, and network-level actions. In other words, the same module responsible for linking compromised machines into a cross-victim relay chain is also fully capable of operating as a traditional ShadowPad backdoor on that same host.

Distributed Relay Network Construction

The most strategically significant capability lies in the module's role in building and maintaining the operator's distributed relay network. The IIS Listener maintains two concurrent registries of peers: a server list and a client list. Nodes can add themselves to either list via dedicated command IDs. Entries in the server list are periodically revalidated every 30 seconds, and the module issues acknowledgments to confirm ongoing availability. Nodes placed into the client list behave differently; if a client remains unpaired for 30 seconds, it is automatically pruned to prevent stale links.

Whenever a new node is inserted into one of the lists and the Listener detects that both lists contain at least one live entry, it pairs the first server node with the first client node. At that moment, it sends the server the victim's IP address. After this handshake, the module establishes bidirectional relaying between the two sockets, shuttling each packet from server → client and client → server with no further processing. The result is a fully transparent hop that allows an upstream operator to deliver commands to a downstream client without ever interacting with that client directly.

```

command_id = ntohl_0(recv_msg_1->command_id);
cur_list = &second_struct->A_list;
if ( command_id != 0xBF2A0000 )
    cur_list = &second_struct->B_list;
insert_to_list(cur_list, conn_struct);
list_A = second_struct->A_list;
list_B = second_struct->B_list;
if ( list_A && list_B )
{
    pop_first_waiting_node(&second_struct->A_list, second_struct->A_list);
    pop_first_waiting_node(&second_struct->B_list, list_B);
}
_InterlockedDecrement(&second_struct->lpAddend);
RtlLeaveCriticalSection(&second_struct->critical_section);
if ( list_A && list_B )
{
    send_msg.command_id = ntohl(0xC5D0030u);
    send_msg.exec_state = ntohl(*&list_B->net_struct->peer_addr.sa_data[2]);
    send_msg.data_size = ntohl(0);
    v16 = build_packet_and_send(list_A->net_struct, &send_msg, 0xFFFFFFFF);
    if ( !v16 )
    {
        v16 = build_packet_and_send(list_A->net_struct, list_B->recv_msg, 0xFFFFFFFF);
        if ( !v16 )
            v16 = transfer_method_0(list_A->net_struct, list_B->net_struct);
    }
}

```

Relay network logic flow

Backdoor Features

The IIS Listener Module of ShadowPad also includes features of the ShadowPad client, letting the attackers run different commands on the IIS machine. This embedded command set provides the operator with extensive control over the compromised host, enabling everything from reconnaissance to interactive access to payload staging.

The breadth of these command IDs illustrates how ShadowPad's IIS Listener is more than a simple traffic forwarder; it is a self-contained control surface capable of both maintaining the distributed relay network and exerting fine-grained control over any machine running it. This duality is central to Ink Dragon's operational philosophy: relay-capable nodes double as fully functional access points, allowing the operator to maintain stealthy persistence, collect intelligence, deploy additional tooling, and issue high-privilege instructions without ever exposing their true command infrastructure.

ShadowPad's backdoor exposes a broad command surface designed to give operators full, hands-on control of a compromised system. It begins with basic orchestration capabilities, retrieving the full command map, gathering a detailed system snapshot, launching an interactive reverse shell, or cleanly shutting down the implant when needed. From there, the malware offers an extensive file-management layer that can enumerate drives, walk directories, manipulate timestamps, create or remove files and folders, and read, write, move, or download data on

demand. This is complemented by rich process and service control, allowing operators to list and kill processes, inspect loaded modules, and start, stop, or delete Windows services with the same precision seen in legitimate administration tools.

```
files_manager_commands(conn_struct, msg_1);
if ( v6 != -1 )
    goto LABEL_26;
temp_val = ntohl_0(msg_1->command_id) - 0x2C7D0000;
if ( temp_val )
{
    temp_val2 = temp_val - 1;
    if ( temp_val2 )
    {
        if ( temp_val2 != 1 )
            goto LABEL_11;
        v9 = enum_modules(conn_struct, msg_1);
    }
    else
    {
        v9 = terminate_process(conn_struct, msg_1);
    }
}
else
{
    v9 = enum_processes(conn_struct, msg_1);
}
if ( v9 != -1 )
    goto LABEL_26;
LABEL_11:
service_commands(conn_struct, msg_1);
if ( v10 != -1 )
    goto LABEL_26;
network_connections_commands(conn_struct, msg_1);
if ( v11 != -1 || ntohl_0(msg_1->command_id) == 0x2F320000 && pipes_commands(conn_struct, msg_1) != -1 )
    goto LABEL_26;
if ( ntohl_0(msg_1->command_id) == 0xF02D0000 && console_commands(conn_struct, msg_1) != -1 )
    goto LABEL_26;
```

Main commands method flow

ShadowPad also supports multiple execution models, including running commands with output capture, spawning interactive console processes with full screen-buffer access, and maintaining long-lived execution channels through pipes. The networking side is equally capable: the implant can inspect local TCP/UDP tables, alter network entries, and even serve as a pivot point by proxying or tunneling traffic through the infected host. In practice, these capabilities turn the backdoor into a complete remote operations platform that blends system administration, espionage tooling, and covert tunneling into a single, tightly integrated module.

ShadowPad Loaders

ShadowPad is a modular, multi-layered backdoor framework widely [attributed](#) to Chinese state-sponsored threat groups and frequently used in long-term espionage campaigns. ShadowPad allows operators to deploy customized modules for data exfiltration, credential harvesting, and lateral movement.

In Ink Dragon’s operations, we observed numerous ShadowPad deployments following a recurring triad sideloading structure: an executable, a malicious DLL, and an encrypted TMP payload. The legitimate or masqueraded executable loads the malicious DLL, which in turn decrypts and executes the ShadowPad core from the TMP file directly in memory, erasing the payload afterward to minimize forensic artifacts.

A notable detail in Ink Dragon’s ShadowPad loader is that many of the malicious DLLs across different incidents shared the same generic name, `DLL.dll`. The DLL loaders were MFC-based binaries, where the malicious code was heavily obfuscated using [ScatterBrain](#).

```
ModuleHandleW = GetModuleHandleW(0);
AfxGetModuleState();
if ( (*(DWORD *)((char *)ModuleHandleW + 0x2305E) ^ 3879556985) == 0xCE5DC31 )
{
    AfxGetModuleState();
    return (char *)ModuleHandleW + 0x2305E;
}
```

Search for ShadowPad obfuscated code entry point

Once executed, these loaders perform on-the-fly decryption of configuration data and payloads stored in the same directory.

Path	Executable	DLL
C:\Users\Public	vncutil64.exe	vncutil64loc.dll
C:\Program Files\Microsoft\Edge	ApplicationLogs.exe	atiadlxy.dll
C:\Program Files\Microsoft\Edge\Application	msedge_proxyLog.exe	msedge_proxyLogLOC.dll

CDBLoader

One of Ink Dragon’s most distinctive TTPs is leveraging the Microsoft debugger (`cdb.exe`) as an execution host.

Rather than launching a typical EXE, the operator runs a WinDbg/CDB scripted session, such as: `c:\users\public\cdb.exe -cf C:\Users\Public\config.ini -o C:\Users\Public\cdb.exe`.

The shipped `config.ini` is not an innocuous INI file. It contains a sequence of memory-edit / write-bytes commands followed by a change to the instruction pointer (RIP) so execution continues at the attacker-controlled shellcode. The shellcode, in turn, reads an auxiliary file (`wmsetup.log`) from disk, decrypts it with an AES key hard-coded into the shellcode, and loads the real payload into memory for execution. After the payload runs, the debugger instance and helper files are often removed, leaving only transient memory-resident code.

```

bp $exentry
g
.foreach /pS 5 ( register { .dvalloc 1638 } ) { r @$t0 = register };
eb @$t0+00 E9;
eb @$t0+01 4B;
eb @$t0+02 06;
eb @$t0+03 00;
eb @$t0+04 00;
eb @$t0+05 CC;

```

The shellcode script in config.ini

LalsDumper

LalsDumper is Ink Dragon's in-house LSASS extraction chain observed during multiple intrusions. The sequence starts with a small loader (**lals.exe**) that manipulates a companion DLL (**fdp.dll**) in the same folder. The loader locates a placeholder (32 'd' characters) inside **fdp.dll**, overwrites that placeholder with the path to an auxiliary file (**rtu.txt**), and produces a patched DLL named **nfdp.dll**.

Crucially, the loader calls **AddSecurityPackageA** to register the DLL as a Security Support Provider (SSP) so that **lsass.exe** will load it, a technique that forces LSASS itself to load attacker code in-process.

```

while ( placeholder[index] == 'd' )
{
    if ( ++index >= 32 )
    {
        new_path = &_amp;_new_path[v14];
        memset(new_path, 0, 0x100u);
        do
            ++length;
        while ( rtu_txt_path[length] );
        memmove(new_path, rtu_txt_path, length);
        DeleteFileA(nfdp);
        hFile = fopen(nfdp, "wb");
        _hFile = hFile;
        if ( hFile )
        {
            fwrite(_new_path, 1u, size, hFile);
            fclose(_hFile);
            memset(&pOptions, 0, sizeof(pOptions));
            AddSecurityPackageA(nfdp, &pOptions);
            print_debug("success\n");
        }
    }
}

```

The replacement of the placeholder by a real name added as
SSP

The registered DLL reads **rtu.txt**, applies a simple XOR with **0x20** to recover a secondary payload, and then maps that payload into memory. The payload implements a custom **MiniDumpWriteDump**-like routine (not using the Windows API directly) to create a compressed dump of **lsass.exe**, writing a ZIP-like dump file named **<%TEMP%>\<pid>.ddt**.

The chain uses direct syscalls (hashed at runtime) to evade API-based hooking and EDR detection.

```
def hash_syscall(name: str) -> int:
```

```
h = 0xCD7815D6 # constant seed
```

```
for (*WORD) ch in name: # iterate Unicode code-points
```

```
h ^= (ord(ch) & 0xFFFF) + ror32(h, 8)
```

```
return h & 0xFFFFFFFF
```

The DLL exposes a function called **Tom**, which serves as the payload's runtime entry point and orchestrates the loader's core logic: when invoked, **Tom** creates a file in the system TEMP folder named using the victim process ID (for example `%TEMP%\<pid>.ddt`), captures a memory dump of `lsass.exe` and writes that dump into the `.ddt` file using a ZIP-style archive format.

```
hDll = (__int64)hinstDLL;
memset(fileName, 0, 0x208u);
memset(temp_path, 0, 0x208u);
process_id = GetCurrentProcessId();
GetTempPathA(0x207u, temp_path);
swprintf_s(fileName, 0x208u, "%s%d.ddt", temp_path, process_id);
DeleteFileA((LPCSTR)fileName);
hFile = CreateFileA((LPCSTR)fileName, 0x10000000u, 0, 0, 2u, 0x80u, 0);
if ( hFile )
{
    MiniDumpWriteDump_logic(v6, process_id, (__int64)hFile);
    ZwClose();
}
```

LalsDumper's main function

032Loader

032Loader is a sideload-based loader that uses host-specific entropy to decrypt and execute payloads. After being sideloaded by a legitimate executable, the loader patches the host EXE's code flow so that control transfers to its own logic immediately after load.

```

change_protection((__int64)v11, (char *)ModuleHandleW + 0xB261B, 0x40u, &old_protect);
if ( v11[0]
    // mov rax, hook_func
    // call rax
    // nop
    // nop
    || (*(_WORD *))(char *)v6 + 730651) = 0xB848,
    *(_QWORD *)(char *)v6 + 730653) = hook_func,
    *(_DWORD *)(char *)v6 + 730661) = 0x9090D0FF,
    v13[0] = 0,
    change_protection((__int64)v11, v7, old_protect, v13),
    v11[0]) )

```

The patching of the executable by 032Loader

The loader then queries the system's **InstallDate** from the registry and derives a decryption key from that value. It uses the **InstallDate** as the seed for RC4 (or RC4-like) decryption of a third-file blob (the encrypted payload), often accessed via the **SystemFunction032** API to perform the decryption step. Once decrypted, the loader maps the payload via **CreateFileMappingW** / **MapViewOfFile** and adjusts memory protections with **VirtualProtect**, then transfers execution to the in-memory payload.

FinalDraft: New Version

FinalDraft is a well-engineered, full-featured remote administration tool with the ability to accept add-on modules that extend functionality and proxy network traffic internally, mostly by abusing Outlook mail service via the Microsoft Graph API. The samples and behavior we analyzed are consistent with previous public write-ups, but Ink Dragon's deployments show continued feature expansion and careful operational tuning to reduce telemetry and maximize resilience.

FinalDraft begins by locating and decrypting its configuration blob. The configuration is XORed with either a hash derived from the host **ProductId** (making the config per-machine) or the hash of a hardcoded string.

```

while ( p_key != (char *) (key_data + key.size) )
    key_hash = *p_key++ + 31 * key_hash;
_key_hash = key_hash;
dec_conf = config;
for ( i = 0; i < 0x14BE; ++i )
{
    result = i & 7;
    dec_conf->refresh_token[i] ^= *((_BYTE *)&_key_hash + result);
}

```

The Decryption of FinalDraft's configuration

Once decrypted, the config exposes a unique GUID for the implant, the preferred communication transport, an AES key for message encryption, C2 endpoints and refresh tokens, and other operational metadata.

00000000 struct __fixed config_structure

```

00000000 {
00000000 char refresh_token[5000];
00001388 char backup_refresh_token_url[200];
00001450 char guid[36];
00001474 int unknown;
00001478 __int16 build_id;
0000147A int sleep_val;
0000147E char communication_method;
0000147F char aes_key[16];
0000148F char external_ip_or_not;
00001490 char self_remove_flag;
00001491 char exit_byte __bin;
00001492 __int64 unknown;
0000149A int interval_val;
0000149E int repeat_val;
000014A2 int[7] time_list;
000014B2 __int64 time1;
000014BA };

```

The most commonly used transport we observed is **COutlookTrans**, which leverages the Microsoft Graph API (the token endpoint at <https://login.microsoftonline.com/common/oauth2/token>) to hide command-and-control traffic inside legitimate cloud mail flows. FinalDraft uses a refresh token embedded in its configuration to acquire OAuth access tokens, which it stores locally under [HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UUID\](#) for later use. Commands and responses are exchanged as email drafts (messages whose subject starts with [r_<session_id>](#) and [p_<session_id>](#)). The payloads inside those messages are base64-encoded, AES-encrypted (with the config key), and compressed, which helps the operator hide instructions inside otherwise normal mailbox data and bypass network filtering that whitelists Microsoft cloud endpoints.

```

copy_buffer(&v31, "/v1.0/me/MailFolders/drafts/messages?$filter=Subject eq 'r_%llu'&$top=5", 0x47u);
sprintf_wrapper(v46, (__int64)&v31, session_id);
memset(&components, 0, 0xC8u);
if ( 0x7FFFFFFFFFFFFFFFFLL - v47 < 0x1B )
    std::vector<void *>::_Xlen(v47);
Src = v46;
if ( v48 > 0xF )
    Src = (void **)v46[0];
build_prefix_suffix(v39, v9, v10, "https://graph.microsoft.com", 0x1Bu, Src, v47);

```

The message FinalDraft sends to get commands

FinalDraft implements a modular command framework in which operators push encoded command documents to the victim's mailbox, and the implant pulls, decrypts, and executes them. Beyond the standard toolkit (process/service control, file/registry access, tunneling, and file transfer), the Ink Dragon variants we analyzed introduce several extensions that increase stealth, operational control, and data-theft efficiency.

Flexible Beacon Scheduling

The malware supports highly granular callback controls, allowing operators to define daily beaoning windows or specify an explicit list of times when the implant will attempt to reach its C2. This schedule-based design helps blend command traffic into predictable operational patterns.

RDP Activity Harvesting

DumpRDPHistory extracts evidence of both outbound and inbound RDP usage.

1. **RDP OUT (Outbound RDP targets)** – MRU-like entries in the user registry hives that record the servers the user has connected to (the built-in RDP client's Recent Server list). For each entry, the tool reads **UserNameHint** and the registry key's last write time to produce lines like: **<ServerName>:<UserNameHint>:<Timestamp>**. This data is queried from **HKCU\SOFTWARE\Microsoft\Terminal Server Client\Servers\<ServerName>**
2. **RDP IN (Inbound RDP sessions)** – Event log entries from the Terminal Services Local Session Manager operational channel that record session connect/disconnect events. The tool queries Windows Event Log (via **EvtQuery** / Get-WinEvent) for Event ID **21** (Remote Desktop Services: Session logon succeeded) and **25** (disconnect/other session events depending on Windows version). From these, it extracts records like: **<UserName>:<IP>:<TimeStamp>**.

Security Control Downgrades

Several commands intentionally weaken Windows security posture to assist persistence and lateral movement:

DisableRestrictedAdmin – makes RDP connections authenticate with reusable credentials or delegated tokens, enabling credential hopping instead of “restricted” mode authentication. It makes RDP more attractive for lateral movement because an attacker who already controls an endpoint can initiate Restricted Admin connections using local credentials or stolen hashes to hop to other systems without needing cleartext passwords.

```
if ( IsUserAnAdmin() )
{
    v3 = RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SYSTEM\\CurrentControlSet\\Control\\Lsa", 0, 2u, &hKey);
    if ( !v3 )
    {
        *(_DWORD *)Data = 0;
        v3 = RegSetValueExA(hKey, "DisableRestrictedAdmin", 0, 4u, Data, 4u);
        RegCloseKey(hKey);
        if ( !v3 )
        {
            v2 = 3;
            goto LABEL_9;
        }
    }
}
```

Enabling Restricted Admin

- **DisableTokenFiltering** – disables remote token filtering so local admin accounts receive full, not restricted, admin tokens over the network. This allows attackers to use reused or harvested local credentials to perform truly administrative remote actions without requiring domain credentials.
- **EnableDSRMAdmin** – allows the powerful DSRM recovery account to log in even when the DC is not in Directory Services Restore Mode. It gives an attacker a powerful, low-audited way to gain full local control of a DC using an otherwise seldom-monitored credential or hash.

```
*(_DWORD *)Data = 2;
Src = 0;
v20 = 0;
v21 = 15;
LOBYTE(Src) = 0;
v3 = 4;
wcscpy(SubKey, L"System\\CurrentControlSet\\Control\\Lsa");
wcscpy(ValueName, L"DSRMAdminLogonBehavior");
if ( IsUserAnAdmin() )
{
    v4 = RegOpenKeyExW(HKEY_LOCAL_MACHINE, SubKey, 0, 2u, &hKey);
    if ( !v4 )
    {
        v13 = v2 ? RegSetValueExW(hKey, ValueName, 0, 4u, Data, 4u) : RegDeleteValueW(hKey, ValueName);
        v4 = v13;
        RegCloseKey(hKey);
        if ( !v4 )
        {
            v3 = 3;
            goto LABEL_9;
        }
    }
}
```

Setting the DSRMAdminLogonBehavior registry key to 2

DisableRunAsPPL – removes Process Protection Level safeguards, enabling injection and tampering with protected processes such as AV/EDR components. Adversaries do this to defeat endpoint defenses (allowing in-memory loaders, hooking, or live patching of AV and EDR components), to persist by replacing signed binaries, or to extract secrets from processes that would normally be protected.

High-Throughput Exfiltration

BackgroundFileTransfer introduces a dedicated asynchronous worker for large-scale exfiltration. It streams data in sizeable chunks, reports progress in the background, and minimizes the number of required outbound connections. This is ideal for moving large archives over cloud-proxied channels without attracting attention.

Host Profiling & Inventory

Ink Dragon consolidates multiple reconnaissance routines such as system fingerprinting, adapter enumeration, and installed software collection into a unified host profile. This captures IP configuration details (similar to `ipconfig /all`), network adapter metadata, installed applications (from HKLM and HKCU uninstall keys), system identifiers (MachineGuid, ProductId, InstallDate, and System uptime), and other indicators useful for victim classification and targeting.

```
aConnectionSpec db '    Connection-specific DNS Suffix . : %ws',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+30F↑o
                                ; sub_7FFD911C1210+775↑o
                                align 20h
aMediaStateMedi db '    Media State . . . . . : Media disconnected',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+277↑o
                                align 20h
aPhysicalAddress db '    Physical Address. . . . . : %s',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+442↑o
                                ; sub_7FFD911C1210+8A8↑o
                                align 10h
aDescriptionWs db '    Description . . . . . : %ws',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+3A7↑o
                                ; sub_7FFD911C1210+80D↑o
                                align 20h
aLinkLocalIpv6A db '    Link-local IPv6 Address . . . . . : %s',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+9F3↑o
                                align 10h
aDhcpEnabledS db '    DHCP Enabled. . . . . : %s',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+4DC↑o
                                ; sub_7FFD911C1210+942↑o
                                align 20h
aIpv4AddressS db '    IPv4 Address. . . . . : %s',0Ah,0
                                ; DATA XREF: sub_7FFD911C1210+B35↑o
                                align 10h
aIpv6AddressS db '    IPv6 Address. . . . . : %s',0Ah,0
```

IP Configuration data strings in FinalDraft

Victimology

Ink Dragon's targeting patterns show a consistent emphasis on government organizations, but beyond that shared characteristic, there are no clear indicators of how the actor selects its victims. Most affected organizations appear to serve a specific operational purpose for the actor rather than reflecting broad, industry-wide targeting.

Geographically, Ink Dragon has focused heavily on government entities in Southeast Asia and Africa, and in recent months has expanded its activity into Europe. Since the ToolShell exploitation wave in July, the actor has steadily increased its operations in the region, with a growing concentration on European government-sector targets. A key aspect of Ink Dragon's tradecraft is its use of compromised organizations as C2 relay nodes. As a result, we have observed European victims being leveraged to launch activity not only against additional European institutions, but also against targets in Africa and Southeast Asia.

Overlap With RudePanda / REF3927 Activity

During our investigation, we also observed evidence of a second intrusion set commonly tracked as RudePanda / REF3927 on several of the same victim environments compromised by Ink Dragon. While we do not assess these activity clusters to be operationally linked, the victimology overlap is notable: both actors exploited the *same* internet-facing server vulnerability to gain footholds in identical organizations and systems.

Upon gaining access, RudePanda relied on its customary toolset, which included:

- **Godzilla-derived webshells** used to execute in-memory .NET payloads and deploy secondary components.
- **Malicious IIS modules** belonging to the **TOLLBOOTH** family embed command-and-control logic directly inside the web server's processing pipeline. Several modules contained PDB references to **dongtai**, consistent with tooling previously [documented](#) in public reporting.
- **Configuration tampering** of IIS's **applicationHost.config**, where a simple diff against previous snapshots exposes the unauthorized module insertions.



```
<add name="ConfigurationValidationModule" image=
"%windir%\System32\inetsrv\validcfg.dll" />
<add name="RewriteModule" image=
"%SystemRoot%\system32\inetsrv\rewrite.dll" />

</globalModules>
```

```
1 <add name="ConfigurationValidationModule" image=
2 "%windir%\System32\inetsrv\validcfg.dll" />
3 <add name="RewriteModule" image=
4 "%SystemRoot%\system32\inetsrv\rewrite.dll" />
5 <add name="ScriptsModule" image="%C:\Windows\system32\inetsrv\scripts.dll"
preCondition="bitness32" />
6 <add name="IsapiCachesModule" image=
7 "%C:\Windows\system32\inetsrv\caches.dll" preCondition="bitness64" />
8
9 </globalModules>
```

In addition, the actor deployed a kernel-mode rootkit driver, **wingtb.sys** (**2965ddbcd11a08a3ca159af187ef754c**), a modified and signed variant of **Winkbj.sys**. Its installation uses an INF file (**Hidden.inf**) naming the service "Wingtb." The driver is derived from the open-source "[Hidden](#)" rootkit project, used to conceal files, processes, and registry entries after being activated on the system.

Finally, several [.lnk](#) artifacts referencing the operator's backdoor kit that is consistent with the GoToHTTP tooling previously documented by both [HarfangLab](#) and [Elastic](#) were recovered on disk. Although the original payloads were deleted, these link files served as additional indicators of RudePanda's presence.

Conclusion

Ink Dragon's operations illustrate a shift in how modern espionage clusters weaponize compromised infrastructure. Rather than treating each victim as an endpoint to be monitored or harvested, the group systematically folds every breached perimeter server into a distributed ShadowPad relay fabric. The IIS Listener Module sits at the center of this strategy: a stealthy, low-friction component that binds hidden URL prefixes, intercepts traffic without disrupting legitimate services, and quietly links victims together into a multi-hop communication grid.

This architecture transforms the threat landscape in several ways. First, it grants the operators resilient command paths that do not rely on fixed infrastructure. Even if a downstream implant has no internet reachability or if upstream servers are blocked, the attacker can simply reroute through other victims. Second, it provides natural operational camouflage. Traffic tunneled between unrelated government or enterprise networks appears outwardly benign and often blends into the expected profile of inter-organizational HTTP flow. Third, it enables strategic re-use of compromised assets: once a host is enrolled as a ShadowPad relay, it becomes an evergreen pivot point that continues to serve campaigns long after the initial intrusion.

The rest of the kill chain reinforces this design philosophy. Mature sideloading patterns, modular loader triads, memory-resident payloads, compartmentalized privilege escalation, and disciplined credential harvesting all serve one purpose: establish durable, high-privilege access long enough to turn the victim into reliable infrastructure. Whether deploying FinalDraft, staging ShadowPad loaders, or extracting domain-wide secrets, each action feeds back into the broader communication mesh that Ink Dragon relies on to maintain persistence across environments.

Taken together, Ink Dragon presents a threat model in which the boundary between "compromised host" and "command infrastructure" no longer exists. Each foothold becomes a node in a larger, operator-controlled network – a living mesh that grows stronger with every additional victim. Defenders must therefore view intrusions not only as local breaches but as potential links in an external, attacker-managed ecosystem, where shutting down a single node is insufficient unless the entire relay chain is identified and dismantled. Ink Dragon's relay-centric architecture is among the more mature uses of ShadowPad observed to date. A blueprint for long-term, multi-organizational access built on the victims themselves.

IOCs

2e84ea5cef8a9a8a60c7553b5878a349a037cffeab4c7f40da5d0873ede7ff72 - dongtai module

e2f6e722c26e19b76396c2502cacf2aaceaaa1486865578c665ebf0065641ffa - dongtai module
f9dd0b57a5c133ca0c4cab3cca1ac8debd4a798b452167a1e5af78653af00c1 - Wingtb.sys
a86e72ca58de6d215a59ae233963eaea27fe47ef0c9f43938e27339df4a86732 - 032Loader
7efe5c1229178c1b48f6750c846575e7f48d17ea817997bd7acba0e5ecf1e577 - 032Loader
D88115113E274071B03A3B4C1DA99EAEA7B8D94ADF833DFD26943AF0A6D78B4D -
032Loader
f094ff83d4b7d06bc17b15db7d7dc0e622778b0eda71e8fc9fdf7db83c460426 - nfdp.dll
36f00887f6c0af63ef3c70a60a540c64040b13a4209b975e96ce239e65548d4a - fdp.dll
ecf0fbd72aac684b03930ad2ff9cdd386e9c13ddf449f27918f337dc8963590e - LalsDumper
2b57deb1f6f7d5448464b88bd96b47c5e2bd6e1c64c1b9214b57c4d35a591279 - LalsDumper
b4a53f117722fb4af0a64d30ec8aa4c4c82f456e3d2a5c5111c63ce261f3b547 - ShadowPad Loader
866fde351251092fb5532e743459ba80968cd5516cce813c8755467f5e8a47a1 - ShadowPad
Loader
188ab2d68f17ecf08a7a4cfc6457c79b0a5117b3277352a7371a525416129114 - ShadowPad
Loader
809ddcbb64d6f2ccc4a8909068da60e6ea8b3ebd9c09dd826def0e188c7a2da2 - config.ini
f438ca355e6888c4c9cd7287b22cfe5773992ef83f0b16e72fb9ae239d85586c - FinalDraft
c305b3b3f9426d024cdd262497a5d196264397bfed445705759d0a793a58fe6e - Encrypted
FinalDraft

[BACK TO ALL POSTS](#)