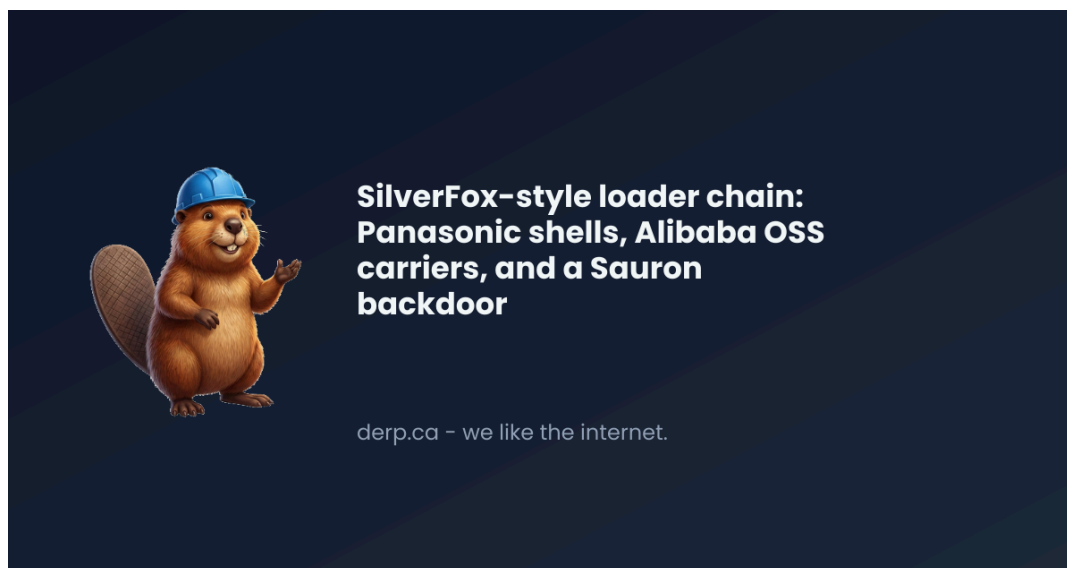


# SilverFox-style loader chain: Panasonic shells, Alibaba OSS carriers, and a Sauron backdoor

 [derp.ca/research/silverfox-panasonic-sauron-loader-chain](https://derp.ca/research/silverfox-panasonic-sauron-loader-chain)

Kirk

June 21, 2026



`ainstaller-86533005.exe` is wearing a Panasonic jacket, and the fit is almost too good.

Inspect the file metadata and everything points at `PPcNotif.Provider.RequiredApp.exe`, Panasonic PC Notification, version 1.10510.0.0. Spin it up and the visible application code is coherent. There is command dispatch, notification IPC, GUI startup, Panasonic device logic, and a mutex for the app. If you stop at the front door, you see legitimate OEM-shaped software.

The malicious path starts in the basement before the Panasonic dispatcher gets the wheel. On execution, the binary allocates a new RWX buffer, copies an encoded stage from its own image, decodes it twice, peels a heap payload with a 19-byte XOR key, and jumps into the loader.

From there, the sample becomes a stack of borrowed coats: Alibaba OSS staging, image-named encrypted carriers, signed side-load hosts, protected DLLs, an RPC Task Scheduler module, a sparse bridge PE, a Philips-branded side-load cluster, and a final `rundll32.dat`!Edge export that implements Sauron service behavior.

The sample source for this writeup is the public [Triage task 260619-v47g4ah12k \(opens in new tab\)](#). The first family clues were thin: Joe Sandbox produced a Gh0stRAT-family signature hit late in the chain, and a VirusTotal community comment from LingGao tied `ainstaller-86533005.exe` to SilverFox. Those clues are pivots. The chain still has to stand on bytes.

## Sample context

| Indicator type   | Value                                                                         | Notes                                                     |
|------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------|
| Sample source    | <a href="#">Triage 260619-v47g4ah12k (opens in new tab)</a>                   | Public source for the sample and Triage runtime artifacts |
| Submitted file   | <code>ainstaller-86533005.exe</code>                                          | PE32+ Windows GUI executable                              |
| SHA-256          | <code>200d9f5d041c870e69f73da32c753a69096742c33e2dab65299767ac31578267</code> | Outer executable                                          |
| Visible identity | <code>PPcNotif.Provider.RequiredApp.exe</code>                                | Panasonic PC Notification                                 |

| Indicator type              |  | Value                                                                  | Notes                                                                                     |
|-----------------------------|--|------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Version                     |  | 1.10510.0.0                                                            | File metadata                                                                             |
| Entry point                 |  | 0x140034998                                                            | Outer PE entry                                                                            |
| Compile timestamp           |  | 2025-03-18 03:20:54 UTC                                                | PE timestamp                                                                              |
| Reported delivery pivot     |  | hxxps://seeo-pc[.]com[.]cn -><br>hxxps://www[.]hed6g8u[.]com/installld | Community-provided context                                                                |
| First reconstructed OSS URL |  | hxxps://jun616[.]oss-cn-beijing[.]aliyuncs[.]com/tad                   | Recovered from static code and marker bytes                                               |
| Later OSS host              |  | 26nn[.]oss-cn-hangzhou[.]aliyuncs[.]com                                | Bridge-correlated later stage                                                             |
| Runtime-loaded config       |  | 47[.]239[.]170[.]54, gqsqq[.]net                                       | Triage-retained mapped rundll32.dat image                                                 |
| Sandbox-report leads        |  | 47[.]238[.]95[.]170:7089, /f.dat, F0M-50.jpg to F0M-53.jpg             | Reported indicators awaiting matching bytes                                               |
| Core reversing tools        |  | Ghidra, r2/radare2                                                     | Static file and memory-image reverse engineering                                          |
| Behavioral references       |  | Triage, Joe Sandbox                                                    | Triage for runtime data; Joe Sandbox for additional behavioral and family-style reference |

## Why the SilverFox-style label fits

External reporting puts this sample near SilverFox/Winos tooling while leaving operator attribution unresolved.

[Huorong's September 2024 SilverFox variant report \(opens in new tab\)](#) is the strongest public overlap. Their chain describes Alibaba OSS-hosted a.gif, b.gif, c.gif, d.gif, s.dat, and s.jpg, a five-byte EOF trailer format, TrueSightKiller driver material, RPC Task Scheduler creation through NdrClientCall3, later drops.jpg staging, 360 security product checks, and a final backdoor export named Edge.

[Picus's medical-software Silver Fox report \(opens in new tab\)](#) gives a second public neighbor: trojanized Philips DICOM viewer material, Alibaba OSS staging, image-named encrypted payloads, TrueSightKiller, RPC task scheduling through \pipe\atsvc, and later F0M-\* module downloads.

[ThreatBook's WinOS ecosystem report \(opens in new tab\)](#) frames SilverFox as a malware-as-a-service toolkit ecosystem with fuzzy actor boundaries. [Fortinet's Winos4.0 analysis \(opens in new tab\)](#) describes Winos4.0 as a modular framework rebuilt from Gh0stRAT and used in Silver Fox campaigns. [MITRE ATT&CK \(opens in new tab\)](#) tracks Gh0st RAT as public remote-access tool code used by multiple groups.

The Gh0stRAT-family sandbox hit has value as lineage gravity. The loader grammar carries the weight: Alibaba OSS, image-named carriers, five-byte trailer extraction, signed side-load hosts, AV-prep material, RPC task creation, and a late Edge backdoor stage.

## Chain overview

Static chain:

ainstaller-86533005.exe

```

-> staged buffer at 0x140086f2e
-> decoded heap payload
-> hxtps://jun616[.]oss-cn-beijing[.]aliyuncs[.]com/tad
-> a.gif / b.gif / c.gif / d.gif / s.dat / s.jpg
-> signed UxEnhanceHost + UxEnhance64.dll + msadox.tb + adoresd.dat
-> msadox.tb RC4 shellcode
-> adoresd.dat trailer-XOR adoresd.dll
-> adoresd.dll!DelegateEnSerialization
-> populated Ux bridge module

```

Runtime-ordered continuation, with static review of each artifact:

populated Ux bridge module

```

-> DIIQf1.exe + XPSLOG.dll + image.png + thumbs.db
-> image.dll prep/dropper stage
-> rundll32.dat!Edge / Sauron

```

Chain summary:

1. The Panasonic host is the wrapper.
2. The loader decodes itself into executable heap memory.
3. Static code reconstructs an Alibaba OSS staging URL.
4. The remote objects are image-named encrypted carriers.
5. s.jpg contains a compressed VirtuOne RPC scheduled-task DLL.
6. a.gif and b.gif form a signed Tencent Ux side-load pair.
7. msadox.tb and adoresd.dat carry shellcode and a sparse bridge DLL.
8. The populated bridge is tied to a later 26nn / drops pivot.
9. Runtime order then reaches a Philips side-load cluster.
10. image.png decodes to a protected prep/dropper DLL.
11. thumbs.db decodes to rundll32.dat, whose Edge export implements Sauron behavior.

## Phase 1: Panasonic shell, staged loader

---

The visible Panasonic application code is a plausible host:

- Command dispatcher at 0x14001ce90
- Handlers for -isMatchStaticRequirement, -register, and -unregister
- GUI startup at 0x14001ca00
- Mutex PanasonicPcNotification\_ProviderRequiredApp\_MultipleInstancePreventer\_MutexName
- Named pipe prefix \\.\pipe\PanasonicPcNotification.Pipe.Request.
- Panasonic device checks through SetupAPI and device IOCTLs

The malicious loader starts before that visible dispatcher.

At 0x1400038e6, the executable calls:

```
VirtualAlloc(NULL, 0xce59, 0x1000, 0x40)
```

It copies 0xce59 bytes from VA 0x140086f2e, then jumps into the allocated buffer. The staged buffer goes through two decoder layers. The second decoder starts with key 0x38, walks 0xce0d bytes, XORs each byte, and feeds the decoded byte back into the key.

The fully decoded stage then copies 0xae05 bytes from offset 0x2d and XOR-decodes them with:

```
4@e!c!bSL2AeimnwYD4x
```

That yields the heap payload.

**Recovered payload**

**SHA-256**

Decoded heap payload 679921a7373fc0a77d60416e3f7ce4454dfd394c12d9c0387be9538b8f030e02

The heap payload brings the real loader behavior:

- PEB/export walking for dynamic API resolution
- Timing and environment checks through QueryPerformanceCounter, Sleep, rdtsc, VirtualAllocExNuma, mutex probes, process checks, heap pressure, and COM/object probes
- ntdll.dll!NtTraceEvent patched with a single 0xc3 byte
- Defender exclusion commands launched through ShellExecuteA and powershell.exe
- WinINet download logic through InternetOpenA, InternetOpenUrlA, InternetReadFile, and InternetCloseHandle
- Elevated relaunch helper using ShellExecuteExA with verb runas
- Hidden VBS helper named checktime.vbs under a generated \Users\Public\ subdirectory

That is the first split between costume and chain. The Panasonic app gives the sample a respectable front door. The staged loader has already moved through the basement.

## Phase 2: `jun616` and the image carriers

The first staging URL is built from static code and marker bytes.

A configuration helper reads the current executable, searches for marker \*/& at file offset 0x410, copies bytes after the marker, and feeds encoded fragments through a Caesar-style decoder:

```
ykkgj://          -> hxxps://
ale616            -> jun616
.fjj-te-svzazex.rczpletj.tfd/ -> .oss-cn-beijing[.]aliyuncs[.]com/
kru               -> tad
```

The reconstructed URL is:

```
hxxps://jun616[.]oss-cn-beijing[.]aliyuncs[.]com/tad
```

That tad object acts as the shelf index for the remote payload set.

| Source object | Decoded output                     | Role                                              |
|---------------|------------------------------------|---------------------------------------------------|
| tad           | tad_dec.bin                        | Object URL and output-name index                  |
| a.gif         | generated_exe_unpatched.bin        | Signed Tencent UxEnhanceHost side-load host       |
| b.gif         | UxEnhance64.dll                    | Unsigned protected side-loaded DLL                |
| c.gif         | msadox.tb                          | SQLite-looking opaque loader material             |
| d.gif         | adoresd.dat                        | PNG carrier for sparse bridge PE                  |
| s.dat         | ranchserv.jpg                      | Signed TrueSight anti-rootkit driver material     |
| s.jpg         | s_jpg_memory_payload_unpatched.bin | x64 shellcode loader with compressed VirtuOne DLL |

The image extensions are camouflage. The recovered code treats these files as opaque byte carriers. JPEG markers, PNG chunks, EXIF, DCT work, and LSB games are noise here. The tail matters.

The carrier grammar is simple and weirdly durable:

```
[payload_start_dword_le][xor_seed_byte]
```

The four bytes before EOF give the payload start offset. The final byte gives the XOR seed. The decoder XORs from payload\_start to EOF-5, incrementing the key for each byte.

This grammar shows up again later in image.png, thumbs.db, and drops.jpg. The actors can repaint file names all day; this five-byte tail keeps squeaking.

### Phase 3: `s.jpg` and RPC Task Scheduler creation

The decoded `s.jpg` file is shellcode carrying VirtuOne metadata and an aPLib-compressed block. Decompression recovers:

| Artifact | Value                                                            |
|----------|------------------------------------------------------------------|
| Module   | RPCScheduleTask.dll                                              |
| SHA-256  | a1e68e7adc5597eae23a6590f1ac345f796abebceb91afcc0f2dd97cc0622740 |
| Type     | Unsigned x64 DLL                                                 |
| Export   | VirtuOne                                                         |

VirtuOne builds a local Task Scheduler RPC binding:

| Field              | Value                                |
|--------------------|--------------------------------------|
| RPC interface GUID | 86D35949-83C9-4044-B424-DB363231FD0C |
| Protocol           | ncacn_np                             |
| Endpoint           | \pipe\atsvc                          |
| Network address    | localhost                            |
| RPC call           | NdrClientCall3                       |

The DLL contains Task Scheduler XML with a logon trigger, a one-minute repeating time trigger, group principal S-1-5-32-545, HighestAvailable run level, a hidden task setting, and command/working-directory fields supplied by the caller.

Caller-side code in the heap payload maps the `s.jpg` payload, receives a function pointer, converts generated ASCII strings to UTF-16, and calls the recovered function pointer with wide-string arguments.

The fixed task machinery is inside `RPCScheduleTask.dll`. The concrete task name, command, working directory, and arguments are caller-generated values.

This overlap is loud. Huorong documented the same `s.jpg` pattern: shellcode, compressed DLL, RPC Task Scheduler logic, and `NdrClientCall3`.

### Phase 4: Ux side-loading and the `adoresd.dll` bridge

The first side-load bundle comes from the `a.gif` through `d.gif` set.

| Runtime path                           | SHA-256                                                          | Static tie                                        |
|----------------------------------------|------------------------------------------------------------------|---------------------------------------------------|
| C:\Users\Public\Y5EiLf\6322eI.exe      | 676a2a7b94ca2f8ec76352ee656e4d075bb342bd7ad6efbc7c19c060001eace7 | Byte-identical to decoded <code>a.gif</code> host |
| C:\Users\Public\Y5EiLf\UxEnhance64.dll | f93906aa27621dfcc20d3a082df2e0b198fdcd1c5b0841310870f3f8a0aeb768 | Runtime-mutated decoded <code>b.gif</code>        |

| Runtime path                       | SHA-256                                                          | Static tie                                                   |
|------------------------------------|------------------------------------------------------------------|--------------------------------------------------------------|
| C:\Users\Public\Y5EiLf\msadox.tb   | eeccbaa9d8ff3f2839eb326cd0ad17661b334b58fb715a674c2ed3e3f1071027 | Runtime-mutated decoded c.gif                                |
| C:\Users\Public\Y5EiLf\adoresd.dat | a01c5cc7081354957baad4ed24719f3c1de9c3e444bb537c66d74f201e05ad9e | Runtime-mutated decoded d.gif; decodes to the same bridge PE |

The host is a signed Tencent UxEnhanceHost executable that imports these UxEnhance64.dll exports:

?EnableUxEnhance@@YA\_NXZ  
?DisableUxEnhance@@YA\_NXZ

Both exports resolve to the same protected entry.

Loaded UxEnhance64.dll memory closes the bridge. The native attach path gates on `edx == 1`, then builds:

RC4 key: 3d e2 3f 7a d2 83 a1 f7  
Marker: bb e6 36 23 33 96 3b 89

It reads sibling msadox.tb, searches for the marker, copies the bytes after it, initializes RC4, and decrypts the tail.

| Recovered shellcode | Value                                                             |
|---------------------|-------------------------------------------------------------------|
| Marker offset       | 0x94                                                              |
| Size                | 8197 bytes                                                        |
| SHA-256             | 56c5bd8df62c54be51312fabcd86ebe9cde896ed015b91cf3e0aa7c941f49d096 |
| Entry pattern       | e9 10 01 00 00                                                    |

That shellcode opens adoresd.dat, reads the five-byte trailer, and applies the same incrementing-XOR carrier decode. Runtime trailer bytes:

d0 05 00 00 f8

That gives payload start 0x5d0 and seed 0xf8.

The decoded payload is a large sparse PE:

| Artifact   | Value                                                            |
|------------|------------------------------------------------------------------|
| Decoded PE | adoresd.dll                                                      |
| SHA-256    | aaa13e34e7c45dc49552e843ee8b611b2373959d8a7cc23265abc353d76235ba |
| Export     | DelegateEnSerialization                                          |

| Artifact    | Value       |
|-------------|-------------|
| Image base  | 0x180000000 |
| SizeOfImage | 0x8be000    |

The shellcode manually maps `adored.dll`, resolves `DelegateEnSerialization`, and calls it with a context pointer.

Then the bridge performs its best haunted-building routine. Earlier memory snapshots show a sparse mapped image with a protected `.bTM` section. Later snapshots show the same image with formerly empty sections populated while `.bTM` remains stable. The bridge is the chain's staging cathedral: lots of empty space, one locked room, and then every pew fills with code.

## Phase 5: the populated bridge and `26nn`

The populated Ux bridge image contains the next major transition:

- Registry marker strings: `Software\0DBDC, revbs`
- Defender exclusion scheduled-task template for task name `Task1`
- Follow-on OSS fragments: `hxxps://26nn[.]oss, -cn-hangzhou[.]ali, [.]yuncs[.]com/drops`
- Code-referenced `GetData` user-agent material

The Triage-retained packet capture confirms TLS SNI for both OSS hosts:

```
jun616[.]oss-cn-beijing[.]aliyuncs[.]com
26nn[.]oss-cn-hangzhou[.]aliyuncs[.]com
```

The Triage capture gives host-level TLS SNI. Cleartext object paths are absent from that capture. Triage runtime summaries record the later request:

```
hxxps://26nn[.]oss-cn-hangzhou[.]aliyuncs[.]com/drops.jpg
User-Agent: GetData
```

The saved Triage PCAP-derived `/drops.jpg` object is available and decodes cleanly with the same five-byte trailer grammar.

| Artifact                | Value                                                            |
|-------------------------|------------------------------------------------------------------|
| Saved object SHA-256    | 667698da115bb3cc322a972dd8fb15280f0e481d5a87779f4c444f535bf17a74 |
| Trailer                 | d0 05 00 00 6a                                                   |
| Start / seed            | 0x5d0 / 0x6a                                                     |
| Decoded payload SHA-256 | 1049a21e9dd6b26b0574c132b12b4565e4bf7096c31f8a21dbe45e5e3fbddc1  |

Decoded `drops.jpg` is x64 shellcode-style material with:

- `kernel32.dll`
- `runas`
- `.lnk`
- `.dat`
- `.exe`
- `SOFTWARE\JDBCC`
- `GetData`
- `360tray.exe`
- `360sd.exe`
- `360safe.exe`
- `Q360SafeMonClass`
- `CreateToolhelp32Snapshot`
- `Process32First`

- Process32Next
- MessageBoxA

A separate Triage-retained mapped PE32+ DLL from the same later phase has SHA-256 12845df5f905fe5345829b2aca546ba8eecbd84f1a2924993b076a5d33bd1307 and contains additional 360 security-tool strings such as 360Tray.exe, 360Safe.exe, LiveUpdate360.exe, and 360sd.exe. The byte tie or loader path from decoded drops.jpg shellcode to that mapped PE remains an evidence need, so we keep it scoped as a separate later-phase artifact.

The later phase has a familiar SilverFox flavor: 360 checks, more shellcode staging, and another cloud object pretending to be an image.

## Phase 6: Philips side-load cluster

Runtime order places a new side-load cluster after the populated Ux bridge:

```
C:\Program Files (x86)\D1IQf1\D1IQf1.exe
C:\Program Files (x86)\D1IQf1\XPSPLOG.dll
C:\Program Files (x86)\D1IQf1\image.png
C:\Program Files (x86)\D1IQf1\thumbs.db
```

Artifacts:

| Artifact                      | SHA-256                                                          | Role                                                 |
|-------------------------------|------------------------------------------------------------------|------------------------------------------------------|
| D1IQf1.exe                    | 6d6ba2bc9ad414837826f7278bc3e0116f1aeda02d0c2284ed65819f5d9180a8 | Signed Philips/Speech Processing Solutions PE32 host |
| XPSPLOG.dll                   | 3317c1e3471de643c7868c7e0821524b0bfdeb4a3d335e1ec5431a9088234ab7 | Protected side-loaded PE32 DLL                       |
| Rebuilt populated XPSPLOG.dll | 63d8b720e35bc8eafc40c37194c7fce270be2b862e04a5278c4aa03d2907b25b | Runtime-populated image for review                   |
| image.png                     | 06d9687f2d7a462d55480037bf1eb0fce9d42d9f56d950695b9d6a2f52241188 | Carrier for image.dll                                |
| thumbs.db                     | 75535087898c38d5da3c16282d008d68df648ae58a3cf8b632a953302544279c | Carrier for rundll32.dat                             |

D1IQf1.exe imports XPSPLOG.dll ordinals 2, 3, 4, 6, 7, and 8. r2 xrefs show direct calls to all imported ordinals plus additional indirect calls through loaded thunks.

The protected handoff through XPSPLOG remains bounded at instruction level. The D1 cluster artifacts and their decoded internals are confirmed.

### `image.png` to `image.dll`

The image.png carrier uses the same five-byte trailer grammar:

```
trailer: d0 05 00 00 6e
start:   0x5d0
seed:    0x6e
```

It decodes to:

```
image.dll
SHA-256: 96aee41363c08acf6a6230d1dda5f7e7dae9f78952993b70a9c8c4949c81bb22
Export: PhilipsCoInitialize
```

A rebuilt populated image.dll snapshot shows the prep/dropper stage:

- Defender Task1 scheduled-task template

- Windows Update disablement body
- Add-MpPreference -ExclusionPath 'C:\ProgramData','C:\Users','C:\Program Files (x86)' -Force
- icacls command builders
- Full D1 bundle path table for D1IQf1.exe, image.png, thumbs.db, and XPSPL0G.dll
- File-table writer for the D1 bundle artifacts
- Path check/decode/write path for C:\Windows\Temp\ranchserv.jpg

Triage-retained mapped-image snapshots show image.dll materializing in phases. Snapshot A to B populates .text, .rdata, .data, and .az\* while .k]\ remains unchanged. Snapshot B to C materializes the D1 path table. Snapshot C to D changes one byte in .data. The stable strings stay stable. The protected .k]\ section acts like the dispatcher/materializer room.

The ranchserv.jpg material comes from the earlier s.dat decode and is a signed TrueSight anti-rootkit driver. File materialization is confirmed. Driver execution remains an open evidence need pending a concrete service-load or driver-load callsite.

Triage process telemetry during this phase recorded:

```
cmd.exe /c vssadmin delete shadows /all /quiet
```

Absence gates across the Triage-retained D1 host memory pages and rebuilt images found zero ASCII or UTF-16LE vssadmin / delete shadows strings. That makes the command a runtime observation with an unrecovered generator or callsite.

The Philips cluster is where the Picus public neighbor becomes useful. Picus documented Silver Fox public-sector campaigns using trojanized Philips DICOM software, Alibaba OSS, a.gif through d.gif, s.dat, s.jpeg, TrueSightKiller, RPC task scheduling, and later F0M-\* modules. Our case starts in a Panasonic host, reaches Philips material later, and lands in the same staging neighborhood.

## Phase 7: `thumbs.db` to `rundll32.dat!Edge`

The thumbs.db carrier also uses the five-byte trailer grammar:

```
trailer: 90 09 00 00 45
start:   0x990
seed:    0x45
```

Decoded output:

| Artifact | Value                                                            |
|----------|------------------------------------------------------------------|
| Module   | rundll32.dat                                                     |
| SHA-256  | 4f67662ae285337b17e537147e4783a6f307b25d61b362e7908b2595c4188f2a |
| Export   | Edge                                                             |

A Triage-retained mapped image from runtime has the same PE core with changed config fields:

| Field   | Decoded thumbs.db payload | Triage-retained mapped image | Offset  |
|---------|---------------------------|------------------------------|---------|
| Address | 8[.]218[.]106[.]149       | 47[.]239[.]170[.]54          | 0x4dd48 |
| Domain  | eeszuu[.]com              | ggsqoq[.]net                 | 0x4e1a4 |

Static review of rundll32.dat!Edge shows Sauron service behavior:

- Uses HKCU\SOFTWARE\Sauron
- Copies itself to C:\Windows\svchost.exe when the Sauron key is absent
- Creates and starts service Sauron
- Deletes marker files such as C:\del, C:\tzfz, C:\1.ini, C:\2.ini, c:\xxxx.ini, and C:\xxxx.inst.ini
- Checks security and analysis tool names

- Writes cleanup batch content
- Launches cleanup, firewall-off, BOOTSECT, marker-file, taskkill, downloader, and service paths
- Uses downloader templates:
  - `hxxp://%s/upx.rar`
  - `hxxp://%s/%d.dll`
  - `hxxp://%s/ip.txt`

The Gh0stRAT-family label has practical value here. Public reporting ties SilverFox/Winos tooling to Gh0stRAT-derived RAT lineages, and this final stage behaves like a service-backed remote-access component with downloader and command-launch paths.

The strongest supported runtime config material for this case is:

```
47[.]239[.]170[.]54
gqsqoq[.]net
```

The Joe Sandbox-reported 47[.]238[.]95[.]170:7089 remains a sandbox-report lead until matching packet bodies, decoded payload bytes, or mapped config material promote it.

### Parallel D1-style branch

A second D1-style branch appears in Triage-retained runtime evidence:

```
C:\Program Files (x86)\2GfN8\ULR23.exe
```

The Triage-retained second XPSPL0G.dll is nearly identical to the D1 XPSPL0G.dll:

| Artifact           | SHA-256                                                          | Notes                                                                       |
|--------------------|------------------------------------------------------------------|-----------------------------------------------------------------------------|
| Second XPSPL0G.dll | f4c4fdf7b6c3a6284ceec83fee339bac826b6e89d08e9609643da61b6dbb91d3 | First 4,812,812 bytes match D1 XPSPL0G.dll; final four overlay bytes differ |
| Second image.png   | 9277983c1beca79e56e16eabba0fd84b17279e21a9e59bda0ed8834cae9b321f | Differs from D1 carrier at one byte; decodes to the same image.dll          |

This branch supports reuse of the same protected carrier set under a second randomized host path. Extra claims stay tied to retained bytes from that branch.

### What made this sample painful

The sample is layered in a way that punishes single-source analysis.

The static file gives the first loader and URL construction. The recovered remote objects give the carriers. The side-loaded Ux path becomes clear from Triage-retained loaded memory. The bridge begins sparse, then populates itself in memory. The D1 stage uses protected dispatch through ordinal exports. `image.dll` has stable strings only after runtime population. `rundll32.dat` carries one config in the decoded carrier and a changed config in the Triage-retained mapped image.

The defensive summary is simple. The research path is swamp ballet.

The recurring design choices are:

- Signed host, protected side-loaded DLL
- Opaque carrier file beside the DLL
- Five-byte EOF trailer for payload extraction
- Incrementing XOR or RC4 transform
- Manual mapping into memory

- Defender and AV-prep commands around the payload
- Cloud object storage used as the payload shelf
- Randomized local directories and filenames
- Late-stage config mutation in mapped memory

Those mechanics matter more than the bucket names. Buckets churn. Carrier grammar tends to leave fingerprints.

## Detection and hunting notes

---

### Carrier grammar

---

Hunt for image-named or data-named files where the last five bytes can be interpreted as:

```
[payload_start_dword_le][xor_seed_byte]
```

Then attempt incrementing-XOR decode from payload\_start to EOF-5. Look for:

- MZ
- Shellcode entry patterns such as near jumps
- High-entropy protected PE material
- Exports such as Edge, PhilipsCoInitialize, DelegateEnSerialization, or task-registration functions

This family of carrier logic appeared across:

```
a.gif
b.gif
c.gif
d.gif
s.jpg
image.png
thumbs.db
drops.jpg
```

### Side-load bundles

---

High-signal local groupings:

```
UxEnhanceHost / UxEnhance64.dll / msadox.tb / adoresd.dat
D1IQf1.exe / XPSPLOG.dll / image.png / thumbs.db
```

The Ux bundle is especially strong when UxEnhance64.dll is adjacent to msadox.tb and adoresd.dat, and the DLL imports only KERNEL32.dll!ReadFile.

The D1 bundle is strong when a signed Philips/Speech Processing Solutions host loads protected XPSPLOG.dll and sits beside image.png and thumbs.db.

### RPC scheduled task creation

---

Cover Task Scheduler RPC alongside command-line schtasks.exe:

```
86D35949-83C9-4044-B424-DB363231FD0C
ncacn_np
\pipe\atsvc
NdrClientCall3
HighestAvailable
S-1-5-32-545
```

### Defender and prep commands

---

Strings and command bodies observed across populated stages include:

```
Task1
Add-MpPreference -ExclusionPath 'C:\ProgramData','C:\Users','C:\Program Files (x86)' -Force
Exclusions\Paths
icacls
Alibaba SecurityHealth
```

## Backdoor service artifacts

For the decoded rundll32.dat!Edge stage:

```
HKCU\SOFTWARE\Sauron
Sauron
C:\Windows\svchost.exe
hxxp://%s/upx.rar
hxxp://%s/%d.dll
hxxp://%s/ip.txt
C:\del
C:\tzfz
C:\1.ini
C:\2.ini
c:\xxx.ini
C:\xxx.inst.ini
```

## Network pivots

| Network IOC                                          | Evidence level                                              | Notes                                             |
|------------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------|
| jun616[.]oss-cn-beijing[.]aliyuncs[.]com             | Static URL reconstruction plus Triage TLS SNI corroboration | First carrier host                                |
| hxxps://jun616[.]oss-cn-beijing[.]aliyuncs[.]com/tad | Static reconstruction                                       | First index object                                |
| 26nn[.]oss-cn-hangzhou[.]aliyuncs[.]com              | Bridge-correlated plus Triage TLS SNI corroboration         | Later drops host                                  |
| /drops.jpg                                           | Triage PCAP-derived object plus Triage path summary         | Decodes to shellcode-style material               |
| 47[.]239[.]170[.]54                                  | Triage-retained mapped rundll32.dat config                  | Strongest runtime config IP for this case         |
| gqsqoq[.]net                                         | Triage-retained mapped rundll32.dat config                  | Strongest runtime config domain for this case     |
| 47[.]238[.]95[.]170:7089                             | Joe Sandbox-report lead                                     | Keep as a lead until promoted                     |
| /f.dat, F0M-50.jpg to F0M-53.jpg                     | Sandbox-report leads                                        | Good public-context pivots pending byte promotion |

## MITRE ATT&CK mapping

| Technique                      | ID        | Evidence in this sample                                                   |
|--------------------------------|-----------|---------------------------------------------------------------------------|
| User Execution: Malicious File | T1204.002 | Trojanized executable delivered as ainstaller-86533005.exe                |
| Masquerading                   | T1036     | Panasonic identity at first stage; signed Tencent and Philips hosts later |
| DLL Side-Loading               | T1574.002 | UxEnhanceHost loads UxEnhance64.dll; D1IQf1.exe loads XPSPL0G.dll         |

| Technique                                 | ID        | Evidence in this sample                                                                  |
|-------------------------------------------|-----------|------------------------------------------------------------------------------------------|
| Obfuscated Files or Information           | T1027     | Multi-layer staged buffer, trailer-XOR carriers, RC4 marker-tail shellcode               |
| Deobfuscate/Decode Files or Information   | T1140     | Incrementing XOR, rolling XOR, RC4, aPLib decompression                                  |
| Virtualization/Sandbox Evasion            | T1497     | Timing checks, VirtualAllocExNuma, heap pressure, process/object probes                  |
| Impair Defenses                           | T1562.001 | Defender exclusions, NtTraceEvent patching, AV process checks, TrueSight driver material |
| Signed Binary Proxy Execution             | T1218     | Trusted signed hosts used to load malicious DLLs                                         |
| Scheduled Task/Job                        | T1053.005 | Task Scheduler XML and RPC task creation                                                 |
| Ingress Tool Transfer                     | T1105     | Alibaba OSS carrier downloads                                                            |
| System Services                           | T1569.002 | Sauron service creation in rundll32.dat!Edge                                             |
| Command and Scripting Interpreter         | T1059     | PowerShell, CMD command templates, cleanup batch content                                 |
| Application Layer Protocol: Web Protocols | T1071.001 | WinINet downloader and HTTP templates in late stage                                      |

## IOC summary

### Hashes

| Artifact                          | SHA-256                                                          |
|-----------------------------------|------------------------------------------------------------------|
| Outer EXE                         | 200d9f5d041c870e69f73da32c753a69096742c33e2dab65299767ac31578267 |
| Encoded staged buffer             | 415611a746fc72025f206164eb61a966125a76125ab1c84ec37e159dfbc8d6b3 |
| First-layer decoded staged buffer | 89ec512835063cb87f1f924f61fe402e9f433de6c194831645cf60a7570b71fc |
| Fully decoded staged buffer       | d612918e6bf49bcd9b840a5f95a874616f48cce309983215c09ccc15cd35891a |
| Decoded heap payload              | 679921a7373fc0a77d60416e3f7ce4454dfd394c12d9c0387be9538b8f030e02 |
| Raw tad object                    | dd69f3a8e89ffb4bd3ee8ebc6dcc786e59cbc47ad8f3e18a6de5de9bdf6d4a1d |
| Decoded tad index                 | f43cc91a545a7e9712e2132e559ed808fcfa80f882c08cf9fa44974fa691bec4 |
| Raw a.gif                         | 6349d18308f458ed290913068752471c453f1c9bbd472a73e9f92b444718bd4d |
| Decoded a.gif host                | 676a2a7b94ca2f8ec76352ee656e4d075bb342bd7ad6efbc7c19c060001eace7 |
| Raw b.gif                         | c2ffcb613698ba2c74614b993f550073f6dd2ba2f1c3f120d68959aa142c78ad |
| Decoded UxEnhance64.dll           | cc6ae6f13d33a7313295ae00a6b0503adc277289fbe615eaf48b6a30ce09f11c |
| Raw c.gif                         | c0627c59e27a670503683d2c5c117e126f4969144815317540895066c819b6f6 |
| Decoded msadox.tb                 | d14e3e61957ba142d64830b63418ce816e76e4c25449539a5cebb5bcd5f84c7f |
| Raw d.gif                         | ab432ec0e732c20c99b6413e7ab9d0ac18bcd5d405a6266ff0c1afced64cb165 |

| Artifact                                       | SHA-256                                                           |
|------------------------------------------------|-------------------------------------------------------------------|
| Decoded adoresd.dat                            | 1a9804780f9b96018961a4100b3ff70d6440a2677ce996209c6be0d581b4677f  |
| Raw s.dat                                      | fe4d5a88fdf7809ab3d45cb3be475a4625a0e6a7e2972c82017411410f40cd6a  |
| Decoded s.dat / ranchserv.jpg                  | 53373b27c784db09449a4c01170acaf99e9316552de54d9be93896738006d60c  |
| Raw s.jpg                                      | 71369dd8ef596d4eefaf21b831dc87f78eeaf2c3436eb95c24981a6d39151dd7  |
| Decoded s.jpg payload                          | 238022615a5d37269bb3826bdfc30429442fcf73b2ed510098183383d3454f6   |
| aPLib block from s.jpg                         | 1cd8b1aa6e26831557b98247c29d2dea27a2b5ae0fced69db1058884d074dfff  |
| RPCScheduleTask.dll                            | a1e68e7adc5597eae23a6590f1ac345f796abebceb91afcc0f2dd97cc0622740  |
| Triage runtime Ux memory image                 | f85321c2cef8b8bf252d314db2422c187f1397641bdc1a3d5a9426a3fd55fd9   |
| RC4 shellcode from msadox.tb                   | 56c5bd8df62c54be51312fabcb86ebe9cde896ed015b91cf3e0aa7c941f49d096 |
| Decoded adoresd.dll                            | aaa13e34e7c45dc49552e843ee8b611b2373959d8a7cc23265abc353d76235ba  |
| Populated bridge image                         | 09ae57f0e6ffa1abed9b868e6693cdba84611b24bc50fc278f3017de7d5c943   |
| D1IQf1.exe                                     | 6d6ba2bc9ad414837826f7278bc3e0116f1aeda02d0c2284ed65819f5d9180a8  |
| XPSPLOG.dll                                    | 3317c1e3471de643c7868c7e0821524b0bfdeb4a3d335e1ec5431a9088234ab7  |
| Rebuilt populated XPSPLOG.dll                  | 63d8b720e35bc8eafc40c37194c7fce270be2b862e04a5278c4aa03d2907b25b  |
| image.png carrier                              | 06d9687f2d7a462d55480037bf1eb0fce9d42d9f56d950695b9d6a2f52241188  |
| Decoded image.dll                              | 96aee41363c08acf6a6230d1dda5f7e7dae9f78952993b70a9c8c4949c81bb22  |
| Mapped populated image.dll                     | 1531fbf2af768bc1ceb8b33039473d18bc186f99c34a29ea395bdb4935c7b4a2  |
| thumbs.db carrier                              | 75535087898c38d5da3c16282d008d68df648ae58a3cf8b632a953302544279c  |
| Decoded rundll32.dat                           | 4f67662ae285337b17e537147e4783a6f307b25d61b362e7908b2595c4188f2a  |
| Mapped rundll32.dat                            | 0272f3055c26287c7e393fcbe8e2273fe39ed0f025daa483b0ca42fe6aafa24f  |
| Saved Triage drops.jpg object                  | 667698da115bb3cc322a972dd8fb15280f0e481d5a87779f4c444f535bf17a74  |
| Decoded drops.jpg payload                      | 1049a21e9dd6b26b0574c132b12b4565e4bfb7096c31f8a21dbe45e5e3fbddc1  |
| Separate Triage-retained later-phase mapped PE | 12845df5f905fe5345829b2aca546ba8eecbd84f1a2924993b076a5d33bd1307  |
| Second XPSPLOG.dll                             | f4c4fdf7b6c3a6284ceec83fee339bac826b6e89d08e9609643da61b6dbb91d3  |
| Second image.png carrier                       | 9277983c1beca79e56e16eabba0fd84b17279e21a9e59bda0ed8834cae9b321f  |

## Network

| Type                    | Value                                                                  | Evidence level                           |
|-------------------------|------------------------------------------------------------------------|------------------------------------------|
| Reported delivery pivot | hxxps://seeo-pc[.]com[.]cn -><br>hxxps://www[.]hed6g8u[.]com/installld | Community-provided context               |
| Initial host            | jun616[.]loss-cn-beijing[.]aliyuncs[.]com                              | Static reconstruction and Triage TLS SNI |

| Type                  | Value                                                  | Evidence level                                      |
|-----------------------|--------------------------------------------------------|-----------------------------------------------------|
| Initial URL           | hxxps://jun616[.]oss-cn-beijing[.]aliyuncs[.]com/tad   | Static reconstruction                               |
| Later host            | 26nn[.]oss-cn-hangzhou[.]aliyuncs[.]com                | Bridge-correlated and Triage TLS SNI                |
| Later path            | /drops.jpg                                             | Triage PCAP-derived object plus Triage path summary |
| Runtime config IP     | 47[.]239[.]170[.]54                                    | Triage-retained mapped rundll32.dat                 |
| Runtime config domain | gqsqq[.]net                                            | Triage-retained mapped rundll32.dat                 |
| Sandbox C2 lead       | 47[.]238[.]95[.]170:7089                               | Joe Sandbox-report lead                             |
| Sandbox URL leads     | /f.dat, F0M-50.jpg, F0M-51.jpg, F0M-52.jpg, F0M-53.jpg | Sandbox-report leads                                |

## Host indicators

| Indicator                         | Context                                            |
|-----------------------------------|----------------------------------------------------|
| */&                               | Marker used to derive first-stage config material  |
| checktime.vbs                     | Hidden VBS helper under generated public-user path |
| C:\Users\Public\Y5EiLf\6322eI.exe | Ux side-load host path from runtime artifacts      |
| UxEnhance64.dll                   | Protected side-loaded DLL                          |
| msadox.tb                         | RC4 marker-tail shellcode carrier                  |
| adoresd.dat                       | Trailer-XOR bridge carrier                         |
| DelegateEnSerialization           | adoresd.dll export                                 |
| VirtuOne                          | RPC scheduled-task DLL export                      |
| D1IQf1.exe                        | Signed Philips/Speech Processing Solutions host    |
| XPSPLOG.dll                       | Protected D1 side-loaded DLL                       |
| image.png                         | Carrier for image.dll                              |
| thumbs.db                         | Carrier for rundll32.dat                           |
| PhilipsCoInitialize               | image.dll export                                   |
| Edge                              | rundll32.dat export                                |
| HKCU\SOFTWARE\Sauron              | Backdoor registry key                              |
| Sauron                            | Service name                                       |

## Research scope

The static chain is strong from the outer executable through the Ux bridge and into decoded D1 artifacts. Several later runtime items remain scoped:

- `/drops.jpg` is bridge-correlated. The bridge has code-referenced `26nn / drops` fragments and `GetData` material; Triage runtime summaries record the matching request; the saved Triage PCAP-derived object bytes decode as documented.
- `vssadmin delete shadows` is runtime-observed. Absence gates found zero generator strings in the reviewed D1 pages and rebuilt images.
- `47[.]238[.]95[.]170:7089, /f.dat`, and `F0M-*` are sandbox-report leads. Treat `47[.]238[.]95[.]170:7089` as a Joe Sandbox lead until byte evidence promotes it.
- `ranchserv.jpg` file materialization is confirmed; driver execution needs loader-path evidence.
- The protected XPSLOG handoff into `image.png` and `thumbs.db` is bounded by imported ordinal calls and decoded artifacts.
- The `VirtualOne` task structure is confirmed; concrete task fields are caller-generated.
- The parallel D1-style branch is supported by the Triage-retained second `XPSLOG.dll` and second `image.png` carrier evidence.

Keep that boundary intact. It separates a defensible teardown from a tidy story. The loader has plenty of masks. The byte trail is still visible.

## Conclusion

---

`ainstaller-86533005.exe` is a layered SilverFox-style loader chain wrapped in a Panasonic host.

The sample moves through an early in-memory loader, reconstructs an Alibaba OSS staging URL, retrieves image-named encrypted carriers, builds a signed Ux side-load bundle, decrypts a bridge DLL through RC4 and trailer-XOR payloads, reaches a Philips side-load cluster, and lands at `rundll32.dat!Edge`, which implements Sauron service/backdoor behavior.

Huorong's SilverFox report gives the clearest public overlap: cloud-storage rhythm, image-carrier names, five-byte trailer format, TrueSightKiller material, RPC Task Scheduler machinery, and late `drops.jpg` / Edge flavor. Picus adds the Philips and medical-themed tradecraft. ThreatBook and Fortinet give the family context: WinOS/Winos4.0 sits in a Gh0stRAT-derived ecosystem used across multiple campaigns.

I would label it SilverFox / Winos-style ecosystem alignment with a Gh0stRAT-family late-stage signal.

For detection, chase the grammar over the costume: Alibaba OSS plus image-named carriers plus five-byte trailer XOR plus signed side-load hosts plus RPC task creation plus late Edge export.

That constellation is the fingerprint. File names can change.

Kirk

I like the internet. Want to get in touch?