Sandman APT | A Mystery Group Targeting Telcos with a LuaJIT Toolkit

Aleksandar Milenkoski ⁝

By Aleksandar Milenkoski, in collaboration with QGroup

Executive Summary

- SentinelLabs has observed a new threat activity cluster by an unknown threat actor we have dubbed Sandman.
- Sandman has been primarily targeting telecommunication providers in the Middle East, Western Europe, and the South Asian subcontinent.
- The activities are characterized by strategic lateral movements and minimal engagements, likely to minimize the risk of detection.
- Sandman has deployed a novel modular backdoor utilizing the LuaJIT platform, a relatively rare occurrence in the threat landscape. We refer to this malware as LuaDream.
- The implementation of LuaDream indicates a well-executed, maintained, and actively developed project of a considerable scale.
- At this time, we don’t have a consistent sense of attribution. LuaDream does not appear to be related to any known threat actors. While the development style is historically associated with a specific type of advanced threat actor, inconsistencies between the high-end development of the malware and poor segmentation practices lead us towards the possibility of a private contractor or mercenary group similar to Metador.

Overview

In collaboration with QGroup GmbH, SentinelLabs observed over August 2023 a threat activity cluster targeting the telecommunication sector. The activities have been conducted by a threat actor of unknown origin using a novel modular backdoor based on the LuaJIT platform. We dub this threat actor and the backdoor Sandman and LuaDream in reference to what we suspect to be the backdoor’s internal name – DreamLand client.

The activities we observed are characterized by strategic lateral movement to specific targeted workstations and minimal engagement, suggesting a deliberate approach aimed at achieving the set objectives while minimizing the risk of detection.

The implementation and architecture of LuaDream suggest a maintained, versioned project under active development. This is a modular, multi-protocol backdoor whose main functionalities are:

- exfiltrating system and user information, paving the way for further precision attacks;
- managing attacker-provided plugins that extend LuaDream’s features.

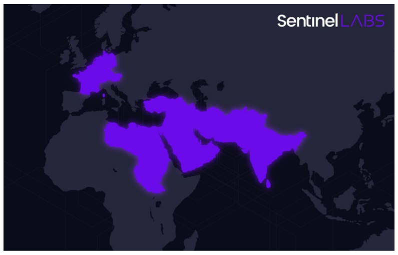
Although the intrusions were detected and interrupted before the threat actor could deploy plugins, our analysis of LuaDream staging samples shared on VirusTotal provided a glimpse into what functionalities the plugins may implement, with command execution capabilities being one example.

The 36 distinct LuaDream components we identified and the support for multiple protocols for C2 communication indicate a project of a considerable scale. The LuaDream staging chain is designed to evade detection and thwart analysis while deploying the malware directly into memory. LuaDream’s implementation and staging process leverage the LuaJIT platform, the just-in-time compiler for the Lua scripting language. This is primarily to make malicious Lua script code difficult to detect.

A Penchant for Telcos

Based on current visibility, accurate clustering remains a challenge. The focussed, strategy-driven activities, and the use of complex malware designed to evade detection point to a motivated and capable adversary. The TTPs, victimology, and the characteristics of the deployed malware indicate that it is highly likely this activity has espionage motivations. Communication providers are frequent targets of espionage activity due to the sensitive data they hold.

The activity cluster we observed and examination of C2 netflow data indicate a pronounced focus on targeting telecommunications providers with a broad geographical distribution, including the Middle East, Western Europe, and the South Asian subcontinent.



Geographical distribution of victims

Compilation timestamps and a string artifact found within LuaDream hint at potential malware development efforts over the first half of 2022, suggesting possible threat actor activity dating back to 2022.

While we cannot associate LuaDream to any known threat actor, we lean towards the possibility of a private contractor or mercenary group. Typically used as a scripting middleware in gaming and specialty embedded applications and appliances, the use of LuaJIT in the context of APT malware is relatively rare but the population using it is becoming broader.

Embedded Lua VMs serve as a mechanism for modularity and extensibility for advanced APTs, historically considered Western or Western-aligned. However, this development paradigm is being embraced by a broader set of threat actors that also target Western countries and deserves further scrutiny as exemplified by the Sandman APT. Our talk at LABScon 2023 described this paradigm of development overtime, bookended by our discovery of Sandman APT as the latest, along with Fast16 as the earliest example dating back to 2005.

In March 2023, new malware was briefly described by Kaspersky during a quarterly roundup actively targeting a government entity in Pakistan. Based on the sparsely described characteristics, we assess that they're referring to a variant of LuaDream –dubbed DreamLand. Note the following string in the LuaDream samples we identified:

```
C:\\project\\tenyears\\DreamLandClient\\Project\\cpp\\HttpClientLj\\testdll.dll
```

Threat Actor Activities

The activities we observed took place over several weeks in August 2023. After stealing administrative credentials and conducting reconnaissance, Sandman infiltrated specifically targeted workstations using the pass-the-hash technique over the NTLM authentication protocol. On one of the targets, all of the workstations were assigned to personnel in managerial positions.

On average, we observed a five-day gap between infiltrations into different endpoints. After gaining access, Sandman limited its activities to deploying folders and files required for loading and executing LuaDream, refraining from any further actions. We observerd the following deployed filesystem artifacts:

```
C:\Windows\System32\ualapi.dll
C:\ProgramData\FaxConfig\fax.dat
C:\ProgramData\FaxConfig\fax.cache
C:\ProgramData\FaxConfig\fax.module
C:\ProgramData\FaxConfig\fax.Application
C:\ProgramData\FaxLib\
```

Sandman abused the DLL hijacking technique to execute LuaDream. The `ualapi.dll` file they placed is a malicious DLL masquerading as its legitimate counterpart (a User Access Logging (UAL) component) and represents the first stage of the intricate LuaDream loading process. The `ualapi.dll` library is loaded by the `Fax` and the `Spooler` Windows service when started. We observed the `Spooler` service loading the malicious `ualapi.dll` on the targeted workstations, executing LuaDream in its context.

It is relevant to note that we did not observe the threat actor restarting the `Fax` and or `Spooler` service to force the execution of LuaDream, likely to evade detection based on service manipulation. Instead, they were patient in waiting for one of these services to load the malicious `ualapi.dll` when started at the next system boot.

LuaDream | Staging

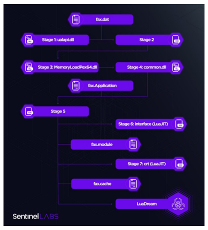
The LuaDream staging process is intricate and designed with a focus on evading detection and thwarting analysis. Initiated by the `Fax` or the `Spooler` service, which would execute the `UalStart` export of the malicious `ualapi.dll` when started, the overall process consists of seven main stages. These are conducted fully in memory and involve a combination of fully-formed DLL PE images, code, and LuaJIT bytecode.

The following table shows DLL images involved in LuaDream staging:

Name	Compilation timestamp	Exports
ualapi.dll	Wed Aug 09 18:24:18 2023	UalInstrument, UalStart, UalStop
MemoryLoadPex64.dll	Wed Mar 22 23:55:07 2023	ProtectMain
common.dll	Wed Aug 09 18:21:18 2023	jsadebugd

Although the DLL timestamps could have been manipulated by the threat actor, given the proximity to the August 2023 intrusion date, it is likely that the timestamps are authentic. Due to the difference of only a few days between the timestamps of `ualapi.dll` and `common.dll`, and their actual deployment dates, it is possible that these images have been built specifically for this intrusion.

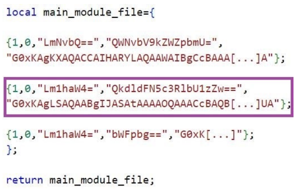
Some of the implemented anti-analysis measures include hiding LuaDream's threads from a debugger using the `NtSetInformationThread` function, file close operation on an invalid handle (`0x123456`), detection of Wine-based sandboxes, and in-memory mapping of malicious PE images to evade EDR API hooks and file-based detections.



LuaDream staging

Next-stage code is typically packed using a combination of XOR-based encryption and compression. The `fax.dat`, `fax.Application`, and `fax.module` files store packed staging code. The code unpacked from `fax.Application` contains a LuaJIT engine enabling the execution of the LuaJIT components internally referred to as `interface` and `crt` as well as LuaDream itself.

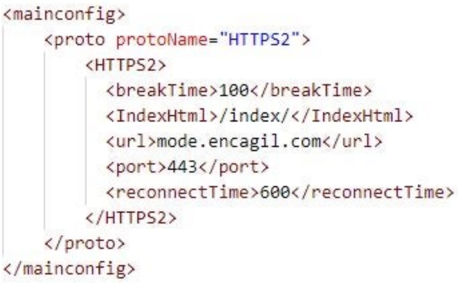
`interface` unpacks crt from `fax.module`, which in turn retrieves XML-formatted configuration and the contents of the `fax.cache` file – an encrypted and compressed Lua function, which returns the reference names and implementations of LuaDream components in Base-64 encoded form.



```
local main_module_file={

{1,0,"LmNvbQ==","QWNvbV9kZWZpbmU=",
"G0xKAgKXAQACCAIHARYLAQAAWAIBgCcBAAA[...]A"};

{1,0,"Lm1haW4=","QkdldFN5c3RlbU1zZw==",
"G0xKAgLSAQAABgIJASAtAAAAOQAAACcBAQB[...]UA"};

{1,0,"Lm1haW4=","bWFpbg==","G0xK[...]"};
};

return main_module_file;
```

fax.cache (unpacked form)

The LuaDream configuration includes C2 and communication protocol information. The LuaDream variant we analyzed is configured to communicate with the `mode.encagil[.]com` domain over the WebSocket protocol.



```
<mainconfig>
    <proto protoName="HTTPS2">
        <HTTPS2>
          <breakTime>100</breakTime>
          <IndexHtml>/index/</IndexHtml>
          <url>mode.encagil.com</url>
          <port>443</port>
          <reconnectTime>600</reconnectTime>
        </HTTPS2>
    </proto>
</mainconfig>
```

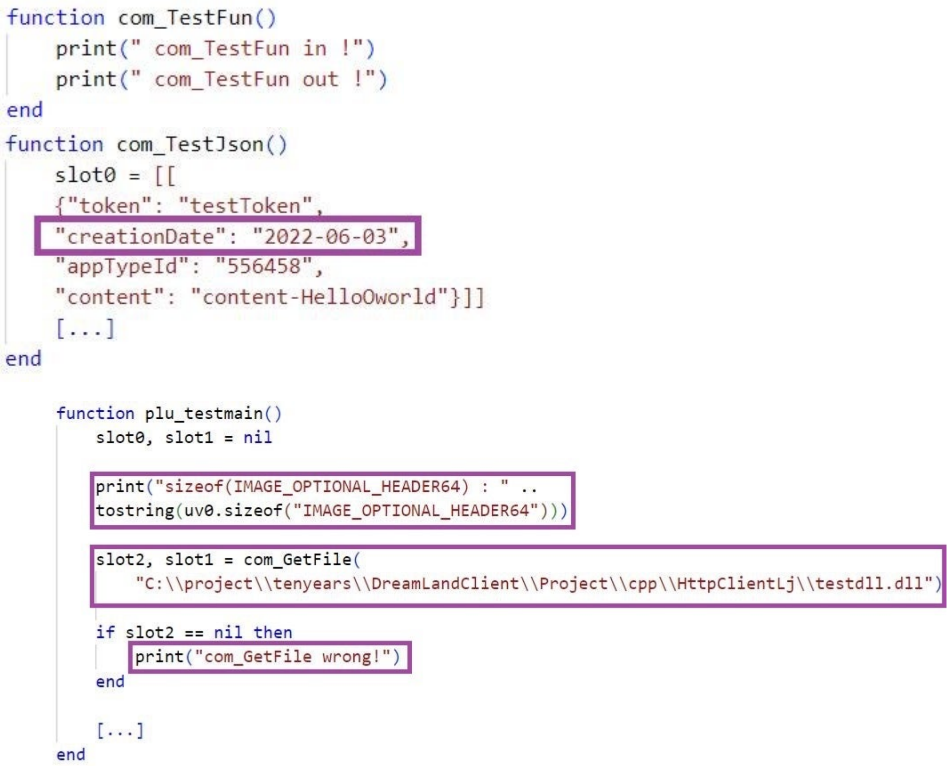
Configuration data

LuaDream | Overview

LuaDream is a multi-component and multi-protocol backdoor, whose main features are managing attacker-provided plugins and exfiltrating system and user information. The implementation and architecture of LuaDream indicates that it is a maintained, actively developed project of a considerable scale.

Throughout our analysis, we observed what is likely a malware version string (`12.0.2.5.23.29`), which the backdoor sends to the C2 server when exfiltrating information. Many LuaDream function and variable definitions follow a naming convention involving the word `fun`, such as `dofun`, `_RUN_FUN_LIST_`, and `FunGetDataCache`.

LuaDream implements testing functions as well as error and execution status logging, which indicates that the malware is likely still in active development. A string artifact in a function labeled `com_TestJson` suggests potential development in June 2022.



```
function com_TestFun()
    print(" com_TestFun in !")
    print(" com_TestFun out !")
end
function com_TestJson()
    slot0 = [[
    {"token": "testToken",
    "creationDate": "2022-06-03",
    "appTypeId": "556458",
    "content": "content-HelloOworld"}]]
    [...]
end
```

```
function plu_testmain()
    slot0, slot1 = nil

    print("sizeof(IMAGE_OPTIONAL_HEADER64) : " ..
    tostring(uv0.sizeof("IMAGE_OPTIONAL_HEADER64")))

    slot2, slot1 = com_GetFile(
        "C:\\project\\tenyears\\DreamLandClient\\Project\\cpp\\HttpClientLj\\testdll.dll")

    if slot2 == nil then
        print("com_GetFile wrong!")
    end

    [...]
end
```

Testing functions (decompiled LuaJIT bytecode)

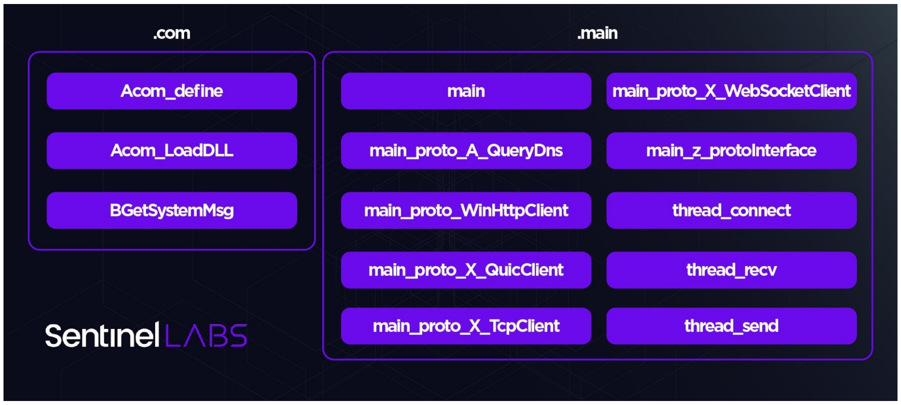
We observed the embedded private IP address `10.2.101[.]99` to which LuaDream binds the communication port `4443`, if so configured. This address does not belong to the IP address spaces of the targeted environments. The IP address may be a leftover from an in-development LuaDream variant or from a previous Sandman engagement.

LuaDream | Components And Features

The LuaDream variant we obtained from the targeted environments consists of 34 components: 13 core and 21 support components. They are implemented in LuaJIT bytecode and use the Windows API through the ffi library using C language bindings.

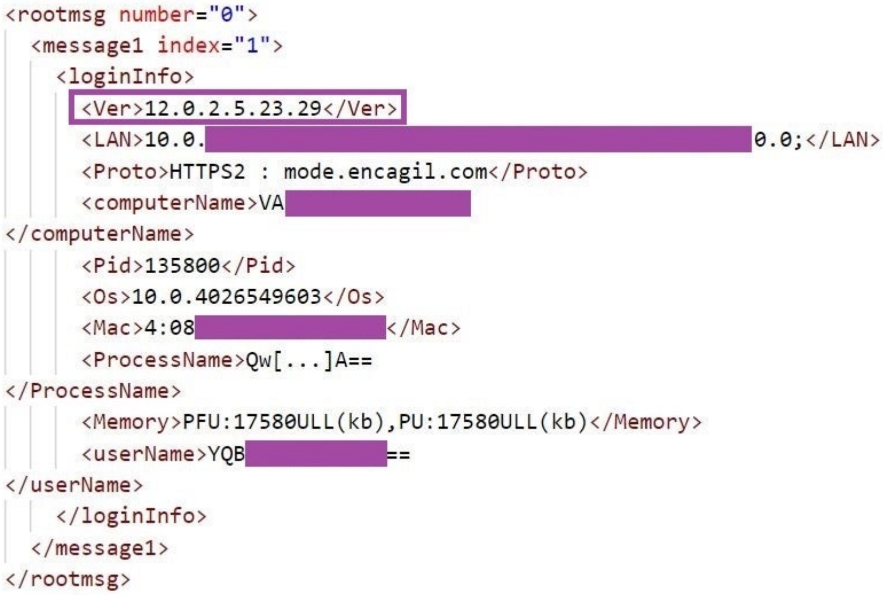
The support components implement Lua libraries as well as Windows API definitions required for LuaDream's operation, such as xml2lua, Windows Sockets, and NtSec API.

The core components implement LuaDream features, such as initialization, gathering system and user information, C2 communication, and plugin management. As per the component definitions from the `fax.cache` file, the core LuaDream components are structured into two categories: `.com` and `.main`.



LuaDream core components

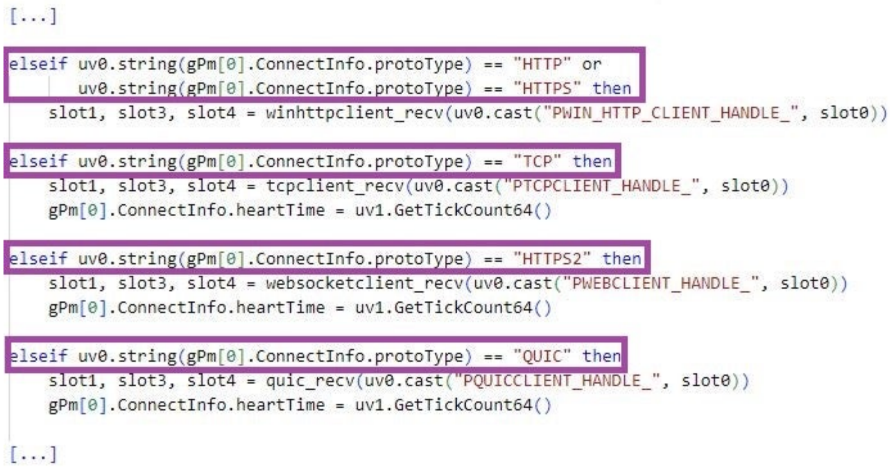
With the `main` component initializing LuaDream, the backdoor connects to the configured C2 server and exfiltrates system, user, and malware-related information gathered by `BGetSystemMsg`. This information includes the malware version, assigned IP and MAC addresses, OS version, available memory, and the name, PID, and username associated with the process in whose context LuaDream runs.



```
<rootmsg number="0">
  <message1 index="1">
    <loginInfo>
      <Ver>12.0.2.5.23.29</Ver>
      <LAN>10.0.[redacted]0.0;</LAN>
      <Proto>HTTPS2 : mode.encagil.com</Proto>
      <computerName>VA[redacted]
</computerName>
      <Pid>135800</Pid>
      <Os>10.0.4026549603</Os>
      <Mac>4:08[redacted]</Mac>
      <ProcessName>Qw[...]A==
</ProcessName>
      <Memory>PFU:17580ULL(kb),PU:17580ULL(kb)</Memory>
      <userName>YQB[redacted]==
</userName>
    </loginInfo>
  </message1>
</rootmsg>
```

Exfiltrated information

LuaDream has the capability to reach out to C2 servers but also to act as an implant listening for incoming connections. The backdoor can communicate over the TCP, HTTPS, WebSocket, and QUIC protocols. The `main_proto_X_TcpClient`, `main_proto_WinHttpClient`, `main_proto_X_WebSocketClient`, and `main_proto_X_QuicClient` components implement support for these protocols, with `main_z_protoInterface` acting as their main handler.



```
[...]

elseif uv0.string(gPm[0].ConnectInfo.protoType) == "HTTP" or
       uv0.string(gPm[0].ConnectInfo.protoType) == "HTTPS" then
    slot1, slot3, slot4 = winhttpclient_recv(uv0.cast("PWIN_HTTP_CLIENT_HANDLE_", slot0))

elseif uv0.string(gPm[0].ConnectInfo.protoType) == "TCP" then
    slot1, slot3, slot4 = tcpclient_recv(uv0.cast("PTCPCLIENT_HANDLE_", slot0))
    gPm[0].ConnectInfo.heartTime = uv1.GetTickCount64()

elseif uv0.string(gPm[0].ConnectInfo.protoType) == "HTTPS2" then
    slot1, slot3, slot4 = websocketclient_recv(uv0.cast("PWEBCLIENT_HANDLE_", slot0))
    gPm[0].ConnectInfo.heartTime = uv1.GetTickCount64()

elseif uv0.string(gPm[0].ConnectInfo.protoType) == "QUIC" then
    slot1, slot3, slot4 = quic_recv(uv0.cast("PQUICCLIENT_HANDLE_", slot0))
    gPm[0].ConnectInfo.heartTime = uv1.GetTickCount64()

[...]
```

Protocol handling (decompiled LuaJIT bytecode)

The `main_proto_A_QueryDns` component resolves domains to IP addresses using the `cloudflare-dns[.]com` service, which `main_proto_X_WebSocketClient` uses for resolving C2 domain names.

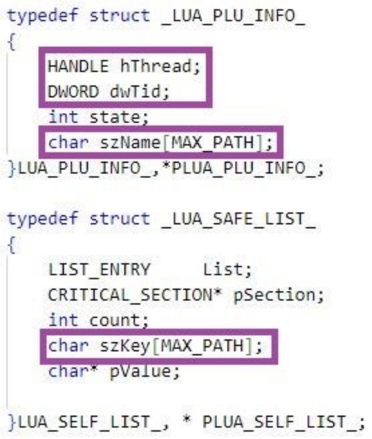
`main_proto_X_QuicClient` draws functionalities from a DLL image which LuaDream maps fully in memory, a functionality implemented by the `Acom_LoadDLL` component.

LuaDream communicates with a C2 server using the `thread_connect`, `thread_send`, and `thread_recv` components, which are responsible for connecting to, sending data to, and receiving data from the C2 server, respectively. These components operate in separate threads. The exchanged data is in JSON and XML format, in an encrypted and compressed form. The `Acom_define` component provides functionalities for inter-thread communication and data manipulation.

The `thread_recv` component handles incoming messages and its main purpose is to manage attacker-provided plugins that extend LuaDream. Some functionalities of this component include:

- taking LuaDream offline (command `offline`);
- loading, executing (command `loadplugin`), unloading (command `unloadplugin`), and saving plugins (command `saveplugin`);
- executing an attacker-specified plugin functionality.

LuaDream maintains a key-based list of plugin information, which includes the handle and the ID of the thread in which the plugin runs, and a plugin-identifying key. Loading of a plugin involves inserting a new entry in this list and executing plugin code in a designated thread. For communicating with plugins, LuaDream leverages inter-thread communication, using the message `1234` for executing plugin functionalities.

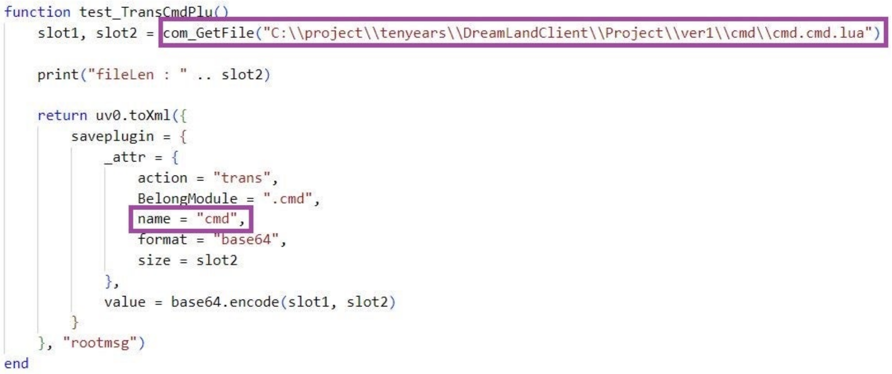


```
typedef struct _LUA_PLU_INFO_
{
    HANDLE hThread;
    DWORD dwTid;
    int state;
    char szName[MAX_PATH];
}LUA_PLU_INFO_,*PLUA_PLU_INFO_;

typedef struct _LUA_SAFE_LIST_
{
    LIST_ENTRY     List;
    CRITICAL_SECTION* pSection;
    int count;
    char szKey[MAX_PATH];
    char* pValue;

}LUA_SELF_LIST_, * PLUA_SELF_LIST_;
```

LuaDream plugin list (from decompiled LuaJIT bytecode)

Our analysis of LuaDream staging samples shared on VirusTotal revealed the existence of two additional components named `main_proto_WinHttpServer` and `thread_test`. `main_proto_WinHttpServer` implements a LuaDream capability to listen for incoming connections based on the Windows HTTP server API.

thread_test implements functions for testing the loadplugin and saveplugin commands. These functions indicate the existence of a plugin named cmd, whose name suggests command execution capabilities.



```
function test_TransCmdPlu()
    slot1, slot2 = com_GetFile("C:\\project\\tenyears\\DreamLandClient\\Project\\ver1\\cmd\\cmd.cmd.lua")

    print("fileLen : " .. slot2)

    return uv0.toXml({
        saveplugin = {
            _attr = {
                action = "trans",
                BelongModule = ".cmd",
                name = "cmd",
                format = "base64",
                size = slot2
            },
            value = base64.encode(slot1, slot2)
        }
    }, "rootmsg")
end
```

cmd plugin references

Network Infrastructure

The LuaDream samples we analyzed communicate with the C2 servers ssl.explorecell[.]com and mode.encagil[.]com. ssl.explorecell[.]com is a Tucows-registered domain with a first-seen resolution date of March 2023. This domain last resolved to 185.82.218[.]230, an IP address of a server hosted in Bulgaria by the ITLDC hosting provider.

mode.encagil[.]com is an Arsys-registered domain with a first-seen resolution date of August 2023. The domain last resolved to 172.67.173[.]208 and 104.21.47[.]226, IP addresses of a server hosted behind a major load balancing platform. The shift from using a directly exposed C2 server IP address to addresses of a load balancing infrastructure marks a change in Sandman's infrastructure management practices – likely to avoid exposing the true hosting location.

Examination of C2 netflow data revealed lack of comprehensive C2 infrastructure segmentation, with several LuaDream deployments at geographically dispersed victim environments communicating with the same C2 server.

Conclusions

Attributing Sandman remains a mystery, placing it in the same enigmatic category as Metador and other elusive threat actors who operate with impunity. LuaDream stands as a compelling illustration of the continuous innovation and advancement efforts that cyber espionage threat actors pour into their ever-evolving malware arsenal.

Navigating the shadows of the threat landscape necessitates consistent cooperation and information sharing within the threat intelligence research community. SentinelLabs remains dedicated to this mission and hopes that this publication will serve as a catalyst for further collaborative efforts. We are grateful for the contributions of Luca Palermo from the SentinelOne EMEA IR TAM team, who assisted with the initial investigations and remediation of the threat.

Indicators of Compromise

SHA1	File name
1cd0a3dd6354a3d4a29226f5580f8a51ec3837d4	fax.dat
27894955aaf082a606337ebe29d263263be52154	fax.Application
5302c39764922f17e4bc14f589fa45408f8a5089	ualapi.dll
77e00e3067f23df10196412f231e80cec41c5253	fax.cache
b9ea189e2420a29978e4dc73d8d2fd801f6a0db2	UpdateCheck.dll
fb1c6a23e8e0693194a365619b388b09155c2183	updater.ver
ff2802cdbc40d2ef3585357b7e6947d42b875884	fax.module

LuaDream Folder File paths
%ProgramData%\FaxConfig
%ProgramData%\FaxLib

C2 Server Domains
mode.encagil[.]com
ssl.explorecell[.]com