🇨🇳 More on DreamLand

Published: 2023-09-22 · Archived: 2026-04-05 17:51:23 UTC

In April, Kaspersky briefly described a new malware dubbed `DreamLand` in their APT trends report Q1 2023. Quote:

> In March, we discovered a new malware strain actively targeting a government entity in Pakistan. We designated this malware “DreamLand”. The malware is modular and utilizes the Lua scripting language in conjunction with its Just-in-Time (JIT) compiler to execute malicious code that is difficult to detect. It also features various anti-debugging capabilities and employs Windows APIs through Lua FFI, which utilizes C language bindings to carry out its activities. This is the first time we have seen Lua used by an APT threat actor since its use by AnimalFarm and Project Sauron.

The same malware is being used in more attacks, as described in a blog post published by SentinelOne yesterday: Sandman APT

The post details an incident happened in August 2023 and gives additional information about two samples submitted to Virustotal. This blog post gives additional information on the samples uploaded to Virustotal.

DreamLand - A brief analysis

File hash (SHA-256)	File name	Description
ceaec139a9370a4cd4eca876e7c4b3d51a013d3739b3f4d526fdfeab27cd2fc2	libcurl.dll	Loader for UpdateCheck.dll
0b962ad02e8eef3c717ce6fcfda9587f92ebe9e7ed6ee93be6bc1103daa4e8bf	UpdateCheck.dll	Loader for the main embedde LuaJIT orchestrator
9bb5e7a76e66d105fa5a65728517b8d8f9465525465f92eb68a89705476b1d26	updater.ver	Contains encrypted/compressed/encod compiled LuaJIT scripts

The loading chain of the avilable samples is as follows: `libcurl.dll` -> `UpdateCheck.dll` -> `updater.ver`

The initial file named `libcurl.dll` is an unconventional loader for the file `UpdateCheck.dll`. It dynamically resolves the API functions `GetConsoleWindow` and `ShowWindow` and calls the latter with the parameter `SW_HIDE` to hide its console window. Next, it patches the entry point of the process it was being loaded into, to call its exported function `curl_easy_cleanup`.

Patched entry point of own process:

```
 sub rsp,28
mov rax, <libcurl.curl_easy_cleanup>
call rax
```

The exported function `curl_easy_cleanup` is an infinite loop:

```
 __int64 curl_easy_cleanup()
{
  DWORD TickCount;
  DWORD v1;
```

```
  do
  {
    TickCount = GetTickCount();
    Sleep(TickCount % 0x43955);
    v1 = GetTickCount();
    Sleep(v1 % 0x433);
    Sleep(0x3005u);
  }
  while ( GetCurrentProcessId() > 4 );
  return 0i64;
}
```

As we don't know the initial process that loads `libcurl.dll`, it's inclear what the purpose of this patch is. It might be made in order to keep the process running infinitely. At last, it loads the second stage DLL `UpdatCheck.dll` and resolves its exported function `curl_get_build_version`. Before calling `curl_get_build_version` to execute `UpdatCheck.dll`, it tries to dynamically resolve the API functions `AzApplicationOpen` (azroles.dll) and `DllEnumClassObjects` (mshtml.dll) subsequently to call them instead of `curl_get_build_version` if they exist in the process. That was probably done to stop the malware from running on certain systems or when a specific security software is present.

When `curl_get_build_version` in `UpdatCheck.dll` is executed, it first decompresses and decrypts an embedded payload that is internally named `HttpClientLJ.dll`. This payload is the main orchestrator and contains the LuaJIT interpreter for the compiled scripts contained in `updater.ver`. It seems to be based on an open-source project called TINN. After a memory module was created out of the raw payload, its entry point (DllMain) is called to run the normal and Lua initialization routines. Afterwards its exported function `GetObjectInterface` is called that in turn loads the first compiled LuaJIT script. This script is a loader that decodes, decompresses, decrypts and runs the main module and configuration data which is as follows:

```
 <mainconfig protoName="HTTPS">
    <HTTPS>
      <breakTime>45</breakTime>
      <IndexHtml>/index/</IndexHtml>
      <url>ssl.explorecell.com</url>
      <reconnectTime>15</reconnectTime>
      <port>443</port>
    </HTTPS>
</mainconfig>
```

The main module contains the remaining compiled LuaJIT scripts that were given the following names:

- Acom_define
- BGetSystemMsg
- main
- main_proto_WinHttpClient
- main_proto_WinHttpServer
- main_z_protoInterface
- thread_connect
- thread_recv
- thread_send
- thread_test

You can find all decompiled scripts in the `Download` section. As none of the current open-source Lua decompilers was able to successfully decompile the LuaJIT scripts, I've used an online service named Lua Decompiler.

File hash (SHA-256)	File name	Description
772293288ddc6c41dbe003e352b22a2c560a56023bc78c87bfef806482f1bf22	Comx64.dll	Loader for shellcode

There was a sample submitted to Virustotal whose list of exported functions look very similar to `UpdateControl.dll`, it's likely from the same developer.

It decrypts the following file path whose content it tries to read, decrypt and execute:

```
C:\ProgramData\Package Cache\{Ff964C81-895B-4433-A23F-42F30B600D93}.v102.sys\sys.dat
```

Unfortunately, we don't have the shellcode, thus this is where the analysis already comes to an end.

Conclusion

The use of (compiled) LuaJIT scripts rather than the more common Lua scripts used in malware in the past makes DreamLand an interesting piece of software. SentinelOne's investigation also concludes that it appears to be a work in progress, thus we will likely see more incidents where this malware will be utilized.

IOCs

Samples (SHA-256)

ceaec139a9370a4cd4eca876e7c4b3d51a013d3739b3f4d526fdfeab27cd2fc2
0b962ad02e8eef3c717ce6fcfda9587f92ebe9e7ed6ee93be6bc1103daa4e8bf
9bb5e7a76e66d105fa5a65728517b8d8f9465525465f92eb68a89705476b1d26
772293288ddc6c41dbe003e352b22a2c560a56023bc78c87bfef806482f1bf22

C2 domain

ssl[.]explorecell[.]com

Download

Samples and compiled/decompiled LuaJIT scripts (pw: "dreamland_infected"): DreamLand.zip

---