Earth Lusca Employs New Linux Backdoor, Uses Cobalt Strike for Lateral Movement

By By: Joseph C Chen Sep 18, 2023 Read time: 9 min (2356 words)

Published: 2023-09-18 · Archived: 2026-04-05 19:05:03 UTC

Malware

While monitoring Earth Lusca, we discovered an intriguing, encrypted file on the threat actor's server — a Linux-based malware, which appears to originate from the open-source Windows backdoor Trochilus, which we've dubbed SprySOCKS due to its swift behavior and SOCKS implementation.

In early 2021, we published a research paper discussing the operation of a China-linked threat actor we tracked as Earth Lusca. Since our initial research, the group has remained active and has even extended its operations, targeting countries around the world during the first half of 2023.

While monitoring the group, we managed to obtain an interesting, encrypted file hosted on the threat actor's delivery server. We were able to find the original loader of the file on VirusTotal and successfully decrypted it. Interestingly, the decrypted payload is a Linux-targeted backdoor that we have never seen before. The main execution routine and its strings show that it originates from the open-source Windows backdoor Trochilus, with several functions being re-implemented for Linux systems. We named this new Linux variant SprySOCKS, referring to the swift behaviors of Trochilus and the new Socket Secure (SOCKS) implementation inside the backdoor.

Analysis of the SprySOCKS backdoor reveals some interesting findings. The backdoor contains a marker that refers to the backdoor's version number. We have identified two SprySOCKS payloads that contain two different version numbers, indicating that the backdoor is still under development. In addition, we noticed that the implementation of the interactive shell is likely inspired from the Linux variant of the Derusbi malware.

Meanwhile, the structure of SprySOCKS's command-and-control (C&C) protocol is similar to one used by the RedLeaves backdoor, a remote access trojan (RAT) reported to be infecting Windows machines. It consists of two components, the loader and the encrypted main payload. The loader is responsible for reading, decrypting, and running the main payload.

Similar to the Windows version, the Linux variant analyzed in this report also consists of these two components. Previously, it was reported that RedLeaves was also built upon the publicly available source code of Trochilus.

So far, we have only observed SprySOCKS used by Earth Lusca. In this blog entry, we will provide more context on Earth Lusca's use of the malware, together with a thorough analysis of its components and capabilities.

Recent Earth Lusca activity

Earth Lusca remained active during the first half of 2023, with its attacks focusing primarily on countries in Southeast Asia, Central Asia, and the Balkans (with a few scattered attacks on Latin American and African countries). The group's main targets are government departments that are involved in foreign affairs, technology, and telecommunications.

Earth Lusca is now aggressively targeting the public-facing servers of its victims. Furthermore, we have seen them frequently exploiting server-based N-day vulnerabilities, including (but not limited to) the following:

Vulnerability	Description
CVE-2022-40684	An authentication bypass vulnerability in Fortinet FortiOS, FortiProxy and FortiSwitchManager
CVE-2022-39952	An unauthenticated remote code execution (RCE) vulnerability in Fortinet FortiNAC
CVE-2021-22205	An unauthenticated RCE vulnerability in GitLab CE/EE
CVE-2019-18935	An unauthenticated remote code execution vulnerability in Progress Telerik UI for ASP.NET AJAX
CVE-2019-9670 / CVE-2019-9621	A bundle of two vulnerabilities for unauthenticated RCE in Zimbra Collaboration Suite
ProxyShell (CVE-2021-34473, CVE-2021-34523v, CVE-2021-31207)	A set of three chained vulnerabilities that perform unauthenticated RCE in Microsoft Exchange

Table 1. The list of vulnerabilities exploited by Earth Lusca

Earth Lusca takes advantage of server vulnerabilities to infiltrate its victim's networks, after which it will deploy a web shell and install Cobalt Strike for lateral movement. The group intends to exfiltrate documents and email account credentials, as well as to further deploy advanced backdoors like ShadowPad and the Linux version of Winnti to conduct long-term espionage activities against its targets.

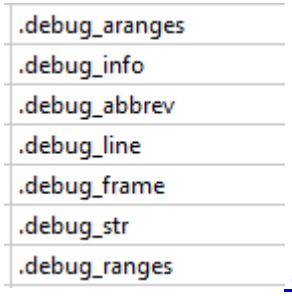
The "mandibule" loader component

At the beginning of our investigation, we observed a file named *libmonitor.so.2* hosted on Earth Lusca's delivery server. Without previous context, this seemed to be a binary file containing only random bytes, indicating that it is likely an encrypted payload. We used the unique file name to perform a search on VirusTotal that allowed us to identify a related ELF file (*SHA256: 65b27e84d9f22b41949e42e8c0b1e4b88c75211cbf94d5fd66edc4ebe21b7359*) named "mkmon". The ELF file could be used to decrypt the *libmonitor.so.2* file and recover its original payload, proving that "mkmon" is the loader bundled with *libmonitor.so.2*.

The loader was not developed from scratch — its developer used a publicly available Linux ELF injector called "mandibule" (the French word for mandible, or lower jaw). The original ELF injector project is a command-line tool with the ability to inject a payload into itself (self-injection) or into another project. As a typical command-

line tool, it prints usage text that lists supported parameters. The original injector also prints various debug messages to inform the user about the progress of the injection.

The threat actor used the mandibule project as a basis for its malware loader. The project creator removed the usage screen and the ability to inject to other processes, and then added a function to load and decrypt the second stage. We consider this job to be sloppily done since the developer did not bother to remove debug messages, and the loader was not stripped (that is, it was distributed with debug symbols). Indeed, the threat actor seemed to put minimum effort into modifying the original injector just to be able to make it load the payload.



open on a new tab

Figure 1. The loader distributed with debug information; note that the ".debug_*" sections are present

The debug messages displayed in Figure 2 have two distinct markers. The ">" marker is from the original mandibule project, while the "[+]" or "[-]" markers are debug messages added to the loader by the threat actor.

The name of the loader's process is set to "kworker/0:22" by the *prctl* command. Normally, kworker is a placeholder process for kernel worker threads. In this scenario, however, the "kworker" name has nothing to do with kernel worker threads. Instead, the loader abuses this name just to avoid suspicion when the user lists all running tasks via commands such as *ps* or *top*.

```
prctl(PR_SET_NAME, "kworker/0:22");
```

Figure 3. The name of the process is set to "kworker/0:22"

```
[redacted]@ubuntu-2004:~$ ps -e | grep kworker
      8 ?        00:00:00 kworker/0:0H-events_highpri
     24 ?        00:00:00 kworker/1:0H-kblockd
     37 ?        00:00:00 kworker/1:1-events
     90 ?        00:00:02 kworker/0:1H-kblockd
    103 ?        00:00:00 kworker/u4:3-events_unbound
    104 ?        00:00:00 kworker/u4:4-events_power_efficient
    105 ?        00:00:00 kworker/1:2-events
    110 ?        00:00:04 kworker/1:1H-kblockd
    121 ?        00:00:00 kworker/u5:0
    259 ?        00:00:02 kworker/0:3-events
   7694 pts/1    00:01:03 kworker/0:22
  30590 ?        00:00:00 kworker/0:0-events
  33473 ?        00:00:00 kworker/u4:0-events_unbound
```

Figure 4. The list of “kworker*” processes on an infected machine; the highlighted process is the analyzed loader

The loader accepts two command-line parameters: the path to the encrypted second stage file and the self-delete flag. The second stage is encrypted with an AES-ECB cipher, with the password being hard-coded in the loader.

```
DecryptString(elf_buf + 16, "uXQLESMXGaRMs6BL", dec_elf_buf, dec_elf_buf_length);
```

Figure 5. Function to decrypt the second stage

The loader is also responsible for setting the persistence. It copies itself and the encrypted second stage to the */usr/sbin/* directory (see debug notes “[+] rename loader ok” and “[+] rename server ok”). It then uses *chkconfig* or *systemd* to start the loader as a service. If the self-delete flag is set to “1”, then the originally executed loader and the encrypted stage files are both deleted.

The SprySOCKS component

While examining the decrypted second stage, visible strings revealed that it was statically compiled with HP-Socket project, a high-performance network framework of Chinese origin.

```
/home/hi/projects/prcsser_linux/HP-Socket/Linux/global/helper.cpp
/home/hi/projects/prcsser_linux/HP-Socket/Linux/src/common/IODispatcher.cpp
/home/hi/projects/prcsser_linux/HP-Socket/Linux/src/common/Thread.cpp
/home/hi/projects/prcsser_linux/HP-Socket/Linux/src/HttpServer.cpp
/home/hi/projects/prcsser_linux/HP-Socket/Linux/src/TcpServer.cpp
```

Figure 6. HP-Socket references among visible strings

Initialization procedure reveals a hard-coded AES-ECB password used for encrypting communication with the C&C server.

```
string_init(v4, "QFTHEYjzX3RBOMgZ");
```

Figure 7. AES password used for C&C communication

The C&C address and port are also hard-coded in the module, but they are not encrypted and are visible in plain text.

```
00197760: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00197770: 00 00 00 00 00 00 00 00|38 30 00 00 00 00 00 00  |        80
00197780: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
00197790: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00  |
001977A0: 00 00 00 00 00 00 00 00|6C 74 37 36 75 78 2E 63  |        lt76ux.c
001977B0: 6F 6E 66 65 6E 6F 73 2E|73 68 6F 70 00 00 00 00  | onfenos.shop
```

Figure 8. C&C server and port configuration

C&C communication consists of packets sent via TCP (Transmission Control Protocol). The packet has a header consisting of *0x12* bytes, followed by a base64-encoded, AES-ECB-encrypted message. Similar to Table B-2 in this previous analysis of the RedLeaves malware, the header contains some random and hard-coded values, plus the length of the payload (highlighted in blue in Figure 9).

```
00000000  42 c8 ef e8 bc bc ac ac  de 00 00 00 8c 8c 63 11   B....... ......c.
00000010  03 00 7a 54 65 2f 57 6e  67 64 78 7a 50 56 4c 6d   ..zTe/Wn gdxzPVLm
00000020  73 50 77 59 6f 38 44 78  71 77 4c 71 6d 6c 7a 66   sPwYo8Dx qwLqmlzf
00000030  76 41 76 64 75 6c 4a 73  79 6a 69 4f 50 41 38 68   vAvdulJs yjiOPA8h
00000040  45 68 30 44 47 59 57 46  62 31 63 49 4a 74 44 45   Eh0DGYWF b1cIJtDE
00000050  44 67 43 54 59 43 42 4c  4f 6b 59 70 55 65 31 63   DgCTYCBL OkYpUe1c
```

Figure 9. Example of a packet sent from the victim's machine to the C&C server

```
(a3 + 8) << 32) | 0xACACBCBC;
```

Figure 10. Fixed value of 0xACACBCBC; occurs in sent packets at offsets 4 – 7

The fixed value used in the original Trochilus is *0xAFAFBFBF*, while in the RedLeaves variant it is *0xBFD9CBAE*.

After decoding and decrypting the message, it reveals keywords such as "__handler", "__msgid", "__serial", and "clientid". Some of these keywords can be found in Trochilus, but more importantly, these messages closely resemble the RedLeaves communication protocol.

```
__handler|0
__msgid|99
__serial|0
clientid|080027189dcb52060A00FFFB8B17
```

Figure 11. Decoded and AES-ECB-decrypted message

The RAT implements several standard backdoor commands, including the collecting system information, starting an interactive shell, listing network connections, creating SOCKS proxy, uploading and downloading files, and other basic file operations (listing, deleting, renaming, and creating a directory). Table 2 shows message IDs and the approximate descriptions of what functions the messages are related to.

Message ID	**Notes**
0x09	Gets machine information
0x0a	Starts interactive shell
0x0b	Writes data to interactive shell
0x0d	Stops interactive shell

0x0e	Lists network connections (parameters: “ip”, “port”, “commName”, “connectType”)
0x0f	Sends packet (parameter: “target”)
0x14, 0x19	Sends initialization packet
0x16	Generates and sets clientid
0x17	Lists network connections (parameters: “tcp_port”, “udp_port”, “http_port”, “listen_type”, “listen_port”)
0x23	Creates SOCKS proxy
0x24	Terminates SOCKS proxy
0x25	Forwards SOCKS proxy data
0x2a	Uploads file (parameters: “transfer_id”, “size”)
0x2b	Gets file transfer ID
0x2c	Downloads file (parameters: “state”, “transferId”, “packageId”, “packageCount”, “file_size”)
0x2d	Gets transfer status (parameters: “state”, “transferId”, “result”, “packageId”)
0x3c	Enumerates files in root /
0x3d	Enumerates files in directory
0x3e	Deletes file
0x3f	Creates directory
0x40	Renames file
0x41	No operation
0x42	Is related to operations 0x3c – 0x40 (srcPath, destPath)

Table 2. List of handled messages and an explanation of their functions

The client information structure resembles the original CLIENT_INFO structure used by Trochilus, with some parameters being the same for both Trochilus and the malware we were analyzing. It is also worth noting the parameter “cpufrep”, which is likely a typo of “cpufreq” (CPU frequency).

```
typedef struct
{
        WCHAR                   clientid[60];   //客户端id
        ULONG                   connectIP;      //外网IP
        WCHAR                   computerName[MAX_COMPUTERNAME_LENGTH + 1];      //计算机名
        WIN_VER_DETAIL  windowsVersion; //操作系统
        BOOL                    bX64;   //是否x64平台
        SYSTEMTIME              installTime;    //客户端安装时间
        WCHAR                   avname[MAX_PATH];       //杀毒软件
        ULONG                   localIPList[MAX_LOCALIP_COUNT]; //本地IP列表
        USHORT                  localIPCount;   //本地IP列表个数
        WCHAR                   groups[MAX_ITEM]; //分组名
        WCHAR                   priv[MAX_ITEM]; //权限
        WCHAR                   proto[MAX_ITEM]; //协议
        WCHAR                   vercode[MAX_ITEM];//操作系统版本号
        WCHAR                   lang[MAX_ITEM];//语言
                int                             cpunum;
                int                             cpufrep;
                int                             memsize;
        WCHAR                   mods[MAX_MODNAMES];     //已安装模块
} CLIENT_INFO;
```

Figure 12. CLIENT_INFO structure in “ClientInfoCallbacks.h,” which is the Trochilus RAT

Further down in *ClientInfoManager.cpp*, the Trochilus RAT, you can see the internal names of the parameters from the CLIENT_INFO structure. Note that most of them have the same values as the listed parameters in Table 2. Furthermore, “cn”, “ip”, “groups”, “vercode”, “priv”, “lang”, “mem”, “os”, “x64”, and “cpufrep” are the same.

```
BOOL ClientInfoManager::ParseData( const CommData& commData,
{
        DECLARE_STR_PARAM(cn);
        DECLARE_STR_PARAM(ip);
        DECLARE_STR_PARAM(groups);
        DECLARE_STR_PARAM(vercode);
        DECLARE_STR_PARAM(priv);
        DECLARE_UINT64_PARAM(lang);
        DECLARE_UINT64_PARAM(mem);
        DECLARE_UINT64_PARAM(instime);
        DECLARE_UINT64_PARAM(os);
        DECLARE_UINT64_PARAM(x64);
        DECLARE_UINT64_PARAM(proto);
        DECLARE_UINT64_PARAM(cpufrep);
```

Figure 13. The internal names of CLIENT_INFO parameters, as defined in ClientInfoManager.cpp, the Trochilus RAT

Table 3. List of fields in the CLIENT_INFO structure of SprySOCKS

When a client is requested to create an interactive terminal, it first interacts with the master endpoint for the pseudo-terminal (PTY) subsystem (*/dev/ptmx*). Afterward, a new slave PTY is created with a unique device node name in the */dev/pts* directory.

After this, an *execve* command is launched with the parameter “[diskio]”, environment variables instructing it not to save session history (*HISTFILE=/dev/null*), and prompt string (*PS1*) containing current username (u), host name of machine (h), and working directory, which is (w) - (PS1=\\u@\\h:\\w \\$).

```
argv[1] = 0LL;
argv[0] = "[diskio]";
v12 = 0LL;
*(__m128i *)envp = _mm_unpacklo_epi64(
                     (__m128i)(unsigned __int64)"TERM=Linux",
                     (__m128i)(unsigned __int64)"HOME=/");
v11 = "PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin";
v10 = _mm_unpacklo_epi64(
        (__m128i)(unsigned __int64)"HISTFILE=/dev/null",
        (__m128i)(unsigned __int64)"PS1=\\u@\\h:\\w \\$ ");
execve("/bin/bash", argv, envp);
```

Figure 14. Creation of the interactive shell

When searching for the aforementioned strings, it’s possible to find a reference to YARA rules matching the Linux version of Derusbi. It is likely that the threat actor gained inspiration from the techniques used by other pieces of malware or possibly even had direct access to the Derusbi source code itself.

The environment ID (client ID) consists of two components. The first part is the MAC address of the first listed interface (the malware gets the first listed interface, but if this interface is "loopback interface", then this interface is skipped and the next interface is considered) with a length of 6 bytes; this, when converted to a hexadecimal string, has a length 12 bytes. The second part corresponds to processor features, returned by the CPUID instruction called with the “CPUID_GETFEATURES” parameter. The length of the result is 8 bytes; when converted to a hexadecimal string, this has a length of 16 bytes. Thus, the generated client ID has 14 bytes, and after its conversion to a hexadecimal string, it has 28 bytes.

Attribution

We observed the encrypted SprySOCKS payload hosted on the delivery server *207[.]148[.]75[.]122* in early June 2023. The server, which was operated by the Earth Lusca threat actor, also delivered executable files of Cobalt Strike and the Linux version of Winnti to its targets.

The SprySOCKS payload contains a version number (1.3.6) and the C&C domain *lt76ux[.]confenos[.]shop*. We found another SprySOCKS payload uploaded by other users on VirusTotal with a version number of 1.1 and which connected to the C&C domain *2e6veme8xs[.]bmssystemg188[.]us*. It’s worth noting that the sibling domain *rvxzn49eghqj[.]bmssystemg188[.]us* resolved to *38[.]60[.]199[.]208*, which overlapped with *793tggz7mw91[.]itcom666[.]live*. The domain *itcom666[.]live* is a known C&C domain attributed to Earth Lusca.

Conclusion

In this report, we discussed the new backdoor SprySOCKS used by Earth Lusca, which expands the group's Linux arsenal. Recently, the threat actor has been highly aggressive in targeting the public-facing servers of its victims by exploiting known vulnerabilities.

It is important that organizations proactively manage their attack surface, minimizing the potential entry points into their system and reducing the likelihood of a successful breach. Businesses should regularly apply patches and update their tools, software, and systems to ensure their security, functionality, and overall performance.

Cutting-edge and adaptable security solutions like Trend Micro XDRproducts play a pivotal role in safeguarding organizations against Earth Lusca and other threat actors. These technologies excel at gathering and connecting activity data across various channels, from emails and endpoints to servers, cloud workloads, and networks. This comprehensive approach empowers organizations with a high level of security detection and investigation capabilities, setting it apart from conventional security solutions.

Indicators of Compromise (IOCs)

The indicators of compromise for this entry can be found here.

Tags
