

## Cyber Challenge ACAD CSIRT 2025 #2



Nama Tim : Fay The Demon King

Yusuf Darmawan

Dhio Zahwan Aryasetyo



Ahmad Fayaadh Baisa

## Challenge 2

### Deskripsi singkat challenge

Pada challenge 2 terdapat sebuah file bernama filependukungchallenge2.zip yang berisi 2 file bernama [sample1.tar.gz](#) dan [sample2.tar.gz](#) kedua file ini masih berupa file hasil kompresi gunzip dan tar.

filependukungchallenge2.zip 2 item

Nama	Terakhir diubah	Ukuran file
 sample1.tar.gz	1 Jun 2025	2 KB
 sample2.tar.gz	1 Jun 2025	321 KB

Gambar 1.1 filependukungchallenge2.zip yang perlu di download

### Sample1

#### Jenis File, Checksum, & Struktur File

Pertama-tama kita cek file sample1 bertipe apa dan kita cek md5sum, dan shasum sebelum kita ekstrak.

```
(kali@kali)-[~/Downloads/acadefense/chall2/new]
$ file sample1.tar.gz
sample1.tar.gz: gzip compressed data, from Unix, original size modulo 2^32 20480

(kali@kali)-[~/Downloads/acadefense/chall2/new]
$ md5sum sample1.tar.gz
78eda2067a1ee97878023a0035eac6b3 sample1.tar.gz

(kali@kali)-[~/Downloads/acadefense/chall2/new]
$ sha1sum sample1.tar.gz
1d800c94b7067aa9565d79849182a7b4401b0036 sample1.tar.gz
```

Gambar 1.2 Cek file sample1.tar.gz

setelah dilakukan file dan checksum kami mendapati kalau file ini masih dikompresi menggunakan gzip dan tar, juga hasil checksumnya yaitu:

- md5 = 78eda2067a1ee97878023a0035eac6b3
- sha1 = 1d800c94b7067aa9565d79849182a7b4401b0036
- sha256 =  
58898bd42c5bd3bf9b1389f0eee5b39cd59180e8370eb9ea838a0b327bd6fe  
47

Selanjutnya kami mengekstrak filenya hingga ke bentuk paling awal sebelum kompresi menggunakan gunzip dan tar, setelah itu kami cek menggunakan file dan mendapatkan kalau file sample1 bertipe PE32 executable yang biasa digunakan untuk OS Windows.

```
(kali@kali)-[~/Downloads/acadefense/chall2]
$ gunzip sample1.tar.gz

(kali@kali)-[~/Downloads/acadefense/chall2]
$ tar -xvf sample1.tar
sample1

(kali@kali)-[~/Downloads/acadefense/chall2]
$ file sample1
sample1: PE32 executable (console) Intel 80386, for MS Windows, 3 sections
```

Gambar 1.3 Ekstrak file sample1.tar.gz

Analisis struktur file :

- *PE32* : Ini adalah file executable Windows 32-bit (Portable Executable format).
- *console* : Berarti ini adalah aplikasi berbasis command-line.
- *3 sections* : Biasanya .text, .data, dan .rdata atau sejenis. Mungkin binary ini sederhana (indikasi malware *custom* atau *dropper* minimalis).

Hal ini juga menandakan kalau file sample1 hanya bisa digunakan pada Sistem Operasi Windows dengan arsitektur 32 bit.

```
(kali@kali)-[~/Downloads/acadefense/chall2/new]
$ md5sum sample1
bb7425b82141a1c0f7d60e5106676bb1 sample1

(kali@kali)-[~/Downloads/acadefense/chall2/new]
$ sha1sum sample1
9dce39ac1bd36d877fdb0025ee88fdaff0627cdb sample1
```

Gambar 1.4 Checksum file sample1.tar.gz

Analisa selanjutnya kami lakukan checksum pada filenya dengan md5sum dan sha1sum dan mendapatkan hasil sebagai berikut:

- md5 = bb7425b82141a1c0f7d60e5106676bb1
- sha1 = 9dce39ac1bd36d877fdb0025ee88fdaff0627cdb

```
(kali@kali)-[~/filependukungchallenge2]
$ upx -t sample1
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2024
UPX 4.2.2      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 3rd 2024

upx: sample1: NotPackedException: not packed by UPX

Tested 0 files.

(kali@kali)-[~/filependukungchallenge2]
$ binwalk -e sample1
/usr/lib/python3/dist-packages/binwalk/core/magic.py:431: SyntaxWarning: invalid escape sequence '\.'
  self.period = re.compile("\.")

DECIMAL      HEXADECEIMAL  DESCRIPTION
-----
0            0x0      Microsoft executable, portable (PE)
```

Gambar 1.5 Cek file tersembunyi sample1

File tidak terkompresi menggunakan UPX (Ultimate Packer for eXecutables), yang merupakan salah satu packer umum untuk executable Windows. Maupun menggunakan binwalk hanya mendeteksi bahwa file adalah PE (Portable Executable).

Tidak ditemukan indikasi adanya kompresi, enkripsi, atau embedded file seperti archive (ZIP, CAB) atau payload tersembunyi di dalam file tersebut.

### Strings sample1 & File API Mencurigakan

```
CloseHandle
UnmapViewOfFile
IsBadReadPtr
MapViewOfFile
CreateFileMappingA
CreateFileA
FindClose
FindNextFileA
FindFirstFileA
CopyFileA
KERNEL32.dll
malloc
exit
MSVCRT.dll
_exit
_XcptFilter
__p__initenv
__getmainargs
__initterm
__setusermatherr
__adjust_fdiv
__p__commode
__p__fmode
__set_app_type
_except_handler3
_controlfp
_stricmp
kerne132.dll
kernel32.dll
.exe
C:\*
C:\windows\system32\kerne132.dll
Kernel32.
Lab01-01.dll
C:\Windows\System32\Kernel32.dll
WARNING_THIS_WILL_DESTROY_YOUR_MACHINE
```

Gambar 1.6 strings file sample1

Saat kami lakukan analisa menggunakan strings sample1 terlihat ada beberapa strings yang mencurigakan seperti :

*WARNING\_THIS\_WILL\_DESTROY\_YOUR\_MACHINE*

Pesan ini bersifat ancaman dan tidak biasa ditemukan pada file executable biasa. Sangat mencurigakan dan menunjukkan niat merusak.

*C:\Windows\System32\kernel32.dll* dan *C:\windows\system32\kerne132.dll* Path file sistem. kerne132.dll adalah penulisan yang mirip dengan kernel32.dll, kemungkinan teknik typosquatting atau DLL hijacking. dan ada fungsi yang biasanya digunakan di Windows API seperti :

- CloseHandle

Menutup handle (akses terbuka) ke file, proses, atau objek sistem lain.

- UnmapViewOfFile

Melepaskan (unmap) file dari memori.

- IsBadReadPtr

Mengecek apakah pointer (alamat memori) bisa dibaca atau tidak untuk memastikan program tidak crash saat mencoba membaca data dari pointer.

- MapViewOfFile

Memetakan bagian file ke dalam memori (RAM) untuk mengakses isi file langsung di memori tanpa baca file manual.

- CreateFileMappingA

Membuat atau membuka file besar dan mengaksesnya seolah-olah datanya ada di RAM.

- CreateFileA

CreateFileA berfungsi untuk membuka atau membuat file.

- FindClose

FindClose berfungsi untuk mengakhiri proses pencarian file (menutup handle).

- FindNextFileA

FindNextFileA untuk melanjutkan pencarian ke file berikutnya yang cocok.

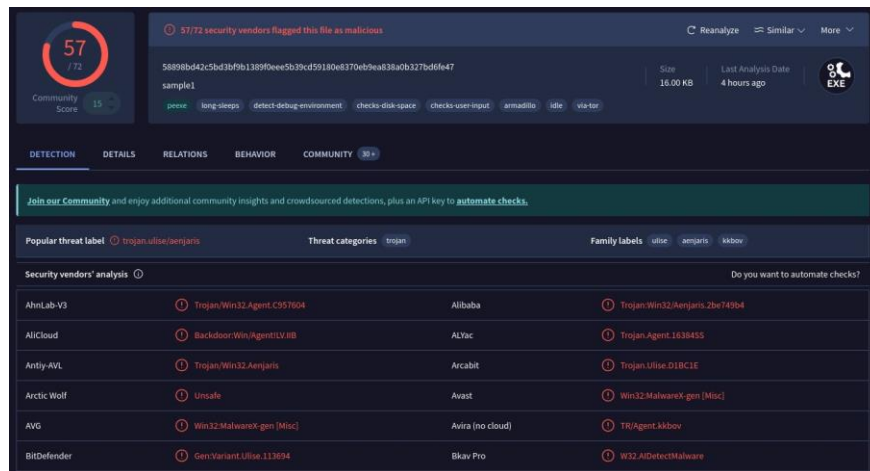
- FindFirstFileA

FindFirstFileA untuk memulai pencarian file yang cocok dengan pola tertentu.

- CopyFileA

CopyFileA berfungsi untuk menyalin suatu file dari satu lokasi ke lokasi yang lain. fungsi-fungsi tersebut adalah fungsi dari Windows API seperti halnya malware ini bertujuan untuk mengakses sistem file dan memori, kami juga mendapatkan kalau malware ini ditulis dalam bahasa C atau C++ karena ada MSVCRT.dll yang merupakan komponen microsoft visual C, ada juga file mencurigakan yang bisa menipu yaitu kernel32.dll sedangkan sistem file yang seharusnya adalah kernel32.dll, ini sangat berbahaya karena kalau pengguna salah buka bisa menjadi masalah yang besar karena ini menyangkut file sistem.

Selanjutnya kami coba masukkan file sample1 ke virustotal dan mendapati bahwa file ini adalah sebuah trojan dengan skor 57/72, ini memastikan kalau filenya adalah malware



Gambar 1.7 hasil virustotal file sample1

History	
Creation Time	2010-12-19 16:16:19 UTC
First Seen In The Wild	2012-01-08 02:19:06 UTC
First Submission	2012-02-16 07:31:54 UTC
Last Submission	2025-06-02 03:42:54 UTC
Last Analysis	2025-06-01 20:58:30 UTC

Gambar 1.8 history file sample1

Kesimpulan awal yang kami dapatkan adalah :

File sudah dipastikan sebuah malware bertipe trojan yang dimaksudkan untuk menyerang sistem operasi windows dengan memasuki bagian sistem file dan memori, malware ditulis dalam bahasa C atau C++.

## Analisa cara kerja malware sample1

```
if (param_1 == 2) {
    pcVar19 = s_WARNING_THIS_WILL_DESTROY_YOUR_M_004030b0;
    pbVar4 = *(byte **)(param_2 + 4);
}
```

Gambar 1.6 Potongan code malware (bag 1)

di awal program malware melakukan cek parameter `param_1 == 2` dan jika benar maka akan memberikan peringatan.



```

44     do {
45         bVar2 = *pbVar4;
46         bVar25 = bVar2 < (byte)*pcVar19;
47         if (bVar2 != *pcVar19) {
48 LAB_00401488:
49             iVar5 = (1 - (uint)bVar25) - (uint)(bVar25 != 0);
50             goto LAB_0040148d;
51         }
52         if (bVar2 == 0) break;
53         bVar2 = pbVar4[1];
54         bVar25 = bVar2 < ((byte *)pcVar19)[1];
55         if (bVar2 != ((byte *)pcVar19)[1]) goto LAB_00401488;
56         pbVar4 = pbVar4 + 2;
57         pcVar19 = (char *)((byte *)pcVar19 + 2);
58     } while (bVar2 != 0);

```

Gambar 1.9 Potongan code malware (bag 2)

melakukan validasi antara data input dengan string, kemungkinan digunakan untuk memastikan payload hanya dijalankan dalam kondisi sesuai.

```

61     if (iVar5 == 0) {
62         hFile = CreateFileA(s_C:\Windows\System32\Kernel32.dll_0040308c,0x80000000,1,
63             (LPSECURITY_ATTRIBUTES)0x0,3,0,(HANDLE)0x0);
64         pvVar6 = CreateFileMappingA(hFile,(LPSECURITY_ATTRIBUTES)0x0,2,0,0,(LPCSTR)0x0);
65         pvVar7 = MapViewOfFile(pvVar6,4,0,0,0);
66         pvVar6 = CreateFileA(s_Lab01-01.dll_0040307c,0x10000000,1,(LPSECURITY_ATTRIBUTES)0x0,3,0,
67             (HANDLE)0x0);
68         if (pvVar6 == (HANDLE)0xffffffff) {
69             /* WARNING: Subroutine does not return */
70             exit(0);
71         }
72         hFileMappingObject = CreateFileMappingA(pvVar6,(LPSECURITY_ATTRIBUTES)0x0,4,0,0,(LPCSTR)0x0);
73         if (hFileMappingObject == (HANDLE)0xffffffff) {
74             /* WARNING: Subroutine does not return */
75             exit(0);
76         }

```

Gambar 1.10 Potongan code malware (bag 3)

Program membuka file kernel32.dll dan memetakan isi file ke memori, lalu membuat file baru bernama kernel132.dll sebagai tempat modifikasi dari kernel32.dll dan menukar isinya. dengan tujuan untuk patching, code injection, dan dll hijacking.

```

82     iVar22 = *(int *)((int)pvVar7 + 0x3c) + (int)pvVar7;
83     puVar9 = (undefined4 *)FUN_00401040(*(undefined4 *)iVar22 + 0x78),iVar22,pvVar7);
84     iVar5 = *(int *)((int)pvVar8 + 0x3c) + (int)pvVar8;
85     puVar10 = (undefined4 *)FUN_00401040(*(undefined4 *)iVar5 + 0x78),iVar5,pvVar8);
86     local_38 = (int *)FUN_00401040(puVar9[7],iVar22,pvVar7);
87     puVar11 = (ushort *)FUN_00401040(puVar9[9],iVar22,pvVar7);
88     puVar12 = (undefined4 *)FUN_00401040(puVar9[8],iVar22,pvVar7);
89     uVar16 = *(uint *)(iVar5 + 0x7c);
90     iVar5 = FUN_00401070(*(undefined4 *)iVar5 + 0x78),iVar5,pvVar8);
91     puVar20 = puVar9;
92     puVar23 = puVar10;
93     for (uVar15 = uVar16 >> 2; uVar15 != 0; uVar15 = uVar15 - 1) {
94         *puVar23 = *puVar20;
95         puVar20 = puVar20 + 1;
96         puVar23 = puVar23 + 1;
97     }
98     for (uVar16 = uVar16 & 3; uVar16 != 0; uVar16 = uVar16 - 1) {
99         *(undefined1 *)puVar23 = *(undefined1 *)puVar20;
100         puVar20 = (undefined4 *)((int)puVar20 + 1);
101         puVar23 = (undefined4 *)((int)puVar23 + 1);
102     }

```

Gambar 1.11 Potongan code malware (bag 4)

Program sedang mengurai file dll asli sambil menangkap informasi seperti offset ke section table, entry point, dan section alignment, juga melakukan transformasi byte per byte yang menunjukkan kalau malware sedang berusaha menyalin isi dari kernel32.dll ke buffer hasil modifikasi, modifikasi dilakukan dalam 2 loop.

```

121 do {
122     if ((*local_38 != 0) && (local_2c = 0, puVar9[6] != 0)) {
123         piVar18 = puVar20 + iVar17;
124         local_44 = (undefined2 *)((int)puVar20 + iVar17 * 2 + iVar3 * 4);
125         local_40 = puVar11;
126         local_3c = puVar12;
127         do {
128             if (*local_40 == local_28) {
129                 pcVar13 = (char *)FUN_00401040(*local_3c, iVar22, pvVar7);
130                 uVar16 = 0xffffffff;
131                 pcVar21 = pcVar13;
132                 do {
133                     if (uVar16 == 0) break;
134                     uVar16 = uVar16 - 1;
135                     cVar1 = *pcVar21;
136                     pcVar21 = pcVar21 + 1;
137                 } while (cVar1 != '\0');
138                 pcVar21 = pcVar13;
139                 pcVar24 = pcVar19;
140                 for (uVar15 = ~uVar16 >> 2; uVar15 != 0; uVar15 = uVar15 - 1) {
141                     *(undefined4 *)pcVar24 = *(undefined4 *)pcVar21;
142                     pcVar21 = pcVar21 + 4;
143                     pcVar24 = pcVar24 + 4;
144                 }
145                 for (uVar16 = ~uVar16 & 3; uVar16 != 0; uVar16 = uVar16 - 1) {
146                     *pcVar24 = *pcVar21;
147                     pcVar21 = pcVar21 + 1;
148                     pcVar24 = pcVar24 + 1;
149                 }
150                 *local_44 = (undefined2)param_2;
151                 piVar18[iVar3 * 2] = (int)(pcVar19 + iVar5);
152                 uVar16 = 0xffffffff;
153                 pcVar21 = pcVar13;
154                 do {
155                     if (uVar16 == 0) break;
156                     uVar16 = uVar16 - 1;
157                     cVar1 = *pcVar21;
158                     pcVar21 = pcVar21 + 1;
159                 } while (cVar1 != '\0');
160                 pcVar19 = pcVar19 + ~uVar16;
161                 *piVar18 = (int)(pcVar19 + iVar5);
162                 *(undefined4 *)pcVar19 = s_Kernel32._00403070._0_4_;
163                 *(undefined4 *)pcVar19 + 4 = s_Kernel32._00403070._4_4_;
164                 pcVar19[8] = s_Kernel32._00403070[8];
165                 uVar16 = 0xffffffff;
166                 pcVar21 = pcVar13;
167                 do {

```

Gambar 1.12 Potongan code malware (bag 5)

Dalam operasi transformasi data kernel32.dll Fungsi FUN\_00401040() digunakan berulang kali, dalam proses modifikasi malware melakukan banyak operasi XOR, penggandaan string, offset pointer, dll. Ini menandakan malware memodifikasi kernel32.dll lalu menyimpan hasilnya di kernel32.dll\_0040304c.

```

207     CloseHandle(hFile);
208     CloseHandle(pvVar6);
209     BVar14 = CopyFileA(s_Lab01-01.dll_0040307c, s_C:\windows\system32\kernel32.dll_0040304c, 0);
210     if (BVar14 == 0) {
211         /* WARNING: Subroutine does not return */
212         exit(0);
213     }
214     FUN_004011e0(&DAT_00403044, 0);
215 }
216 }
217 return 0;
218 }

```

Gambar 1.13 Potongan code malware (bag 6)

Pada akhir program terdapat fungsi penutup, untuk menjalankan payload utama untuk mendaftarkan file dll baru, memuat ke dalam proses, dan menghapus jejak.



## API yang terkonfirmasi mencurigakan:

### 1. CopyFileA

CopyFileA yang digunakan untuk menyalin file dari satu lokasi ke lokasi lain. Dalam konteks malware ini, CopyFileA digunakan untuk menyalin file DLL hasil modifikasi (Lab01-01.dll) ke direktori sistem Windows, menggantikan kernel32.dll, salah satu file inti sistem operasi Windows.

Mitre att&ck terkait

Tactic:	Defense Evasion
Technique:	T1036: Masquerading
Sub-Technique:	T1036.003: Masquerade Task or Service

### 2. CreateFileMappingA + MapViewOfFile

Dengan menggunakan CreateFileMappingA dan MapViewOfFile, malware dapat mengakses isi file DLL sistem dan memodifikasinya di dalam memori. Ini memungkinkan penggantian atau penyisipan kode tanpa jejak langsung ke disk, atau menghindari deteksi antivirus berbasis disk scanning.

Mitre att&ck terkait

Tactic:	Defense Evasion
Technique:	T1055: Process Injection

CreateFileMappingA digunakan untuk membuat pemetaan file ke memori, dan MapViewOfFile digunakan untuk membuat representasi (view) file dalam memori virtual proses, dengan proses sebagai berikut:

- File penting sistem (kernel32.dll) dipetakan ke memori.
- Kemudian, file Lab01-01.dll juga dipetakan ke memori.
- Setelah itu, konten Lab01-01.dll disalin ke lokasi file kernel32.dll, baik seluruhnya atau sebagian, melalui manipulasi langsung terhadap isi memori.
- Teknik ini memungkinkan modifikasi konten file sistem secara langsung, tanpa menulis file baru secara eksplisit terlebih dahulu, sehingga lebih stealthy.

Mitre att&ck terkait

Tactic:	Defense Evasion
Technique:	T1027: Obfuscated Files or Information
Sub-Technique:	T1027.004: Compile After Delivery

## **Kesimpulan hasil analisa sample1**

dari hasil analisa program malware sample1 kami menemukan tujuan dari programnya:

1. Memodifikasi kernel32.dll menjadi versi yang telah disusupi.
2. Melakukan dekripsi / decoding terhadap bagian tertentu dari PE file.
3. Menghindari deteksi dengan pemrosesan XOR dan loop kompleks.
4. Menyimpan versi patch kernel32.dll baru untuk nanti digunakan kembali atau disalahgunakan.
5. Mungkin menyisipkan shellcode, hook API, atau backdoor dalam DLL sistem.

## **Potensi Dampak:**

1. Eksekusi Payload Berbahaya:  
Trojan jenis "Agent" atau "Dropper" berpotensi men-download dan menjalankan malware tambahan secara diam-diam.
2. Backdoor / Remote Access:  
Dapat membuka celah bagi penyerang untuk mengakses sistem secara remote, termasuk mengambil kendali penuh.
3. Pencurian Data / Credential:  
Potensi mencuri data sensitif seperti username, password, token login, atau dokumen penting pengguna.
4. Perusakan Sistem dan File Penting  
String seperti WARNING\_THIS\_WILL\_DESTROY\_YOUR\_MACHINE dan referensi ke file sistem seperti kernel32.dll menunjukkan potensi kerusakan destruktif. Malware ini bisa saja dirancang untuk menghapus file sistem, membuat OS gagal booting, atau memicu kerusakan permanen.
5. Persistence & Evasion:  
Malware ini dapat menggunakan teknik seperti "self-delete", "long sleep", dan anti-debug untuk menghindari deteksi dan bertahan lama di sistem.
6. Penyebaran Lateral:  
Bisa digunakan sebagai pintu masuk untuk menyebar ke sistem lain dalam jaringan internal (lateral movement).

## **Rekomendasi Mitigasi:**

1. Isolasi Sistem Terdampak

Putuskan koneksi internet dan jaringan dari sistem yang terinfeksi untuk mencegah penyebaran lebih lanjut.

2. Lakukan Full Scan Antivirus/EDR

Gunakan antivirus atau Endpoint Detection & Response (EDR) yang up-to-date untuk pembersihan menyeluruh.

3. Analisis File di Sandbox / Virtual Machine

Jika perlu memahami perilaku file lebih lanjut, jalankan di lingkungan terisolasi (VM atau sandbox).

4. Reset Credential

Ganti semua password yang mungkin telah dikompromikan dari sistem yang terdampak.

5. Perkuat Endpoint Protection

Aktifkan firewall, proteksi real-time, dan pembaruan OS serta perangkat lunak.

6. Catat dan Simpan Hash File

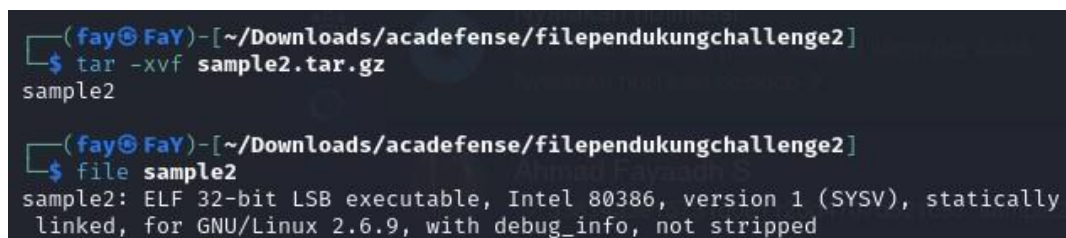
Gunakan hash (SHA256/MD5) untuk membuat rule pada firewall, IDS/IPS, dan antivirus agar mendeteksi file serupa di masa depan.

7. Pulihkan Sistem dari Backup

Jika sistem sudah terinfeksi parah atau file penting sudah rusak, solusi terbaik adalah melakukan pemulihan dari backup bersih yang dibuat sebelum infeksi terjadi. Pastikan backup tidak mengandung file malware tersebut.

## Sample2

Ekstrak file sample2.tar.gz menggunakan perintah tar. Setelah berhasil diekstrak, kami melakukan identifikasi jenis file dengan perintah file dan mendapatkan informasi bahwa file ini merupakan executable Linux dengan format ELF 32-bit. Untuk memastikan integritas dan mencatat hash file.



```
(fay@FaY)-[~/Downloads/acadefense/filependukungchallenge2]
$ tar -xvf sample2.tar.gz
sample2

(fay@FaY)-[~/Downloads/acadefense/filependukungchallenge2]
$ file sample2
sample2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically
linked, for GNU/Linux 2.6.9, with debug_info, not stripped
```

Gambar 2.1 Ekstrak file sample2.tar.gz

Analisis awal file :

- *ELF 32-bit LSB executable* : Ini adalah binary Linux 32-bit dalam format ELF.
- *Intel 80386* : Kompatibel dengan arsitektur x86.

- *Statically linked* : Semua library dibundel dalam binary, jadi analisis akan dilakukan langsung dari file tanpa dependency eksternal.
- *With debug\_info* : Ada simbol debugging yang memudahkan reverse engineering (nama fungsi, variabel bisa terlihat).
- *Not stripped* : Simbol-simbol internal belum dihapus yang sangat menguntungkan buat analisis. Artinya simbol debug masih ada sehingga sangat memudahkan proses reverse engineering.

File sample2 merupakan executable ELF 32-bit untuk sistem operasi Linux. Oleh karena itu, file tersebut hanya bisa dijalankan pada Sistem Operasi Linux berbasis arsitektur Intel 80386 (x86). Sistem operasi yang kompatibel minimal adalah GNU/Linux kernel versi 2.6.9.



```
(fay@FaY)-[~/Downloads/acadefense/filependukungchallenge2]
$ md5sum sample2
53814a3e15731d0e1125d4707d521ce3 sample2

(fay@FaY)-[~/Downloads/acadefense/filependukungchallenge2]
$ sha1sum sample2
40a50e4d2003c70257ddf36d4ebb5a2e80dc4cf5 sample2
```

Gambar 2.2 Checksum file sample2.tar.gz

Analisa selanjutnya kami lakukan checksum pada filenya dengan md5sum dan sha1sum dan mendapatkan hasil sebagai berikut:

- md5 = 53814a3e15731d0e1125d4707d521ce3
- sha1 = 40a50e4d2003c70257ddf36d4ebb5a2e80dc4cf5

Saat kami lakukan analisa menggunakan strings sample2 terdapat

#### 1. Path Mencurigakan

- /boot/.IptabLes
- /usr/.IptabLes
- /etc/rc.d/init.d/IptabLes
- /etc/init.d/IptabLes
- /tmp/IptabLes
- /etc/rc3.d/S55IptabLes
- /etc/rc.d/rc\*.d/\*IptabLes

```

/proc
/proc/%s/exe
tasklistlock
/proc/self/exe
/boot/.IptabLes
/usr/.IptabLes
xxxx
:KGCG~PE,
/rc.d/r
/boot/IptabLes xxxx xxx
/etc%sc%d.d/S55IptabLes
/etc/rc3.d/S55IptabLes
/etc/rc.d/rc3.d/S55IptabLes
/delallmykkk
nohup cp %s %s>/dev/null
nohup chmod 777 %s>/dev/null
/etc/rc.d/init.d/IptabLes
/etc/rc.d/IptabLes
/boot/IptabLes
/delxxaazz

```

Gambar 2.3 strings file sample2 (bag 1)

File ini mengkloning dirinya ke banyak lokasi, lokasi ini sering digunakan untuk memastikan malware dijalankan saat sistem booting, yang sesuai dengan karakteristik persistence yang disebutkan di laporan VirusTotal.

## 2. Perintah Linux Shell

- #!/bin/sh
- ps -axu | grep .IptabLes | awk '{print \$2}' | xargs kill -9
- rm -f %s
- chmod 777 %s
- echo %08xT%dT%d>>/tt.txt
- nohup sh /delallmykkk > /dev/null

Analisis digunakan untuk:

- Membunuh proses yang sedang berjalan dengan nama .IptabLes
- Menghapus diri sendiri atau file lain (self-destruct) ● Memberikan hak akses penuh (777) ke file tertentu
- Menjalankan script di background agar tetap aktif meskipun terminal ditutup

## 3. Fungsi dan Modul serangan DDOS

- SynFloodThread
- DnsFloodThread
- SynFloodSendThread
- DnsFloodSendThread
- DnsFloodBuildThread
- SynFloodBuildThread

```

(kali@kali) [~/Filependukungchallenge2]
$ strings sample2 | grep "Flood"
SynFloodSendThread
DnsFloodSendThread
DnsFloodBuildThread
SynFloodThread
SynFloodBuildThread
DnsFloodThread
DnsFloodBuildThread
DnsFloodSendThread
DnsFloodThread
SynFloodThread
SynFloodBuildThread
SynFloodSendThread
SynFloodBuildThread
DnsFloodSendThread
SynFloodSendThread
DnsFloodBuildThread
DnsFloodThread
SynFloodThread

```

Gambar 2.4 strings file sample2 (bag 2)

### Analisis:

File ini berisi modul serangan DDoS jenis SYN Flood dan DNS Flood, dengan thread khusus yang dibuat untuk mengirim paket berbahaya secara paralel. Nama-nama fungsi ini umum ditemukan dalam malware botnet.

#### 4. Url yang mencurigakan

```

#!/bin/bash
sleep 3
kill %d
sleep 1
rm -f %s
rm -rf "$0"
nohup sh /delxxaazz>/dev/null&
http://www.yahoo.com.http://www.baidu.com.http://www.china.com.http://www.ife
ng.com

```

Gambar 2.5 strings file sample2 (bag 3)

### Indikator Jaringan dan Serangan DDoS:

- String seperti :
  1. <http://www.yahoo.com>.
  2. <http://www.baidu.com>.
  3. <http://www.china.com>.
  4. <http://www.ifeng.com>.

menunjukkan adanya daftar domain yang kemungkinan menjadi target serangan. Trojan DDoS sering kali memiliki daftar target seperti ini untuk melakukan flooding.

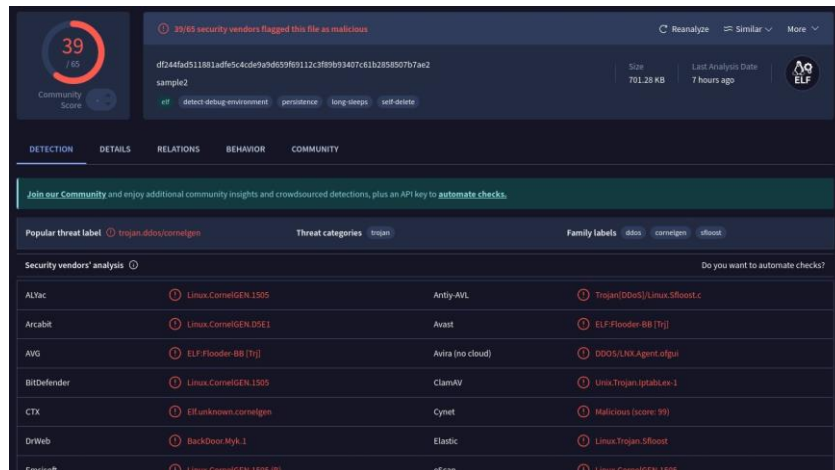
### Perilaku Trojan:

- `rm -rf "$0"` menunjukkan mekanisme penghapusan diri untuk menghindari deteksi, yang umum pada trojan.
- `sleep 3` dan `sleep 1` sering digunakan untuk menunda aktivitas agar serangan tidak langsung terdeteksi.

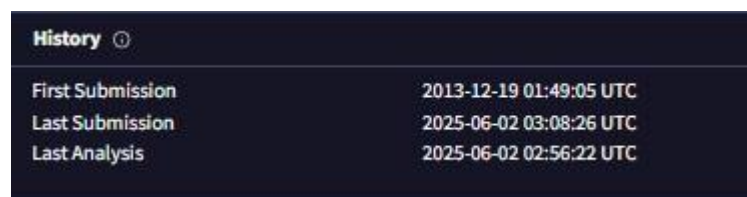


- `nohup sh /delxxaazz>/dev/null&` menjalankan proses di latar belakang tanpa output, menunjukkan upaya untuk bertahan (persistence) di sistem.

Selanjutnya kami coba masukkan file sample1 ke virustotal dan mendapati bahwa file ini adalah sebuah trojan dengan skor 39/65, ini memastikan kalau filenya adalah malware



Gambar 2.6 hasil virustotal file sample2



Gambar 2.7 history file sample2

Beberapa informasi penting dari hasil tersebut adalah:

- File ini termasuk jenis trojan, yaitu program jahat yang menyamar seperti program biasa, tetapi diam-diam menjalankan aktivitas berbahaya di sistem korban.
- Banyak vendor keamanan mengelompokkan file ini sebagai trojan yang digunakan untuk serangan DDoS, dan diberi label trojan.ddos/cornelgen.
- File ini termasuk dalam keluarga malware "correlgen" dan "sfloost", yang dikenal menyerang sistem berbasis Linux dengan melakukan serangan DDoS atau mengendalikan sistem dari jarak jauh.

Beberapa label deteksi mencakup:

- Trojan[DDoS]/Linux.Sfloost.c
- Linux.CornelGEN.1505
- Unix.Trojan.IptabLex-1
- ELF:Flooder-BB [Trj]
- BackDoor.Myk.1

Isi string menunjukkan berbagai aktivitas mencurigakan, yang konsisten dengan perilaku malware atau backdoor pada sistem Linux, terutama bot DDoS seperti Trojan Iptables.

File ini juga memiliki beberapa karakteristik seperti:

- detect-debug-environment
- persistence
- long-sleeps
- self-delete

Karakteristik tersebut menunjukkan bahwa malware ini mencoba menghindari analisis, bertahan lama di sistem korban, dan bisa menghapus dirinya sendiri untuk menghilangkan jejak.

Kesimpulan awal yang kami dapatkan adalah:

File sample2 dipastikan sebagai malware bertipe trojan correlgen yang menyerang sistem operasi Linux, dengan tujuan utama menjadikan sistem botnet untuk melakukan serangan DDoS dan mempertahankan keberadaannya di sistem korban melalui mekanisme persistence, long-sleeps, dan self-delete.

## Analisa cara kerja malware sample2

```
2int main(int argc, char **argv)
3
4{
5    uint uVar1;
6    __pid_t __pid;
7    int iVar2;
8    char *pcVar3;
9    char *pcVar4;
10   undefined1 uVar5;
11
12   uVar5 = &stack0x00000000 == (undefined1 *)0x30;
13   init_daemon();
14   init_daemon();
15   memset(&g_mainsrvinfo, 0, 600);
16   deinfo(AAAAAA);
17   uVar1 = time((time_t *)0x0);
18   srandom(uVar1);
19   memset(g_mainsrvinfo.mypropath, 0, 0x100);
20   readlink("/proc/self/exe", g_mainsrvinfo.mypropath, 0x100);
21   iVar2 = 0xe;
22   pcVar3 = g_mainsrvinfo.mypropath;
23   pcVar4 = "/boot/.IptabLes";
24   do {
25       if (iVar2 == 0) break;
26       iVar2 = iVar2 + -1;
27       uVar5 = *pcVar3 == *pcVar4;
28       pcVar3 = pcVar3 + 1;
29       pcVar4 = pcVar4 + 1;
30   } while ((bool)uVar5);
31   if (!(bool)uVar5) {
32       iVar2 = 0xd;
33       pcVar3 = g_mainsrvinfo.mypropath;
34       pcVar4 = "/usr/.IptabLes";
35       do {
36           if (iVar2 == 0) break;
37           iVar2 = iVar2 + -1;
38           uVar5 = *pcVar3 == *pcVar4;
39           pcVar3 = pcVar3 + 1;
40           pcVar4 = pcVar4 + 1;
41       } while ((bool)uVar5);
42       if (!(bool)uVar5) {
43           initsrv();
44           rundelmeCmd();
45           /* WARNING: Subroutine does not return */
46           exit(1);
47       }
```

Gambar 2.8 hasil decompile pada ghidra (bag 1)

Bagian ini menginisialisasi program sebagai daemon (dengan `init_daemon()` yang dipanggil dua kali) dan memeriksa path eksekusi file menggunakan `readlink("/proc/self/exe")`. Program membandingkan path-nya dengan `/boot/.IptabLes` dan `/usr/.IptabLes`, yang merupakan lokasi mencurigakan, menunjukkan upaya malware untuk memastikan keberadaannya di lokasi tertentu. Jika path tidak sesuai, ia menjalankan `initsrv()` dan `rundelmecmd()` (kemungkinan untuk menghapus jejak, konsisten dengan sifat self-delete), lalu keluar.

```

49  iVar2 = promutex(LOCKFILEX);
50  if (iVar2 != 1) {
51      while( true ) {
52          getpid();
53          __pid = fork();
54          if (__pid < 1) break;
55          waitpid(__pid, (int *)0x0, 0);
56          kill(__pid, 9);
57      }

```

Gambar 2.9 hasil decompile pada ghidra (bag 2)

Bagian ini menggunakan mekanisme mutex (promutex) untuk memastikan hanya satu instance malware yang berjalan (menggunakan `LOCKFILEX` dan `LOCKFILE`). Jika mutex gagal, program melakukan `fork()` untuk membuat proses anak, lalu membunuh proses sebelumnya (`kill(__pid,9)`), menunjukkan upaya persistence agar malware tetap aktif meskipun ada proses lain yang serupa. Ini adalah taktik umum trojan untuk menghindari konflik instance.

```

58  if (__pid == 0) {
59      getpid();
60      iVar2 = promutex(LOCKFILE);
61      if (iVar2 != 1) {
62          memset(&g_mainsrvinfo, 0, 600);
63          deinfo(AAAAAA);
64          uVar1 = time((time_t *)0x0);
65          srandom(uVar1);
66          memset(g_mainsrvinfo.mypropath, 0, 0x100);
67          g_mainsrvinfo.srvver = 0x4b1;
68          g_mainsrvinfo.lanspeed = 0;
69          readlink("/proc/self/exe", g_mainsrvinfo.mypropath, 0x100);
70          g_mainsrvinfo.mainhostip = 0;
71          g_mainsrvinfo.udpport = 0;
72          g_mainsrvinfo.MyMainSocket = -1;
73          g_mainsrvinfo.MainIsrun = 0;
74          g_mainsrvinfo.TestIsrun = 0;
75          g_mainsrvinfo.CtlIsActivated = 0;
76          g_mainsrvinfo.srandipb = 0;
77          g_mainsrvinfo.srandipe = 0;
78          g_mainsrvinfo.tsrandipb = 0;
79          g_mainsrvinfo.tsrandipe = 0xffffffff;
80          g_mainsrvinfo.localip = 0;
81          memset(&g_mainsrvinfo.srvsend, 0, 0x9c);
82          g_mainsrvinfo.checkcount = 0;
83          g_mainsrvinfo.xmlfile = (char *)0x0;
84          getpthinfo();
85          MyTaskListCreat();
86          mainxxxx("xxxx");
87          /* WARNING: Subroutine does not return */
88          exit(1);
89      }
90  }
91  }
92  /* WARNING: Subroutine does not return */
93  exit(0);
94  }

```

Gambar 2.10 hasil decompile pada ghidra (bag 3)

Bagian ini dijalankan pada proses anak (`__pid == 0`) dan menginisialisasi struktur `g_mainsrvinfo`, yang berisi konfigurasi malware seperti versi server (`srvver`), port UDP (`udpport`), dan IP acak (`tsrandipe`). Fungsi seperti `MyTaskListCreat()` dan

mainxxxx("xxxx") menunjukkan pembuatan daftar tugas dan eksekusi utama malware, kemungkinan untuk serangan DDoS (konsisten dengan label seperti Trojan[DDoS]/Linux.Sfloost.c). Inisialisasi ini juga mencakup mekanisme untuk menghindari deteksi dengan menggunakan randomisasi (srand).

## **Kesimpulan Hasil Analisa Sample2**

Dari hasil analisa program malware sample2, kami menemukan tujuan dari programnya:

1. Berjalan sebagai daemon untuk melakukan serangan DDoS dengan menargetkan domain seperti yahoo.com dan baidu.com.
2. Memastikan hanya satu instance aktif menggunakan mekanisme mutex dan proses fork.
3. Menghindari deteksi dengan mekanisme seperti long-sleeps, self-delete (menghapus diri dengan `rm -rf "$0"`), dan mendeteksi lingkungan debug.
4. Mengumpulkan informasi sistem (melalui `/proc/net/dev`, `/proc/meminfo`) untuk mengatur serangan jaringan.
5. Menginisialisasi konfigurasi untuk serangan jaringan (via struktur `g_mainsrvinfo`) dan berpotensi mengendalikan sistem dari jarak jauh sebagai bagian dari keluarga malware correlgen/sfloost.

## **Potensi Dampak:**

1. Serangan DDoS (Distributed Denial of Service)  
Malware ini dirancang untuk membanjiri target (website/server) dengan trafik sampah dari mesin terinfeksi.
2. Botnet Enrollment  
Sistem terinfeksi bisa menjadi bagian dari botnet dan dikendalikan secara terpusat oleh C2 server (Command & Control).
3. Peningkatan Beban Sistem & Crash  
CPU dan bandwidth akan habis digunakan untuk mengirim serangan, membuat sistem tidak responsif atau crash.
4. Kemungkinan Serangan Lanjutan  
Malware ini bisa menjadi pintu masuk untuk malware lain, atau digunakan untuk menjatuhkan kredibilitas organisasi.

## **Rekomendasi Tindakan:**

1. Putuskan Jaringan Server yang Terinfeksi  
Hindari agar server tidak terus digunakan dalam serangan DDoS tanpa disadari.
2. Hapus Malware dan Bersihkan Sistem  
Hapus executable mencurigakan, lalu lakukan reinstall OS jika perlu (karena malware ELF bisa sulit dibersihkan sempurna).
3. Pantau Koneksi ke C2 Server  
Gunakan firewall/log jaringan untuk mendeteksi komunikasi ke IP/domain mencurigakan yang dikendalikan penyerang.
4. Update Kernel & Paket Sistem  
Pastikan tidak ada kerentanan yang bisa dieksploitasi ulang.
5. Audit Keamanan Server  
Cek semua akun user, proses berjalan, crontab, dan file startup untuk deteksi persistence.
6. Pasang Intrusion Detection System (IDS)  
Gunakan IDS (seperti Snort/Suricata) untuk mendeteksi pola serangan DDoS atau aktivitas mencurigakan lain di masa depan.

## **Tools yang digunakan**

selama pengerjaan challenge 2 baik file sample1 maupun sample2 kami menggunakan beberapa tools untuk membantu memudahkan pekerjaan dan meningkatkan efisiensi analisa kami, tools-tools yang kami gunakan yaitu:

1. Strings
2. File
3. 7z
4. Gunzip
5. Tar
6. upx
7. binwalk
8. Ghidra
9. Virustotal.com
10. att&ck.mitre.org
11. Chatgpt
12. Grok

**Deklarasi Penggunaan AI**

Dalam pengerjaan challenge 2 ini kami menggunakan AI untuk memperbaiki kata-kata yang typo dan masih berbahasa inggris dalam laporan kami, kami juga menggunakan AI untuk membantu kami dalam menganalisa dan membaca file malware sample1 dan sample2, kami juga menggunakan AI untuk mencari laporan-laporan serupa yang sebagai referensi cara penulisan.