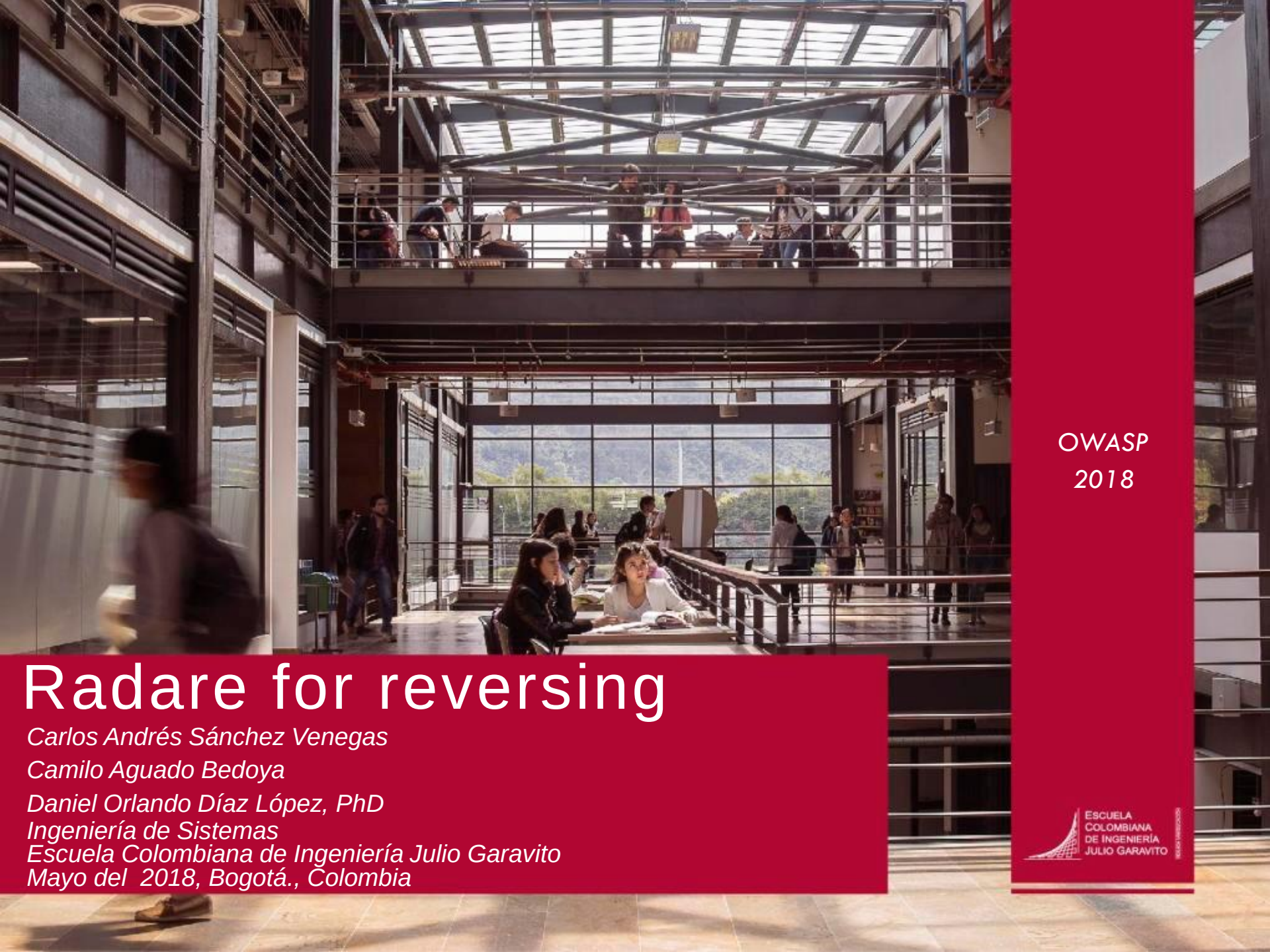

OWASP
Bogotá Chapter



Radare for reversing

Carlos Andrés Sánchez Venegas

Camilo Aguado Bedoya

Daniel Orlando Díaz López, PhD
Ingeniería de Sistemas
Escuela Colombiana de Ingeniería Julio Garavito
Mayo del 2018, Bogotá., Colombia

OWASP
2018

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

CONTENT

1. Project description
   - What is Reversing engineering
   - Functionality and utility of the reversing engineering
   - What is IoCs
   - Radare vs IDA Pro
2. Use case description
   - Hacking with Radare
   - Analyzing an Android app
3. Project improvements
   - r2yara
   - yarGen

1. PROJECT DESCRIPTION

WHAT IS REVERSING ENGINEERING

- ✓Is the process of deconstruction of an object made by man to reveal his designs, architecture or extract knowledge of the object.

- ✓The method is named like that because it moves in the opposite direction to the usual engineering tasks.

- ✓**"When malware is discovered on their systems, they want to know what it might have done, if the threat is still ongoing, and what they might have lost to the infection"**

1. PROJECT DESCRIPTION

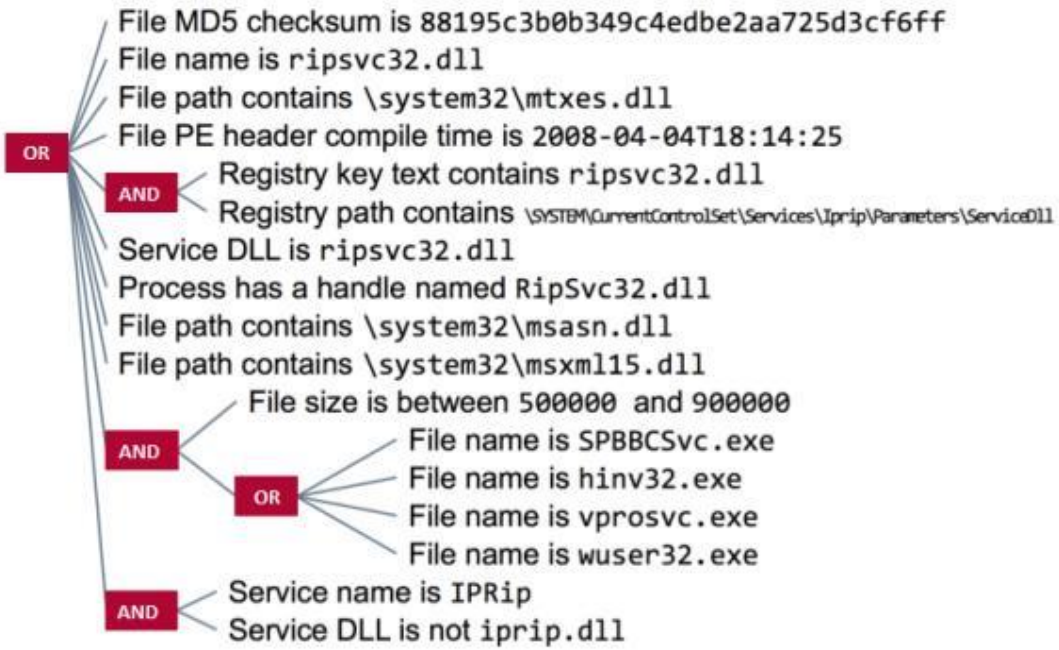
FUNCTIONALITY AND UTILITY OF THE REVERSING ENGINEERING

- With reverse engineering we can discover IOCs and create Yara´s rule
- Discover that a file is a malware
- Hacking
- Discover C&C
- Find traces of attackers
- Discover vulnerabilities in our code

1. PROJECT DESCRIPTION

WHAT IS IOCS

An Indicator of Compromise is the description of an cybersecurity incident, activity and/or malicious artifact through patterns to identify them in a network or endpoint, improving by this way the capacities of the incident management



1. PROJECT DESCRIPTION

RADARE VS IDA PRO

Radare is a reversing framework that can:

- Disassemble (machine language into assembly language) and assemble for many different architectures
- Perform forensics on filesystems
- Visualize data structures of several file types
- Aid in software exploitation
- Open source code

R2CON

Comercial alternative

IDA Pro
by Ilfak Guilfanov

Interactive and multi-processor disassembler written in C++ for Windows, Linux or MAC. It can be coupled to a debugger.

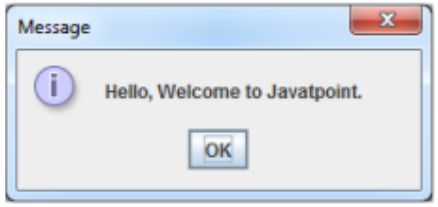
https://github.com/radare/radare2

CONTENT

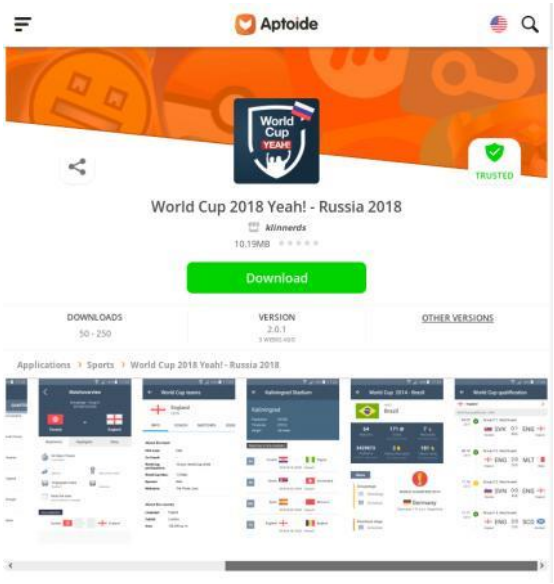
1. Project description
   - What is Reversing engineering
   - Functionality and utility of the reversing engineering
   - What is IoCs
   - Radare vs IDA Pro
2. Use case description
   - Hacking with Radare
   - Analyzing an Android app
3. Project improvements
   - r2yara
   - yarGen

2. USE CASE DESCRIPTION

1. Hacking “malware” to enable sandbox analysis



https://cc-csirt.policia.gov.co/Sandbox



2. Analysis of malicious Android applications addressed to the FIFA world cup 2018

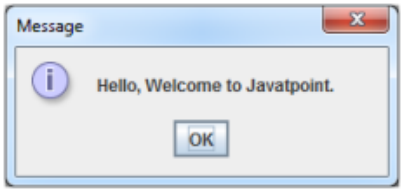
2. USE CASE DESCRIPTION

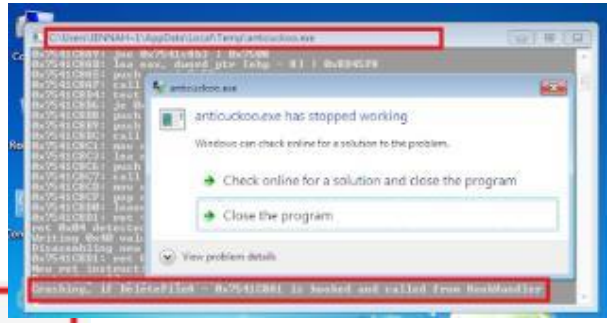
There are many anti sandboxing techniques:

1. Sandboxing timeout due a required user interaction (click, dialog, etc.)
2. Detect a typical routine followed by the sandbox
3. Malicious behavior just present after reboot
4. Detection of non end user Workstation (Based on browser historial or other)
5. More and more

We have focused on malware using technique 1 (dialog)

We will hack “malware” to enable the sandbox analysis

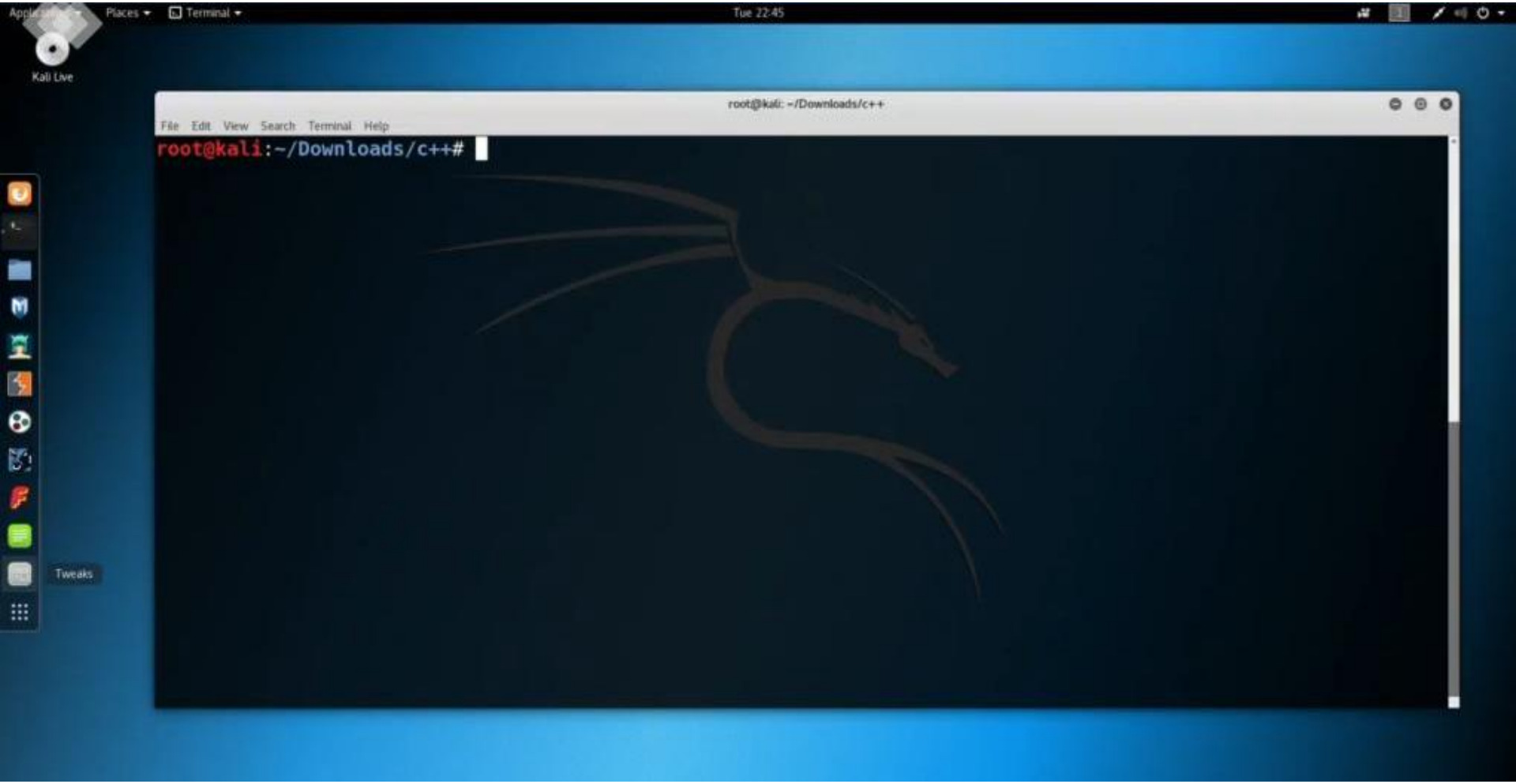





```
C:\Users\bird\AppData\Local\Temp\hook_api_hotpatch_jmp_indirect
C:\Users\bird\AppData\Local\Temp\hook_api_native_jmp_indirect
C:\Users\bird\AppData\Local\Temp\Hooks
C:\Users\bird\AppData\Local\Temp\Suspicius_string_found_HookHandle
C:\Users\bird\AppData\Local\Temp\Suspicius_string_found_Cuckoo
C:\Users\bird\AppData\Local\Temp\Suspicius_string_found_New_NtCreateThreadEx
C:\Users\bird\AppData\Local\Temp\Suspicius_string_found_cuckoomon
C:\Users\bird\AppData\Local\Temp\Suspicius_string_found_unhook
C:\Users\bird\AppData\Local\Temp\SuspiciusDataInMyMemory
```

2. USE CASE DESCRIPTION

DEMOSTRATION OF USE

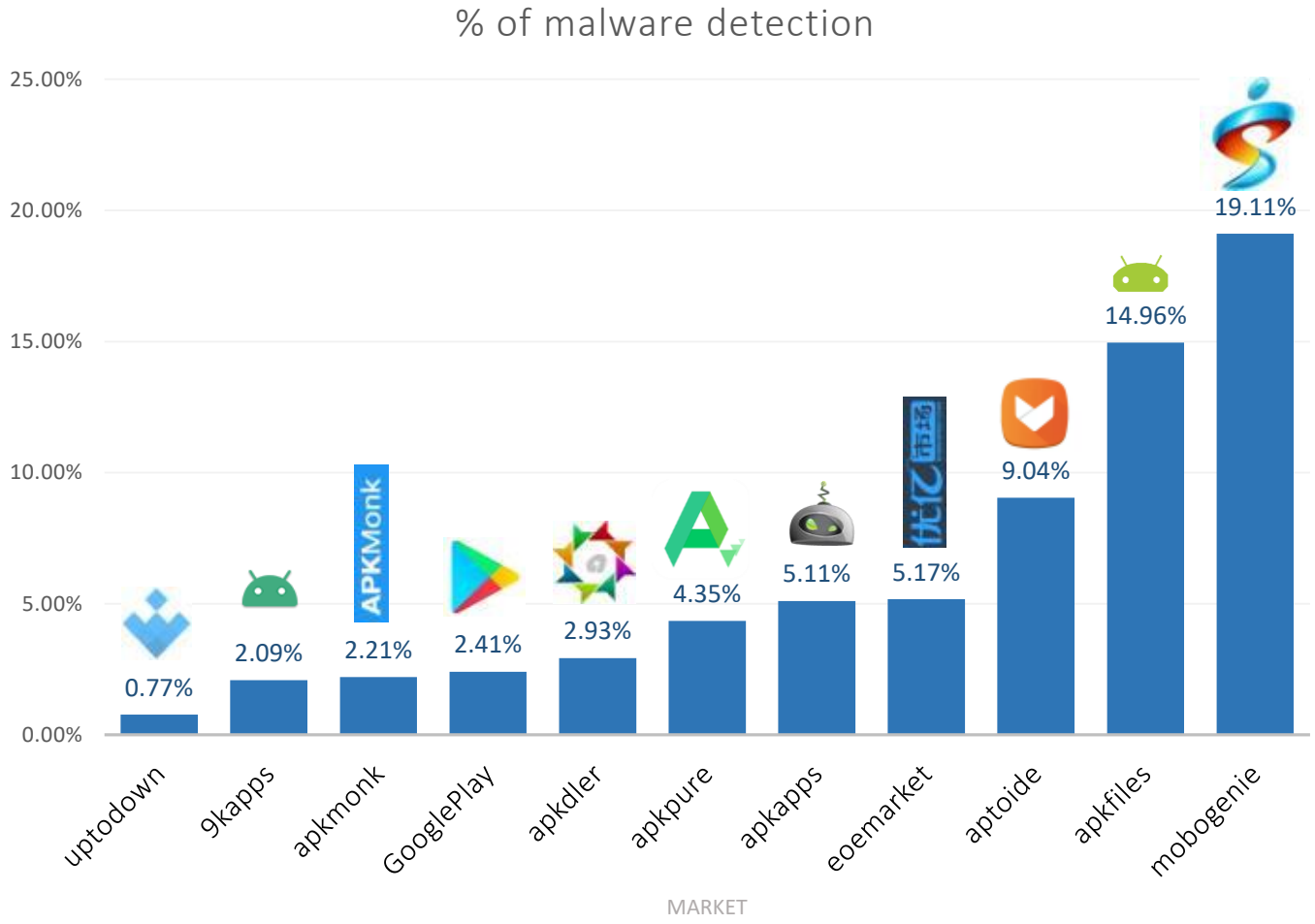


CONTENT

1. Project description
   - What is Reversing engineering
   - Functionality and utility of the reversing engineering
   - What is IoCs
   - Radare vs IDA Pro
2. Use case description
   - Hacking with Radare
   - Analyzing an Android app
3. Project improvements
   - r2yara
   - yarGen

2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP



2. USE CASE DESCRIPTION

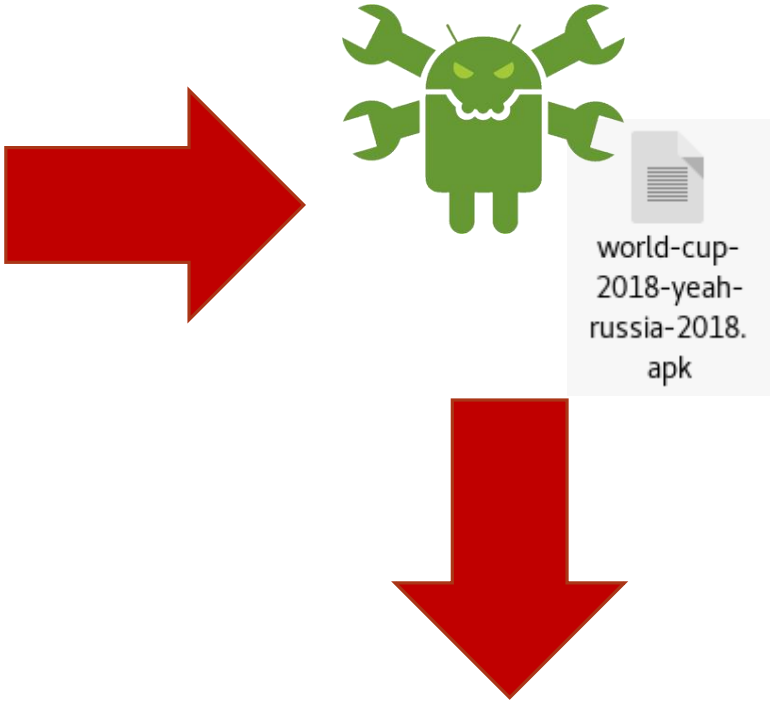
ANALYZING AN ANDROID APP

In which store would an attacker publish a malware?

Aptoide
Your Android App Store

2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP



"Esta completa aplicación es un compañero perfecto para la Copa Mundial de Fútbol 2018 en Rusia. En la aplicación, encontrarás la información más importante sobre la Copa del Mundo."

2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP

Unzipped

```
root@kali:~/Downloads# unzip world-cup-2018-yeah-russia-2018.apk -d unziped
```

AndroidManifest!!!!!!

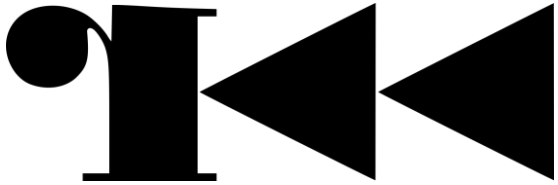
```
root@kali:~/Downloads/unziped# ls
AndroidManifest.xml  assets  build-data.properties  classes2.dex  classes.dex  com  fabric  jsr305_annotations  lib  META-INF  res  resources.arsc  version.properties
root@kali:~/Downloads/unziped#
```

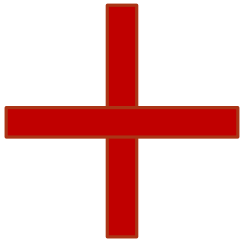
Two files obtained from an APK extraction

1. **AndroidManifest.xml:** Permissions and activities
2. **Classes.dex:** Dalvik code of the app (java to dalvik to assembled code) If we analyze that file, we can find own classes, imports, strings

2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP






```
root@kali:~/Downloads/unziped# r2pm -r axml2xml AndroidManifest.xml
```

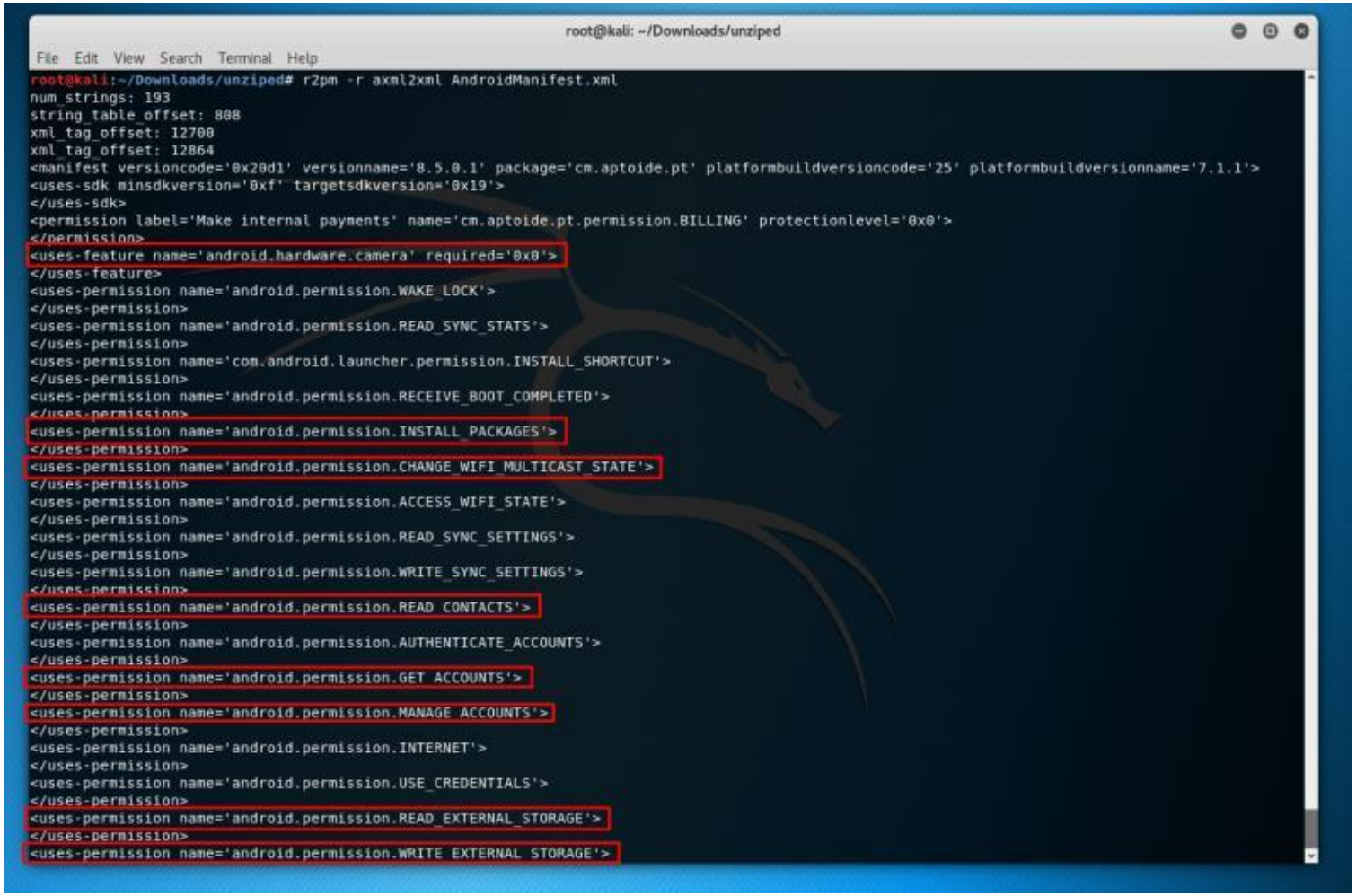
Radare
Open source
GitHub Link:
https://github.com/radare/radare2
It is a forensics tool aimed to make reversing on applications

Androguard
Open source
GitHub Link:
https://github.com/androguard/androguard
It is a library aimed to make reversing on android applications

2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP

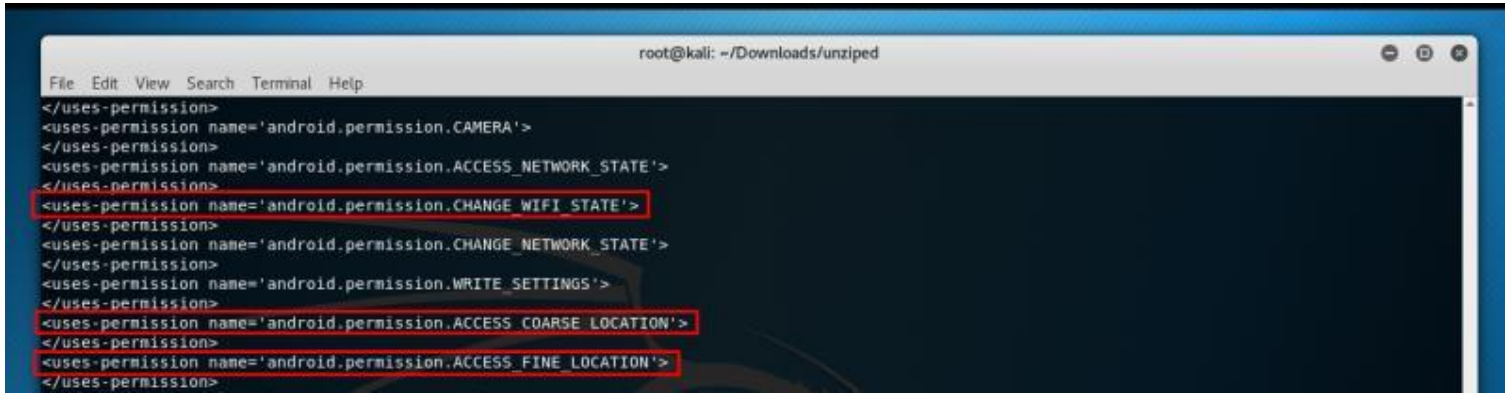
R2pm –r axml2xml



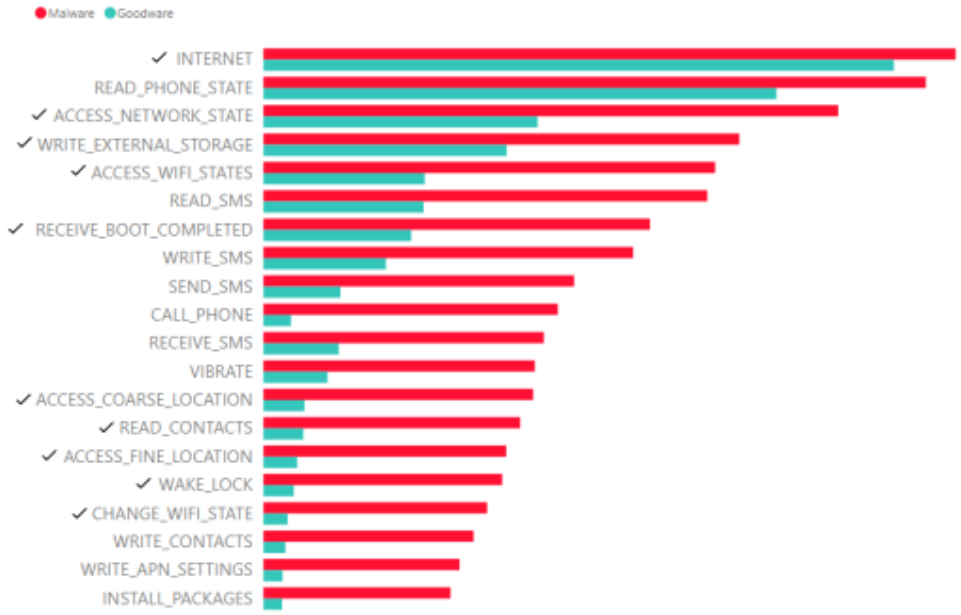
2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP

R2pm –r axml2xml



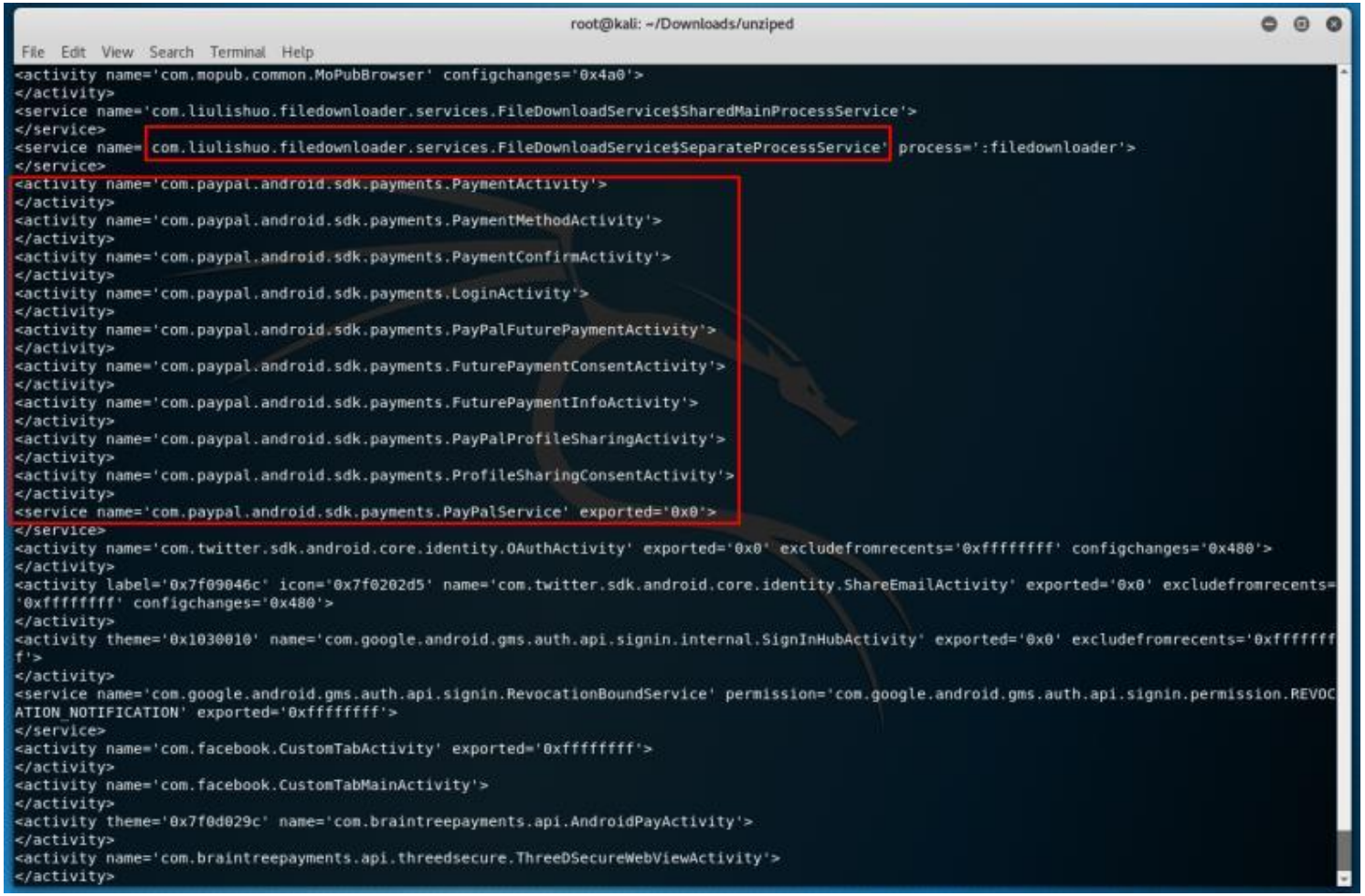
Common malware permissions



2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP

R2pm –r axml2xml



2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP

Classes.dex: Dalvik code of the app (java to dalvik to assembled code)

If we analyze that file, we can find own classes, imports, strings.

Analyze imports to see if the malware is using something for SMS, Bluetooth, NFC, datagram, telephony.

Rabin2 –qi

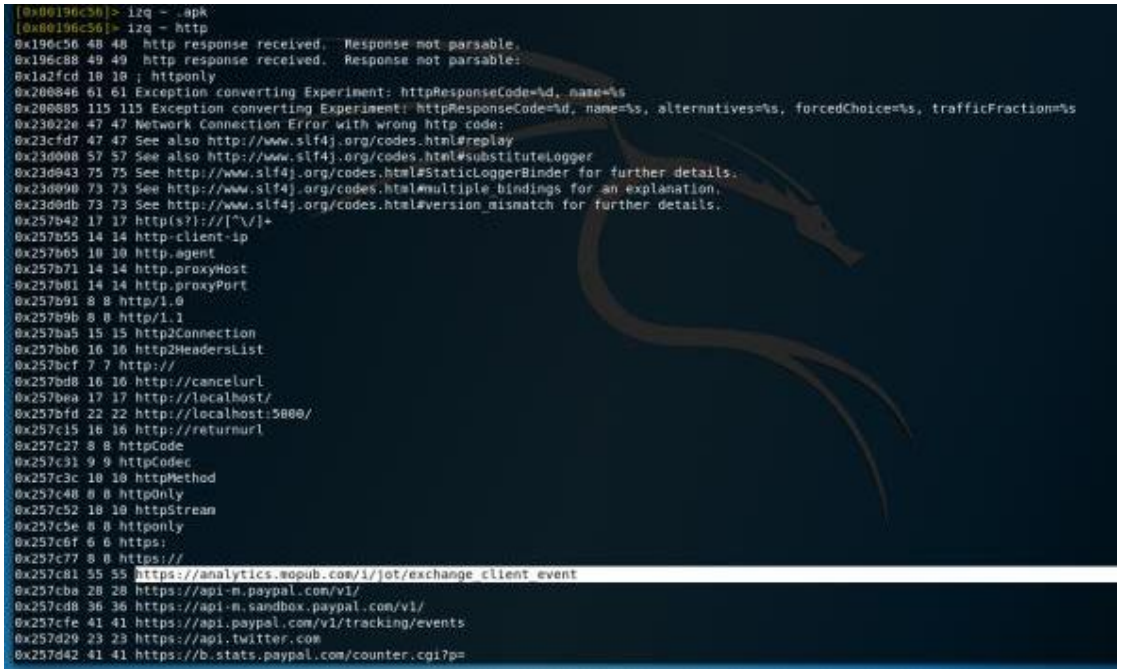
```
root@kali:~/Downloads/unziped# rabin2 -qi classes.dex | grep -i -e sms
root@kali:~/Downloads/unziped#
```

```
root@kali:~/Downloads/unziped# rabin2 -qi classes.dex | grep -i -e Telephony
Landroid/telephony/PhoneNumberFormattingTextWatcher.method.<init>()V
Landroid/telephony/PhoneNumberUtils.method.stripSeparators(Ljava/lang/String;)Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getNetworkCountryIso()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getNetworkOperator()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getNetworkOperatorName()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getPhoneType()I
Landroid/telephony/TelephonyManager.method.getSimCountryIso()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getSimOperator()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getSimOperatorName()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getSimState()I
root@kali:~/Downloads/unziped#
```

2. USE CASE DESCRIPTION

ANALYZING AN ANDROID APP

Strings



izq ~ .apk

izq ~ .http

izq ~ /system

- Http Strings

- Root access strings

```
[0x00196c56]> izq ~ /system
0x1a1acd 25 25 /system/app/Superuser.apk
0x1a1ae8 15 15 /system/xbin/su
[0x00196c56]>
```

CONTENT

1. Project description
   - What is Reversing engineering
   - Functionality and utility of the reversing engineering
   - What is IoCs
   - Radare vs IDA Pro
2. Use case description
   - Hacking with Radare
   - Analyzing an Android app
3. Project improvements
   - r2yara
   - yarGen

3. PROJECT IMPROVEMENTS

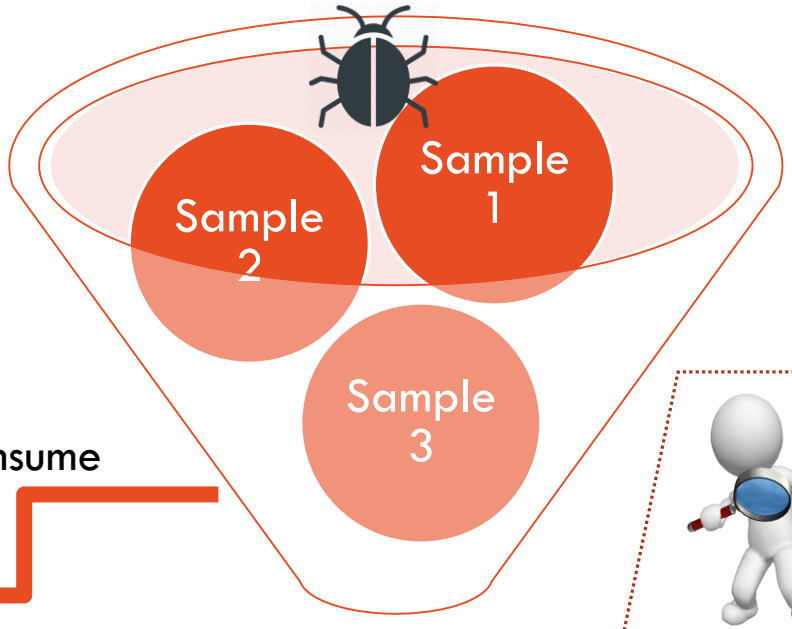
What can we do with all these information obtained from a malware analysis?

3. PROJECT IMPROVEMENTS

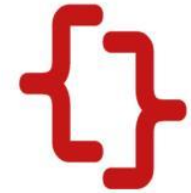
This is a Yara Rule

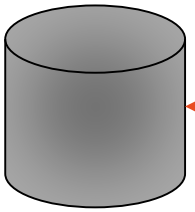
```
rule DarkRAT
{
    meta:
        date = "11/02/2015"
        tags = "rat"
    strings:
        $a = "@1906dark1996coder@"
        $b = "SHEmptyRecycleBinA"
        $c = "mciSendStringA"
        $d = "add_Shutdown"
        $e = "get_SaveMySettingsOnExit"
        $f = "get_SpecialDirectories"
    condition:
        all of them
}
```

We can analyze unknown samples and discover that the sample is very similar to some malware



consume





Yara Rules Database

A match with a rule has been found!

3. PROJECT IMPROVEMENTS

These commands are included in Radare

r2yara

This is another Yara Rule

```
rule rule_info
{
condition:
    r2.info.havecode == 1 and
    r2.info.pic == 0 and
    r2.info.canary == 1 and
    r2.info.nx == 1 and
    r2.info.crypto == 0 and
    r2.info.va == 1 and
    r2.info.intrp contains "linux-x86" and
    r2.info.bintype == "elf" and
    r2.info.class contains "ELF64" and
    r2.info.lang == "c" and
    r2.info.arch == "x86" and
    r2.info.bits == 64 and
    r2.info.machine == "AMD x86-64 architecture" and
    r2.info.os == "linux" and
    r2.info.minopsz == 1 and
    r2.info.maxopsz == 16 and
    r2.info.pcalign == 0 and
    r2.info.subsys == "linux" and
    r2.info.endian == "little" and
    r2.info.stripped == 1 and
    r2.info.static == 0 and
    r2.info.linenum == 0 and
    r2.info.lsyms == 0 and
    r2.info.relocs == 0 and
    r2.info.binsz > 100000 and
    r2.info.rpath == "NONE" and
    r2.info.compiled != "Sat Sep 9 11:32:42 2006" and
    r2.info.dbg_file not contains "test" and
    r2.info.guid == ""
}
```

Create Yara rules from **rabin2** and **rahash2** information:

- ❑Imports
- ❑Sections
- ❑Exports
- ❑List arch
- ❑Headers fields
- ❑Binary info
- ❑Libraries
- ❑Md5
- ❑Sha1
- ❑Sha256
- ❑Sha384
- ❑Sha512
- ❑Crc16
- ❑...

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
 GNU nano 2.9.1                    r2yara.yar

import "r2"

rule resources {
    condition:
        r2.export("ADVAPI32.dll_WmiQuerySingleInstanceW", "FUNC") and
        (r2.hash.md5 != "945fedb3a3c290d69f075f997e5320fc" or
            r2.hash.crc32 contains "b053d")

        for any i in ( 0..r2.number_of_resources ) :
            (r2.resources[i].size > 2KB and
             r2.resources[i].paddr > 1024 and
             r2.resources[i].type == "ICON" and
             r2.resources[i].lang contains "JAPANESE")
}
```

3. PROJECT IMPROVEMENTS

Yara rules are **useful** because helps to identify malwares

Otherwise Radare is great because let is see what is **inside** of a malware

Yara and Radare can be used **together** to easily identify many types of malware

However creating Yara rule can be an slow and arduous task for a malware analyst

¿Is it possible to automate the Yara rule generation?

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
 GNU nano 2.9.1                    r2yara.yar

import "r2"

rule resources {
    condition:
        r2.export("ADVAPI32.dll_WmiQuerySingleInstanceW", "FUNC") and
        (r2.hash.md5 != "945fedb3a3c290d69f075f997e5320fc" or
            r2.hash.crc32 contains "b053d")

        for any i in ( 0..r2.number_of_resources ) :
            (r2.resources[i].size > 2KB and
             r2.resources[i].paddr > 1024 and
             r2.resources[i].type == "ICON" and
             r2.resources[i].lang contains "JAPANESE")
}
```

3. PROJECT IMPROVEMENTS

YARGEN

- yarGen is an opensource **generator for YARA rules**
- The main principle is the creation of YARA rules from **strings found in malware files** while removing all strings that also appear in **goodware files**
- Uses its own database of strings
- It can be used with custom databases

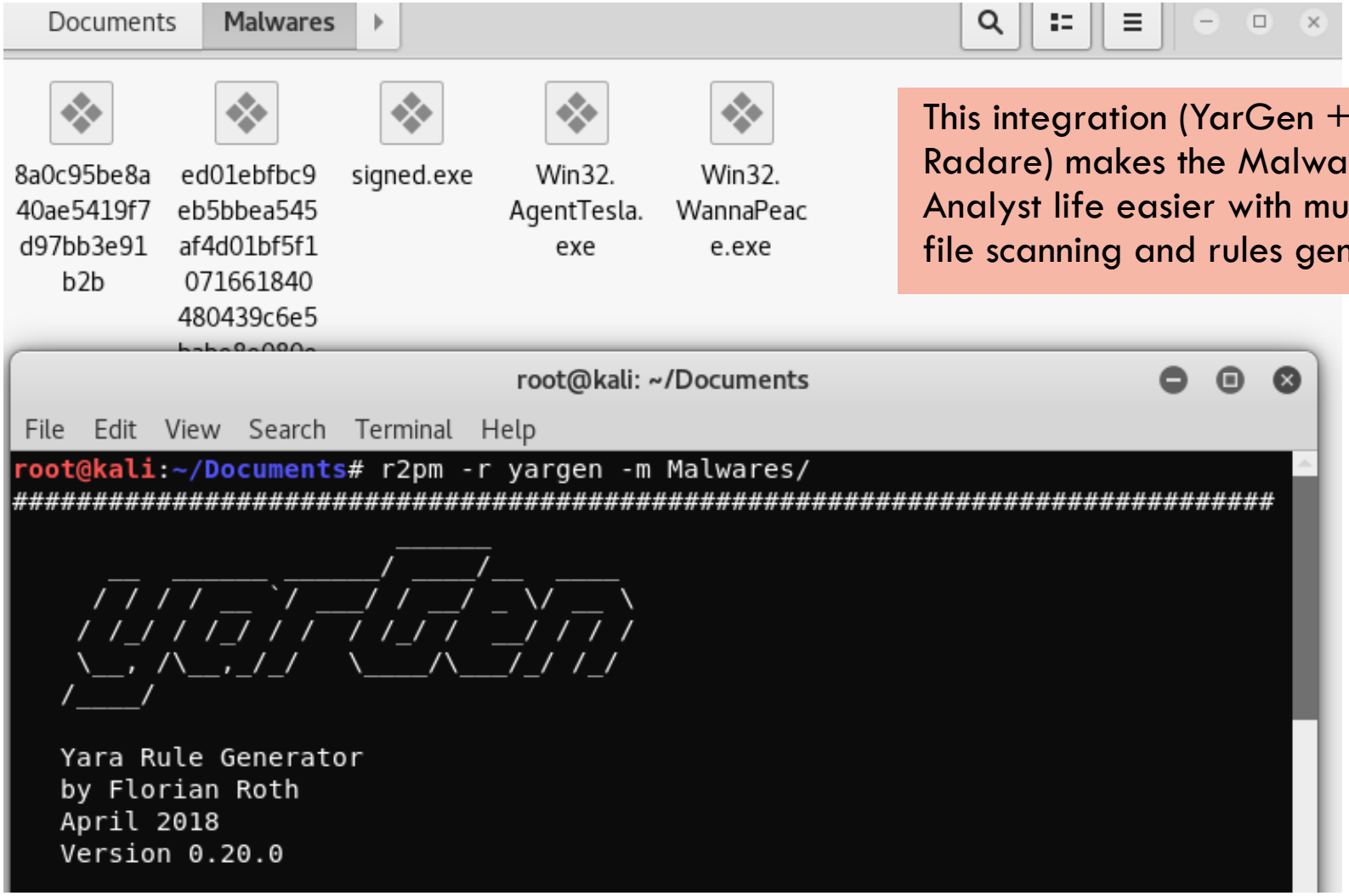
3. PROJECT IMPROVEMENTS
RADARE AND YARGEN INTEGRATION

- The objective is to create a **Radare plugin** to make a fast and easy installation and use of **yarGen** to help the user in the creation of YARA rules.

- The generated rule need to be cleaned after its generation!

```
/*
    Yara Rule Set
    Author: YarGen Rule Generator
    Date: 2015-07-09
    Identifier: bin
*/

/* Rule Set ----------------------------------------------------------------- */

rule backdoor {
    meta:
        description = "Auto-generated rule - file backdoor.exe"
        author = "YarGen Rule Generator"
        reference = "not set"
        date = "2015-07-09"
        hash = "bad8c7e6836b9a5679bfac0bc7483091e8e168f2"
    strings:
        $s0 = "%systemroot%\\System32\\rundll32.exe \"" fullword ascii /* PEStudio Blacklist: str
        $s1 = "c:\\Agenti\\SimpleVector\\Release\\SimpleVector.pdb" fullword ascii /* score: '28.
        $s2 = "GetCurrentProcessID" fullword ascii /* PEStudio Blacklist: strings */ /* score: '2
        $s3 = "<requestedExecutionLevel level=\"highestAvailable\" uiAccess=\"false\"/>" fullword
        $s4 = "SOFTWARE\\Microsoft\\VisualStudio\\9.0\\Setup\\VS" fullword ascii /* PEStudio Blac
        $s5 = "BG:\"oMpp" fullword ascii /* score: '12.00' */
        $s6 = "vvKPP80k.mKn" fullword ascii /* score: '12.00' */
        $s7 = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\" ?><assembly xmlns=\"u"
        $s8 = "0b55581e0a49451a01584c2a1d52232245595663182 44a41405b172e11161932" fullword ascii /
        $s9 = "SimpleVector, Version 1.0" fullword wide /* score: '9.00' */
        $s10 = "* GN3?" fullword ascii /* score: '7.00' */
        $s11 = "SIMPLEVECTOR" fullword wide /* score: '6.50' */
        $s12 = ".OcL/2" fullword ascii /* score: '6.00' */
        $s13 = "VH.IYl" fullword ascii /* score: '6.00' */
        $s14 = "uKmtnQzd78" fullword ascii /* score: '5.00' */
        $s15 = "About SimpleVector" fullword wide /* score: '5.00' */
    condition:
        uint16(0) == 0x5a4d and filesize < 3785KB and all of them
}
```

3. PROJECT IMPROVEMENTS
RADARE AND YARGEN INTEGRATION

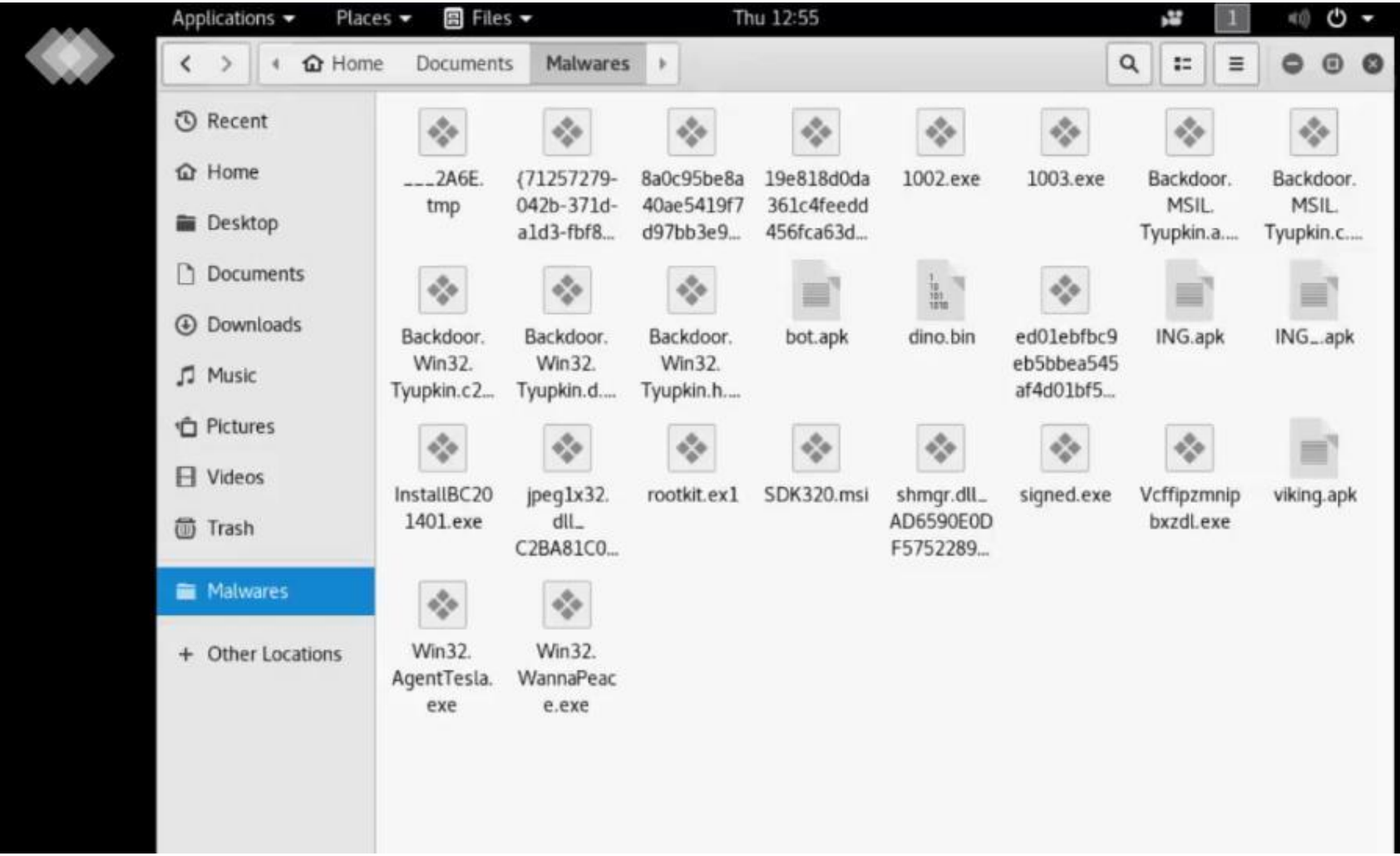


This integration (YarGen + Radare) makes the Malware Analyst life easier with multiple file scanning and rules generation

3. PROJECT IMPROVEMENTS INTEGRATION FILE

```
 GNU nano 2.9.1                                    yarGen

R2PM_BEGIN

R2PM_DESC "[app] Yara generator for malware"

SOURCE="https://raw.githubusercontent.com/Neo23x0/yarGen/master/yarGen.py"
GOOD="https://raw.githubusercontent.com/Neo23x0/yarGen/master/lib/good.txt"
PARTY="https://raw.githubusercontent.com/Neo23x0/yarGen/master/3rdparty/strings.xml"
B="${R2PM_BINDIR}"/yargen

R2PM_INSTALL() {
        ${R2PM_SUDO} pip install pefile
        ${R2PM_SUDO} pip install scandir lxml naiveBayesClassifier
        mkdir -p "${R2PM_BINDIR}"
        rm -f "${B}"
        wget -O "${B}" -c "${SOURCE}"
        chmod +x "${B}" || R2PM_FAIL "Making executable yarGen"
        mkdir "${R2PM_BINDIR}"/lib
        wget -O "${R2PM_BINDIR}"/lib/good.txt -c "${GOOD}"
        wget -O "${R2PM_BINDIR}"/3rdparty/strings.xml -c "${PARTY}"
        "${B}" --update || R2PM_FAIL "Downloading database"
        mv dbs "$R2PM_BINDIR"
}

R2PM_UNINSTALL() {
        rm -f "${B}"
        rm -fr "${R2PM_BINDIR}"/lib/good.txt
        rm -fr "${R2PM_BINDIR}"/dbs
}

R2PM_END

^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mark Text
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo   M-6 Copy Text
```

3. PROJECT IMPROVEMENTS



4. CONCLUSIONS

- It is recommended to **analyze the binaries files** before installing any application or executable or executing them
- With Radare (reverse engineering) we can easily **discover malicious applications** for Android devices
- Using Radare we can **solve sandboxing weaknesses** like timeout
- Through reverse engineering we can **discover IoC**
- Radare is complemented with different plugins such like the creation of Yara's rules with Rabin and Rahash (**r2bin**) and the total automation of the creation of Yara's rules (**yarGen**).

BIBLIOGRAPHY

I. Radare2 source code. Retrieved from https://github.com/radare/radare2

II. Radare documentation. Retrieved from https://rada.re/r/

III. Reverse Engineering definition. Retrieved from https://en.wikipedia.org/wiki/Reverse_engineering

IV. Security: The beauty of ...malware reverse engineering. Retrieved from https://www.networkworld.com/article/2712315/security/security--the-beauty-of-----malware-reverse-engineering.html

V. IoCs y sus capacidades. Retrieved from https://www.pandasecurity.com/spain/mediacenter/seguridad/iocs-y-sus-capacidades/

VI. YarGen source code. Retrieved from https://github.com/Neo23x0/yarGen


GRACIAS
ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO


OWASP
LATAM
2018
LATIN AMERICA TOUR
05 ABRIL - 25 DE MAYO