





OWASP  
2018

# 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*

# 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

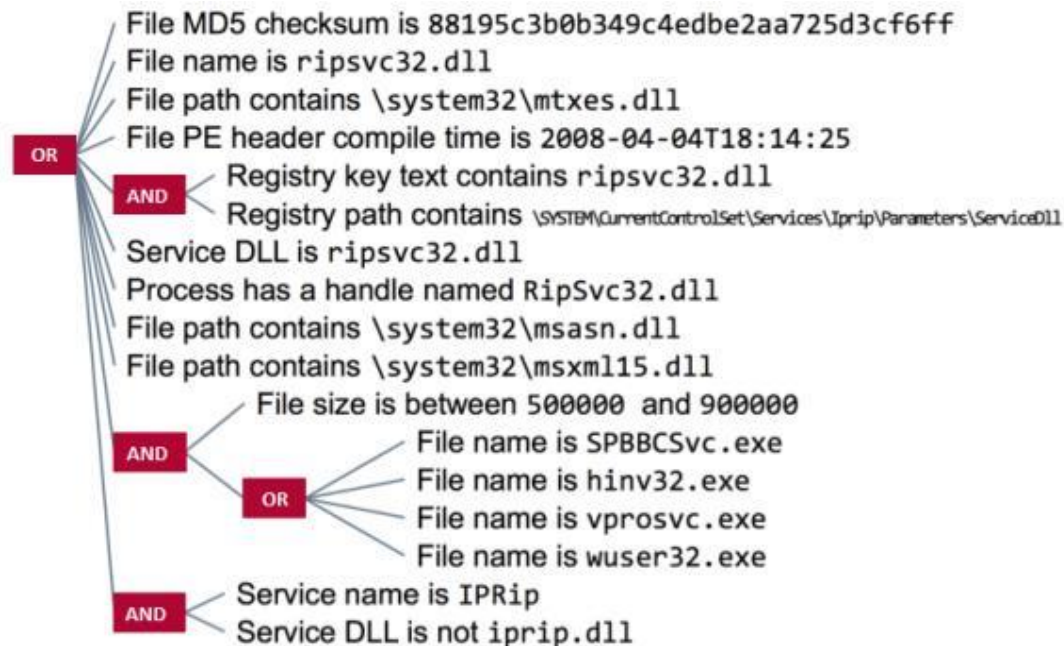
## FUNCTIONALITY AND UTILITY OF THE REVERSING ENGINEERING

- 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

Comercial alternative



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



R2CON



<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

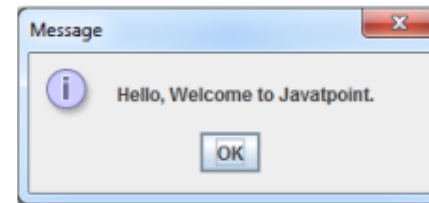
## 3. Project improvements

- 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



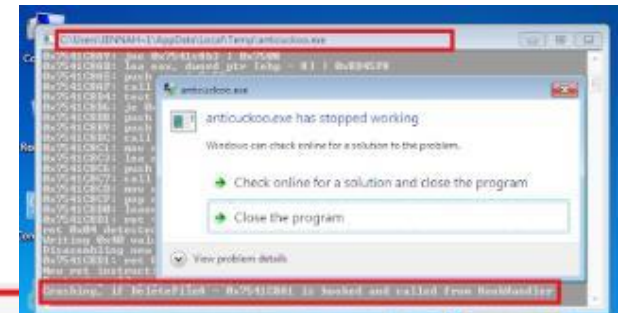
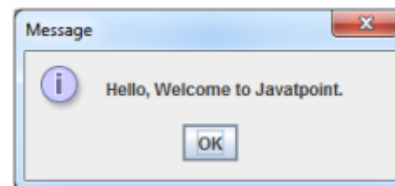
## 2. USE CASE DESCRIPTION

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)

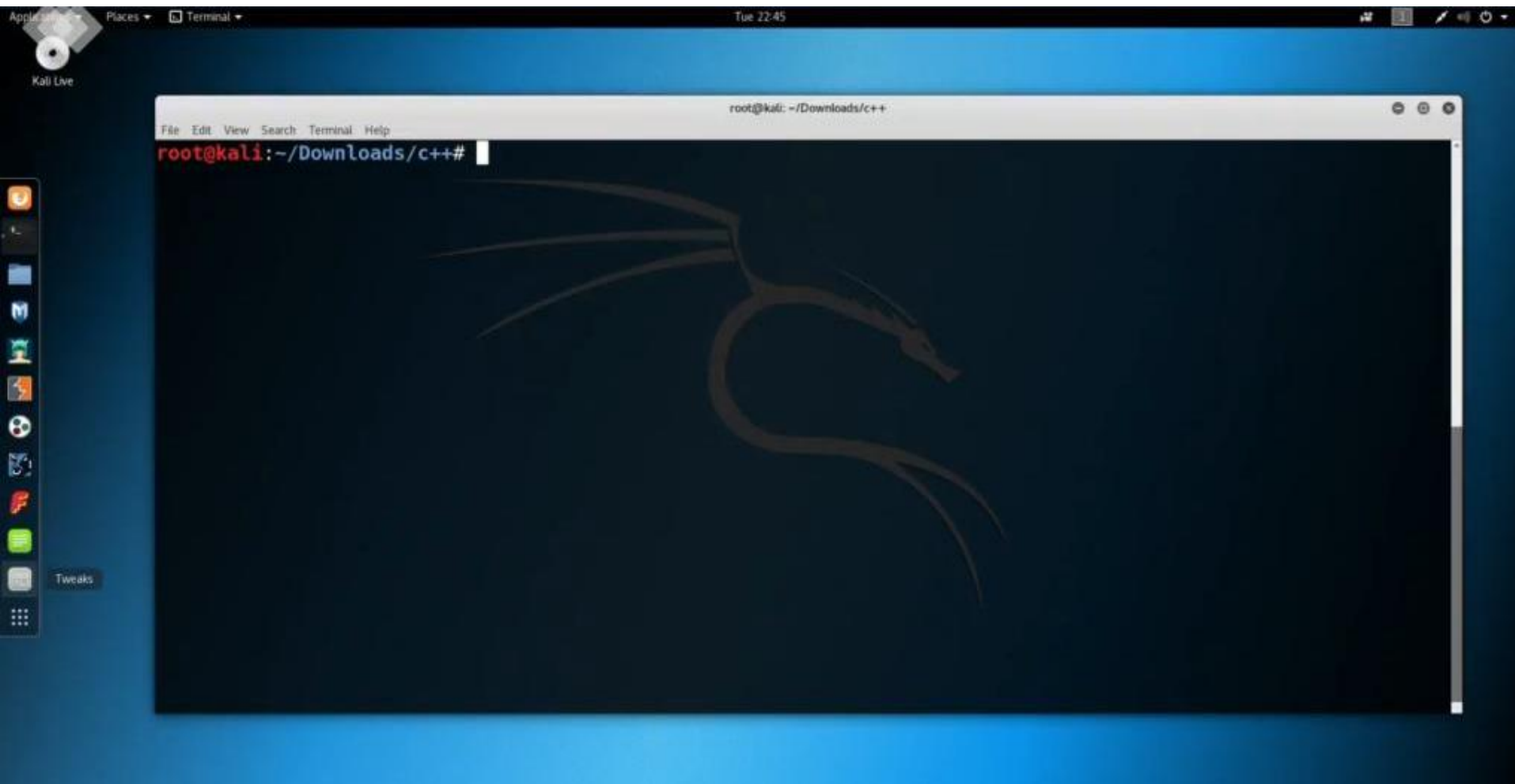
We will hack “malware” to enable the sandbox analysis



```
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\Suspicious_string_found_HookHandle
C:\Users\bird\AppData\Local\Temp\Suspicious_string_found_Cuckoo
C:\Users\bird\AppData\Local\Temp\Suspicious_string_found_New_NtCreateThreadEx
C:\Users\bird\AppData\Local\Temp\Suspicious_string_found_cuckoocomon
C:\Users\bird\AppData\Local\Temp\Suspicious_string_found_unhook
C:\Users\bird\AppData\Local\Temp\SuspiciousDataInMyMemory
```

## 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

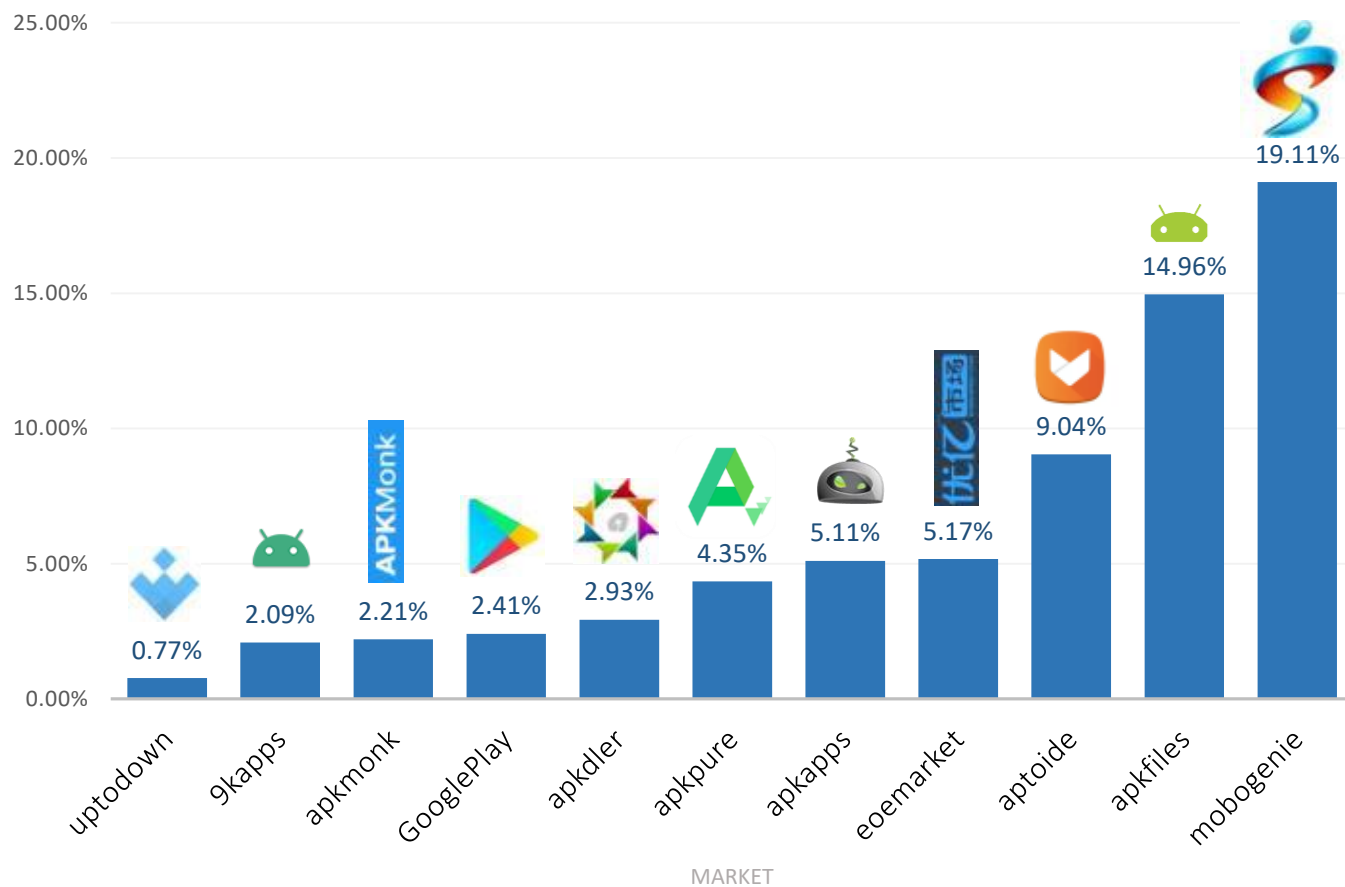
## 3. Project improvements

- r2yara
- yarGen

## 2. USE CASE DESCRIPTION

### ANALYZING AN ANDROID APP

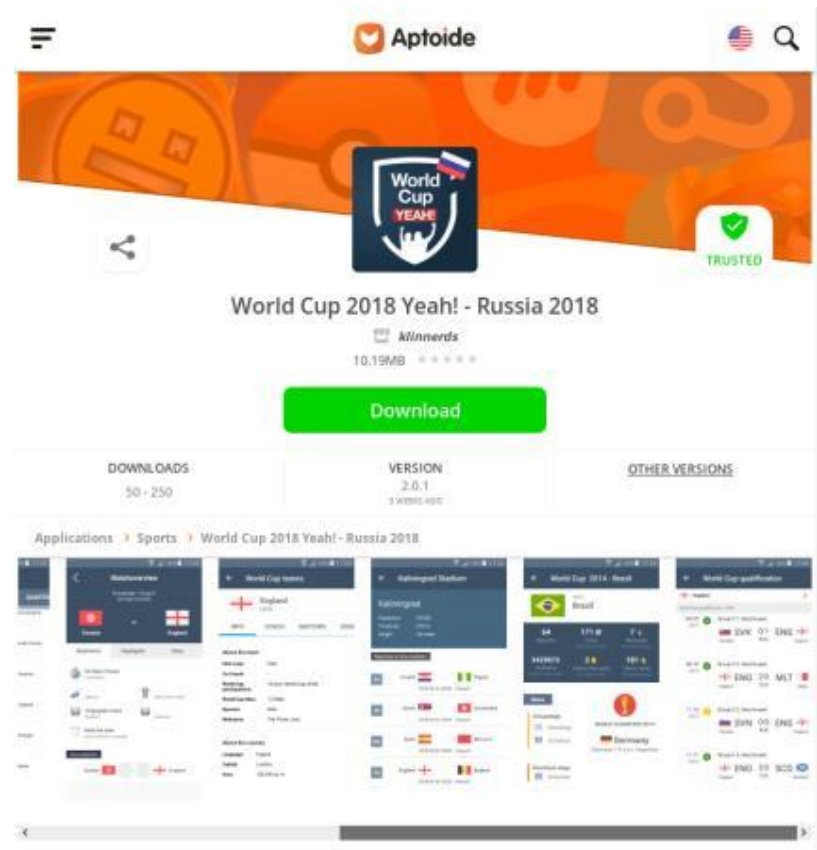
% of malware detection



## 2. USE CASE DESCRIPTION

### ANALYZING AN ANDROID APP

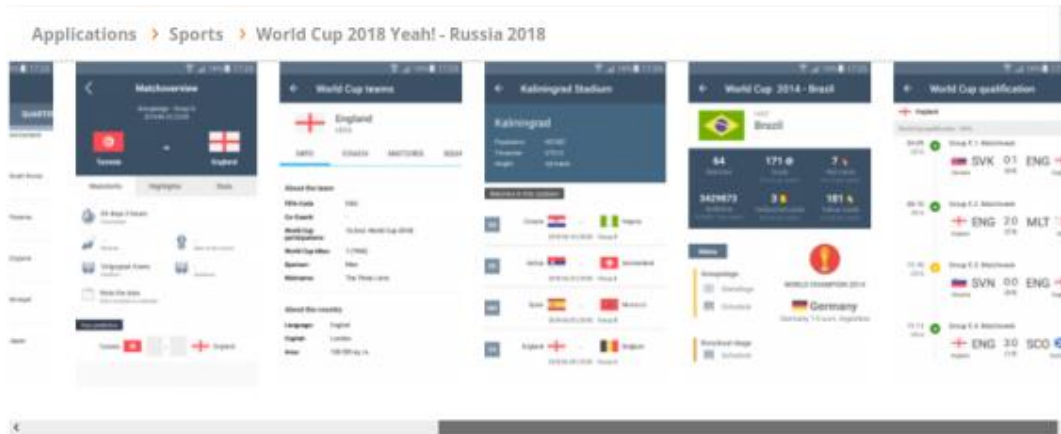
In which store would an attacker publish a malware?





## 2. USE CASE DESCRIPTION

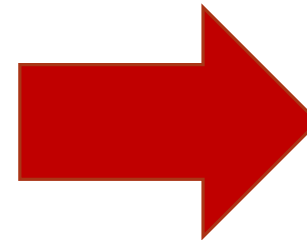
### ANALYZING AN ANDROID APP



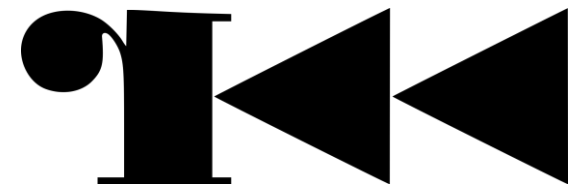
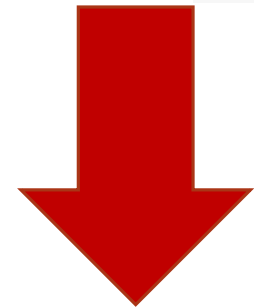
#### Description of World Cup 2018 Yeah! - Russia 2018

This comprehensive application is a perfect companion for the Football World Cup 2018 in Russia. In the application, you will find the most important information about the World Cup.

“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.”



world-cup-  
2018-yeah-  
russia-2018.  
apk



## 2. USE CASE DESCRIPTION

# ANALYZING AN ANDROID APP

## Unzipped

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

## AndroidManifest!!!!!!

```
root@kali:~/Downloads/unzipped# 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/unzipped#
```

AndroidManifest.xml (--/Downloads/ungziped) - VIM

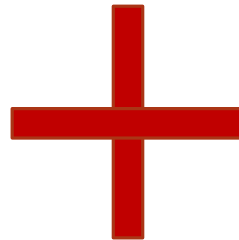
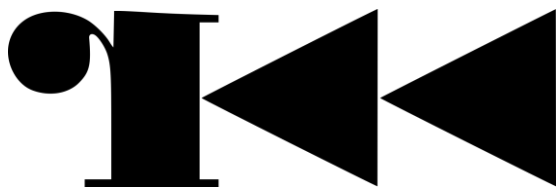
[illegible]

## 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/unzipped# 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

```

root@kali: ~/Downloads/unzipped
File Edit View Search Terminal Help
root@kali:~/Downloads/unzipped# r2pm -r axml2xml AndroidManifest.xml
num_strings: 193
string_table offset: 808
xml_tag offset: 12700
xml_tag offset: 12864
<manifest versioncode='0x20d1' versionname='8.5.0.1' package='cm.aptoide.pt' platformbuildversioncode='25' platformbuildversionname='7.1.1'>
<uses-sdk minsdkversion='0xf' targetsdkversion='0x19'>
</uses-sdk>
<permission label='Make internal payments' name='cm.aptoide.pt.permission.BILLING' protectionlevel='0x0'>
</permission>
<uses-feature name='android.hardware.camera' required='0x0'>
</uses-feature>
<uses-permission name='android.permission.WAKE_LOCK'>
</uses-permission>
<uses-permission name='android.permission.READ_SYNC_STATS'>
</uses-permission>
<uses-permission name='com.android.launcher.permission.INSTALL_SHORTCUT'>
</uses-permission>
<uses-permission name='android.permission.RECEIVE_BOOT_COMPLETED'>
</uses-permission>
<uses-permission name='android.permission.INSTALL_PACKAGES'>
</uses-permission>
<uses-permission name='android.permission.CHANGE_WIFI_MULTICAST_STATE'>
</uses-permission>
<uses-permission name='android.permission.ACCESS_WIFI_STATE'>
</uses-permission>
<uses-permission name='android.permission.READ_SYNC_SETTINGS'>
</uses-permission>
<uses-permission name='android.permission.WRITE_SYNC_SETTINGS'>
</uses-permission>
<uses-permission name='android.permission.READ_CONTACTS'>
</uses-permission>
<uses-permission name='android.permission.AUTHENTICATE_ACCOUNTS'>
</uses-permission>
<uses-permission name='android.permission.GET_ACCOUNTS'>
</uses-permission>
<uses-permission name='android.permission.MANAGE_ACCOUNTS'>
</uses-permission>
<uses-permission name='android.permission.INTERNET'>
</uses-permission>
<uses-permission name='android.permission.USE_CREDENTIALS'>
</uses-permission>
<uses-permission name='android.permission.READ_EXTERNAL_STORAGE'>
</uses-permission>
<uses-permission name='android.permission.WRITE_EXTERNAL_STORAGE'>

```



## 2. USE CASE DESCRIPTION

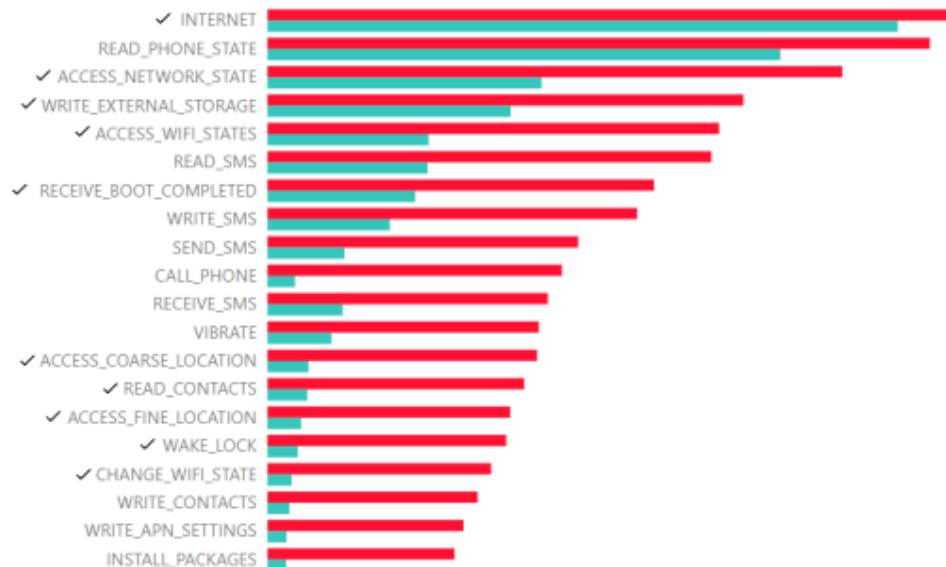
### ANALYZING AN ANDROID APP

R2pm -r axml2xml

```

root@kali: ~/Downloads/unzipped
File Edit View Search Terminal Help
</uses-permission>
<uses-permission name='android.permission.CAMERA'>
</uses-permission>
<uses-permission name='android.permission.ACCESS_NETWORK_STATE'>
</uses-permission>
<uses-permission name='android.permission.CHANGE_WIFI_STATE'>
</uses-permission>
<uses-permission name='android.permission.CHANGE_NETWORK_STATE'>
</uses-permission>
<uses-permission name='android.permission.WRITE_SETTINGS'>
</uses-permission>
<uses-permission name='android.permission.ACCESS_COARSE_LOCATION'>
</uses-permission>
<uses-permission name='android.permission.ACCESS_FINE_LOCATION'>
</uses-permission>
  
```

● Malware ● Goodware



Common malware permissions

## 2. USE CASE DESCRIPTION

### ANALYZING AN ANDROID APP

R2pm -r axml2xml

```

root@kali: ~/Downloads/unzipped
File Edit View Search Terminal Help
<activity name='com.mopub.common.MoPubBrowser' configChanges='0x4a0'>
</activity>
<service name='com.liulishuo.filedownload.services.FileDownloadService$SharedMainProcessService'>
</service>
<service name='com.liulishuo.filedownload.services.FileDownloadService$SeparateProcessService' process=':filedownloader'>
</service>
<activity name='com.paypal.android.sdk.payments.PaymentActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.PaymentMethodActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.PaymentConfirmActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.LoginActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.PayPalFuturePaymentActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.FuturePaymentConsentActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.FuturePaymentInfoActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.PayPalProfileSharingActivity'>
</activity>
<activity name='com.paypal.android.sdk.payments.ProfileSharingConsentActivity'>
</activity>
<service name='com.paypal.android.sdk.payments.PayPalService' exported='0x0'>
</service>
<activity name='com.twitter.sdk.android.core.identity.OAuthActivity' exported='0x0' excludeFromRecents='0xffffffff' configChanges='0x480'>
</activity>
<activity label='0x7f09046c' icon='0x7f0202d5' name='com.twitter.sdk.android.core.identity.ShareEmailActivity' exported='0x0' excludeFromRecents='0xffffffff' configChanges='0x480'>
</activity>
<activity theme='0x1030010' name='com.google.android.gms.auth.api.signin.internal.SignInHubActivity' exported='0x0' excludeFromRecents='0xffffffff'>
</activity>
<service name='com.google.android.gms.auth.api.signin.RevocationBoundService' permission='com.google.android.gms.auth.api.signin.permission.REVOCATION_NOTIFICATION' exported='0xffffffff'>
</service>
<activity name='com.facebook.CustomTabActivity' exported='0xffffffff'>
</activity>
<activity name='com.facebook.CustomTabMainActivity'>
</activity>
<activity theme='0x7f0d029c' name='com.braintreepayments.api.AndroidPayActivity'>
</activity>
<activity name='com.braintreepayments.api.threedsecure.ThreeDSecureWebViewActivity'>
</activity>

```



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

```
root@kali:~/Downloads/unzipped# 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/unzipped#
```

## 2. USE CASE DESCRIPTION

### ANALYZING AN ANDROID APP

#### Strings

```
[0x00196c56]> izq ~ .apk
[0x00196c56]> izq ~ http
0x196c56 48 48 http response received. Response not parsable.
0x196c88 49 49 http response received. Response not parsable.
0x1a2fcd 10 10 ; httponly
0x200846 61 61 Exception converting Experiment: httpResponseCode=%d, name=%s
0x200885 115 115 Exception converting Experiment: httpResponseCode=%d, name=%s, alternatives=%s, forcedChoice=%s, trafficFraction=%s
0x23022e 47 47 Network Connection Error with wrong http code:
0x23cf7d 47 47 See also http://www.slf4j.org/codes.html#replay
0x23d008 57 57 See also http://www.slf4j.org/codes.html#substituteLogger
0x23d043 75 75 See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
0x23d090 73 73 See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
0x23d0db 73 73 See http://www.slf4j.org/codes.html#version_mismatch for further details.
0x257b42 17 17 http(s)://[^\/*]
0x257b55 14 14 http-client-ip
0x257b65 10 10 http.agent
0x257b71 14 14 http.proxyHost
0x257b81 14 14 http.proxyPort
0x257b91 8 8 http/1.0
0x257b9b 8 8 http/1.1
0x257ba5 15 15 http2Connection
0x257bb6 16 16 http2HeadersList
0x257bc7 7 7 http://
0x257bd8 16 16 http://cancelurl
0x257bea 17 17 http://localhost/
0x257bfd 22 22 http://localhost:5000/
0x257c15 16 16 http://returnurl
0x257c27 8 8 httpCode
0x257c31 9 9 httpCodec
0x257c3c 10 10 httpMethod
0x257c48 8 8 httpOnly
0x257c52 10 10 httpStream
0x257c5e 8 8 httpOnly
0x257c6f 6 6 https:
0x257c77 8 8 https://
0x257c81 55 55 https://analytics.mopub.com/i/ot/exchange client event
0x257cba 28 28 https://api-m.paypal.com/v1/
0x257cd8 36 36 https://api-m.sandbox.paypal.com/v1/
0x257cfe 41 41 https://api.paypal.com/v1/tracking/events
0x257d29 23 23 https://api.twitter.com
0x257d42 41 41 https://b.stats.paypal.com/counter.cgi?p=
```

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?

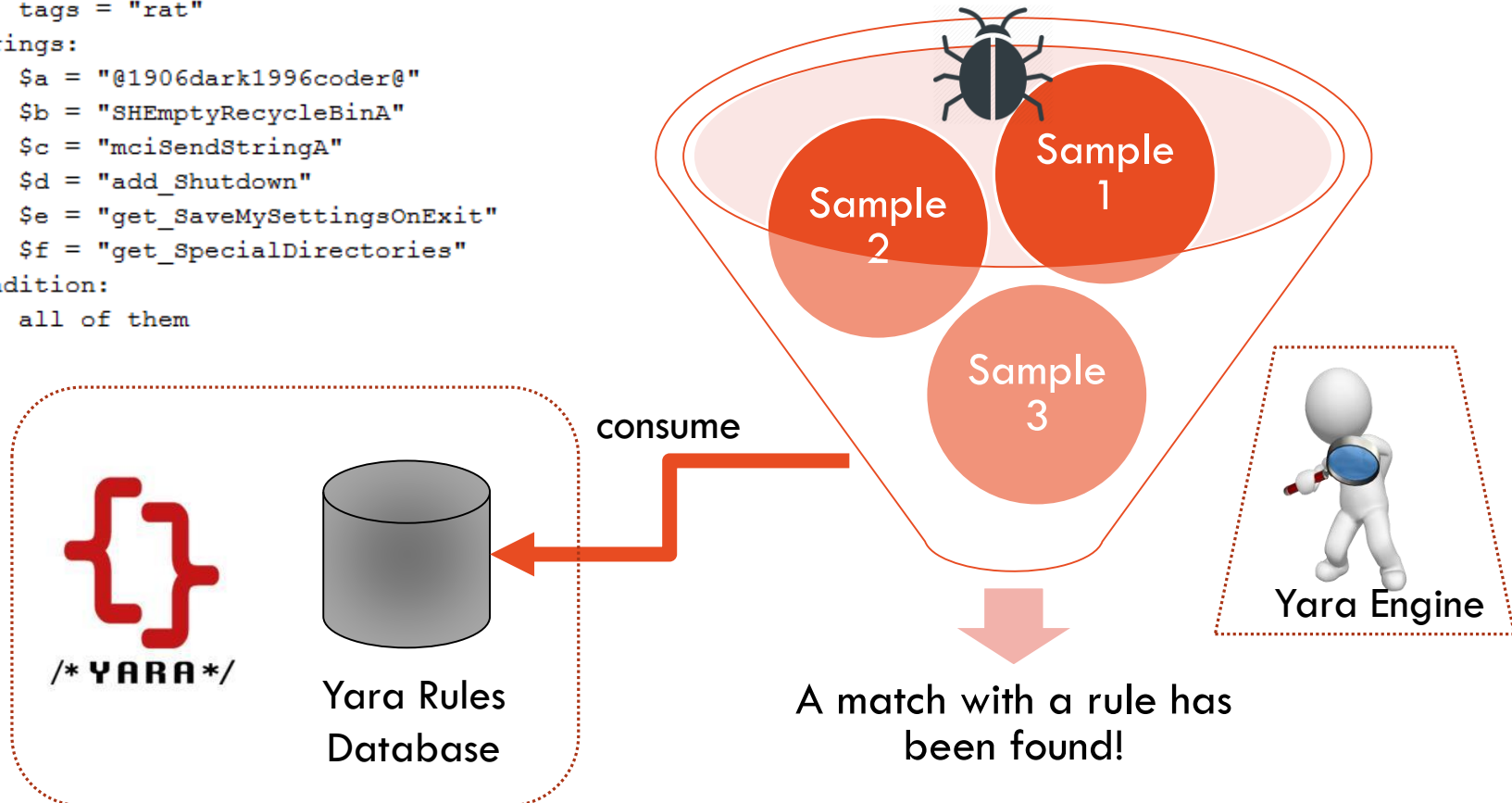


# 3. PROJECT IMPROVEMENTS

This is a Yara Rule

```
1 rule DarkRAT
2 {
3   meta:
4     date = "11/02/2015"
5     tags = "rat"
6   strings:
7     $a = "@1906dark1996coder@"
8     $b = "SHEmptyRecycleBinA"
9     $c = "mciSendStringA"
10    $d = "add_Shutdown"
11    $e = "get_SaveMySettingsOnExit"
12    $f = "get_SpecialDirectories"
13  condition:
14    all of them
15 }
```

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



# 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.striped == 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

```
prometheus:yarGen neo$ python yarGen.py -c -g ~/Downloads/GoogleChromePortable -i "GoogleChrome"
#####
          YARGEN
          Yara Rule Generator
          by Florian Roth
          February 2017
          Version 0.17.0
          #####
[+] Processing PEStudio strings ...
[+] Processing goodwill files ...
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/GoogleChromePortable.exe
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/help.html
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/readme.txt
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appcompactor.ini
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appicon.ico
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appicon_128.png
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appicon_16.png
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appicon_256.png
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appicon_32.png
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appicon_75.png
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/appinfo.ini
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/EULA.txt
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/AppInfo/installer.ini
[-] Extracting Strings: /Users/neo/Downloads/GoogleChromePortable/App/Chrome-bin/chrome.exe
```

# 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!

```

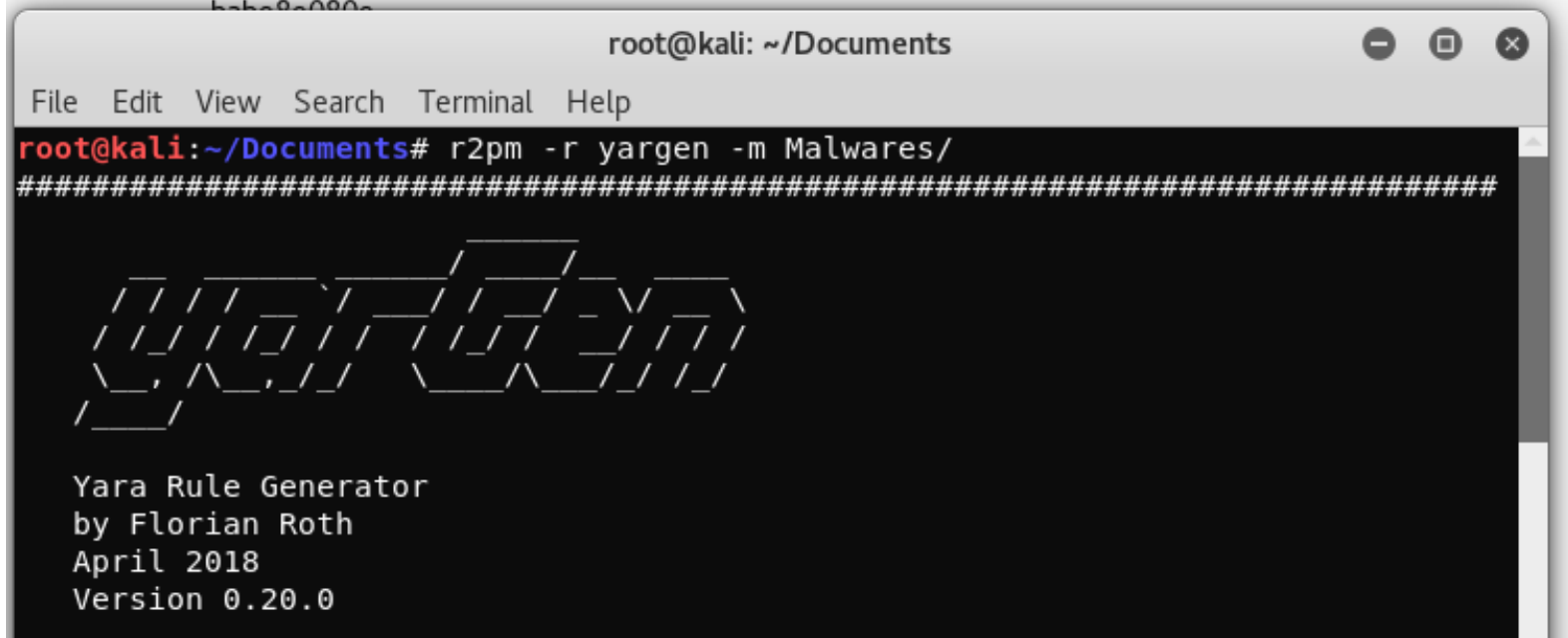
1  /*
2  Yara Rule Set
3  Author: YarGen Rule Generator
4  Date: 2015-07-09
5  Identifier: bin
6  */
7
8  /* Rule Set ----- */
9
10 rule backdoor {
11   meta:
12     description = "Auto-generated rule - file backdoor.exe"
13     author = "YarGen Rule Generator"
14     reference = "not set"
15     date = "2015-07-09"
16     hash = "bad8c7e6836b9a5679bfac0bc7483091e8e168f2"
17   strings:
18     $s0 = "%systemroot%\System32\rundll32.exe \"\" fullword ascii /* PEStudio Blacklist: str
19     $s1 = "c:\\Agenti\\SimpleVector\\Release\\SimpleVector.pdb" fullword ascii /* score: '28.
20     $s2 = "GetCurrentProcessID" fullword ascii /* PEStudio Blacklist: strings */ /* score: '2
21     $s3 = "<requestedExecutionLevel level='\"highestAvailable\" uiAccess='\"false\"/>" fullword
22     $s4 = "SOFTWARE\\Microsoft\\VisualStudio\\9.0\\Setup\\VS" fullword ascii /* PEStudio Blac
23     $s5 = "BG:\\oMpp" fullword ascii /* score: '12.00' */
24     $s6 = "vKPP80k.mKn" fullword ascii /* score: '12.00' */
25     $s7 = "<?xml version='\"1.0\" encoding='\"UTF-8\" standalone='\"no\"' ?><assembly xmlns='\"u"
26     $s8 = "0b55581e0a49451a01584c2a1d5223224559566318244a41405b172e11161932" fullword ascii /
27     $s9 = "SimpleVector, Version 1.0" fullword wide /* score: '9.00' */
28     $s10 = "* QN37" fullword ascii /* score: '7.00' */
29     $s11 = "SIMPLEVECTOR" fullword wide /* score: '6.50' */
30     $s12 = ".Ocl/2" fullword ascii /* score: '6.00' */
31     $s13 = "VH.IYl" fullword ascii /* score: '6.00' */
32     $s14 = "uKmtnQzd78" fullword ascii /* score: '5.00' */
33     $s15 = "About SimpleVector" fullword wide /* score: '5.00' */
34   condition:
35     uint16(0) == 0x5a4d and filesize < 3785KB and all of them
36 }
37

```

### 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

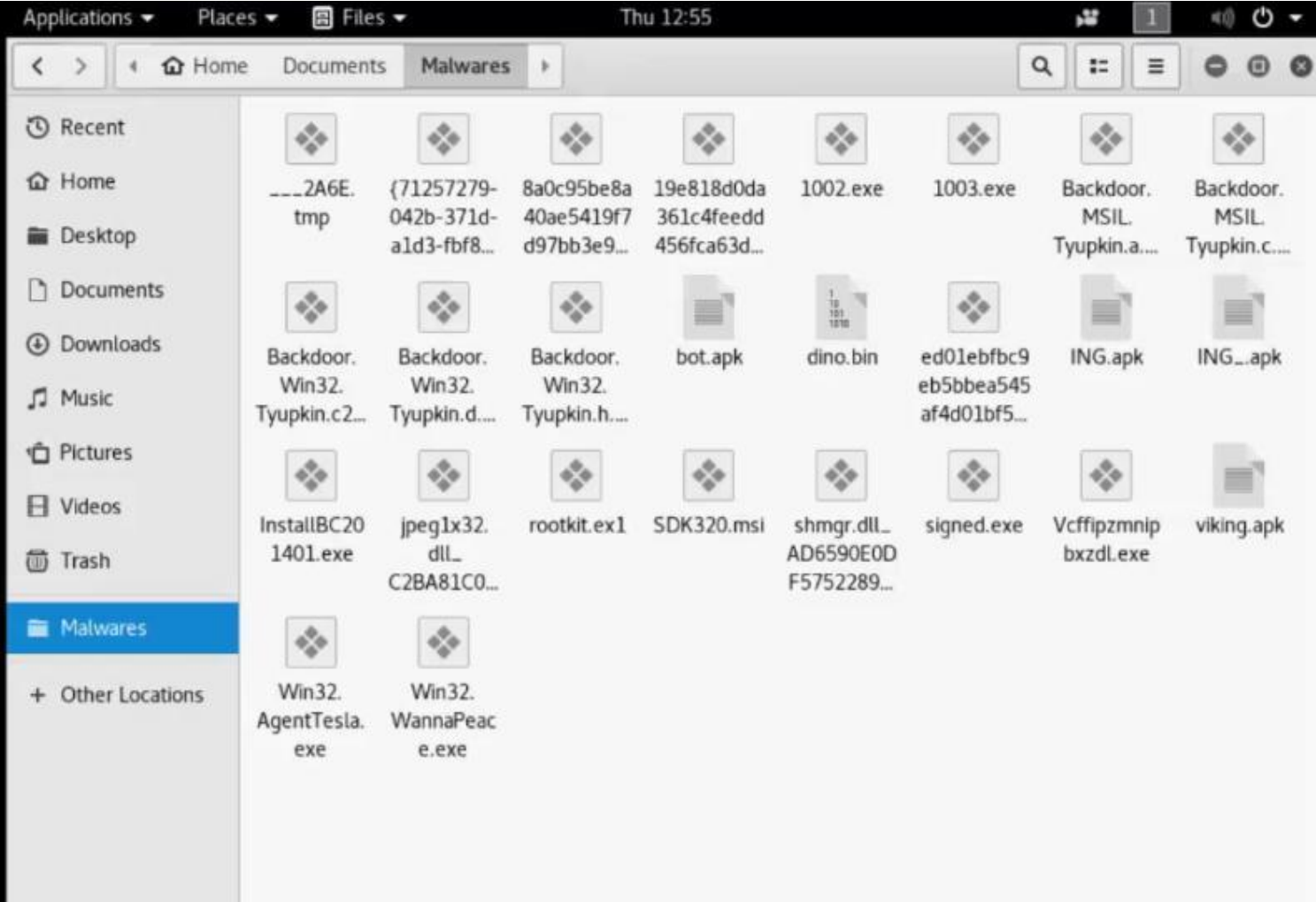
R2PM_INSTALL() {
    ${R2PM_SUDO} pip install pefile
    ${R2PM_SUDO} pip install scandir lxml naiveBayesClassifier
    mkdir -p "${R2PM_BINDIR}"
    rm -f "${B}"
    wget -O "${B}" -c "${SOURCE}"
    chmod +x "${B}" || R2PM_FAIL "Making executable yarGen"
    mkdir "${R2PM_BINDIR}"/lib
    wget -O "${R2PM_BINDIR}"/lib/good.txt -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](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**



OWASP  
**LATAM**  
2018  
LATIN AMERICA TOUR

**05 ABRIL - 25 DE MAYO**

