Android专项漏洞消灭之五杀

许雷永

@360 Vulpecker Team

360
www.360.cn

$whoami

许雷永

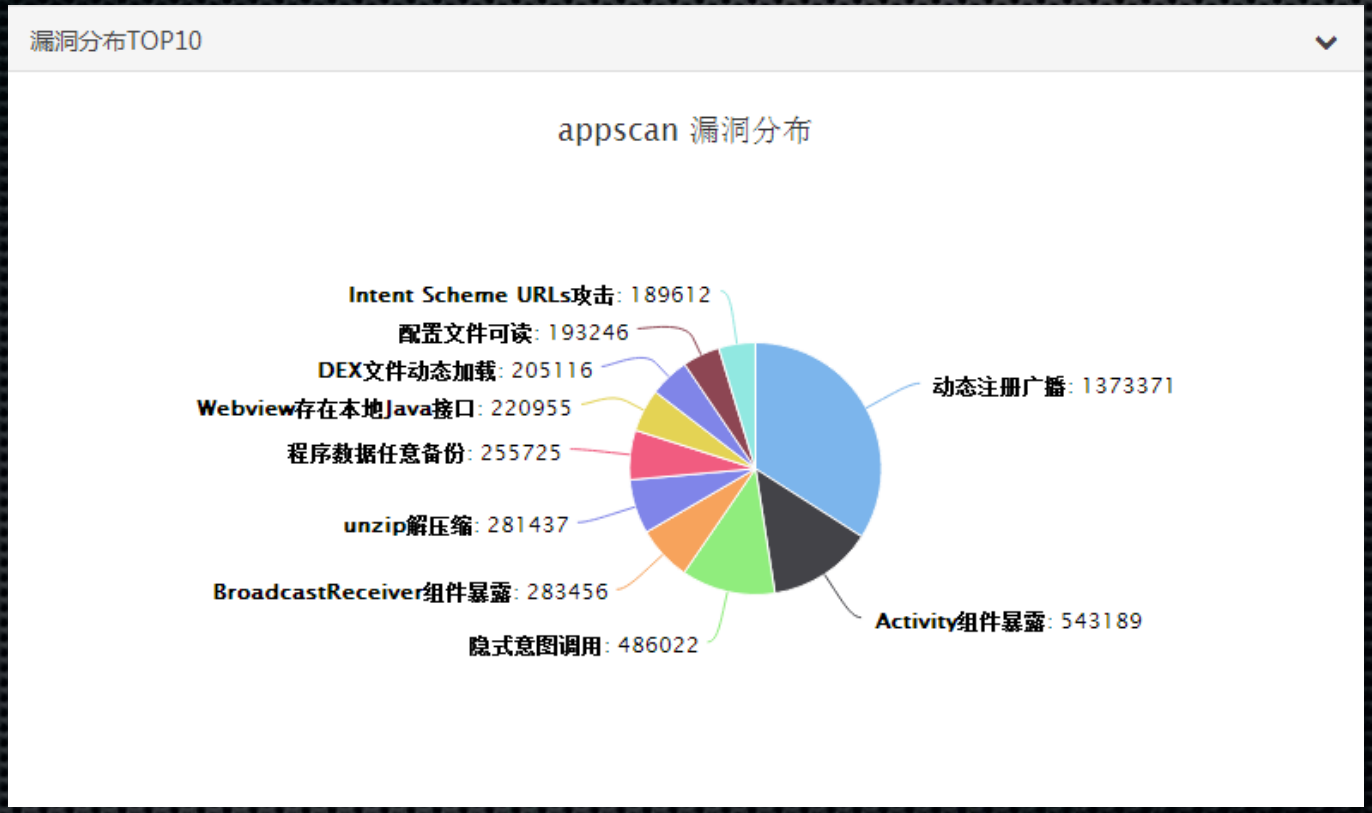
ID : BigFaceCat

来自360 Vulpecker Team的Android安全研究员，擅长Android APP的协议分析、逆向破解和算法还原以及漏洞挖掘、代码审计等。主要从事公司内部产品的安全审计，及时发现其中存在的安全漏洞。同时，也致力于Android应用层及系统层漏洞的挖掘工作，目前已挖掘多个分配CVE编号的漏洞，获得谷歌、三星、华为等多家手机厂商的致谢。

360
www.360.cn

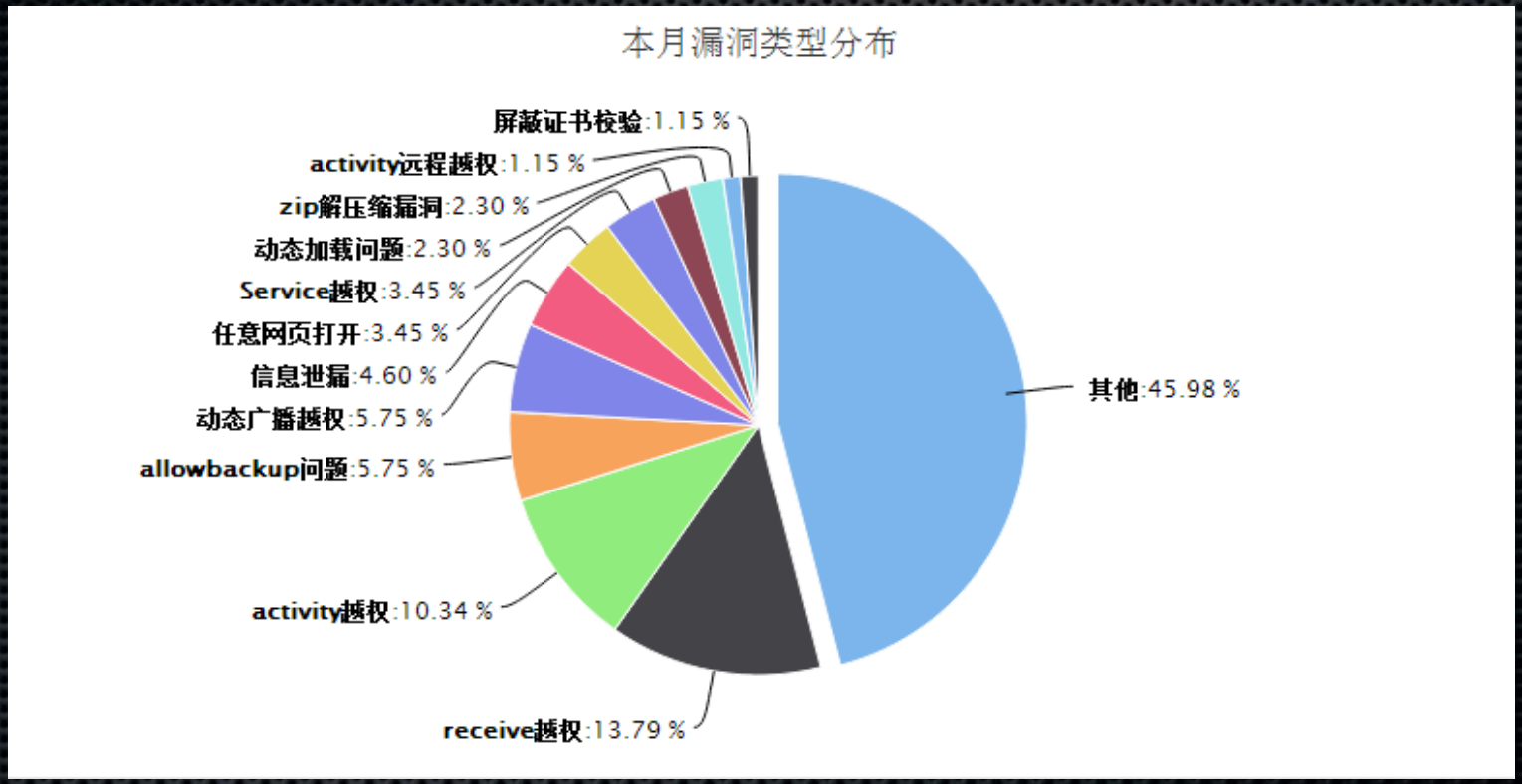
移动安全审计现状

Appscan外网扫描发现漏洞TOP10



移动安全审计现状

Appscan内网扫描 漏洞类型分布



360 www.360.cn

结论

在Android App安全审核中，大量时间花费在了常规漏洞的人工检测上，浪费大量劳动力

怎么办？

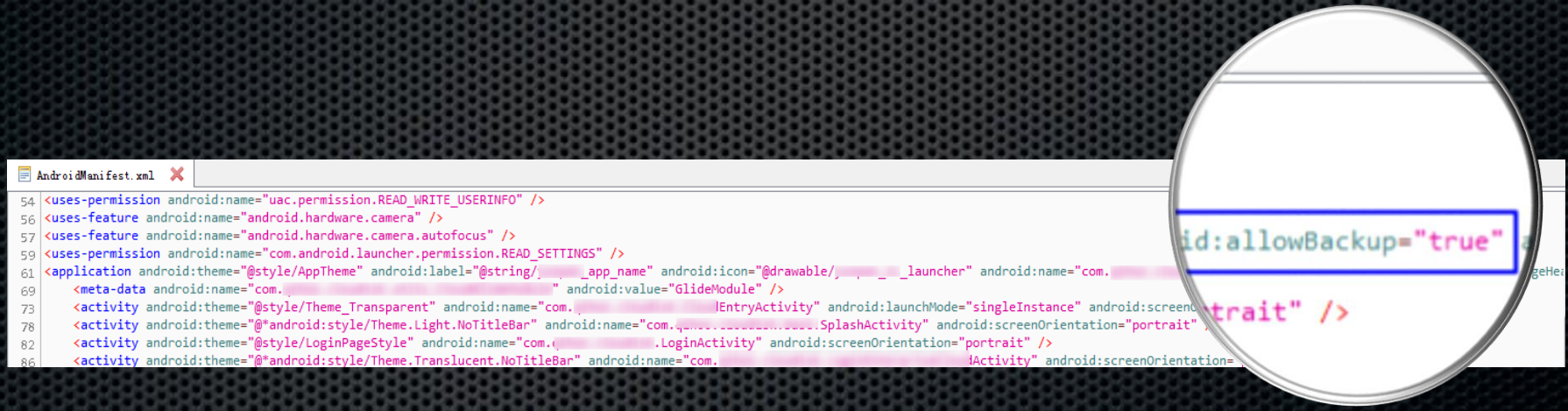
专项漏洞检测工具的诞生

- 针对每种特定类型的漏洞，分析其成因、特征，并据此开发出对应的检测工具
- QA同学使用漏洞扫描工具自测，在提交安全审核前反馈开发者，从根源上杜绝常见简单类型漏洞的产生

都有哪些检测呢？

First Blood —— 允许备份与可被调试漏洞

- 允许备份可导致用户数据泄露；
- 允许调试可导致应用被调试分析，极大降低了攻击门槛



First Blood —— 允许备份与可被调试漏洞

检测方法

```
private void startCheckBackup() {
    String pkgName = getIntent().getStringExtra("packageName");
    PackageInfo packageinfo = null;
    try {
        packageinfo = getPackageManager().getPackageInfo(pkgName, 0);
    } catch (Exception e) {
        e.printStackTrace();
    }
    if (packageinfo == null) {
        return;
    }
    ApplicationInfo info = packageinfo.applicationInfo;
    Boolean isDebuggable = false;
    if (0 != (info.flags & ApplicationInfo.FLAG_DEBUGGABLE)) {
        isDebuggable = true;
    }
    Boolean allowBackup = false;
    if (0 != (info.flags & ApplicationInfo.FLAG_ALLOW_BACKUP)) {
        allowBackup = true;
    }
```

Double Kill —— 通用拒绝服务漏洞

使用Intent接收数据时，未对传入的畸形数据进行异常处理，导致组件异常崩溃，存在本地拒绝服务漏洞。

360VulpeckerTeam于15年发现的通用型漏洞。

Double Kill —— 通用拒绝服务漏洞

```
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    if (VERSION.SDK_INT >= 19) {
        getWindow().addFlags(67108864);
        QkLog.d(TAG, "VERSION.SDK_INT =" + VERSION.SDK_INT);
    } else {
        QkLog.d(TAG, "SDK < 19");
        getWindow().addFlags(1024);
    }
    ScreenUtils.setDarkStatusIcon(this, true);
    this.mAudioManager = (AudioManager) getSystemService("audio");
    int sortMode = 1;
    if (icicle == null) {
        this.mSelectedUri = (Uri) getIntent().getParcelableExtra("android.intent.extra.ringtone.EXISTING_URI");
    } else {
        this.mSelectedUri = (Uri) icicle.getParcelable("android.intent.extra.ringtone.EXISTING_URI");
        this.mListState = icicle.getParcelable(LIST_STATE_KEY);
        this.mListHasFocus = icicle.getBoolean(FOCUS_KEY);
        sortMode = icicle.getInt(SORT_MODE_KEY, 1);
    }
```

修复建议：使用try/catch捕获所有异常

Double Kill —— 通用拒绝服务漏洞

检测方法

```
Intent intent = new Intent();
intent.setComponent(componentName);
intent.putExtra("Serializable_key", new IntentTest());

Message msg = Message.obtain();
msg.what = startWhich;
msg.obj = name;
handler.sendMessage(msg);

switch(listName.get(componentName)){
    case Util.MODULE_ACTIVITY:
        startActivity(intent);
        break;
    case Util.MODULE_SERVICE:
        startService(intent);
        break;
    case Util.MODULE_RECEIVER:
        sendBroadcast(intent);
        break;
}
```

```
public void handleApplicationCrash(IBinder app, ApplicationErrorReport.CrashInfo crashInfo) {
    ProcessRecord r = findAppProcess(app, "Crash");
    final String processName = app == null ? "system_server"
            : (r == null ? "unknown" : r.processName);

    handleApplicationCrashInner("crash", r, processName, crashInfo);
}

/* Native crash reporting uses this inner version because it needs to be somewhat
 * decoupled from the AM-managed cleanup lifecycle
 */
void handleApplicationCrashInner(String eventType, ProcessRecord r, String processName,
        ApplicationErrorReport.CrashInfo crashInfo) {
    EventLog.writeEvent(EventLogTags.AM_CRASH, Binder.getCallingPid(),
            UserHandle.getUserId(Binder.getCallingUid()), processName,
            r == null ? -1 : r.info.flags,
            crashInfo.exceptionClassName,
            crashInfo.exceptionMessage,
            crashInfo.throwFileName,
            crashInfo.throwLineNumber);

    addErrorToDropBox(eventType, r, processName, null, null, null, null, null, crashInfo);

    crashApplication(r, crashInfo);
}
```

Triple Kill —— 自升级中间人劫持漏洞

漏洞起因

应用在检查更新时由于使用不安全的传输协议且未对安装包进行校验导致可被中间人劫持替换为恶意apk，在用户手机上安装任意应用。

漏洞样例

```
private void installBySystem(final ApullDownloadArgs apullDownloadArgs) {
    this.mObserver.onInstalling(apullDownloadArgs);
    new Handler(Looper.getMainLooper()).postDelayed(new Runnable() {
        public void run() {
            ApullInstallHelper.this.mObserver.onInstall(apullDownloadArgs);
        }
    }, 2000);
    File file = new File(apullDownloadArgs.fileSavePath);
    Intent intent = new Intent("android.intent.action.VIEW");
    intent.setDataAndType(Uri.fromFile(file), "application/vnd.android.package-archive");
    intent.addFlags(268435456);
    this.mContext.startActivity(intent);
}
```

Triple Kill —— 自升级中间人劫持漏洞

漏洞检测

通过hook替换APK下载的链接，并监控系统安装事件。

```
private void hook_intent_install(LoadPackageParam lpparam){
    XposedHelpers.findAndHookMethod(
            "android.app.Activity",
            lpparam.classLoader,
            "startActivity",
            Intent.class,
            new intent_install_check(lpparam.packageName)
    );
}
```

```
private void hook_url_replace(LoadPackageParam lpparam){
    XposedHelpers.findAndHookMethod(
            "android.webkit.WebView",
            lpparam.classLoader,
            "loadUrl",
            String.class,
            new url_replece_hook(lpparam.packageName)
    );
    XposedHelpers.findAndHookMethod(
            "java.net.URL",
            lpparam.classLoader,
            "openConnection",
            new url_replece_hook(lpparam.packageName)
    );

    XposedHelpers.findAndHookMethod(
            "java.net.URL",
            lpparam.classLoader,
            "openConnection",
            java.net.Proxy.class,
            new url_replece_hook(lpparam.packageName)
    );
}
```

Quadra Kill —— 本地代码执行

漏洞起因

应用使用的本地的so、jar包、apk等可执行文件被设置成全局可读写的权限，可被修改为恶意内容，造成本地代码执行。

漏洞样例

```
root@hammerhead:/data/data/c                                    k/lib # ll
-rwxrwxrwx u0_a95    u0_a95       38064 2017-06-15 18:20 libgif.so
```

```
public final boolean a() {
    boolean v0_1;
    try {
        this.a(this.b.getInputStream(this.c), new FileOutputStream(new File(g.a(), this.a)));
        h.a(new File(g.a(), this.a));
        v0_1 = true;
    }
    catch(IOException v0) {
        v0.printStackTrace();
        v0_1 = false;
    }

    return v0_1;
}

private static int a(File arg3) {
    int v0;
    try {
        v0 = Runtime.getRuntime().exec("chmod 777 " + arg3.getAbsolutePath()).waitFor();
    }
    catch(IOException v1) {
        v1.printStackTrace();
    }
    catch(InterruptedException v1_1) {
```

360 www.360.cn

Quadra Kill —— 本地代码执行

漏洞检测

```
public void CheckPackageFiles(String Path, boolean IsIterative)
{
    File[] files = new File(Path).listFiles();
    for (int i = 0; i < files.length; i++)
    {
        File f = files[i];
        if (f.isFile())
        {
            FileInfo info = getFileInfo(f.getPath());
            if(checkOtherIsVul(info)) {
                listFile.add(f.getPath());
            }
            if (!IsIterative)
                break;
        }
        else if (f.isDirectory() && f.getPath().indexOf("/.") == -1)
            CheckPackageFiles(f.getPath(), IsIterative);
    }
}
```

Penta Kill —— 弱加密漏洞

漏洞样例

开发中对需要加密的敏感数据常采用硬编码的秘钥，
导致反编译apk后可拿到密钥并解密敏感数据。

```
private String d(String str) {
    if (str == null) {
        return null;
    }
    return g.b(str, "[jk1f%v6");
}
```

```
public static String b(String str, String str2) {
    String str3 = TokenKeyboardView.BANK_TOKEN;
    try {
        return new String(b.c(a(str.getBytes(), str2.getBytes()), 0));
    } catch (Throwable th) {
        f.b("Utils", TokenKeyboardView.BANK_TOKEN, th);
        return str3;
    }
}
```

```
public static byte[] a(byte[] bArr, byte[] bArr2) {
    byte[] bArr3 = null;
    if (bArr != null) {
        try {
            if (bArr.length > 0) {
                AlgorithmParameterSpec ivParameterSpec = new IvParameterSpec(bArr2);
                Key secretKeySpec = new SecretKeySpec(bArr2, "DES");
                Cipher instance = Cipher.getInstance("DES/CBC/PKCS5Padding");
                instance.init(1, secretKeySpec, ivParameterSpec);
                bArr3 = instance.doFinal(bArr);
            }
        } catch (Throwable th) {
            th.printStackTrace();
            f.b("Utils", "encryptDES error: ", th);
        }
    }
    return bArr3;
}
```

Penta Kill —— 弱加密漏洞

修复建议

- 网络传输敏感数据可通过动态协商确定加密算法的秘钥
- 本地存储的敏感数据，可根据设备硬件信息生成加密算法的秘钥

Penta Kill —— 弱加密漏洞

漏洞检测

```
//hook crypto related methods
XposedHelpers.findAndHookConstructor(SecretKeySpec.class, byte[].class, String.class, new XC_MethodHook() {
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {
        String key = Util.byteArrayToString((byte[]) param.args[0]);
        XposedBridge.log(TAG + "SecretKeySpec key " + key);
    }
});
XposedHelpers.findAndHookMethod(Cipher.class, "doFinal", byte[].class, new XC_MethodHook() {
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {
        String value = Util.byteArrayToString((byte[]) param.args[0]);
        XposedBridge.log(TAG + "value " + value);
    }
});
XposedHelpers.findAndHookMethod(Cipher.class, "getIV", new XC_MethodHook() {
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {
        XposedBridge.log(TAG + "IV " + param.getResult());
    }
});
XposedHelpers.findAndHookConstructor(IvParameterSpec.class, byte[].class, new XC_MethodHook() {
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {
        XposedBridge.log(TAG + "IV " + Util.byteArrayToString((byte[]) param.args[0]));
    }
});
XposedHelpers.findAndHookMethod(Cipher.class, "getInstance", String.class, new XC_MethodHook() {
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {
        XposedBridge.log(TAG + "Cipher " + param.args[0]);
    }
});
```

需手动触发
加密过程

总结

- 人工参与检测，手动触发相关功能，从而检测该过程是否存在漏洞。

- 通过专项检测工具，检测出APP中大量的常规漏洞，将审核人员从大量重复劳动中解救出来。

未来安全审计的发展

平台化扫描检测

360显危镜：appscan.360.cn 免费，支持静态分析、动态扫描

其他情况？

360 IoT安全守护计划

360行车记录仪

360智能摄像机

360手机

360儿童手表

360安全路由

竟然给CVE

360安全卫士

360手机卫士

360安全路由

360 www.360.cn

Thanks

许雷永

@360 Vulpecker Team